UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria dell'informazione

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

GESTIONE DI FLUSSI VIDEO A DESCRIZIONI MULTIPLE SU RETI P2P TRAMITE GIOCHI COOPERATIVI

Relatore:
Giancarlo Calvagno

Laureando:
STEFANO BUSATO

Correlatore:
Simone Milani

14 marzo 2011

Indice

Sommario

In questa tesi sono presentati dei meccanismi di classificazione di pacchetti video, codificati tramite codifica a descrizioni multiple (MDC). Lo scopo del lavoro riguarda la trasmissione efficiente di flussi video attraverso reti P2P. A tal fine, la sequenza video viene dapprima divisa in diversi sottoinsiemi di informazione chiamati *descrizioni*. Quando tutte le descrizioni sono ricevute correttamente, il decodificatore è in grado di ricostruire il segnale originario, mentre se si verificano degli errori di trasmissione è possibile sfruttare la ridondanza condivisa tra tutte le descrizioni per stimare i dati persi e ricostruire una approssimazione del segnale originale. Questo tipo di codifica si è dimostrato particolarmente efficace per la distribuzione di contenuti audio/video attraverso canali affetti da perdite di vario genere. Nel primo capitolo sono quindi definite le problematiche tipiche di una trasmissione su reti *peer-to-peer* che portano ad una inevitabile perdita di pacchetti. La natura di queste perdite può infatti dipendere dallo stato del canale, dalle congestioni della rete e dai ritardi di trasmissione. Nel secondo capitolo sono illustrati nel dettaglio diversi meccanismi di codifica a descrizioni multiple e lo schema utilizzato per la codifica di sequenze video. Lavori di studi precedenti, hanno dimostrato come la suddivisione dei pacchetti in diverse classi di servizio a diversa priorità possa portare ad un miglioramento delle prestazioni ottenute attraverso la sola applicazione della codifica MD. Questo lavoro di tesi si propone di unire i vantaggi della codifica MD all'utilizzo di diverse classi di servizio, attraverso l'introduzione di algoritmi efficienti di classificazione per i pacchetti appartenenti alle diverse descrizioni. Questi algoritmi fanno uso di quella teoria nota come *Teoria dei Giochi*, definendo il problema di classificazione come un gioco in cui le descrizioni della codifica MDC rappresentano i giocatori, mentre le classi di servizio dei diversi pacchetti corrispondono alle possibili strategie da adottare. In particolare, i meccanismi di classificazione sviluppati utilizzano un approccio *cooperativo*, in cui i diversi giocatori hanno la possibilità di coalizzarsi tra loro per migliorare il proprio guadagno. La teoria dei giochi è descritta nel terzo capitolo, mentre nel quarto capitolo è definito un particolare meccanismo chiamato *DiffServ*, che permette di classificare i pacchetti all'interno di una

rete in base al loro livello di importanza. La classificazione prevede la suddivisione dei pacchetti in tre diverse classi di servizio secondo le specifiche definite dal protocollo srTCM (introdotto anch'esso nel quarto capitolo). Nel quinto capitolo è stato definito nel dettaglio lo schema di codifica implementato e i diversi meccanismi di classificazione sviluppati attraverso la teoria dei giochi cooperativi. Infine, nel Capitolo 6, sono illustrati i modelli di rete realizzati per la simulazione dell'intero sistema. Le diverse architetture sono state create attraverso l'ambiente di simulazione NS2 (*Network Simulator version 2*). Sono state realizzate varie topologie di rete di diversa complessità, in modo da riprodurre più fedelmente possibile le strutture tipiche di una rete P2P. Infine, per ogni modello di rete e meccanismo di classificazione implementato, sono stati riportati i risultati ottenuti, i grafici e le conclusioni.

Capitolo 1

Introduzione

Negli ultimi anni, le elevate prestazioni ottenute attraverso algoritmi di codifica sempre più sofisticati, combinate con la crescita continua di internet e delle reti cellulari, hanno portato allo sviluppo di una nuova famiglia di servizi di comunicazione che riguarda la distribuzione di contenuti audio/video attraverso canali affetti da perdite di vario genere.

Uno dei principali contesti in cui il fenomeno di scambio di contenuti audio/video (es. video *streaming*) si è sviluppato maggiormente è quello delle reti *peer to peer* (P2P). Recentemente, i servizi *peer-to-peer* hanno avuto una notevole crescita e sono diventati degli strumenti molto popolari in internet, specialmente per lo scambio di file [1]. L'idea è stata prima largamente sviluppata da "Napster" che fu poi seguito da molti altri servizi come "Gnutella" ed "E-Donkey". In particolare una rete peer-to-peer permette ad un gruppo di utenti di connettersi fra loro e di condividere le risorse in loro possesso. La condivisione di queste risorse permette una distribuzione efficace e flessibile dei servizi offerti dalla rete (ad esempio condivisione di file). La caratteristica principale di una rete peer to peer è che, al suo interno, ogni nodo (peer) ha funzioni equivalenti. Un *peer* è accessibile direttamente da un qualsiasi altro nodo della rete senza passare attraverso altre entità, in questo modo i partecipanti alla rete possono sia fornire che richiedere servizi. Inoltre qualsiasi nodo può essere rimosso dalla rete senza che nessuno dei servizi offerti precedentemente sia compromesso. Questo va ad eliminare parte dei problemi esistenti in un'architettura di tipo *client/server*. Un'architettura client/server infatti è generalmente composta da un server e più client. Il server possiede una potenza di calcolo superiore a tutti gli altri nodi ed è l'unico in grado di fornire tutti i servizi e i contenuti presenti nella rete. Un client può solo richiedere file o l'esecuzione di servizi al server senza condividere alcuna delle sue risorse. Una rete di questo tipo, inoltre, non è in grado di gestire un elevato numero di richieste simultanee di connessione: persino i più grandi server di contenuti multimediali non sono in

grado di gestire più di poche centinaia di sessioni attive contemporaneamente. Inoltre come ogni altro modello client/server, il server è un "punto critico" in quanto se si guasta, la trasmissione viene interrotta e la risorsa rimane isolata. Questi problemi, nelle reti P2P possono essere risolti in quanto ogni nodo della rete può recitare la parte sia di client, ma anche di server ed inoltre, con l'aumentare dei partecipanti alla rete, raggiungere prestazioni superiori ad una rete centralizzata. Inoltre una rete P2P è molto più tollerante ai guasti in quanto uno stesso contenuto multimediale può essere duplicato ed apparire presente in più nodi della rete eliminando la criticità di avere un solo server contenente una data informazione.

Sebbene presentino degli indubbi vantaggi rispetto alle architetture client/server, anche nelle reti P2P esistono problemi legati alla perdita di pacchetti. La natura di queste perdite può infatti dipendere dallo stato del canale, dalle congestioni della rete e dai ritardi di trasmissione. Le trasmissioni su reti P2P prevedono, infatti, il passaggio di informazione attraverso *link* di diverse capacità. In questo caso, se la capacità di un link è inferiore rispetto al rate di traffico che lo attraversa, alcuni pacchetti saranno inevitabilmente persi. Un secondo problema è legato alle congestioni. Nel caso in cui, all'arrivo di pacchetti in un nodo di scambio, il buffer di ricezione è pieno, i pacchetti saranno scartati. Una delle principali tecniche adottate per sopperire a questa perdita di informazione è quella nota come *Automatic Repeat reQuest* (ARQ), che consiste nella ritrasmissione dei pacchetti persi. In questo caso, utilizzando adeguati codici di canale, si riesce infatti a riconoscere quali pacchetti sono stati persi e si procede alla richiesta di ritrasmissione. Se le perdite sono sporadiche questa tecnica è molto efficiente: in caso di perdita, un pacchetto è ritrasmesso una sola volta. Se invece le perdite sono frequenti, le continue ritrasmissioni possono incrementare il livello di congestione e di conseguenza la percentuale di perdita di pacchetti, portando ad un'amplificazione dell'effetto. Le ritrasmissioni sono molto utili in comunicazioni punto-punto dove è disponibile un canale di *feedback*. Un protocollo che fa uso di ARQ è TCP (*Trasmission Control Protocol*) [2], il quale consente la numerazione dei pacchetti e la loro ritrasmissione in caso di perdita, garantendo così il corretto arrivo di tutte le informazioni. Per applicazioni come video streaming, il ritardo di arrivo dei pacchetti è un fattore rilevante. Limitare la variazione del tempo di interarrivo tra i pacchetti (*jitter*) è uno dei problemi principali da risolvere per ottenere una riproduzione fluida del video ricevuto. Se il livello di congestione è troppo elevato il protocollo TCP non è più adatto dal momento che in applicazioni multimediali, è preferibile perdere un pacchetto piuttosto che riceverlo con un forte ritardo. Per questi motivi, nelle applicazioni multimediali si utilizza il protocollo UDP (User Datagram Protocol) il quale non stabilisce alcuna connessione e non garantisce l'ar-

rivo a destinazione dei pacchetti ma consente una loro più rapida consegna. Ad esso viene affiancato un protocollo di livello superiore RTP (*Real Time Protocol*) [3] che fornisce informazioni aggiuntive non implementate da UDP (es. la numerazione dei pacchetti) che permettono di capire quali pacchetti sono stati persi e ricostruire l'esatta riproduzione della sequenza ricevuta.

Una soluzione alternativa alla ritrasmissione dei pacchetti è quella di affiancare alla codifica di sorgente una codifica di canale più efficiente di quella utilizzata nel modello ARQ, per cercare di ricostruire i pacchetti erroneamente ricevuti o persi.

Per poter trasmettere un video, una codifica di sorgente è sempre necessaria per ridurre la quantità di bit da trasmettere attraverso una compressione mirata ad eliminare la ridondanza spaziale e temporale del segnale video. Gli standard di codifica (ad esempio MPEG-4 [4] e H.264/AVC [5]) definiscono un formato per il video codificato ed hanno attualmente raggiunto degli ottimi livelli di compressione a discapito dell'introduzione di una certa distorsione anche se molto limitata (compressione con perdite). La codifica di canale invece mira a ricostruire i pacchetti ricevuti erroneamente o persi attraverso la trasmissione di pacchetti ridondanti che permettano il recupero di quelli corrotti. Questa tecnica è nota come FEC (Forward Error Correction). Utilizzando schemi di tipo FEC però, la complessità aumenta notevolmente: codificare e decodificare pacchetti ridondanti richiede della memoria aggiuntiva ed elevate risorse computazionali. Inoltre il recupero di pacchetti al ricevitore richiede spesso l'elaborazione di un'elevata quantità di dati, introducendo un'altra causa di ritardo. La Codifica a Descrizioni Multiple può essere classificata come una codifica congiunta sorgente-canale che non prevede né l'inoltro di pacchetti ridondanti e nemmeno la ritrasmissione dei pacchetti persi. L'idea è quella di incrementare la robustezza della trasmissione dividendo il segnale d'ingresso in più parti chiamate *descrizioni*, inviare le descrizioni attraverso canali distinti e poi ricostruire il segnale originario a partire dalle descrizioni ricevute correttamente. Questa tecnica sarà trattata nel prossimo capitolo e si dimostrerà molto efficace per la trasmissione di sequenze video in reti peer-to-peer.

Capitolo 2

La Codifica a Descrizioni Multiple

2.1 Introduzione

La perdita di informazione dovuta a guasti, congestioni e un peggioramento del rapporto segnale/rumore sui canali di trasmissione sono comuni in molti sistemi di comunicazione. In particolare, nella codifica video, se uno o più pacchetti dello *stream* vengono persi, il video decodificato risulterà fortemente corrotto. Negli ultimi anni, molte tecniche sono state studiate per sopperire a queste inevitabili perdite di dati attraverso meccanismi di ritrasmissione dei pacchetti persi (Automatic Repeat reQuest, ARQ) o recupero delle informazioni attraverso l'utilizzo di particolari codici di canale (Forward Error Correction, FEC).
La tecnica di Automatic Repeat reQuest (ARQ) [6] è adottata per comunicazioni punto-punto in cui è disponibile un canale di *feedback*, mentre non trova utilizzo per comunicazioni *broadcast*, *video streaming* o *real-time* a causa dei forti ritardi che la ritrasmissione di pacchetti comporterebbe e della difficoltà di gestire più canali di ritorno. La tecnica *Forward Error Correction* (FEC) invece non richiede la presenza di un canale di *feedback*, non introduce particolari ritardi ed è quindi preferibile per trasmissioni video *broadcast* e *real-time*. Essa però comporta un aumento considerevole della complessità di codificatore e decodificatore che implementano il codice di correzione. Inoltre questa è una tecnica le cui prestazioni sono del tipo "all-or-nothing", il che significa che se gli errori sono contenuti entro la soglia di protezione del codice essi saranno corretti perfettamente, mentre nel caso in cui gli errori superino la "capacità" di correzione del codice, esso non riesce a recuperare alcuna informazione e molti pacchetti saranno scartati. Lo scopo principale della codifica a Descrizioni Multiple è quindi quello di consentire una trasmissione robusta dell'informazione eliminando gli svantaggi delle due tecniche precedenti.

2.1.1 Codifica a descrizioni multiple (MDC)

La codifica a descrizioni multiple consiste nel dividere il segnale d'ingresso in *N* flussi di dati correlati tra loro chiamati *descrizioni*. Quest'ultime saranno poi codificate indipendentemente l'una dall'altra ed inoltrate su canali diversi [7].
In ricezione, se tutte le descrizioni sono ricevute correttamente, il segnale originale codificato può essere ricostruito. In caso contrario è possibile sfruttare la correlazione tra le varie descrizioni per stimare l'informazione persa dalle descrizioni ricevute correttamente ed ottenere così una versione approssimata del segnale d'ingresso.
La codifica a descrizioni multiple non richiede quindi nè un canale di *feedback* nè la conoscenza del canale di trasmissione al fine di generare un codice per proteggere lo stream da errori. Inoltre un'eventuale ridondanza di informazione è introdotta dal codificatore della sorgente al momento della generazione delle descrizioni in quanto i vari sottosegnali risultano correlati tra loro. La codifica a descrizioni multiple, inoltre, non ha un comportamento del tipo *all-or-nothing*: a differenza della tecnica FEC, in caso di perdita di pacchetti le prestazioni subiscono un peggioramento graduale e non esiste una soglia di errori "permessi" oltre la quale le prestazioni decadono vertiginosamente.
Per capire meglio l'utilizzo effettivo della tecnica di codifica a descrizioni multiple è però necessario definire meglio l'associazione tra il concetto di "descrizione" ed un'entità logica. L'associazione più comune è quella con la singola unità di informazione inviata da un mittente e ricevuta da uno o più destinatari attraverso una rete di trasmissione dati. La codifica a descrizioni multiple si dimostra quindi particolarmente utile quando:

- un destinatario non riceve una o più descrizioni a causa di perdite o congestioni nella rete di trasmissione;
- i destinatari del flusso d'informazione richiedono differenti livelli di qualità di servizio.

Come già accennato, nel caso in cui il destinatario dell'informazione riceva tutti i dati inoltrati, l'informazione codificata potrebbe essere perfettamente recuperata sia con, che senza l'utilizzo della scomposizione in diverse descrizioni. Nel caso in cui, però, il mezzo trasmissivo sia soggetto a perdite, la tecnica MDC permette il recupero dell'informazione trasmessa ad una qualità inferiore (proporzionalmente alla percentuale di pacchetti persi), il che è preferibile ad una totale mancanza dell'informazione persa. La tecnica di codifica a descrizioni multiple, inoltre, permette di fornire un unico servizio ad un insieme eterogeneo di utenti. Alcuni di essi, infatti, mediante la sottoscrizione di un

contratto di servizio e grazie alla capacità di decodificare più descrizioni, possono usufruire di un livello qualitativo maggiore rispetto a chi è in grado di decodificare solo alcune elle descrizioni ricevute.

2.2 Il Modello channel splitting

La tecnica di codifica a descrizioni multiple è nata negli anni ’70 nei “Bell Laboratories” come meccanismo di trasmissione vocale su rete telefonica. Il problema che le ricerche nei Bell Laboratories miravano a risolvere era quello di migliorare l’affidabilità dei collegamenti telefonici in modo da evitare l’interruzione del servizio in caso di guasto. La prima soluzione ideata consisteva nel raddoppiare tutte le linee esistenti mediante linee di riserva sulle quali veniva dirottato il traffico voce in caso di guasto. Tuttavia, in condizioni normali queste linee supplementari rimanevano inutilizzate. Per sopperire a tale spreco venne quindi proposta una soluzione alternativa nota con il nome di “channel splitting”. In base a questo modello, il segnale veniva diviso in due flussi (descrizioni) che erano poi trasmessi su due canali distinti. In assenza di guasti, le due descrizioni, potevano essere ricombinate a destinazione, per ricreare il segnale originario, mentre nel caso in cui una delle due descrizioni fosse andata persa, l’altra avrebbe impedito l’interruzione della comunicazione, anche se con una qualità inferiore.

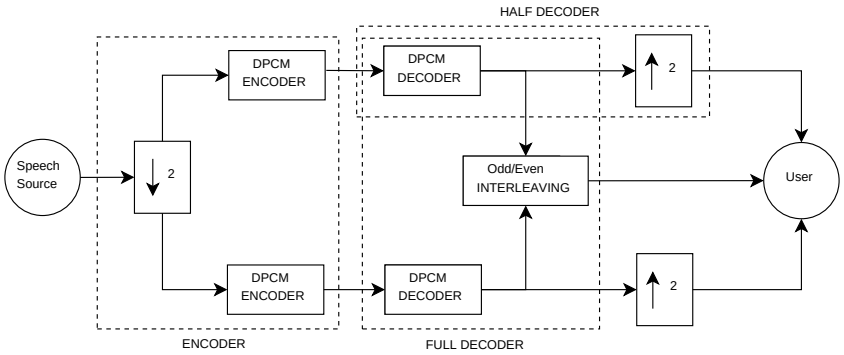


Figura 2.1. Il Modello Channel Splitting proposto da Jayant.

Lo schema di Figura 2.1 fu proposto da Jayant, un ricercatore dei Bell Laboratories, nel 1980 [7]. Secondo questo modello, i coefficienti del segnale vocale campionato vengono dapprima divisi in campioni pari e dispari in modo da ottenere un sottocampionamento di un fattore 2. I due insiemi di campioni costituenti le due descrizioni vengono poi codificati separatamente utilizzando due decodificatori DPCM (Differential Pulse Code Modulation) ed inviati su canali

diversi.
Il decodificatore è invece composto da tre diverse parti. Collegato a ciascuno dei due canali in cui arrivano le descrizioni è presente un decodificatore DPCM seguito da un interpolatore. Questi due blocchi identici costituiscono i due decodificatori laterali chiamati anche "half decoder" che sono in grado di ricostruire il segnale inoltrato a partire da una sola delle due descrizioni. Il terzo blocco è invece composto da un *interleaver* che lavora sui campioni all'uscita dei due decodificatori DPCM in modo da "unire" le due descrizioni e ricostruire il segnale originario. I due decodificatori DPCM assieme all'interleaving formano il decodificatore centrale che è anche chiamato "full decoder". Nel caso in cui entrambe le descrizioni arrivano correttamente a destinazione esso riesce a ricostruire il segnale d'ingresso con il rate effettivo di campionamento. Utilizzando solo una descrizione, invece, i due decodificatori laterali riusciranno a ricostruire il segnale ad un rate dimezzato, da qui il nome di half decoder. Nello schema proposto da Jayant la sorgente è campionata a 12 kHz, a differenza dei normali standard di codifica dell'epoca che prevedevano una frequenza di campionamento di 8 kHz. In questo modo, i due flussi costituenti le due descrizioni avranno una frequenza di campionamento di 6 kHz, il che comporta un livello di interferenza di intersimbolo (*aliasing*) abbastanza contenuto, avendo il segnale vocale d'ingresso una banda generalmente compresa entro i 3.2 kHz.

Passiamo ora ad un'analisi più approfondita dello schema, considerando una sorgente d'informazione discreta del tipo:

$$x[k] = \rho x[k-1] + w[k] \tag{2.1}$$

dove $k \in \mathbf{Z}$, $x[k]$ sono i campioni del segnale d'ingresso e $w[k]$ è una sequenza di variabili aleatorie gaussiane i.i.d. e a media nulla. La quantità ρ, $|\rho| < 1$ è la correlazione tra tra due campioni consecutivi del segnale d'ingresso. Ponendo la varianza di $w[k]$ pari a $1 - \rho^2$ il processo $x[k]$ diviene a potenza unitaria.

Una misura della distorsione che si incontra attraverso la quantizzazione di un processo di questo tipo si ottiene esaminando la funzione *Rate Distortion*:

$$D(R) = (1-\rho^2)2^{-2R}, \ \text{ per } \ \mathrm{R} \geq \log_2(1+\rho) \tag{2.2}$$

La separazione in campioni pari e dispari porta a due nuovi processi con correlazione ρ^2 minore della precedente e più difficili da "comprimere", causando, a parità di rate, una maggior distorsione. La funzione *Rate Distortion* del decodificatore centrale è ora data da:

$$D_{full}(R) = (1-\rho^4)2^{-2R}, \ \text{ per } \ \mathrm{R} \geq \log_2(1+\rho^2) \tag{2.3}$$

Ciascuno dei due decodificatori laterali ha la stessa distorsione del decodificatore centrale per i campioni ricevuti, alla quale si somma però un errore aggiuntivo causato dall'interpolazione lineare per ottenere i campioni mancanti. L'errore quadratico medio portato dall'operazione di interpolazione è dato da:

$$D_{interp}(R) = (1-\rho)^2 + \frac{1}{2}(1-\rho^2) + \omega D_q \tag{2.4}$$

dove D_q è la varianza dell'errore di quantizzazione e $\omega \in [0,1]$ dipende dalla correlazione con l'errore di quantizzazione.
Mediando i due contributi di distorsione otteniamo la funzione *Rate Distortion* per i due decodificatori laterali che risulta pari a :

$$D_{half}(R) = \frac{1}{2}\left[(1-\rho)^2 + \frac{1}{2}(1-\rho^2)\right] + \frac{1+\omega}{2} D_{full}(R) \tag{2.5}$$

Dall'espressione si può notare che per rate elevati il primo termine costante risulta dominante sul secondo che invece tende a zero, portando ad un degrado delle prestazioni nel caso di ricezione di una sola descrizione.
Nonostante questo, test percettivi hanno mostrato che usando dai 2 ai 5 bit per campione (bit-rate da 24 a 60 kbit/s) la qualità del segnale vocale ottenuto da un singolo canale laterale è paragonabile a quella ottenuta unendo le due descrizioni attraverso il decodificatore centrale [7].

2.3 Teoria dell'informazione per MD

La realizzazione del modello channel splitting pose la seguente questione: se una sorgente di informazione è descritta da due descrizioni separate quali limitazioni subisce la qualità di tali descrizioni prese separatamente o congiuntamente? Wyner sollevò questa questione nel 1979 e nel settembre dello stesso anno, con l'aiuto di Witsenhausen, Wolf e Ziv, al workshop sulla teoria dell'informazione, presentò i primi risultati sperimentali. Questa questione divenne poi nota con il nome di "Problema MD" [7].

2.3.1 Il Problema MD

La tipica situazione per uno schema di codifica a descrizioni multiple è illustrata in Figura 2.2. Un codificatore codifica una sequenza di simboli $\{X_k\}_{k=1}^{N}$ e la trasmette su due canali ideali (non corrotti da rumore). In ricezione, un decodificatore (decodificatore centrale) riceve l'informazione inoltrata su entrambi i canali mentre i restanti due decodificatori (decodificatori laterali) ricevono

l'informazione trasmessa solo sui loro rispettivi canali. Il rate trasmissivo sul canale i-esimo è definito come $R_i,\ i = 1, 2$ ed è espresso in bit per campione. La sequenza ricostruita prodotta dal decodificatore i-esimo è indicata da $\{\hat{X}_k^{(i)}\}_{k=1}^N,\ \ i = 0, 1, 2$ e le distorsioni ottenute attraverso queste ricostruzioni sono indicate da $D_i,\ \ i = 0, 1, 2$.

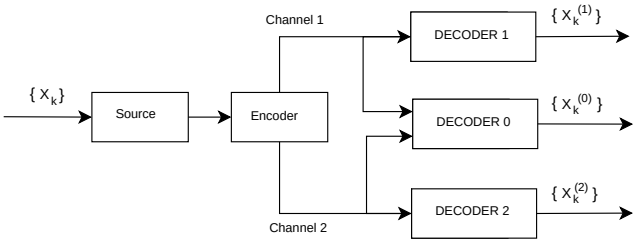


Figura 2.2. Schema del sistema di codifica a descrizioni multiple con due canali e tre ricevitori.

Una situazione del genere si può avere sia quando ci sono tre diversi utenti o classi di utenti in modo da realizzare una trasmissione broadcast, sia quando c'è invece un solo utente che può trovarsi in uno dei tre stati a seconda di quante e quali descrizioni riceve correttamente. L'ottimizzazione della codifica a descrizioni multiple è complessa a causa di due richieste contrastanti: Se si costruisce un buon codificatore per la prima descrizione al rate R_1 da inoltrare sul primo canale ed un altro altrettanto buono al rate R_2 per la seconda descrizione non è garantito che questo sia un buon modo per utilizzare $R_1 + R_2$ bit per la codifica complessiva. Allo stesso modo, se si cerca di ottimizzare il rate totale di codifica $R = R_1 + R_2$ potrebbe non essere semplice costruire due descrizioni "utili" dividendo opportunamente il rate R. Euristicamente si può pensare che ottimizzando separatamente i due rate di codifica per le due descrizioni R_1 ed R_2, le due descrizioni ottenute saranno tra loro simili e combinandole al decodificatore centrale non si otterrà un vantaggio così superiore rispetto a considerare la migliore delle due. Creare descrizioni individualmente ottime ma diverse tra loro è il *trade-off* fondamentale della codifica MD. Il problema teorico principale è dunque quello di determinare, per una data sorgente e per una data misura della distorsione, l'insieme dei valori ammissibili per la quintupla $(R_1, R_2, D_0, D_1, D_2)$ e tra essi scegliere quelli che portano alle migliori prestazioni per una successiva implementazione pratica di codificatori e decodificatori.

2.4 Quantizzazione Scalare

In questa sezione si valuta l'utilizzo di un modello channel splitting per una sorgente priva di "ridondanza" per capire quali sono le difficoltà nel creare le descrizioni. A tal fine si considera la trasmissione di un numero reale $x \in [-1, 1]$ con 4 bit.

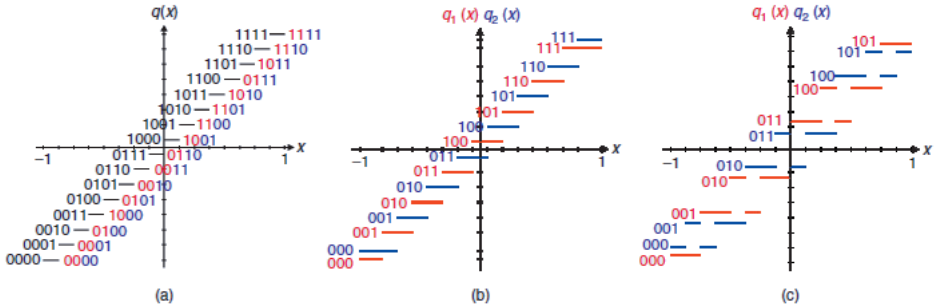


Figura 2.3. (a) Quantizzatore scalare a 4 bit, (b) Due quantizzatori uniformi a 3 bit e con diverso offset, (c) Due quantizzatori più complicati che combinati assieme permettono di ottenere una risoluzione di 4 bit/campione.

La scelta naturale, che è anche la migliore per una sorgente uniformemente distribuita nell'intervallo è quella di utilizzare un quantizzatore uniforme come mostrato in Figura 2.3(a). Ad ogni modo non ci sono modi semplici ed efficienti per dividere i 4 bit in due descrizioni in modo che ognuna di esse rappresenti un buona approssimazione del valore trasmesso. Una possibile soluzione, la si può vedere sempre in Figura 2.3(a) in cui si assegnano i due bit più significativi alla prima descrizione e i due meno significativi alla seconda descrizione. In questo modo la prima descrizione è decisamente più significativa della seconda in quanto una sua perdita porterebbe ad una sensibile riduzione della qualità del segnale. Per risolvere questo problema si è quindi deciso di ricorrere ad un secondo modello rappresentato in Figura 2.3(b). In questo caso si utilizzano due quantizzatori uniformi a 3 bit con diverso *offset* per creare le due descrizioni. I due quantizzatori (rosso e blu in figura) presentano due caratteristiche di trasferimento traslate l'una rispetto all'altra e quindi combinando l'informazione di entrambi, in ricezione, si ottiene un bit aggiuntivo di risoluzione. Tuttavia

questo modello presenta un certo livello di inefficienza: il numero totale di bit inoltrati sui due canali è di 6 bit per campione, mentre, in ricostruzione, la qualità che si ottiene combinando entrambi i canali equivale a quella di un quantizzatore a 4 bit. Il primo ad introdurre un modello di channel splitting senza incrementare il numero di bit trasmessi fu Reudink. In questo modello (Figura 2.3(c)), sono ancora utilizzati due diversi quantizzatori ma con intervalli di quantizzazione non connessi. Quando le due caratteristiche sono combinate esse si complementano ma presentano anche piccole intersezioni che fanno aumentare notevolmente il grado di precisione. Inoltre ogni quantizzatore ha solo 6 valori d'uscita che equivalgono a $\log_2 6 = 2.6$ bit/campione.
Il lavoro di Reudink fu archiviato nel 1980 come "report tecnico" [8] e non fu pubblicato. Le sue tecniche per la creazione di due descrizioni per sorgenti senza memoria furono analizzate in dettaglio e pubblicate molti anni dopo da Vaishampayan [9]. Reudink ideò inoltre altri modelli per sorgenti correlate, che saranno poi costruiti ed esaminati nel lavoro di Gersho. Gersho propose l'utilizzo di una codifica "modulo PCM" per channel splitting [10]. Tornando al primo modello di quantizzatore di Figura 2.3(a), si considera il problema di stimare x supponendo di ricevere solo i due bit meno significativi. Essendoci due bit mancanti sappiamo che x starà in uno di quattro intervalli di ampiezza $\frac{1}{8}$. Supponendo però di conoscere il campione precedente ed essendo esso fortemente correlato con il campione attuale, è possibile sfruttare questa correlazione per scegliere quale tra i quattro intervalli possibili scegliere. Un modo per implementare questo modello su channel splitting consiste nell'inoltrare sul primo canale i due bit più significativi di tutti i campioni pari e i due meno significativi dei campioni dispari, e viceversa sul secondo canale. Questa assegnazione dei bit fu proposta da Goodman e successivamente studiata da Quirk. Nella successiva sezione viene esaminato in modo più formale e generale il problema della quantizzazione scalare per modelli a descrizioni multiple.

2.4.1 Quantizzazione scalare MD

Un quantizzatore MD a rate fisso è generalmente composto da un codificatore α_0 e tre decodificatori β_0, β_1 e β_2. Il codificatore α_0 trasforma ogni campione reale x in una coppia di indici di quantizzazione (i_1, i_2) mentre i tre decodificatori producono delle stime di tale campione a partire rispettivamente da (i_1, i_2), i_1 e i_2. In Figura 2.3(b) e (c) per esempio, gli indici i_1 e i_2 sono illustrati attraverso i due colori rosso e blu. La mappatura dell'indice nel valore ricostruito fatta dai due decodificatori β_1 e β_2 corrisponde, invece, al valore corrispondente sull'asse delle ordinate. L'azione di β_0 è invece implicita e si ottiene mettendo assieme il funzionamento dei primi due decodificatori. Vaishampayan [11] introdusse un modo conveniente per visualizzare le operazioni di codifica in

forma tabulare. Prima di tutto il codificatore α_0 è scomposto in due parti: un codificatore α e una funzione f. Un codificatore iniziale α divide l'asse reale in vari intervalli ed assegna ad ognuno di essi un indice che lo rappresenta. La funzione f assegna ad ogni indice trovato da α una coppia di indici (i_1, i_2). Questa assegnazione deve essere invertibile in modo che il decodificatore centrale β_0 possa recuperare il valore in uscita da α. La tecnica di visualizzazione si basa su una matrice di assegnazione in cui sono rappresentati i valori di f^{-1}. Per il quantizzatore di Figura 2.3(b), ad esempio, la matrice di assegnazione è riportata in Tabella 2.1

	000	001	010	011	100	101	110	111
000	0							
001	1	2						
010		3	4					
011			5	6				
100				7	8			
101					9	10		
110						11	12	
111							13	14

Tabella 2.1. Matrice di assegnazione per modello di figura 2.3(b).

I numeri da 0 a 14 denotano gli indici all'uscita del primo quantizzatore α. La matrice mostra la ridondanza di questo modello in quanto sono occupate solo 15 celle su 64 disponibili, ridondanza che si traduce nell'inefficienza in termini di bit/campione utilizzati per codificare le due descrizioni come già descritto nella sezione precedente. Ad ogni modo, una matrice con una percentuale elevata di celle vuote per ogni riga o colonna comporta che la distorsione ottenuta decodificando una sola descrizione è contenuta. Un esempio di matrice con una ridondanza minore rispetto a quella di Tabella 2.1 è quella associata al modello di Figura 2.3(c) che permette di ridurre i bit/campione inoltrati sui due canali a discapito di un aumento della distorsione dei due decodificatori laterali. Tale matrice è rappresentata in Tabella 2.2.

Se la matrice di assegnazione è invece completamente piena come per il modello di Figura 2.3(a) non c'è alcuna ridondanza e quindi non vengono introdotti bit aggiuntivi per la creazione delle due descrizioni ma si ha una forte distorsione laterale nel caso di ricezione di una sola delle due rappresentazioni dello stesso valore (Tabella 2.3).

Per creare un modello di quantizzatore MD l'operazione più difficile è proprio quella di ottimizzare la funzione di assegnazione f in modo da trovare il giusto *trade-off* tra ridondanza e distorsione.

	000	001	010	011	100	101
000	0	1				
001	2	3	5			
010		4	6	7		
011			8	9	11	
100				10	12	13
101					14	15

Tabella 2.2. Matrice di assegnazione per modello di figura 2.3(c).

	00	01	10	11
00	0	1	5	6
01	2	4	7	12
10	3	8	11	13
11	9	10	14	15

Tabella 2.3. Matrice di assegnazione per modello di figura 2.3(a).

2.5 Applicazioni della codifica MD

La codifica a descrizioni multiple, come già accennato, si rivela particolarmente utile nei casi in cui uno o più utenti non ricevano una o più descrizioni a causa delle perdite nel canale di trasmissione e quando sono disponibili diversi livelli di servizio (o distorsione ammissibile) per diverse categorie di utenti. Queste due condizioni determinano rispettivamente il “cosa” (tipologie di sorgenti di informazione) e il “dove” (tipi di mezzi di comunicazione) per l’applicazione delle tecniche di codifica MD. Vediamo quindi quali sono alcuni dei principali scenari in cui la codifica a descrizioni multiple si dimostra una valida soluzione.

2.5.1 Sorgenti d’informazione

Di seguito sono elencate alcune delle principali sorgenti d’informazione che meglio si prestano all’applicazione di tipologie di codifica a descrizioni multiple.

Audio

La codifica a descrizioni multiple è nata proprio per la codifica di un segnale vocale da poter trasmettere attraverso la linea telefonica in quanto era necessario poter trasmettere efficientemente senza perdite totali di informazione e senza ritardi dovuti a ritrasmissioni. Un segnale audio, inoltre, risulta il più semplice

da considerare in quanto unidimensionale e dotato di una forte correlazione tra campioni temporalmente adiacenti. Per questo motivo, una prima soluzione adottata anche nei primi modelli di channel splitting consisteva nel modello riportato nella Sezione 2.2. Configurazioni più recenti utilizzano invece tecniche di predizione più evolute e modelli percettivi in modo da allocare in maniera ottimale i bit nella creazione delle due o più descrizioni.

Immagini

Il modello Channel Splitting discusso per la codifica di segnali audio è stato il primo ad essere utilizzato anche per l'applicazione della tecnica MD ad immagini e a segnali bidimensionali in genere. Anche in questo caso, infatti, i campioni del segnale presentano un'ottima correlazione spaziale che sta alla base per l'applicazione efficace della codifica a descrizioni multiple. Maggiori spiegazioni sull'applicazione di questo tipo di codifica ad immagini saranno definiti in modo dettagliato nel prossimo paragrafo dedicato alle sorgenti video.

Video

Uno tra gli scenari più importanti ed attuali in cui la codifica a descrizioni multiple trova maggiore applicazione è quello della codifica e trasmissione di sequenze video. L'uso della codifica MD porta a notevoli vantaggi nella compressione di un segnale tridimensionale come un video, in quanto gli elevati *bit-rate* necessari per la sua trasmissione su reti, specialmente *peer-to-peer* ed in modalità *streaming*, determinano l'utilizzo di altre tecniche per combattere l'inevitabile perdita di informazione. Le restrizioni, in termini di massimo ritardo tollerabile, impediscono l'utilizzo di tecniche basate su ritrasmissione di pacchetti persi o l'uso di codici di canale su blocchi di lunghezza considerevole che andrebbero a complicare notevolmente le operazioni di codifica e decodifica ed introdurre un ritardo considerevole. Per questo motivo, la codifica video MD ha avuto un enorme sviluppo negli ultimi anni e viene studiata in numerose configurazioni. Sono state infatti proposte molte tecniche di codifica video a descrizioni multiple adottando i codificatori già esistenti (es. MPEG-2, MPEG-4, H.263, H.264/AVC). Nella successiva sezione l'utilizzo della codifica a descrizioni multiple per sequenze video sarà illustrato nel dettaglio indicando più precisamente i vantaggi rispetto ad altre tecniche in uso.

2.5.2 Codifica per sequenze video: il modello Indipendent Flux Polyphase Downsampling Multiple Description (IF-PDMD)

Non è semplice progettare ed implementare uno schema di codifica a descrizioni multiple per una sequenza video. Ci sono molti standard di codifica esistenti e che hanno raggiunto un notevole grado di sviluppo: MPEG-2, MPEG-4, H.263, H.264/AVC. Per questo motivo è difficile imporre lo sviluppo di nuovi standard appositi per codifica MD che siano più complessi di quelli già esistenti. Tuttavia ci sono molte tecniche per creare descrizioni multiple per una sorgente video che fanno uso degli standard di codifica esistenti: quantizzatori MD scalari o vettoriali, trasformate correlanti e sottocampionamento polifase spaziale o temporale. I requisiti da soddisfare per la realizzazione di un modello di codifica video a descrizioni multiple sono quindi i seguenti:

- **compatibilità**: la possibilità di utilizzare codificatori standard per ogni descrizione;
- **semplicità**: minima aggiunta di memoria e potenza computazionale ai vari blocchi di codifica e decodifica standard;
- **efficienza**: Data una certa disponibilità di banda e in assenza di perdite, si deve avere la minima perdita di qualità in decodifica rispetto alla migliore qualità raggiunta oggi dai più evoluti standard di codifica.

Un modello che rispetta tutti questi requisiti è quello che usa la tecnica del sottocampionamento con scomposizione polifase. Esso si dimostra infatti molto semplice e può essere facilmente implementato con i principali codificatori che costituiscono lo stato dell'arte.

Secondo questa tecnica, la sequenza video è dapprima suddivisa in sottosequenze che saranno poi codificate separatamente. Questa suddivisione, nel caso di tecniche MD basate su campionamento spaziale, può essere attuata nei seguenti modi:

- due descrizioni possono essere ottenute separando le righe pari e dispari della sequenza: il video è quindi decomposto in due sottosequenze a risoluzione ridotta, che potranno essere decodificate indipendentemente;
- quattro descrizioni possono invece essere create separando righe pari e dispari e successivamente colonne pari e dispari (Figura 2.4). In questo modo un video ad elevata risoluzione, può essere ridotto a quattro stream video che possono essere codificati usando dei codificatori standard.

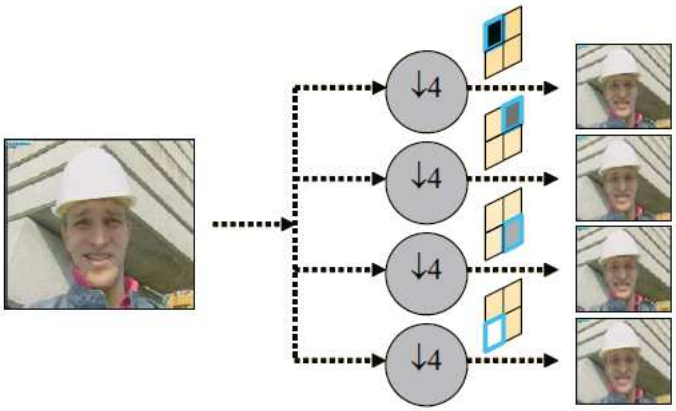


Figura 2.4. Primo stadio del modello di codifica MD a 4 descrizioni: sottocampionamento spaziale. Righe pari e dispari sono separate e lo stesso viene fatto per le colonne in modo da creare le 4 descrizioni.

Al lato decodificatore, le sottosequenze, dopo essere state decodificate vengono ricomposte per ricreare il video originale. Questa operazione è quella che permette di ottenere i migliori risultati in termini di **robustezza**. Persino

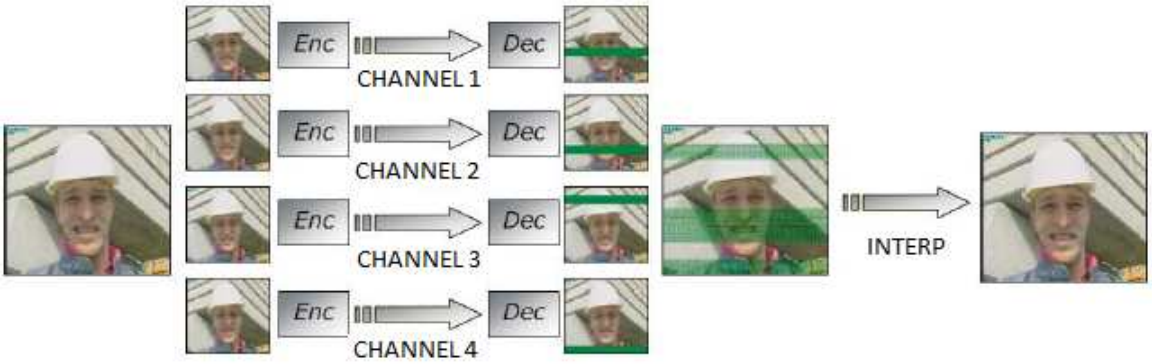


Figura 2.5. Modello IF-PDMD completo per codifica a 4 descrizioni.

per elevate probabilità di perdita, è molto difficile che porzioni corrispondenti risultino corrotte in tutte le descrizioni. Un frame è ricostruito "fondendo" le descrizioni e ricostruendo i pixel mancanti tramite interpolazione bilineare tra pixel vicini. Una porzione di immagine mancante in una descrizione si trasforma in una mancanza distribuita di pixel nell'immagine originale. Pertanto, nel caso in cui almeno un descrizione sia disponibile, è possibile ricostruire le descrizioni mancanti da quelle disponibili. Lo schema completo è rappresentato in Figura 2.5 ed è noto come *Indipendent Flux Polyphase Downsampling Multiple Description* (IF-PDMD). Questo schema è inoltre completamente indipendente dal tipo di codificatore video usato.

2.5.3 Codifica per sequenze video: codifica MD vs codifica scalabile

Un altro vantaggio che la codifica a descrizioni multiple permette di ottenere è legato al concetto di **scalabilità** [12]. Si pensi, ad esempio, alla trasmissione su terminali mobili: utilizzando una codifica standard, l'intero *bitstream* deve essere prima decodificato e poi adattato per essere visualizzato su display diversi. Contrariamente, se si usa una codifica a descrizioni multiple, un terminale potrà decodificare soltanto una descrizione o il numero di descrizioni che sono più adatte alle sue caratteristiche di potenza, memoria, display e banda. Un altro motivo per usare solo alcune delle descrizioni create può essere dato da una temporanea carenza di banda al lato del trasmettitore. Inoltre per bassi *bit rate* la qualità ottenuta applicando la codifica a descrizioni multiple piò risultare più elevata di quella fornita da una codifica standard. Questo accade in quanto, con una codifica MD, i bassi *bit rate* possono essere ottenuti attraverso l'utilizzo di una sola descrizione. Al contrario, con una codifica standard, si dovrebbe ricorrere ad una quantizzazione meno fine. Le corruzioni dovute ad una quantizzazione più grossolana sono infatti maggiormente visibili rispetto a quelle introdotte dal solo utilizzo di una o poche descrizioni. Sebbene la scalabilità sia un indubbio vantaggio esso è da ritenersi come un beneficio "non voluto", in quanto non è questo il principale motivo per cui è stata ideata la codifica MD.

Un meccanismo di codifica ideato per permettere la scalabilità è invece la Codifica Scalabile (Layered Coding, LC). Questo tipo di codifica è simile a alla codifica a descrizioni multiple ma con la differenza che i flussi in cui lo *stream* principale viene diviso non sono tra loro indipendenti. Lo scopo della codifica LC è proprio quello di creare n layer dipendenti tra loro: un layer base (*base layer*) e $n-1$ livelli di raffinamento (*enhancement layer*). In questo modo, a differenza della codifica MD, i layer non hanno tutti la stessa importanza e non possono essere persi indistintamente l'uno dall'altro. Il livello base non

può mai essere perso in quanto costituisce lo stream più importante del video originale e viene solitamente protetto da una codifica di canale. Un'eventuale perdita del livello base porta alla perdita dell'intero flusso video. I livelli di raffinamento servono a migliorare la qualità del livello base ed anch'essi devono essere ricevuti secondo un certo ordine per poter essere utilizzati. Ricevuto il layer base è necessario ricevere il primo enhancement layer per migliorare l'immagine, poi il secondo e così via fino all'ultimo layer. Nella trasmissione video *real-time* o *streaming* questa differenza è rilevante. Supponiamo di voler trasmettere il frame di una sequenza video secondo tre tipologie di codifica:

- codifica standard, non scalabile e a singola descrizione;
- codifica scalabile (LC);
- codifica a descrizioni multiple (MDC).

Supponiamo che in ricezione il frame possa essere costruito in tempo reale man mano che arrivano a destinazione i pacchetti che lo compongono e supponiamo di perdere uno di questi pacchetti.

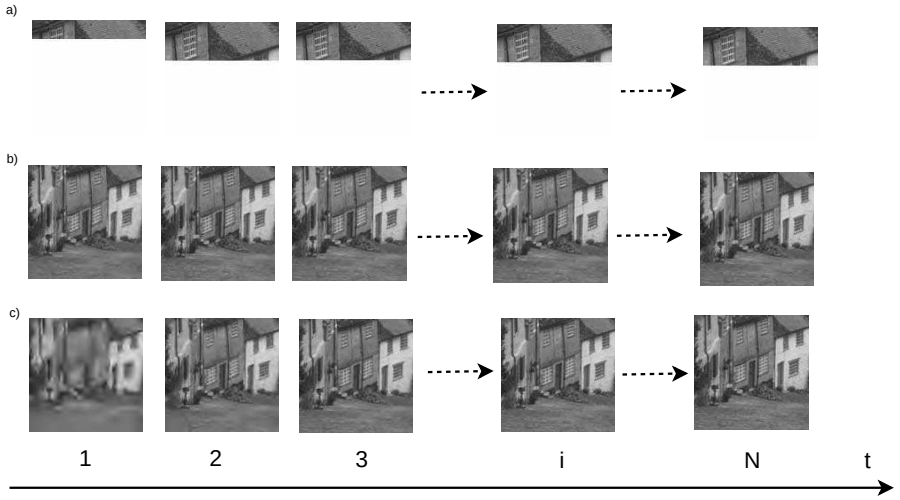


Figura 2.6. (a) modello di codifica standard. (b) modello di codifica LC. (c) modello di codifica MDC.

Come si può vedere in Figura 2.6(a), con il primo modello di codificatore l'immagine non può essere completata in quanto i successivi pacchetti non possono essere utilizzati dovendo rispettare l'ordine di invio per poter essere "assemblati", il flusso video subisce quindi un'interruzione. Il secondo modello di codifica utilizza invece la scomposizione in più livelli (base layer ed enhancement layers). La ricezione dei pacchetti del livello base permette l'immediata costruzione del frame ad una qualità bassa, qualità che è destinata ad

aumentare quando arrivano i successivi pacchetti. L'arrivo deve essere però ordinato e quindi i pacchetti successivi a quello perso non potranno essere utilizzati per aumentare il livello di dettaglio e quindi la qualità dell'immagine (Figura 2.6(b)). Il terzo modello invece è quello che fa uso della codifica a descrizioni multiple. Anche in questo caso l'immagine può essere ricostruita subito dopo l'arrivo dei primi pacchetti come il modello di codifica a livelli, ma il suo grado di qualità è destinato ad aumentare all'arrivo di ogni pacchetto in quanto essi hanno tutti la stessa importanza e l'ordine di arrivo è irrilevante (Figura 2.6(c)). Questo ci fa capire come la perdita di un pacchetto che risulta essere critica per i primi due modelli di codifica, diventa quasi impercettibile in un modello a descrizioni multiple che quindi si dimostra particolarmente adatto a trasmissioni video multimediali, su di un mezzo affetto da perdite e dove i ritardi dovuti alla ritrasmissione di pacchetti non sono permessi.

2.5.4 Codifica a descrizioni multiple per reti *peer-to-peer*

I principali supporti per la trasmissione di contenuti multimediali (audio/video) codificati sono le reti a pacchetto e più in dettaglio le reti peer-to-peer. Negli ultimi anni, le reti peer-to-peer hanno avuto un notevole sviluppo per la condivisione e lo scambio di file di diversa natura. Queste architetture di rete sono presto diventate la base anche per lo sviluppo di applicazioni come *video streaming* e trasmissione video *real time* [1]. Una rete peer-to-peer (P2P) è infatti un tipo di rete che permette a un gruppo di utenti di connettersi fra loro e di condividere le risorse in loro possesso. All'interno di tali reti, ciascun nodo (peer) ha capacità e responsabilità equivalenti. Ciò differisce dalle architetture di tipo *client/server* in cui uno o più computer (server) sono dedicati al servizio di altri computer (client). In reti P2P ogni nodo può recitare il ruolo sia di server che di client potendo quindi sia fornire che richiedere servizi ad altri peer della rete. In questo modo il sistema diventa molto più flessibile e robusto in quanto scompare la figura del server centrale che in caso di guasto porterebbe all'immediata cessazione di alcuni servizi presenti nella rete. Nelle reti peer-to-peer, invece, un qualsiasi nodo potrà liberamente lasciare la rete senza che essa perda alcuna delle sue funzionalità in quanto i file e i servizi in essa presenti sono distribuiti equamente tra i vari nodi che compongono la rete. Un utente che effettua la richiesta di un file otterrà risposta da tutti i nodi contenenti quel file che contribuiranno con le loro risorse a trasmetterlo a destinazione con successo. In questo modo, anche se la potenza computazionale dei singoli nodi di una rete P2P non può competere con quella di un server centralizzato in grado di gestire migliaia di accessi, la loro unione e collaborazione nell'adempire ad un servizio richiesto, permette di ottenere prestazioni elevate. Durante la trasmissione, alcuni pacchetti nei quali viene suddivisa l'informazione da trasmettere possono

essere persi o non arrivare correttamente a destinazione. La natura di queste perdite può essere legata principalmente allo stato del canale, alle congestioni della rete e ai ritardi di trasmissione. I link che compongono una rete possono essere estremamente eterogenei, ognuno con una capacità di trasmissione diversa. Se i dati trasmessi superano la capacità del link più lento del percorso di rete dalla sorgente alla destinazione è inevitabile che alcuni pacchetti vengano scartati. La probabilità di perdita di un pacchetto è inoltre imprevedibile e tempo variante in quanto dipende dal livello effettivo di traffico nella rete e dalla congestione dei link in un preciso momento. Per questo motivo risulta impossibile sapere quali pacchetti andranno persi e quali invece arriveranno correttamente a destinazione: in altre parole i pacchetti per la rete hanno tutti la stessa importanza. Questo concetto sta alla base della tecnica di codifica a descrizioni multiple e pone le basi per un suo possibile utilizzo nel contesto delle reti P2P.
Il modo più comune per gestire la perdita di pacchetti è sicuramente quello di ricorrere all'utilizzo di un protocollo che ne preveda una loro ritrasmissione in caso di perdita (es. TCP). Quando però le perdite di pacchetti sono frequenti, la ritrasmissione può contribuire ad aumentare il livello di congestione della rete creando un circolo vizioso che porta ad un consecutivo aumento dei pacchetti persi e ad eccessivi ritardi dovuti alle continue ritrasmissioni. Il ritardo è un fattore rilevante per molte applicazioni multimediali come la trasmissione video *real-time* e il video *streaming*. Nelle trasmissioni in tempo reale un pacchetto che arriva con un ritardo eccessivo diventa inutilizzabile ed è quindi equivalente ad una perdita. Per le applicazioni interattive come lo *streaming* invece, è possibile accettare un leggero ritardo in ricezione a patto che non vari molto nel tempo e sia conosciuto, in modo da dimensionare il buffer di ricezione e permettere una visione fluida del video. Ritardi eccessivi dovuti alle continue ritrasmissioni non sono quindi permessi nemmeno in quest'ultimo caso. Un modo per evitare di ricorrere alle ritrasmissioni di pacchetti è quello di utilizzare protocolli che cercano di recuperare l'informazione attraverso la trasmissione di pacchetti ridondanti che saranno poi utilizzati in ricezione per ricostruire l'informazione mancante. Questa aggiunta di informazione è ottenuta attraverso metodi di codifica FEC (Forward Error Correction) che causano un necessario aumento di complessità e potenza necessaria sia al lato trasmettitore che ricevitore. L'informazione ridondante introdotta nello stream spesso richiede l'adozione di blocchi di informazione molto lunghi introducendo un'ulteriore ritardo (ritardo di decodifica di canale).
La tecnica di codifica a descrizioni multiple descritta nelle precedenti sezioni è per questo molto usata nel contesto delle reti peer-to-peer nelle quali sono presenti tutti i suddetti problemi.

In particolare la codifica a descrizioni multiple si presenta come una tecnica

adeguata in tutte quelle situazioni in cui:

- la ritrasmissione non è possibile;
- il ritardo è un fattore rilevante;
- esiste la possibilità di instradare i pacchetti contenenti le diverse descrizioni su percorsi diversi.

Proprio la possibilità di instradare pacchetti su percorsi diversi rende le reti P2P l'ambiente dove meglio si presta la codifica a descrizioni multiple. Se i pacchetti contenenti le varie descrizioni seguono percorsi diversi c'è un'alta probabilità che almeno una descrizione arrivi a destinazione correttamente permettendo una buona ricostruzione dell'informazione trasmessa anche se a qualità inferiore. Questo concetto è fondamentale nelle applicazioni multimediali come videoconferenza o *live streaming* in cui una perdita temporanea di qualità è assolutamente preferibile ad un'interruzione del flusso video.

2.5.5 Memorizzazione distribuita su reti peer-to-peer

L'ambiente in cui operano le applicazioni P2P è caratterizzato da un'importante asimmetria dovuta al meccanismo di accesso dei terminali alla rete. La banda disponibile in *downlink* è infatti decisamente superiore a quella riservata per le operazioni di *uplink*. Ad esempio, un tipico servizio ADSL offre una capacità di downlink di $7\ Mbit/s$ e $356\ kbit/s$ per l'uplink. Per questo motivo i peer costituenti la rete possono ricevere informazioni da altri nodi ad un rate decisamente superiore a quello che ognuno di essi utilizza per spedirle. Tramite codifica MD è quindi possibile memorizzare in sistemi diversi descrizioni differenti. Al momento della richiesta del contenuto, i peer che contengono le n descrizioni iniziano a trasmetterle ad un rate n volte inferiore a quello con il quale avrebbero dovuto trasmettere l'intero flusso. Il destinatario potrà quindi utilizzare la sua capacità superiore in downlink per ricevere contemporaneamente tutte le descrizioni e ricostruire lo *stream*. Inoltre l'archiviazione distribuita delle diverse descrizioni permette di ottenere un tipo di codifica scalabile, in cui la qualità del flusso ricevuto varia a seconda del numero di descrizioni ricevute. Se un utente ha bisogno di effettuare un accesso veloce ad una informazione senza privilegiare la qualità potrebbe essere interessato a ricevere solo una descrizione dal nodo più vicino che la contiene. Se invece è richiesta una qualità superiore la richiesta si propaga anche ai nodi contenenti le altre descrizioni. In una rete peer-to-peer questa situazione si può ottenere ricorrendo ad una topologia ad albero, in cui le descrizioni sono memorizzate nei diversi nodi della rete che stanno alla radice di diversi alberi.

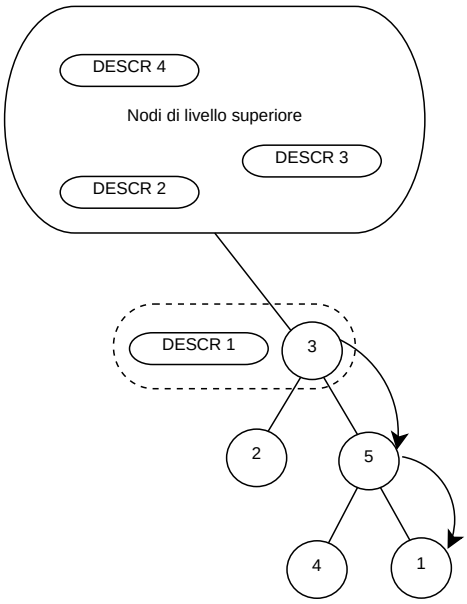


Figura 2.7. Memorizzazione distribuita di 4 descrizioni in una rete peer-to-peer con topologia ad albero.

In Figura 2.7 è illustrato un modello di questo tipo in cui il peer (1) prima accede alla prima descrizione contenuta nel nodo (3) e se la qualità non è sufficiente farà richiesta per ulteriori descrizioni contenute in nodi che stanno a livelli superiori della struttura ad albero.

2.5.6 Codifica a descrizioni multiple con diverse classi di QoS

La casualità dell'eliminazione di pacchetti in reti peer-to-peer porta la tecnica di codifica a descrizioni multiple ad essere una tecnica adeguata in questo contesto. Le prestazioni però potrebbero subire un netto miglioramento se la rete fosse in grado di riconoscere l'importanza dell'informazione portata da ogni pacchetto ed in caso di congestioni, scartare solamente i pacchetti che possono essere stimati più facilmente. A tale scopo, una classificazione dei pacchetti al momento del loro ingresso nella rete, potrebbe definire il livello di importanza di ciascuno di essi ed essere utilizzata dai nodi interni della rete in caso di congestioni o restrizioni di banda nei link. Un modello che permette questa classificazione è noto come *Differentiated Services (DiffServ)*: esso è un meccanismo di accesso alla rete che offre differenti livelli di qualità di servizio (QoS) a diversi *stream* di traffico. Una rete *DiffServ* si basa sul comportamento dei nodi che la compongono quando ricevono traffico da diversi flussi [13]. Essi applicano

per ogni pacchetto quello che viene definito come *Per-Hop Behaviours* (PBH), in base al quale:

- per ogni pacchetto viene fatta una classificazione solo all'ingresso di tale pacchetto nella rete e tale classificazione non può più essere modificata;
- il processo di instradamento dei pacchetti avviene tenendo conto della loro classe di appartenenza che equivale al loro livello di priorità rispetto agli altri pacchetti che viaggiano nella rete;
- il numero di classi è generalmente limitato in quanto devono essere limitati i comportamenti dei nodi a seconda della classe di pacchetto da instradare;
- *DiffServ* non ha bisogno di comunicare alla rete nessuna allocazione di risorse per i pacchetti trasmessi in quanto la classificazione è definita nell'*header* di ogni pacchetto e valutata al momento del suo arrivo in un nodo della rete;

Il lavoro di tesi si propone di unire i vantaggi della tecnica MD per la codifica e trasmissione di sequenze video in reti P2P , con una classificazione efficiente dei pacchetti appartenenti alle varie descrizioni secondo le classi imposte dal modello *DiffServ*. Il meccanismo *DiffServ* sarà meglio definito nel capitolo 4, mentre nel prossimo capitolo saranno date le basi teoriche per creare una classificazione efficiente dei pacchetti delle varie descrizioni attraverso la "Teoria dei Giochi".

2.5.7 Altri scenari applicativi

Vediamo ora quali sono altri scenari applicativi, oltre a quello delle reti peer-to-peer in cui la codifica a descrizioni multiple trova efficientemente utilizzo.

Sistemi wireless e lo standard IEEE 802.11e

Nei sistemi wireless le perdite di pacchetto sono frequenti rendendo anche in questo caso la tecnica di codifica MD estremamente vantaggiosa. Lo standard 802.11e supporta inoltre diversi livelli di *Quality of Service* ed è particolarmente utilizzato per trasmettere contenuti multimediali (audio e video) su rete *wireless*. Lo standard prevede 4 classi di servizio, ciascuna con la propria coda (Figura 2.8). Ogni coda possiede una diversa probabilità di perdita dei pacchetti contenuti al suo interno ed anche in questo caso, classificando opportunamente i pacchetti delle varie descrizioni, possiamo definire il livello di priorità di ciascuno di essi. L'articolo [14] mostra che assegnando i pacchetti RTP video di

due descrizioni a canali diversi si ottengono dei risultati migliori rispetto alla tecnica che non prevede l'utilizzo di diverse classi di QoS.

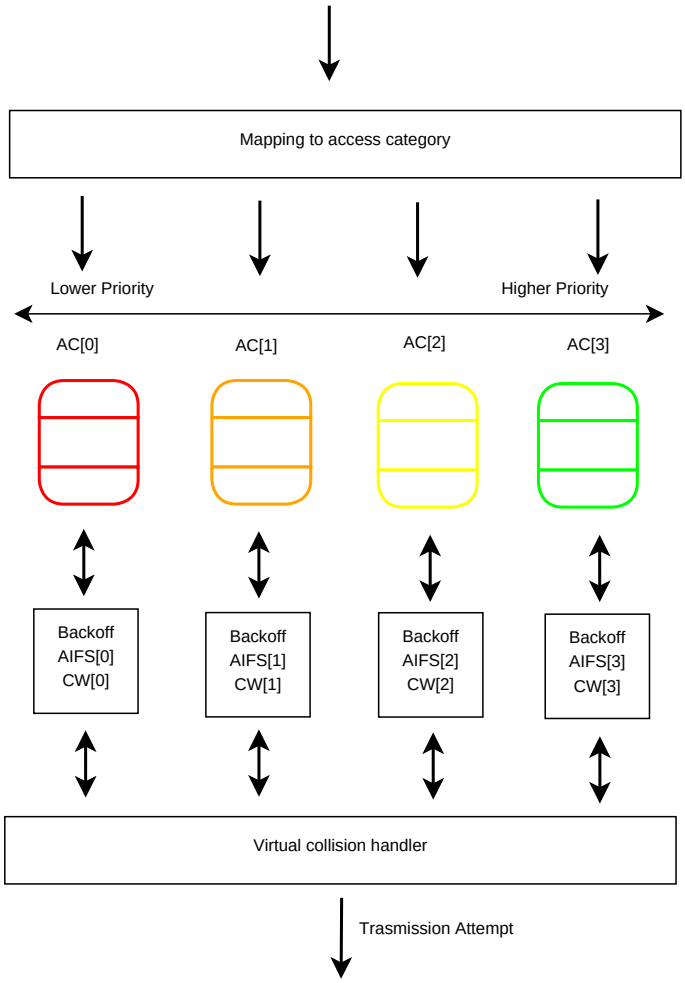


Figura 2.8. funzionamento standard IEEE 802.11e.

Sistemi di trasmissione digitale ibridi

Nelle trasmissioni radio, le stazioni sono tipicamente separate in frequenza e le frequenze centrali di trasmissione sono assegnate in modo da evitare che altre stazioni adiacenti usino la stessa frequenza. Nonostante questo, al confine tra due stazioni che usano la stessa frequenza, si generano delle interferenze. Per questo motivo, sono state introdotte delle tecniche per la trasmissione di dati digitali all'interno delle comuni trasmissioni analogiche, ponendole alle due estremità della banda stessa per non degradare il segnale. Alcuni utenti quindi

potrebbero ricevere, a causa delle interferenze, solo uno o l'altro dei due canali posti all'estremità della banda, oppure entrambi, favorendo le condizioni ideali per l'utilizzo della codifica a descrizioni multiple.

Capitolo 3

Teoria dei Giochi

Questo capitolo si propone di analizzare, dal punto di vista teorico, gli strumenti matematici che saranno utilizzati per effettuare una classificazione ottimale dei pacchetti video, ottenuti tramite codifica a descrizioni multiple. Come già accennato nel precedente capitolo, una classificazione dei pacchetti in base al loro livello di importanza porta ad un incremento delle prestazioni in reti peer-to-peer. La Teoria dei Giochi mira ad utilizzare le differenze di importanza tra i vari pacchetti per assegnare ad ognuno di essi un diverso livello di priorità.

3.1 Introduzione

La Teoria dei Giochi fornisce le basi matematiche per analizzare situazioni in cui sono coinvolte più identità interagenti tra loro ed in grado di prendere delle decisioni in modo autonomo. In particolare, questa teoria fornisce gli strumenti per analizzare il problema e proporre una soluzione. La ricerca operativa classica suppone che le decisioni relative a un problema siano prese da un unico decisore che può operare in completa autonomia e libertà. La teoria dei giochi tratta situazioni in cui il risultato finale dipende dalle scelte fatte da più persone, dette giocatori, che operano perseguendo obiettivi che possono risultare comuni, differenti ed eventualmente contrastanti. In queste situazioni, la programmazione matematica risulta insufficiente a descrivere, e quindi risolvere, un dato problema.

Alcuni concetti teorici che stanno alla base della teoria dei giochi furono studiati già nel diciottesimo secolo. Lo sviluppo maggiore della teoria ebbe però inizio nel 1920 con il lavoro dei matematici Emile Borel e John Von Neumann e continuò con la pubblicazione, nel 1944, del libro *Theory of games and economic behavior*, di Von Neumann e Oskar Morgenstern. Da quel momento i primi modelli di teoria dei giochi furono applicati in ambiti politici ed eco-

nomici. Il più famoso studioso ad essersi occupato, successivamente, di teoria dei giochi, fu il matematico John Forbes Nash jr. Egli, studiò nell'università di Princeton e nel suo lavoro di tesi, nel 1950, introdusse la nozione di "Equilibrio di Nash", delineando le classi di giochi principali in cui tale equilibrio era presente. Questo concetto era destinato a cambiare l'evoluzione della teoria, tanto che nel 1994 John Nash vinse il premio nobel per l'economia, condividendolo con John C. Harsanyi e Reinhard Selten, altri due studiosi di teoria dei giochi [15].

Esempio 3.1.1 ***(Dilemma del prigioniero)*** *Questo è uno dei problemi più noti in letteratura e sarà utilizzato come esempio preliminare per illustrare una tipica situazione in cui la Teoria dei Giochi trova utilizzo. Introdotto nel 1950 da Dresher e Flood, questo problema può essere definito nel seguente modo. Due individui I e II sono stati arrestati per lo stesso reato e vengono interrogati separatamente. Ognuno di essi può scegliere, indipendentemente dall'altro, di confessare (C) o non confessare (NC). Se entrambi gli individui non confessano, vengono condannati a due anni ciascuno; se entrambi confessano, vengono condannati a cinque anni ciascuno; se uno confessa e l'altro no, quello che confessa viene condannato ad un anno, mentre l'altro a sei anni. Le pene sono riportate nella Tabella 3.1, in quella che in seguito chiameremo forma strategica.*

I/II	*C*	*NC*
C	$-5, -5$	$-1, -6$
NC	$-6, -1$	$-2, -2$

Tabella 3.1. Dilemma del Prigioniero.

Lo scenario ora è il seguente: ciascun prigioniero non è a conoscenza della scelta dell'altro e ognuno di essi sarà tentato a confessare, in modo da conseguire una condanna minore, qualunque sia la scelta dell'avversario. Questo si può facilmente vedere esaminando la Tabella 3.1. Il primo valore di tutti gli elementi delle due righe rappresenta la condanna del giocatore I nel caso in cui egli confessi (prima riga) o non confessi (seconda riga). Ora, se il giocatore II confessa si può notare come $-5 > -6$*; se invece non confessa* $-1 > -2$*. Indipendentemente dalla scelta del secondo giocatore, il primo giocatore sarà quindi portato a confessare. Un ragionamento analogo lo si può fare anche per il secondo giocatore che quindi sceglierà l'opzione C. La decisione attesa è quindi (C, C), con una condanna di* 5 *anni ciascuno, mentre per entrambi sarebbe stata più vantaggiosa la scelta (NC, NC) che avrebbe portato ad una pena di* 2 *anni.*

3.2 Rappresentazione di un gioco

Un gioco può essere rappresentato principalmente in tre forme:

- *forma estesa*, introdotta da von Neumann (1928) e formalizzata da Kuhn (1953);
- *forma strategica*, definita da Shubik (1982) ma già nota come forma normale nei lavori di von Neumann e Morgestern;
- *forma caratteristica*, dovuta a von Neumann e Morgenster ed utilizzata per la rappresentazione di giochi cooperativi.

Prima di passare all'analisi delle tre diverse forme di rappresentazione definiamo alcuni concetti fondamentali e che sono indipendenti dalla rappresentazione adottata.

Definizione 3.2.1 *Si chiama funzione dei pagamenti (payoff) una funzione f che assegna ad ogni giocatore la sua vincita per ogni possibile terminazione del gioco.*

Definizione 3.2.2 *Si definisce strategia del giocatore i una funzione σ_i che assegna al giocatore i una mossa per ogni possibile situazione del gioco.*

3.2.1 Forma estesa

Questa rappresentazione definisce in modo puntuale tutte le mosse e identifica la situazione del gioco dopo ognuna di esse. In generale si utilizza una rappresentazione ad albero, in cui ad ogni nodo si associa una possibile situazione di gioco. Gli archi uscenti da un nodo rappresentano le possibili mosse del giocatore che è chiamato a decidere, mentre ai nodi terminali si associano i valori delle vincite (*payoff*) di ciascun giocatore.

In Figura 3.1 è illustrata la rappresentazione in forma estesa del gioco definito nell'Esempio 3.1.1. Come si può notare, questa rappresentazione è di solito molto ricca e dettagliata ma poco maneggevole.

3.2.2 Forma strategica

Questa rappresentazione è una descrizione matematica del gioco. Per un gioco ad n giocatori, si utilizza una $2n$-upla $(\Sigma_1, \Sigma_2, \ldots, \Sigma_n, f_1, f_2, \ldots, f_n)$ dove:

- $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$ sono insiemi non vuoti contenenti tutte le possibili strategie per ogni giocatore;

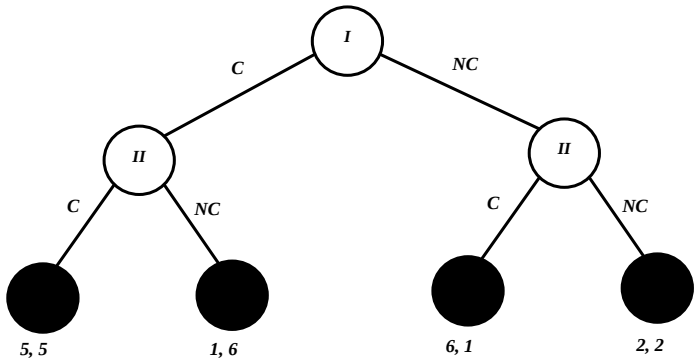


Figura 3.1. Rappresentazione in forma estesa dell'Esempio 3.1.1: *dilemma del prigioniero*

- $f_1, f_2, \ldots, f_n$ sono le funzioni di *payoff*, esse sono funzioni reali, definite sul prodotto cartesiano degli insiemi Σ_i:

$$f_i : \prod_{k=1}^{n} \Sigma_k \to \mathbb{R} \quad i = 1, \ldots, n \tag{3.1}$$

In questo caso, tutti i giocatori scelgono contemporaneamente la strategia da adottare e f_i indica il guadagno del giocatore i-esimo per quella data configurazione di strategie chiamata anche *profilo di strategie*.

Definizione 3.2.3 *Una n-upla $(\sigma_1, \sigma_2, \ldots, \sigma_n) \in \prod_{k=1}^{n} \Sigma_k$ è detta profilo di strategie.*

Nel caso di gioco a due giocatori, gli elementi della forma strategica possono essere rappresentati in forma tabulare. In questo modo, sulle righe saranno rappresentate le possibili strategie per il primo giocatore e sulle colonne le strategie del secondo giocatore. all'interno di ogni cella, in corrispondenza di una coppia di strategie saranno indicati i rispettivi payoff f_1 ed f_2. Un esempio di rappresentazione in forma strategica è illustrato in Tabella 3.1 ed è relativo al gioco "il dilemma del prigioniero". Come si può facilmente intuire, mentre risulta generalmente agevole passare da una rappresentazione in forma estesa ad una in forma strategica, il passaggio inverso risulta più complesso in quanto quest'ultima è una rappresentazione puramente matematica.

3.2.3 Forma caratteristica

Questa forma può essere utilizzata solo per giochi cooperativi in quanto fa riferimento alla nozione di coalizione.

Definizione 3.2.4 *Detto N l'insieme dei giocatori, ogni sottoinsieme $S \subseteq N$ è detto coalizione. Se $S = N$ si ha la grande coalizione, in cui tutti i giocatori possono comunicare e stipulare accordi tra loro.*

Definizione 3.2.5 *Si dice funzione caratteristica di un gioco ad n giocatori, una funzione indicata con v tale che:*

$$v : \phi(N) \to \mathbb{R} \ \ con \ \ v(\emptyset) = 0 \tag{3.2}$$

Definizione 3.2.6 *Una funzione caratteristica v è detta additiva se, per ogni coppia di coalizioni disgiunte S e T, $S \cap T = \emptyset$ si ha $v(S \cup T) = v(S) + v(T)$. Se $v(S \cup T) \geq v(S) + v(T)$ la funzione è detta superadditiva, mentre se $v(S \cup T) \leq v(S) + v(T)$ la funzione v è detta subadditiva.*

In questo modo , la funzione caratteristica v assegna ad ogni coalizione S, la massima vincita possibile, indipendentemente dal comportamento dei giocatori non appartenenti ad S. Un gioco descritto tramite funzione caratteristica è detto in *forma caratteristica o coalizionale*. Se la funzione caratteristica è additiva, superadditiva o subadditiva, anche il gioco è detto additivo, superadditivo e subadditivo rispettivamente.

Definizione 3.2.7 *Se per ogni coalizione $S \subseteq N$, si ha $v(S) + v(N \setminus S) = v(N)$ il gioco è detto a somma costante.*

Tornando all'Esempio 3.1.1, la sua rappresentazione in forma caratteristica è la seguente:

$$N = \{I, II\}$$
$$v(\emptyset) = 0; \ v(I) = v(II) = 5; \ v(I, II) = 4$$

La forma caratteristica costituisce una descrizione limitata del gioco, in quanto non permette di definire la vincita di ogni singolo giocatore della coalizione, ma solo la vincita complessiva.

3.3 Teoria dell'utilità

Nelle precedenti sezioni, si fa riferimento al concetto di guadagno, che ogni giocatore cerca di massimizzare. In questo paragrafo si definiscono i concetti di *preferenza* e di *utilità* di *von Neumann-Morgenstern* che permettono di dare un'interpretazione ai valori numerici definiti dalle funzioni di *payoff* [16]. Tutti i giocatori cercano di adottare una strategia che permetta loro di massimizzare la loro utilità. Il concetto di utilità è però relativo in quanto può dipendere da diversi fattori (ad es. economico, sentimentale, sociale). per questo motivo si introduce un ulteriore operatore, detto di *preferibilità*.

Definizione 3.3.1 *Dati due esiti A e B, si dice che A è preferibile a B, per un giocatore, se egli cerca di conseguire A piuttosto che B e si indica con $A \succ B$.*

Definizione 3.3.2 *Dati due esiti A e B si dice che A è indifferente a B, per un giocatore, se nessuno è preferibile all'altro e si indica con $A \equiv B$.*

Gli esiti possono essere certi oppure incerti secondo una probabilità nota. Tale situazione viene rappresentata tramite il concetto di lotteria.

Definizione 3.3.3 *Dati due esiti A e B, si definisce lotteria l'esito $rA + (1 - r)B,\ 0 \leq r \leq 1$, in cui A si verifica con probabilità r e B con probabilità $1-r$.*

Si noti che la lotteria non è una combinazione lineare di esiti, in quanto il risultato può essere solo A o B.

Dato un insieme di esiti E, una relazione di preferenza su E può essere rappresentata come una funzione di utilità $u : E \to \mathbb{R}$ tale che per ogni coppia di esiti $(E_1,\ E_2) \in E$ si ha:

$$E_1 \succ E_2 \Leftrightarrow u(E_1) > u(E_2) \tag{3.3}$$

$$u(rE_1 + (1 - r)E_2) = ru(E_1) + (1 - r)u(E_2) \tag{3.4}$$

La funzione di utilità permette quindi di quantificare le preferenze. Per capire come le preferenze sono legate alle strategie e ai payoff finali di ogni giocatore è necessario introdurre il concetto di *Game Form*.

Definizione 3.3.4 *Si consideri un gioco ad n giocatori in forma strategica $(\Sigma_1, \Sigma_2, \ldots, \Sigma_n, f_1, f_2, \ldots, f_n)$. La Game Form è data da $(\Sigma_1, \Sigma_2, \ldots, \Sigma_n, E, h)$, dove E è l'insieme degli esiti finali e $h : \prod_{i=1}^{n} \Sigma_i \to E$ è una funzione che permette di individuare a quale esito si perviene per ogni profilo di strategie.*

Per studiare il comportamento dei giocatori e risolvere il gioco è necessario conoscere le preferenze dei giocatori sui possibili esiti, rappresentate da una funzione di utilità

$$u_i : E \to \mathbb{R}, \quad i = 1, \ldots, n \tag{3.5}$$

da cui si ottengono i payoff:

$$f_i = u_i \circ h, \ \ f_i : \prod_{k=1}^{n} \Sigma_k \to \mathbb{R}, \ \ i = 1, \ldots, n \tag{3.6}$$

con i quali costruire la rappresentazione in forma strategica.

3.4 Soluzione di un gioco

Dopo aver definito gli elementi di cui è costituito un gioco ed aver illustrato le varie forme di rappresentazione, è necessario descrivere meglio il concetto di soluzione. Risolvere un gioco consiste nel fornire delle indicazioni a tutti i giocatori, sulla strategia da adottare. Tali indicazioni non possono però essere assolute, in quanto bisogna tenere conto di altri fattori, aleatori o legati alle preferenze del singolo giocatore. Con il termine "soluzione" si indica quindi una scelta che può risultare accettabile a tutti i giocatori secondo i loro criteri soggettivi.

Esistono inoltre due diverse tipologie di gioco:

- **giochi non cooperativi**;
- **giochi cooperativi**.

Nel primo caso i giocatori non possono comunicare tra loro o stipulare accordi vincolanti. Nel secondo caso, invece, sono possibili accordi tra i giocatori che potranno formare una o più coalizioni. Nella sezione successiva sarà trattata nel dettaglio la tipologia di gioco *non cooperativo*, definendo il concetto di *equilibrio di Nash* come possibile soluzione.

3.5 Giochi non cooperativi

Questa classe di giochi non permette ai giocatori, di comunicare tra loro e stipulare accordi vincolanti, indipendentemente dal fatto che i loro obiettivi siano comuni o contrastanti. In questo caso, ogni giocatore sceglie la strategia migliore da adottare per massimizzare il proprio guadagno, senza sapere quali saranno le scelte degli altri giocatori.

3.5.1 Equilibrio di Nash

Il concetto più importante di soluzione per un gioco non cooperativo è quello di *Equilibrio di Nash*.

Definizione 3.5.1 *Dato un gioco G, si dice che la n-upla di strategie $(\sigma_1^*, \sigma_2^*, \ldots, \sigma_n^*)$, con $\sigma_i^* \in \Sigma_i$, è in equilibrio, se nessun giocatore ha interesse ad essere l'unico a cambiare strategia, ovvero:*

$$f_i(\sigma_1^*, \ldots, \sigma_i^*, \ldots, \sigma_n^*) \geq f_i(\sigma_1^*, \ldots, \sigma_i, \ldots, \sigma_n^*), \qquad \forall \sigma_i \in \Sigma_i, \ \ \forall i \in \mathbb{N} \quad (3.7)$$

In questa situazione ogni giocatore non ha interesse a cambiare la propria strategia, in quanto, se lo facesse andrebbe incontro ad un guadagno inferiore a quello di equilibrio. Se ne deduce, quindi, che se i giocatori raggiungono un equilibrio di Nash, nessuno può migliorare il proprio risultato modificando solo le proprie scelte, ed è quindi vincolato dalle strategie degli altri. Poiché questo vale per tutti i giocatori, è evidente che se esiste un equilibrio ed è unico, esso rappresenta la soluzione del gioco, in quanto nessuno dei giocatori ha interesse a cambiare strategia. Un esempio di equilibrio lo si può vedere esaminando l'Esempio 3.1.1. La coppia di strategie (C, C) rappresenta un equilibrio che porta alla coppia di guadagni $(-5, -5)$ per i due giocatori. In questo caso, se uno solo dei due giocatori decidesse di cambiare la sua strategia, otterrebbe un guadagno inferiore, passando da -5 a -6.

Si può notare, però, come la strategia (NC, NC) sarebbe stata più vantaggiosa per i due giocatori ma non rappresentava un equilibrio. L'equilibrio è quindi inefficiente dal punto di vista di un *supervisore* esterno, ma rappresenta le scelte più "razionali", che i due giocatori avrebbero fatto in quella situazione di gioco. Inoltre, se i due giocatori si fossero coalizzati, potendo così comunicare tra loro, si sarebbero potuti accordare per la coppia di strategie (NC, NC). L'equilibrio inoltre potrebbe non essere unico, in questi casi, generalmente, si sceglie quello che massimizza il guadagno complessivo per tutti i giocatori.

3.5.2 Giochi a somma zero

Definizione 3.5.2 *Un gioco G si dice a somma zero se, per ogni terminazione del gioco la somma dei payoff è nulla*

In questo modo, tutto quello che viene guadagnato da un giocatore viene perso dagli altri giocatori. Nel caso di gioco a due giocatori, la rappresentazione più utilizzata è la forma strategica che in questo caso si trasforma in una rappresentazione matriciale. In questa matrice A, la riga i-esima è associata alla strategia σ_i del primo giocatore, mentre la colonna j-esima corrisponde alla strategia σ_j del secondo giocatore. Ogni elemento $a_{i,j}$ della matrice rappresenta il payoff del primo giocatore, ovvero quanto il primo giocatore riceve dal secondo se essi giocano la coppia di strategie (σ_i, σ_j). Il payoff del secondo giocatore non ha bisogno di essere indicato in quanto è l'opposto di quello del primo, dovendo la loro somma essere nulla. Questa rappresentazione è detta anche *forma normale*.

Equilibrio di Nash per giochi a due giocatori e a somma zero

Per giochi a due giocatori, a somma zero ed espressi in forma normale, l'equilibrio di Nash è composto dalla coppia di strategie (σ_i, σ_j) tali che l'elemento

$a_{i,j}$ risulta essere il più grande della colonna j-esima ed il più piccolo della riga i-esima.

Esempio 3.5.3 *Sia dato il gioco rappresentato in forma normale dalla seguente matrice:*

$$A = \begin{pmatrix} 6 & 3 & 4 \\ 5 & 2 & -2 \\ 7 & -1 & 3 \end{pmatrix}$$

Applicando la definizione di equilibrio di Nash si può osservare come la coppia di strategie (σ_1, σ_2)*, corrispondente alla coppia di indici* $(1, 2)$ *della matrice e a cui è associata la coppia di payoff* $(3, -3)$*, sia di equilibrio. Se infatti il primo giocatore cambiasse strategia guadagnerebbe* 2 *o* -1*, entrambi minori di* 3*, e analogamente, se il secondo giocatore cambiasse strategia avrebbe come payoff* -6 *e* -4*, entrambi minori di* -3*.*

Anche per i giochi a somma zero continua ad esistere il problema della possibile presenza di punti di equilibrio multipli che risulta ulteriormente aggravato dal seguente teorema.

Teorema 3.5.4 *In un gioco a due giocatori e a somma zero, se le coppie di strategie* (σ_i, σ_j) *e* (σ_h, σ_k) *sono di equilibrio, allora lo sono anche* (σ_i, σ_k) *e* (σ_h, σ_j)

Ricapitolando, anche per i giochi a somma zero, l'obiettivo dei giocatori è quello di massimizzare la propria vincita o minimizzare la perdita. In particolare, se i giocatori scelgono la coppia di strategie (σ_i, σ_j), con payoff $a_{i,j}$, il giocatore I cerca di massimizzarlo, intervenendo sulla scelta di σ_i, mentre il giocatore II cerca di minimizzarlo, intervenendo sulla scelta di σ_j. L'assenza di comunicazione tra due giocatori fa in modo che nessuno dei due abbia informazioni sulla scelta dell'altro, ma l'esistenza di un punto di equilibrio fa si che, ragionevolmente, entrambi scelgano la corrispondente coppia di strategie. Il problema nasce quando un gioco risulta privo di equilibri di Nash. In questo caso, il concetto di soluzione è più complesso e sarà definito nel prossimo paragrafo.

Gioco a due giocatori e a somma zero senza equilibri di Nash

Alcuni giochi non presentano punti di equilibrio. In questo caso è necessario introdurre i concetti di *vincita minima* e *perdita massima*. Dato un gioco espresso in forma normale attraverso la matrice A:

- si definisce vincita minima per il giocatore I e si indica con v'_I

$$v'_I = \max_i \min_j \{a_{i,j}\};$$

- si definisce perdita massima per il giocatore II e si indica con v'_{II}

$$v'_{II} = \min_j \max_i \{a_{i,j}\} .$$

Con il termine vincita minima si intende la massima vincita che il primo giocatore può ottenere agendo sulla propria strategia σ_i, se contemporaneamente il secondo giocatore fa di tutto per minimizzarla, agendo su σ_j. Data la matrice A questo corrisponde nel trovare il più piccolo valore per ogni riga della matrice e tra gli elementi trovati scegliere il maggiore.

Con il termine perdita massima per il giocatore II, invece, si intende la minima perdita che il secondo giocatore può ottenere agendo sulla propria strategia σ_j, se, allo stesso tempo, il primo giocatore interviene sulla propria σ_i, per cercare di massimizzare tale perdita. Data la matrice A, questo corrisponde nel trovare il più grande valore per ogni colonna della matrice e tra gli elementi trovati scegliere il più piccolo.

Si può verificare che se $v'_I = v'_{II}$ allora esiste un punto di equilibrio.

Esempio 3.5.5 *Dato il seguente gioco a due giocatori e a somma zero:*

$$A = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$$

si può notare come esso non presenti punti di equilibrio. Si possono però trovare i punti di vincita minima e perdita massima rispettivamente per il primo e secondo giocatore. Applicando la definizione si ottiene:

$$v'_I = \max\{2, 1\} = 2 \quad e \quad v'_{II} = \min\{4, 3\} = 3$$

in questo caso, $v'_I \leq v'_{II}$.

In generale, un comportamento "razionale" dei due giocatori fa in modo che, il primo giocatore vinca almeno v'_I, mentre il secondo giocatore perda al più v'_{II}. Se però un giocatore sfrutta la razionalità dell'altro può prevederne la mossa e cambiare la sua per trarne vantaggio. Pertanto, in questa situazione, per migliorare il risultato è necessario non giocare razionalmente, ovvero non giocare sempre le strategie di vincita minima o perdita massima.

Riassumendo, l'utilizzo dell'equilibrio di Nash come soluzione del gioco ha cambiato la storia della teoria dei giochi, in quanto ci permette di raggiungere una situazione di gioco "favorevole" a tutti i giocatori. Ad ogni modo, è possibile evidenziare alcuni limiti dell'equilibrio di Nash:

- inefficienza (Esempio 3.1.1);

- non unicità;
- non esistenza (Esempio 3.5.5).

Il modello di soluzione che abbiamo esaminato in questa sezione è privo di un'analisi sull'aleatorietà delle possibili scelte dei giocatori ed è anche noto come equilibrio di Nash per *strategie pure*. Un modello più complesso che sarà definito di seguito è invece chiamato a *strategie miste.*

3.5.3 Strategie Miste

Definizione 3.5.6 *Si chiama strategia mista, per un giocatore, una distribuzione di probabilità sull'insieme delle sue strategie pure.*

Se l'insieme delle strategie pure è composto da n elementi, una strategia mista si può indicare con un vettore $x = (x_1, x_2, \ldots, x_n)$ con $x_i \geq 0$ e $\sum_{i=1}^{n} x_i = 1$. Per un gioco a due giocatori, chiameremo l'insieme delle strategie miste del primo giocatore con X e quello del secondo giocatore con Y.

Definizione 3.5.7 *Dato un gioco G a due giocatori e a somma zero, in forma normale, con matrice A, è detta vincita attesa se il giocatore I gioca la strategia mista $x \in X$ e il giocatore II gioca la strategia $y \in Y$, la quantità:*

$$A(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} x_i a_{ij} y_j = x^T A y. \tag{3.8}$$

Anche in questo caso è possibile definire la vincita minima per il giocatore I, se sceglie la strategia mista $x \in X$ come:

$$v(x) = \min_{x \in X} \left\{ x^T A y \right\} = \min_{j} \left\{ x^T A_j \right\}; \tag{3.9}$$

e la perdita massima per il giocatore II se sceglie la strategia mista $y \in Y$ come:

$$v(y) = \max_{y \in Y} \left\{ x^T A y \right\} = \max_{i} \left\{ A_i y \right\}; \tag{3.10}$$

dove A_j e A_i sono la colonna j-esima e la riga i-esima di A e le seconde uguaglianze derivano dal fatto che il minimo ed il massimo cercati si ottengono con strategie pure.

L'obiettivo del giocatore I è quindi quello di massimizzare la sua vincita minima $v(x)$, ottenendo la quantità:

$$v_I = \max_{x \in X} \min_{j} \left\{ x^T A_j \right\}; \tag{3.11}$$

e quello del giocatore II di minimizzare la sua massima perdita $v(y)$, ottenendo la quantità:

$$v_{II} = \min_{y \in Y} \max_{i} \{A_i y\} . \tag{3.12}$$

Definizione 3.5.8 *La strategia mista x che permette al giocatore I di ottenere v_I è detta maximin, mentre la strategia mista y che permette al giocatore II di ottenere v_{II} è detta minimax. v_I e v_{II} sono detti invece, valori del gioco per i giocatori I e II rispettivamente.*

Teorema 3.5.9 ***Teorema del minimax*** *(von Neumann 1928)*

$$v_I = v_{II}. \tag{3.13}$$

Il risultato più importante è però il seguente teorema.

Teorema 3.5.10 *Qualsiasi gioco in cui ogni giocatore ha un numero finito di strategie, ammette almeno un equilibrio di Nash in strategie miste.*

Per questo motivo, anche i giochi che non presentano punti di equilibrio in strategie pure, ne hanno sicuramente uno in strategie miste. Inoltre è possibile che un gioco abbia punti di equilibrio sia in strategie pure che miste come vedremo nel prossimo paragrafo.

3.5.4 Calcolo dell'equilibrio di Nash in strategie miste

Per capire come calcolare gli equilibri di Nash in strategie miste, in un gioco a due giocatori si considera il seguente esempio.

Esempio 3.5.11 ***(Battaglia dei sessi)*** *Due fidanzati devono scegliere tra andare al teatro (T) o alla partita (P). La ragazza (giocatore I) preferisce il teatro, mentre il ragazzo (giocatore II) preferisce la partita, ma entrambi non hanno interesse a restare da soli. Il seguente gioco può essere rappresentato in forma caratteristica come riportato in Tabella 3.2. Come si può notare, questo gioco*

I/II	*T*	*P*
T	2, 1	0, 0
P	0, 0	1, 2

Tabella 3.2. Battaglia dei sessi

presenta due punti di equilibrio di Nash in strategie pure, che sono le coppie di strategie (T, T) e (P, P).

Procediamo ora al calcolo delle strategie miste per l'Esempio 3.5.11. Se il giocatore I gioca la strategia mista $(p, 1-p)$ e il giocatore II gioca la strategia $(q, 1-q)$, la vincita attesa del giocatore I è data da:

$$v_I(p) = 2pq + 0(1-p)q + 0p(1-q) + 1(1-p)(1-q) = (3q-1)p - (q-1)$$

Il secondo termine non dipende da p, cioè dal giocatore I; si hanno quindi tre casi:

$$3q - 1 > 0 \Rightarrow p = 1 \ \ (strategia\ pura)$$
$$3q - 1 = 0 \Rightarrow p \ \ qualsiasi$$
$$3q - 1 < 0 \Rightarrow p = 0 \ \ (strategia\ pura)$$

Analogamente, la vincita attesa per il giocatore II è data da:

$$v_{II}(q) = 1pq + 0(1-p)Q + 0p(1-q) + 2(1-p)(1-q) = (3p-2)q - 2(p-1)$$

a cui corrispondono tre casi:

$$3p - 2 > 0 \Rightarrow q = 1 \ \ (strategia\ pura)$$
$$3p - 2 = 0 \Rightarrow q \ \ qualsiasi$$
$$3p - 2 < 0 \Rightarrow q = 0 \ \ (strategia\ pura)$$

Oltre agli equilibri di Nash in strategie pure (T, T) e (P, P), si ha un equilibrio in strategie miste se:

$$3q - 1 = 0 \Rightarrow q = \frac{1}{3}$$
$$3p - 2 = 0 \Rightarrow q = \frac{2}{3}$$

portando al punto

$$\left(\left(\frac{2}{3}, \frac{1}{3}\right), \left(\frac{1}{3}, \frac{2}{3}\right)\right).$$

3.6 Giochi Cooperativi

In un gioco cooperativo, alcuni giocatori, possono voler perseguire un fine comune, pertanto è possibile che alcuni di essi abbiano interesse ad associarsi per migliorare il proprio guadagno. Pertanto deve essere possibile:

- permettere ai giocatori di comunicare e stipulare accordi tra loro;
- far rispettare gli accordi stipulati, deve esistere un'autorità accettata da tutti i giocatori.

Un'ulteriore suddivisione di giochi cooperativi fa riferimento a come i giocatori di una coalizione possono spartirsi la vincita ottenuta. A tal fine si distinguono due sottoclassi:

- giochi cooperativi senza pagamenti laterali (NTU);
- giochi cooperativi a pagamenti laterali (TU).

Per i giochi NTU, tutti i giocatori ricevono un payoff assegnato, in caso di vincita. In un gioco TU, invece, i giocatori di una coalizione possono ripartirsi la vincita in qualsiasi modo.

3.6.1 Giochi cooperativi senza pagamenti laterali

Questa tipologia di gioco è stata introdotta da Aumann e Peleg nel 1960. Ogni giocatore utilizza le proprie strategie in accordo con gli altri giocatori con cui ha formato una coalizione, ma consegue una vincita indipendentemente dagli altri.

Definizione 3.6.1 *Un gioco NTU è una coppia $G = (N, V)$ dove N è l'insieme dei giocatori e V è la funzione che ad ogni coalizione $S \subset N$, associa l'insieme dei payoff ammissibili per i giocatori di S, tali che:*

- $V(S) \subset \mathbb{R}^S$
- *V(S) è chiuso e non vuoto*

3.6.2 Problema di contrattazione a due giocatori senza pagamenti laterali

Un'applicazione dei giochi cooperativi NTU è data dal problema di contrattazione a due giocatori (Nash, 1950). I due giocatori possono accordarsi per la strategia da adottare e giocare qualunque elemento dello spazio delle strategie $X \times Y$. Sotto opportune ipotesi di compattezza dell'insieme delle strategie possibili e di comportamento delle funzioni di utilità, l'immagine dello spazio delle utilità è l'insieme V, chiuso e convesso (Figura 3.2). Al giocatore i-esimo si assegna un valore di riferimento $d_i, \ \ i = 1, 2$, che generalmente, corrisponde alla soluzione non cooperativa di Nash, e si definisce il punto $d = (d_1, d_2)$, che costituisce il payoff dei giocatori nel caso non raggiungano un accordo. Si considera quindi il sottoinsieme indicato con $F = V \cap \{(x_1, x_2) \mid x_1 \geq d_1, \ x_2 \geq d_2\}$, chiuso, convesso, limitato e non vuoto. Esso costituisce l'insieme dei payoff che i giocatori possono raggiungere contrattando (Figura 3.2).

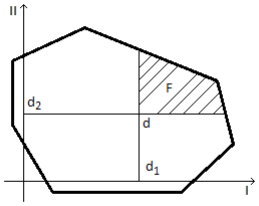


Figura 3.2. Problema di contrattazione a due giocatori

Definizione 3.6.2 *Un problema di contrattazione a due giocatori è rappresentato dalla coppia* (F, d) *con* $F \subset \mathbb{R}^2$ *chiuso, convesso, limitato e non vuoto, e* $d = (d_1, d_2)$.

3.6.3 Soluzione assiomatica di Nash

Una soluzione $\Phi(F, d)$ di un problema di contrattazione a due giocatori $(F, d) \in C$, dove C è l'insieme dei problemi di contrattazione, si determina con una funzione $\Phi : C \to \mathbb{R}^2$ tale che $\Phi(F, d) \in F$ e che soddisfa i seguenti *assiomi di Nash*:

1. **efficienza stretta:** la soluzione appartiene alla frontiera paretiana stretta, cioè non può essere migliorabile per ogni giocatore:

$$x \in F, x \geq \Phi(F, d) \Rightarrow x = \Phi(F, d) \tag{3.14}$$

2. **razionalità individuale:**

$$\Phi(F, d) \geq d \tag{3.15}$$

   con la relazione d'ordine di $\mathbb{R}^2$.

3. **invarianza**: la soluzione è invariante per le trasformazioni lineari, cioè $\forall \lambda_1, \lambda_2 \in \mathbb{R}_>$ e $\forall \mu_1, \mu_2 \in \mathbb{R}$, siano

$$\hat{F} = \{(\lambda_1 x_1 + \mu_1, \lambda_2 x_2 + \mu_2) \mid (x_1, x_2) \in F\} \tag{3.16}$$

   e

$$\hat{d} = (\lambda_1 d_1 + \mu_1, \lambda_2 d_2 + \mu_2) \tag{3.17}$$

   allora

$$\Phi(\hat{F}, \hat{d}) = (\lambda_1 \Phi_1(F, d) + \mu_1, \lambda_2 \Phi_2(F, d) + \mu_2) \tag{3.18}$$

4. **simmetria:** se F è simmetrico per i due giocatori, cioè entrambi possono ottenere gli stessi payoff, $(a,b) \in F \Rightarrow (b,a) \in F \; e \; d_1 = d_2$, allora si ha:
$$\Phi_1(F,d) = \Phi_2(F,d) \tag{3.19}$$

5. **indipendenza dalle alternative irrilevanti:** se si elimina un sottoinsieme di F non contenente $\Phi(F,d)$, la soluzione resta invariata, cioè
$$d,\ \Phi(F,d) \in G \subset F \Rightarrow \Phi(G,d) = \Phi(F,d). \tag{3.20}$$

Teorema 3.6.3 *Esiste un'unica funzione $\Phi : C \to \mathbb{R}^2$ che soddisfa gli assiomi di Nash, quella che massimizza il prodotto di Nash:*
$$\Phi(F,d) = \arg \max_{x_1,x_2 \in F} \{(x_1 - d_1)(x_2 - d_2) \mid x \in F\} = N_S. \tag{3.21}$$

3.6.4 Giochi cooperativi a pagamenti laterali

I giochi cooperativi a pagamenti laterali (TU), sono stati introdotti da von Neumann e Morgestern nel 1944. In un gioco TU, i giocatori possono stipulare accordi vincolanti tra loro, ma, a differenza dei giochi NTU, possono ripartirsi la vincita con un accordo al di fuori delle regole del gioco.

Definizione 3.6.4 *Un gioco TU è una coppia $G = (N,v)$, dove N è l'insieme dei giocatori e v la funzione caratteristica, con $v(\emptyset) = 0$.*

Se i valori della funzione v sono negativi si ha un gioco di costi o *cost game* (N,c), in cui si pone $c = -v$, in modo da operare con quantità non negative.

Definizione 3.6.5 *Un gioco $G = (N,v)$ si dice monotono se $v(S) \leq v(T),\ \forall S \subseteq T$.*

Definizione 3.6.6 *Un gioco $G = (N,v)$ si dice convesso se vale una delle seguenti condizioni equivalenti:*

- $v(s) + v(T) \leq v(S \cup T) + v(S \cap T),\ \forall S,T \subseteq N$;
- $v(S \cup \{i\}) - v(S) \leq v(T) \cup \{i\}) - v(T),\ \forall S \subset T \subseteq N \setminus \{i\},\ \forall i \in N$.

Definizione 3.6.7 *Un gioco $G = (N,v)$ si dice semplice se le coalizioni possono assumere solo i valori 0 e 1. Se una coalizione ha valore 1 è detta vincente, se ha valore 0 è detta perdente.*

Definizione 3.6.8 *Un gioco $G = (N, v)$ si dice coesivo se per ogni partizione di N, $\{S_1, S_2, \ldots, S_k\}$ si ha:*

$$\sum_{i=1}^{k} v(S_i) \leq v(N) \tag{3.22}$$

Le soluzioni di un gioco TU possono essere raggruppate in due categorie:

- *Soluzioni insiemistiche* che individuano un insieme di vettori di payoff che ripartiscono il valore del gioco tra tutti i giocatori;
- *Soluzioni puntuali* che individuano una sola ripartizione ottimale e che quindi riprendono l'idea classica di soluzione di un problema.

3.6.5 Soluzioni insiemistiche di un gioco TU

Un idea per dividere le vincite del gioco tra i membri di una coalizione, può essere quella di risolvere un sottogioco ristretto ai giocatori di ciascuna coalizione, oppure quella di suddividere, in parti uguali, la vincita, trascurando il contributo dei singoli giocatori della coalizione. Tuttavia esistono metodi più complessi che meglio tengono conto del ruolo svolto da ciascun giocatore e che definiscono altri concetti di soluzione.

Definizione 3.6.9 *Dato un gioco $G = (N, v)$ si dice imputazione del valore del gioco o soluzione del gioco, un vettore $x = (x_1, x_2, \ldots, x_n)$ tale che:*

$$\sum_{i \in N} x_i = v(N) \tag{3.23}$$

$$x_i \geq v(i), \quad i = 2, \ldots, n \tag{3.24}$$

La prima espressione identifica l'ipotesi di efficienza, mentre la seconda quella di razionalità individuale dei membri della coalizione. Nel caso di cost game, la razionalità individuale richiede $x_i \leq c(i)$.

L'insieme di tutte le imputazioni si indica con $E(v)$.

Definizione 3.6.10 *Se per un gioco $G = (N, v)$ si ha:*

$$\sum_{i \in N} v(i) = v(N) \tag{3.25}$$

allora $E(v)$ ha come unico elemento $x = (v(1), v(2), \ldots, v(n))$ ed il gioco è detto inessenziale. In caso contrario il gioco è detto essenziale.

Per l'ipotesi di razionalità individuale, un'imputazione deve assegnare, ad ogni giocatore, almeno quanto egli riuscirebbe ad ottenere non coalizzandosi. Per questo motivo, ogni concetto di soluzione dovrà soddisfare questa condizione, ovvero dovrà essere un'imputazione. Se il gioco è essenziale, esistono, però, più imputazioni possibili e si ripresenta il problema di scegliere la soluzione. L'ipotesi 3.23, afferma che la somma degli elementi di ogni imputazione è costante. Per questo motivo, se due imputazioni x e y sono distinte, esiste almeno un giocatore k per cui $x_k > y_k$ e almeno un giocatore h per cui $x_h < y_h$.

Definizione 3.6.11 *Date $x, y \in E(v)$ e una coalizione S, si dice che x domina y mediante S, e si indica con $x \succ_S y$, se*

- $x_i > y_i \ \forall i \in S$
- $x(S) \leq v(S)$ *dove* $x(S) = \sum_{i \in S} x_i$

Definizione 3.6.12 *Date $x, y \in E(v)$ si dice che x domina y, e si indica con $x \succ y$, se esiste almeno una coalizione S, tale che $x \succ_S y$.*

3.6.6 Insiemi stabili

Un insieme stabile definisce un primo concetto di soluzione per un gioco TU, privilegiando alcune imputazioni rispetto ad altre.

Definizione 3.6.13 *Un insieme $V \subset E(v)$ si dice stabile se:*

- *dati $x, y \in E(v)$ si ha $x \not\succ y$ e viceversa $\Rightarrow$ stabilità interna*
- *dato $x \notin V$, $\exists y \in V$ per il quale si ha $y \succ x \Rightarrow$ stabilità esterna.*

Un insieme stabile contiene la soluzione del gioco ma la decisione dipende da altre informazioni del gioco, non espresse dalla forma caratteristica.

3.6.7 Nucleo

Il nucleo è considerato il concetto di soluzione insiemistica più interessante per numerose classi di giochi TU. Introdotta da Gillies nel 1953, l'idea di base consiste nel considerare il comportamento delle imputazioni rispetto alle coalizioni, richiedendo che sia verificata l'ipotesi di razionalià per ogni coalizione:

$$x(S) \geq v(S) \quad S \subset N. \tag{3.26}$$

Definizione 3.6.14 *Si dice nucleo di un gioco, l'insieme:*

$$C(v) = \{x \in E(v) \mid x(S) \geq v(S),\ \forall S \subset N\} \tag{3.27}$$

Nel caso di un cost game (N, c) *la razionalità della coalizione richiede* $x(S) \leq c(S),\ \forall S \subset N$.

Le imputazioni non dominate costituiscono il nucleo del gioco e quindi rappresentano delle possibili soluzioni. Il nucleo potrebbe, però, essere vuoto. In questo caso le coalizioni saranno caratterizzate da un certo grado di instabilità. Risulta quindi interessante capire se un gioco ha o meno nucleo non vuoto.

Definizione 3.6.15 *Una collezione* $B = \{S_1, S_2, \ldots, S_m\}$ *di sottoinsiemi di* N *è detta bilanciata se esistono* m *numeri non negativi* $y_1, y_2, \ldots, y_m$*, detti coefficienti di bilanciamento, tali che:*

$$\sum_{S_j \ni i} y_j = 1 \quad \forall i \in N. \tag{3.28}$$

Definizione 3.6.16 *Una collezione bilanciata è detta minimale se nessuna sottocollezione è bilanciata.*

Definizione 3.6.17 *Un gioco è detto bilanciato se, per ogni collezione bilanciata minimale* $B = \{S_1, S_2, \ldots, S_m\}$ *con coefficienti di bilanciamento* $y_1, y_2, \ldots, y_m$*, si ha:*

$$\sum_{j=1}^{m} y_j v(S_j) \leq v(N). \tag{3.29}$$

Inoltre si possono definire le seguenti proprietà:

- Una collezione bilanciata è minimale se e solo se i coefficienti di bilanciamento sono unici.
- Le collezioni bilanciate non dipendono dalla funzione caratteristica, ma solo da N.
- Ogni collezione bilanciata è unione di collezioni bilanciate minimali.

Teorema 3.6.18 ***(Bondareva - Shapley)*** *Un gioco TU* $G = (N, v)$ *ha nucleo non vuoto se e solo se è bilanciato.*

Per questo motivo un gioco a nucleo non vuoto viene anche detto bilanciato. Oltre a dire se un gioco ha nucleo non vuoto, il teorema 3.6.18 risulta particolarmente utile per dimostrare che un gioco ha nucleo vuoto, in quanto è sufficiente trovare una collezione bilanciata che non verifica la 3.29.

3.6.8 Soluzioni puntuali di un gioco TU

Le soluzioni puntuali di un gioco TU sono anche definite come "indici di potere" poiché permettono di identificare il potere di ogni giocatore all'interno del gioco.

3.6.9 Il valore di Shapley

Questo concetto di soluzione si basa sul valore che ogni giocatore è in grado di aggiungere alle possibili coalizioni a cui appartiene, ovvero il suo contributo marginale.

Definizione 3.6.19 *Si chiama valore di Shapley il vettore $\phi(v)$, la cui componente i-esima ϕ_i identifica il contributo marginale medio del giocatore i-esimo rispetto alle possibili permutazioni dei giocatori:*

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi} [v(P(\pi, i) \cup \{i\}) - v(P(\pi, i))] \tag{3.30}$$

dove $n = |N|$, π è una permutazione di N e $P(\pi, i)$ è l'insieme dei giocatori che precedono i nella permutazione π.

Il valore di Shapley per un gioco TU esiste ed è unico. Inoltre, se il gioco è subadditivo questo valore è un'imputazione in quanto verifica:

$$\sum_{i \in N} \phi_i(v) = v(N) \tag{3.31}$$

$$\phi_i(v) \geq v(i) \quad \forall i \in N \tag{3.32}$$

Nella precedente sezione è stato presentato il concetto di soluzione insiemistica, come imputazione appartenente al nucleo. Se il gioco è convesso (concavo nel caso di cost game) il valore di Shapley è un elemento del nucleo. In generale, applicando la definizione, il valore di Shapley risulta molto difficile da calcolare. A tal fine, è infatti necessario determinare i contributi marginali dei giocatori in tutte le possibili coalizioni ordinate, che sono $n!$ (es. per 10 giocatori è necessario considerate $10! = 3.628.800$ permutazioni per ogni giocatore). Una semplificazione si può ottenere considerando tutte le $2^n - 1$ coalizioni non vuote, e per ciascuna considerare ogni giocatore come l'ultimo arrivato e pesare il suo contributo marginale con le permutazioni degli altri giocatori, facenti e non, parte della coalizione. In questo modo si ottiene la seguente espressione:

$$\phi_i(v) = \sum_{S \subseteq N,\ i \in S} \frac{(s-1)!(n-s)!}{n!} [v(S) - v(S \setminus \{i\})] \tag{3.33}$$

Utilizzando l'espressione (3.33), nel caso di 10 giocatori si considerano quindi, solo $2^{10} - 1 = 1.023$ coalizioni.

3.7 Applicazione della teoria dei giochi

In questo capitolo sono stati introdotti i concetti che stanno alla base della teoria dei giochi e gli strumenti matematici per risolvere un problema per mezzo di questa teoria. Molte situazioni, nella vita reale, possono essere rappresentati in forma di gioco. Alcuni degli ambiti in cui la teoria dei giochi trova applicazione sono quello economico e politico: grandi marchi che competono per la supremazia nel mercato o candidati politici che competono per ottenere il voto, sono esempi di situazioni di gioco con diversi giocatori e diverse strategie da adottare. Sono state inoltre identificate due importanti classi di giochi, in base alla possibilità dei giocatori di comunicare e stipulare accordi tra loro. In molti casi un gioco cooperativo può essere conveniente a tutti i giocatori, che accordandosi possono ottenere guadagni maggiori rispetto a quelli che avrebbero ottenuto giocando individualmente. Nel seguito, sarà illustrata un'altra situazione in cui la teoria dei giochi trova applicazione. Questo ambito riguarda il problema della classificazione dei pacchetti video, con codifica MD, in reti P2P. Questa tesi si propone di utilizzare la teoria dei giochi, ed in particolare i giochi con coalizioni, per migliorare le prestazioni ottenute dagli algoritmi di classificazione precedenti. In particolare, nel prossimo capitolo sarà definito il meccanismo di accesso *DiffServ* ed introdotte le classi di servizio da esso supportate. Queste classi, rappresenteranno le possibili strategie che potranno essere adottate dai giocatori del gioco, ovvero i pacchetti delle diverse descrizioni del flusso video.

Capitolo 4

Differentiated Services

4.1 Introduzione

In questo capitolo si introduce un particolare meccanismo di accesso alla rete chiamato *Differentiated Service (DiffServ)*, che permette di definire diverse classi di Servizio (QoS) che saranno utilizzate per la classificazione dei pacchetti video. In seguito definiremo il protocollo *srTCM (single rate Three Color Marker)* utilizzato da DiffServ per implementare la classificazione utilizzando tre differenti livelli di priorità.

4.2 Qualità del servizio (QoS)

Prima di definire un meccanismo per la classificazione dei pacchetti video è necessario definire cosa si intende per qualità del servizio (QoS) ed in particolare per classe di servizio (CoS). Il Termine QoS è oggi utilizzato con una vasta varietà di significati. I *service providers* utilizzano il termine QoS per indicare l'efficienza del servizio offerto, mentre, per un utente comune, QoS indica la disponibilità del servizio richiesto. L'introduzione di reti basate sulla possibilità di offrire diversi livelli di QoS è nata dall'idea di poter differenziare e misurare le caratteristiche di ogni comunicazione e, per ognuna di esse, garantire un dato livello di qualità. Il gestore di rete garantisce, ad ogni utente, il corretto trasferimento delle informazioni in accordo con i parametri stabiliti in fase di negoziazione delle caratteristiche richieste. I requisiti della comunicazione sono generalmente indicati dall'utente in termini di banda necessaria, ritardo massimo consentito e percentuale di pacchetti persi. Le richieste dell'utente saranno quindi approvate nel caso in cui esse possano essere supportate dalla rete valutando eventuali congestioni sui link. La rete dovrà quindi garantire i parametri negoziati per tutta la durata della conversazione.

Per classe di servizio (CoS) si intende un meccanismo di gestione del traffico nella rete attraverso la distinzione di diverse tipologie di traffico. Categorie di servizi ed applicazioni con caratteristiche di traffico differenti (es. e-mail, video streaming, traffico voce, file sharing) sono suddivise in classi e ad ognuna di esse è assegnato un diverso livello di priorità di servizio. A differenza del concetto di qualità del servizio, CoS non garantisce determinate caratteristiche in termini di banda disponibile o ritardo massimo consentito, ma definisce solamente il livello di priorità rispetto agli altri flussi di dati presenti nella rete. QoS e CoS possono quindi essere considerati come due parametri distinti [**?**], con i quali la rete cerca di offrire il miglior servizio possibile ad una data categoria di traffico in accordo con il continuo cambiamento delle condizioni della rete. Nella prossima sezione saranno definite meglio quali sono le richieste dell'utente e come avviene la negoziazione del servizio con il gestore di rete.

4.3 Service Level Agreement (SLA)

Con il termine service level agreement (SLA) si intende un contratto tra il gestore di rete e l'utente che, in questo contesto, diventa un cliente della rete. Questo contratto permette di descrivere i servizi che la rete è in grado di offrire al cliente e di parametrizzare quanto quest'ultimo è disposto a pagare per usufruirne. Lo SLA viene quindi negoziato quando l'utente effettua la richiesta di un nuovo servizio o intende modificare le caratteristiche di un servizio già a sua disposizione. Stipulato il contratto, ogni connessione per un dato utente avrà le caratteristiche definite ed approvate in fase di negoziazione. Gli elementi che caratterizzano lo SLA sono generalmente i seguenti:

1. Lato cliente:

    - **Specifiche di traffico:** caratterizzazione del traffico che l'utente può inoltrare nella rete. Questo generalmente si traduce in un opportuno dimensionamento dei buffer di ricezione al caso peggiore.
    - **Garanzia:** descrizione delle garanzie sulle specifiche di traffico definite e livello di servizio garantito.
    - **Prezzo da pagare per usufruire del servizio**

2. Lato gestore di rete:

    - **Descrizione del servizio:** descrizione del servizio offerto al cliente in forma procedurale o comportamentale.

- **Descrizione della qualità:** caratterizzazione delle risorse di rete riservate all'utente e definizione dei parametri quantificabili del servizio (es. throughput, delay. jitter, tempo di risposta).
- **Disponibilità del servizio:** definizione di eventuali interruzioni del servizio (es. link guasti) e degradazioni temporanee del servizio (es. alcune risorse sono momentaneamente inutilizzabili o non raggiungibili).
- **Supporto tecnico:** livello di supporto garantito e prezzo per usufruirne (es. disponibilità telefonica e prezzo per la chiamata).
- **Rimborso:** Quantitativo di denaro eventuale pagato dal gestore di rete al cliente per occasionali interruzioni o degradazioni del servizio.

4.4 Differentiated Services

Differentiated Services (DiffServ) è un meccanismo in base al quale il gestore di rete è in grado di offrire diversi livelli di QoS a diversi flussi di traffico. L'idea del modello DiffServ è quella di fornire un certo grado di granularità alla qualità del servizio dei diversi flussi che attraversano la rete. Questi flussi saranno poi raggruppati in diverse classi di servizio. Ad ogni pacchetto è quindi assegnata una data classe di servizio che sarà mappata all'interno di un campo dell'intestazione del pacchetto IP o RTP chiamato *DiffServ Code Point (DSCP)*. All'interno della rete, la classe di appartenenza si traduce in un determinato meccanismo di *scheduling* e *routing* chiamato *Per Hop Behaviour (PHB)*. Il meccanismo di scheduling indica un insieme di regole che definiscono l'importanza di una classe rispetto alle altre. Queste regole caratterizzano il quantitativo di risorse dedicate al traffico appartenente ad una data classe ed identificano il comportamento dei nodi interni alla rete per quel flusso, in caso di congestione. Le caratteristiche principali del meccanismo di classificazione sono le seguenti:

- la classificazione dei pacchetti avviene solo al momento dell'ingresso dei pacchetti nella rete;
- i meccanismi si scheduling e routing dei pacchetti sono basati solo sul valore del campo DSCP che ne identifica le classi di appartenenza;
- nessuna risorsa è pre-allocata all'interno della rete per una determinata classe di servizio.

4.4.1 Traffic Conditioning Agreement (TCA)

Nella sezione 4.3 è stato definito il concetto di SLA. Nel modello DiffServ esso però assume un significato diverso. DiffServ non è un servizio che un cliente acquista ma è un meccanismo che il gestore di rete utilizza per garantire uno o più servizi. Per questo motivo, è stato definito il concetto di *Traffic Conditioning Agreement (TCA)*. Esso svolge l'azione di filtro per i diversi flussi di traffico nella rete separando e classificando i pacchetti appartenenti a diverse classi di servizio. TCA, inoltre, fa uso di un insieme di parametri che specificano le regole per effettuare la classificazione. Questi parametri sono noti come *Traffic Conditioning Specification (TCS)*.

4.4.2 Domini di servizio nel modello DiffServ

Un dominio DiffServ è un insieme di nodi che operano seguendo una politica comune di classificazione e di comportamento da seguire per i pacchetti appartenenti a diverse classi di servizio. Un dominio DiffServ è composto da due diverse classi di nodi: i nodi di accesso e i nodi interni alla rete. I nodi di accesso si occupano di classificare i pacchetti al loro ingresso nella rete, ovvero impostare il valore del campo DSCP. I nodi interni alla rete utilizzano una diversa politica di scheduling ed instradamento per i vari pacchetti in base alla loro classe di appartenenza. Questa politica è chiamata *Per Hop Behaviour (PHB)* ed è in corrispondenza biunivoca con il valore del campo DSCP. I nodi di accesso, si occupano inoltre di collegare un dominio DiffServ con un altro dominio DiffServ o non DiffServ. I nodi interni invece possono essere collegati soltanto ad altri nodi appartenenti allo stesso dominio di amministrazione (vedi Figura 4.1).

4.4.3 Regioni DiffServ

Una regione DiffServ è composta da uno o più domini DiffServ contigui. I domini che compongono una regione DiffServ possono supportare diversi PHB e persino diverse mappe DSCP $\rightarrow$ PHB. La definizione dei diversi PHB per i vari domini è contenuta all'interno del TCA dove saranno indicate le differenze di comportamento dei nodi interni alla rete a seconda del dominio di appartenenza. La classificazione dei pacchetti è invece effettuata all'interno dei nodi di accesso ad un dato dominio.

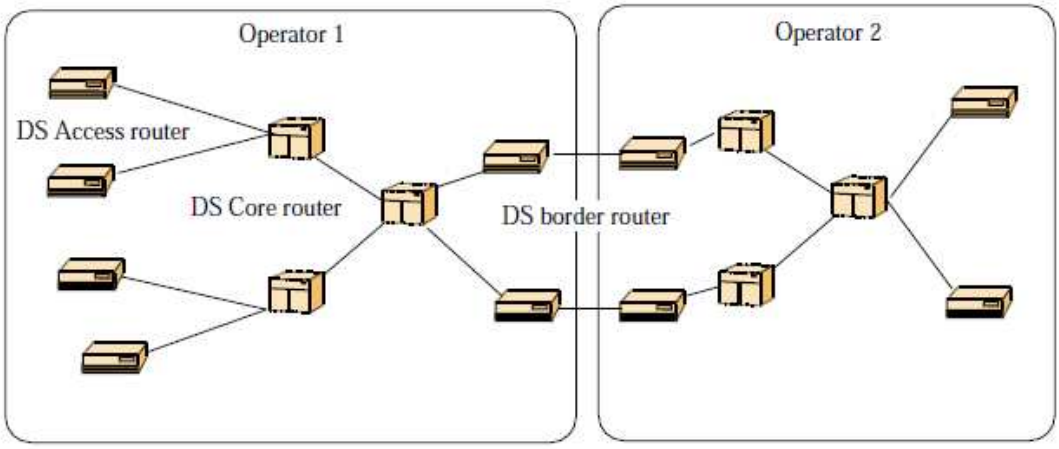


Figura 4.1. Struttura di un dominio DiffServ ed interconnessione di più domini.

4.4.4 Profili di traffico

Il profilo di traffico definisce le regole per determinare se un particolare pacchetto è conforme o meno alle specifiche definite all'interno del TCS. Un esempio di profilo è il seguente:

$$DSCP = X \Rightarrow \ \text{token bucket} \ \ \text{b}, \ \ \text{r}$$

Questo profilo indica che le caratteristiche di tutti i pacchetti marcati con codepoint X saranno confrontate con i valori di rate r e di buffer b definite all'interno del TCA. Un esempio più concreto sarà definito nel seguito del capitolo con l'introduzione del protocollo srTCM. I pacchetti che non rispettano le specifiche potranno essere messi in coda, in attesa che rispettino i requisiti (*shaped*), scartati (*policed*) o riclassificati con un diverso codepoint (*re-marked*). L'operazione di riclassificazione consiste nel definire un nuovo codepoint che corrisponde ad una classe di servizio a più bassa priorità.

4.4.5 Un nodo DiffServ

Un nodo appartenente ad un dominio DiffServ deve essere in grado di implementare sia le operazioni di classificazione dei pacchetti (nodi di accesso), sia di separazione dei flussi di traffico appartenenti a diverse classi di servizio. Per questi motivi, un nodo o router della rete è composto da due componenti principali (Figura 4.2):

- *Traffic Conditioning* (TC) con il compito di svolgere le principali operazioni di controllo e classificazione dei pacchetti.
- *Per Hop Behaviour* (PHB) con il compito di separare i pacchetti appartenenti a diverse classi di servizio e per ognuna di esse implementare un diverso meccanismo di *forwarding* del traffico.

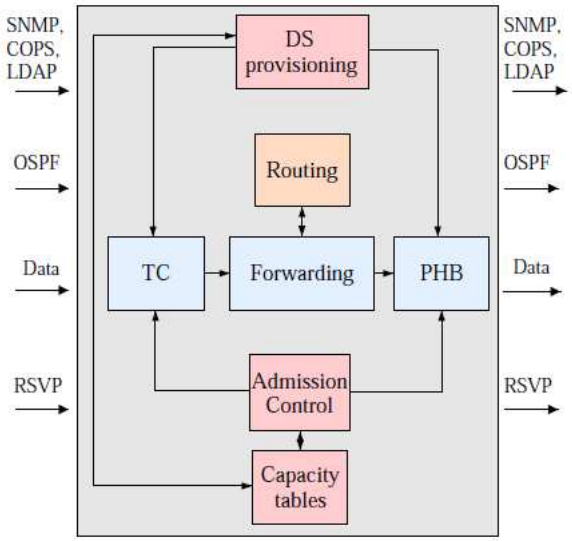


Figura 4.2. Schema dei blocchi principali contenuti all'interno di un nodo DiffServ e loro interconnessioni logiche.

4.4.6 Il blocco di Traffic Conditioning (TC)

In un modello DiffServ, il blocco TC ha diversi compiti in funzione del tipo di nodo in cui risiede. In un nodo di accesso esso ha il compito di classificare

i pacchetti seguendo le regole specificate all'interno del TCA. In un nodo interno alla rete, invece, il blocco TC separa il flusso di traffico, raggruppando i pacchetti appartenenti alla stessa classe di servizio. Le funzioni principali del blocco TC sono quindi:

1. classificazione dei pacchetti;
2. analisi del traffico.

Classificazione dei pacchetti

La politica di classificazione dei pacchetti definisce diverse classi di servizio per diversi flussi di traffico che possono attraversare i nodi di un dominio DiffServ. Si possono definire due diverse tipologie di classificatori:

- *Behaviour Aggregate* (BA)
- *Multi-Field* (MF)

Il classificatore BA separa i pacchetti dello stream soltanto in base al valore contenuto all'interno del campo DSCP. Questo valore è l'unico elemento di cui necessitano i nodi interni alla rete per distinguere le diverse classi di servizio ed applicare diversi PHB. Il classificatore MF classifica i pacchetti anche in base a più campi contenuti all'interno dell'header del pacchetto (es. sorgente, destinazione, numeri di porta ed altre informazioni). Entrambi i classificatori devono essere configurati attraverso procedure che rispettano le specifiche definite all'interno del TCA.

Analisi del traffico

Il blocco di analisi del traffico, oltre a separare i pacchetti appartenenti a diverse CoS, si occupa di verificare che la classificazione sia ancora valida, ovvero rispetti le specifiche definite nel TCA. Questo blocco è composto da un insieme di componenti, ognuno con una precisa funzione:

- **Meter:** questo componente ha il compito di misurare i rate dei vari stream di traffico appartenenti a diverse classi di servizio e di confrontarli con i profili di traffico specificati nel TCA.
- **Marker:** esso ha il compito di impostare o modificare il valore del campo DSCP. In un nodo di accesso alla rete, il marker definisce il codepoint per la prima volta in fase di classificazione. In un nodo interno alla rete, invece, ricevute le informazioni dal blocco di meter, il marker può operare una riclassificazione dei pacchetti che non rispettano il profilo di traffico imposto dalla classificazione precedente.

- **Shaper:** questo blocco offre un'alternativa alla riclassificazione. Esso si occupa di ritardare la trasmissione di uno o più pacchetti che non rispettano il profilo di traffico assegnatogli. In questo modo i pacchetti torneranno a essere conformi alla loro classe di appartenenza. I pacchetti che non rispettano le specifiche saranno quindi posti in una coda in attesa di essere trasmessi. Se la coda è piena i nuovi pacchetti in arrivo saranno scartati dal blocco di *dropper*.
- **Dropper:** esso ha il compito di scartare uno o più pacchetti che non rispettano il profilo di traffico nel caso non siano possibili né riclassificazioni, né operazioni di *shaping*.

Lo schema degli elementi dei blocchi di classificazione e analisi sono illustrati in Figura 4.3.

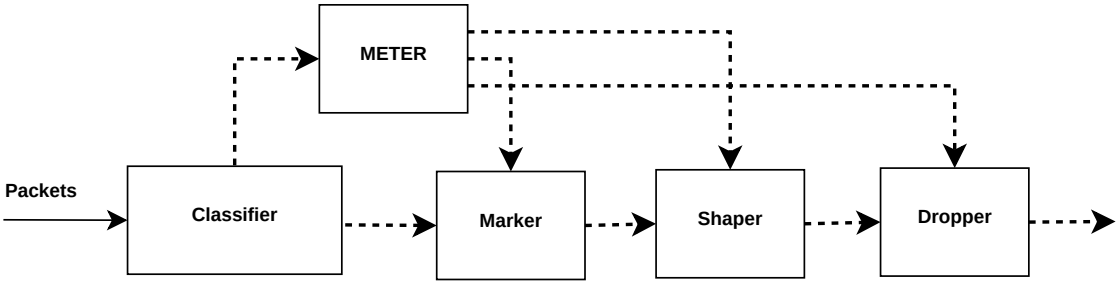


Figura 4.3. Modello logico delle operazioni di classificazione dei pacchetti ed analisi del traffico

4.4.7 Per Hop Behaviour (PHB)

Un Per Hop Behaviour (PHB) rappresenta una descrizione del comportamento che un nodo DiffServ assume nei confronti del traffico appartenente ad una determinata classe di servizio. Ognuno di questi comportamenti è chiamato *Behaviour Aggregate* (BA). Un PHB associa quindi, ad una classe di servizio, caratteristiche specifiche di allocazione di banda, probabilità di perdita di pacchetti, ritardi di trasmissione e jitter. Il compito di un nodo è quindi quello di allocare le opportune risorse per ogni BA implementato al suo interno. Un esempio di PHB consiste nel garantire una minima allocazione di banda X, al traffico appartenente ad una data CoS. La banda in eccesso sui vari link potrà poi rimanere inutilizzata o essere suddivisa proporzionalmente tra i diversi BA. Un PHB deve essere inoltre specificato in termini di priorità rispetto agli altri PHB implementati in un nodo e PHB con caratteristiche analoghe sono raggruppati

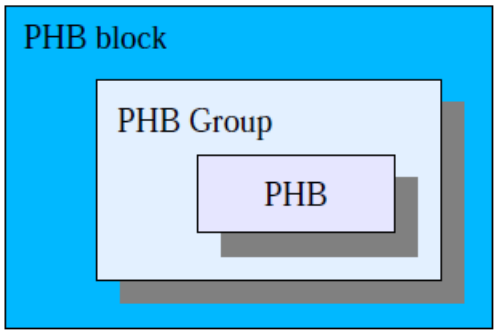


Figura 4.4. Struttura di un PHB all'interno di un nodo della rete

in gruppi (Figura 4.4). PHB appartenenti ad uno stesso gruppo condividono stesse politiche di scheduling e gestione dei buffer nella rete. In generale, un PHB può essere implementato in vari modi all'interno di in nodo e può corrispondere ad uno o più valori di DSCP. In ogni caso, tutti i codepoint disponibili devono essere associati ad un solo PHB. I codepoint non mappati saranno assegnati ad un PHB di default.

4.5 srTCM

In questa sezione si introduce un componente utilizzato dal meccanismo DiffServ che adotta una determinata politica di analisi del traffico e classificazione dei pacchetti all'interno della rete. Questo protocollo è noto con il nome di *single rate Three Color Marker* (srTCM). Questo modello specifica il funzionamento del blocco TC, introdotto nelle precedenti sezioni, e descrive in maniera dettagliata le operazioni di *metering* e *marking* dei pacchetti. Esso utilizza tre diverse classi di servizio che sono rappresentate da tre diversi colori:

- **Rosso:** classe a minor priorità
- **Giallo:** classe a priorità intermedia
- **Verde:** classe a maggior priorità

La classificazione dei pacchetti avviene utilizzando un modello di tipo *token bucket* per l'analisi del traffico. Il modello token bucket modella la capacità del link tramite un budget di trasmissione rappresentato da dei gettoni (token) a disposizione dei pacchetti. La trasmissione di 1 byte di 1 pacchetto comporterà il costo di un gettone che verrà tolto dal contenitore (bucket). Nel caso in cui

il contenitore risulti vuoto nessun altro byte può essere trasmesso fintanto che il contenitore non si riempie di nuovo. Il bucket viene riempito periodicamente a seconda del rate di trasmissione supportato. Questo contenitore è usato per rappresentare il buffer di ricezione opportunamente dimensionato in funzione della dimensione media dei pacchetti e del rate di traffico stabilito in fase di negoziazione tra gestore della rete ed utente. L'analisi dei pacchetti avviene quindi attraverso l'utilizzo dell'informazione portata da tre parametri principali:

- **Committed Information Rate (CIR)**: esso rappresenta il rate medio di traffico ed è espresso in byte al secondo.
- **Committed Burst Profile (CBS)**: questo parametro indica la dimensione massima, espressa in byte, che il buffer di ricezione può raggiungere entro la quale i pacchetti rispettano le regole imposte dal profilo di traffico.
- **Excess Burst Size (EBS)**: questo parametro, espresso in byte, definisce la massima dimensione che il buffer di ricezione può raggiungere prima di indicare che che il traffico trasmesso non è conforme alle specifiche negoziate.

I valori di CBS ed EBS, inoltre non devono mai essere nulli ed il minimo valore che possono assumente deve essere maggiore o uguale alla massima dimensione che un pacchetto può raggiungere. Il funzionamento di base è ora il seguente: nel caso in cui, dopo l'inserimento del pacchetto nel buffer, la dimensione di quest'ultimo è contenuta entro il valore di CBS, il pacchetto sarà classificato come verde; se invece la dimensione del buffer risulta superiore al valore di CBS ma inferiore a quello di EBS, il pacchetto sarà classificato come giallo; infine se la dimensione è maggiore del valore di EBS, il pacchetto sarà classificato come rosso. Di seguito sono descritte in modo dettagliato le operazioni di meternig e buffering per il modello srTCM.

4.5.1 Metering e Marking

Il blocco di Meter può operare in due diversi modi a seconda che un pacchetto sia stato o meno già classificato precedentemente. Nel caso in cui un pacchetto non sia ancora stato classificato, il meter funziona in un modo chiamato *color-blind*. Se invece un pacchetto è stato precedentemente classificato da un'altra entità esso lavora nella tipologia *color-aware*. Il funzionamento del blocco di Meter è specificato dai parametri di due token bucket C ed E, che condividono lo stesso rate definito dal valore del CIR. Le dimensioni dei due buffer di gettoni sono però differenti: il primo modello utilizza un buffer di dimensione pari a CBS, mentre il secondo token bucket utilizza un buffer di dimensione EBS.

I due buffer dei modelli C ed E sono supposti inizialmente pieni: $T_c(0) = \mathrm{CBS}$, $T_e(0) = \mathrm{EBS}$. Successivamente, i valori di T_c e T_e, che rappresentano il contenuto dei due buffer in un certo istante, sono aggiornati CIR volte al secondo nel seguente modo:

---

if $T_c < CBS$ **then** $T_c = T_c + 1$
else if $T_e < EBS$ **then** $T_e = T_e + 1$
else T_c e T_e rimangono inalterati

Questa prima parte dell'algoritmo è comune alle due tipologie di funzionamento. Nella seconda parte dell'algoritmo si utilizza anche il blocco di Marker, che ricevute le informazioni dal Meter può procedere con la classificazione dei pacchetti. Nel caso in cui il modello srTCM funzioni nella modalità color-blind, quando un pacchetto di B byte arriva in un nodo a procedura adottata è la seguente:

---

if($T_c - B \geq 0$) **then** $T_c = T_c - B$ e il pacchetto è classificato come verde
else if($T_e - B \geq 0$) **then** $T_e = T_e - B$ e il pacchetto è classificato come giallo
else T_c e T_e invariati ed il pacchetto è classificato come rosso

---

Nel caso in cui il modello srTCM sia configurato nella modalità color-aware, quando un pacchetto di B byte arriva in un nodo a procedura adottata è la seguente:

---

if(pacchetto verde **and** $T_c - B \geq 0$) **then**
$T_c = T_c - B$ e il pacchetto rimane classificato come verde

else if(pacchetto verde **or** pacchetto giallo **and** $T_e - B \geq 0$) **then**
$T_e = T_e - B$ e il pacchetto è riclassificato come giallo

else T_c e T_e invariati ed il pacchetto è riclassificato come rosso

---

In questo secondo modello, i gettoni del buffer di dimensione CBS sono generalmente utilizzati per i pacchetti classificati con il colore verde, mentre quelli del buffer di dimensione EBS sono utilizzati per i pacchetti di colore giallo. Il modello srTCM trova utilizzo per implementare un servizio in cui i pacchetti non hanno tutti la stessa importanza, ma ci sono pacchetti che devono necessariamente giungere a destinazione o essere persi con una probabilità molto bassa

(pacchetti verdi) e pacchetti che invece, se necessario, possono essere scartati per far spazio a quelli di priorità maggiore (pacchetti gialli e rossi). Il modello che sarà trattato nel prossimo capitolo e che rappresenta l'obiettivo di questa tesi, utilizza un modello srTCM in configurazione color-aware, con alcune modifiche, per permettere una classificazione iniziale dei pacchetti al momento del loro ingresso nella rete, basata sull'utilizzo della teoria dei giochi illustrata nel Capitolo 3.

Capitolo 5

Applicazione della teoria dei giochi cooperativi alla classificazione di video a descrizioni multiple

5.1 Introduzione

Nel secondo capitolo è stata introdotta la tipologia di codifica a descrizioni multiple (MDC) per sequenze video in reti peer to peer. Questa codifica si è rivelata particolarmente efficiente per trasmettere sequenze video attraverso canali affetti da perdite. Nel quarto capitolo, invece, è stato definito un particolare meccanismo chiamato DiffServ che permette di classificare i pacchetti all'interno della rete in base al loro livello di importanza. Lavori di studio precedenti, hanno dimostrato come la differenziazione delle classi di servizio per i vari pacchetti possa portare ad un miglioramento delle prestazioni della codifica MD [17] [18]. Questo lavoro di tesi si propone di unire i vantaggi della codifica MD, all'utilizzo di diverse classi di servizio, attraverso l'introduzione di algoritmi efficienti di classificazione per i pacchetti appartenenti alle diverse descrizioni. L'idea per lo sviluppo di questi algoritmi è stata quella di definire il problema di classificazione dei pacchetti attraverso diverse tipologie di gioco, secondo la teoria definita nel terzo capitolo. Per ognuno di questi modelli, le descrizioni della codifica MD avranno il ruolo di giocatori e le classi di servizio del meccanismo DiffServ rappresenteranno le possibili strategie da adottare. Definiti i giocatori e le strategie, la soluzione del gioco consiste nell'assegnare ad ogni giocatore la strategia migliore, ovvero ad ogni pacchetto la miglior classificazione possibile. Di seguito saranno descritti nel dettaglio i sistemi di codifica e decodifica utilizzati.

5.2 Schema di trasmissione

Lo schema adottato può essere suddiviso in due stadi: codifica e decodifica. Lo stadio di codifica consiste nel suddividere la sequenza video originale in quattro descrizioni che saranno poi codificate separatamente da quattro codificatori H.264/AVC indipendenti[1]. I pacchetti RTP (Real-Time Transport Protocol) prodotti, saranno poi classificati ed inoltrati nella rete. Nel blocco di decodifica, si utilizzano quattro decodificatori H.264/AVC, uno per ogni descrizione, ed un'unità di *error concealment*, indicata come MD concealment in Figura 5.3, con il compito di stimare e ricostruire eventuali pacchetti persi in alcune descrizioni da quelli ricevuti correttamente. Di seguito saranno definiti meglio gli stadi di codifica e decodifica, descrivendo nel dettaglio i blocchi principali di cui sono composti.

5.2.1 Codifica

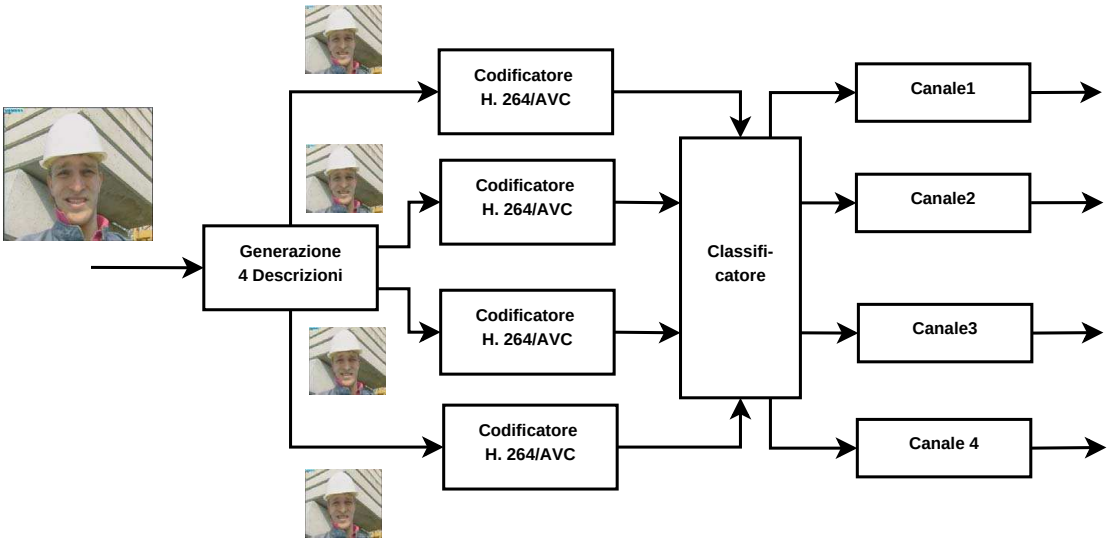


Figura 5.1. Schema di codifica di una sequenza video MD a quattro descrizioni, con classificazione dei pacchetti.

Lo stadio di codifica è illustrato in Figura 5.1. La prima operazione da compiere sulla sequenza video originale consiste nel generare le quattro descrizioni,

[1]È opportuno notare come in questo caso la complessità computazionale non cambia. I quattro codificatori infatti elaborano un quarto dei dati elaborati da un singolo codificatore in una codifica standard. La suddivisione in 4 codificatori e decodificatori è quindi puramente schematica

questa parte viene anche chiamata *pre-codifica*. Dato un frame di dimensioni $n \times m$ pixel, esso viene prima suddiviso in due parti dividendo righe pari e righe dispari, in modo da ottenere due blocchi di dimensioni $n \times \frac{m}{2}$. i due blocchi ottenuti saranno poi suddivisi ulteriormente, separando le colonne pari dalle dispari, ottenendo così 4 blocchi di dimensione $\frac{n}{2} \times \frac{m}{2}$ che rappresentano le quattro descrizioni. Come si può notare in Figura 5.2, questo procedimento equivale a suddividere l'immagine in blocchetti adiacenti 4×4 pixel ed assegnare il pixel in alto a sinistra alla prima descrizione (D1), quello in alto a destra alla seconda descrizione (D2), quello in basso a sinistra alla terza descrizione (D3) e quello in basso a destra alla quarta descrizione (D4).
Le quattro descrizioni sono poi inviate a quattro codificatori H.264/AVC in-

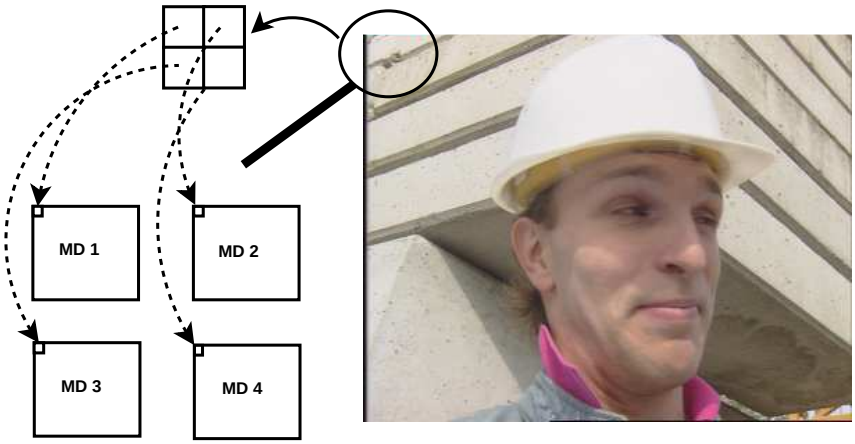


Figura 5.2. Procedimento di generazione delle quattro descrizioni.

dipendenti. Per ogni descrizione, ogni frame viene suddiviso in blocchi 16×16 pixel chiamanti *macroblocchi*.I macroblocchi possono essere raggruppati in *slice* le quali rappresentano il più piccolo sottoinsieme del frame decodificabile indipendentemente dagli altri. Ogni pacchetto RTP dello *stream*, generato dal codificatore H.264/AVC, corrisponde esattamente ad una slice. dopo la codifica, i pacchetti saranno inoltrati al blocco di classificazione che sarà descritto dettagliatamente nelle successive sezioni.

5.2.2 Decodifica

Il sistema di decodifica (Figura 5.3) è composto da quattro decodificatori H.264/AVC in grado di decodificare separatamente le quattro descrizioni. I quattro decodificatori sono però collegati ad un blocco chiamato MD Concealment, con il quale sono costantemente in comunicazione. Quest'ultimo blocco, funziona in supporto ai decodificatori, con il compito di stimare i pacchetti persi in una

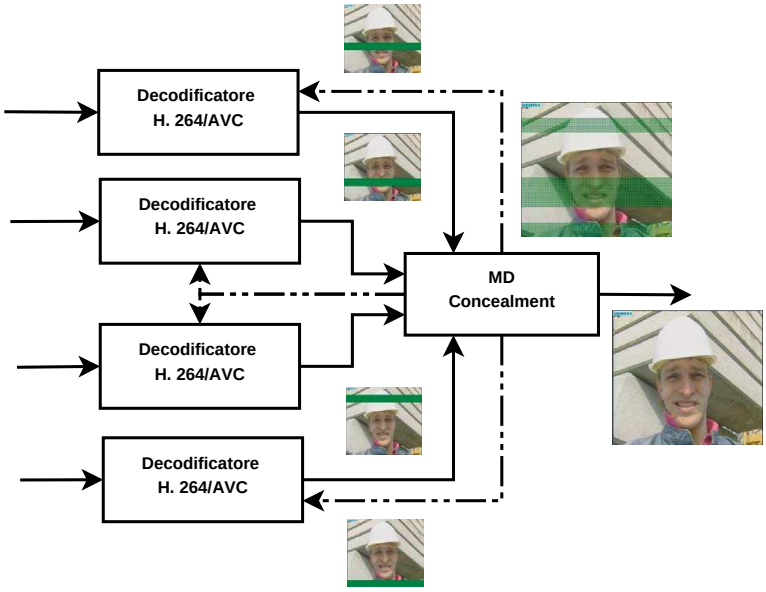


Figura 5.3. Schema di decodifica di una sequenza video MD a quattro descrizioni.

o più descrizioni da quelli ricevuti correttamente nelle altre descrizioni. Nel caso in cui non ci sia alcuna perdita di pacchetti, la qualità del segnale è la stessa presente al codificatore. Nel caso in cui un pacchetto sia perso in una descrizione, il blocco di concealment utilizza un sistema di interpolazione bilineare per stimare i pixel mancanti nell'immagine, da quelli ad essi adiacenti, ricevuti e decodificati correttamente dalle altre descrizioni. Il frame potrà quindi sempre essere ricostruito interamente, anche se ad una qualità eventualmente inferiore a quella presente in fase di codifica.

5.3 Il codificatore H.264/AVC

Il codificatore H.264/AVC è uno standard internazionale per la codifica video. Sviluppato da un comitato di standardizzazione congiunto nell'ambito delle organizzazioni ISO/IEC e ITU-T, esso è anche noto come MPEG-4 Part 10 o MPEG-4 AVC (Advanced Video Coding). Lo standard H.264/AVC descrive il decodificatore, definendo gli elementi sintattici che permettono di ricostruire il segnale video e caratterizzano il bitstream codificato. In questo modo, ogni decodificatore conforme allo standard produce un'uscita analoga a parità di bit-stream fornito dal codificatore (conforme anch'esso allo standard). Lo standard, concede quindi massima libertà sull'implementazione del codificatore lasciando ad esso un elevato grado di libertà sui parametri di complessità, qualità della codifica e costi di implementazione. Lo standard H.264/AVC è stato

progettato per essere usato in molteplici applicazioni in cui la trasmissione dell'informazione codificata può avvenire su reti di tipo ethernet, LAN, wireless o combinazioni di esse. Per ottenere questa flessibilità, lo standard H.264/AVC prevede due stadi:

- VCL (Video Coding Layer): progettato per ottenere la massima efficienza di codifica;

- NAL (Network Adaptation Layer): progettato con lo scopo di adattare il bit-stream codificato dall'unità VLC ad un'ampia varietà di reti eterogenee. In particolare esso permette di interfacciare l'uscita del VLC con i protocolli del livello di trasporto e di rete come RTP ed IP per ogni tipo di servizio internet real-time sia cablato che wireless.

Come si può notare dallo schema di Figura 5.4, il codificatore H.264/AVC è organizzato come una struttura ad anello chiuso. I frame possono essere di tipo Intra o Inter. Mentre per la codifica dei frame Intra non non è utilizzata alcuna correlazione temporale, per i frame Inter, sono necessari dei riferimenti al frame precedente per attuare la compensazione del moto. Attuata una predizione spaziale e/o temporale, la differenza tra il segnale originale e la predizione è trasformata attraverso DCT e opportunamente quantizzata, prima di subire una compressione tramite codifica entropica.

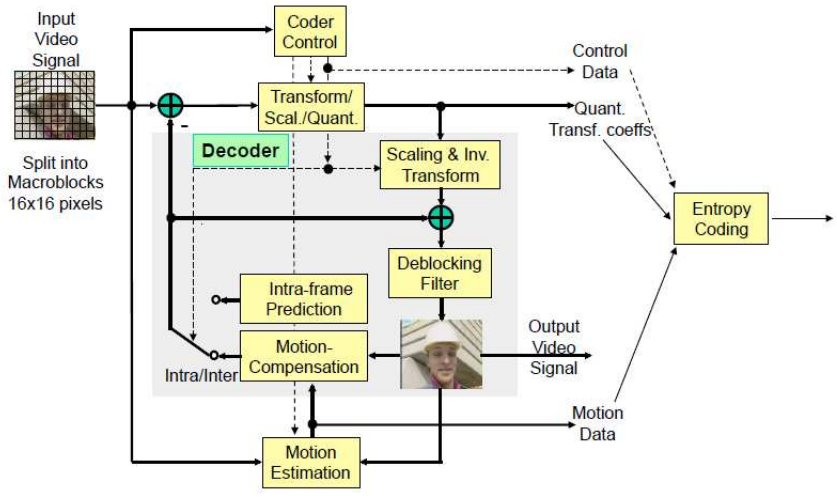


Figura 5.4. Schema di codifica H.264/AVC.

Il formato 4:2:0

Il sistema visivo umano percepisce il contenuto di una scena con una maggiore sensibilità alla componente luminosa rispetto a quelle di colore. H.264/AVC usa uno spazio di colore chiamato $Y\ U\ V$, in cui la componente Y, chiamata luminanza, rappresenta la versione in scala di grigi dell'immagine, mentre U e V sono definite crominanze e indicano la deviazione di colore verso il blu e il rosso. H.264/AVC usa una struttura di campionamento in cui le componenti U e V contengono ciascuna un quarto dei pixel della luminanza. Questo schema di campionamento prende il nome di 4:2:0.

Macroblocchi e Slice

In H.264/AVC, i macroblocchi sono costituiti da 16×16 campioni per immagine, che nello spazio 4:2:0 si traducono in 16×16 pixel di luminanza e 8×8 pixel per ciascuna componente di crominanza U e V. Una riga di macroblocchi forma una slice, che rappresenta un sottoinsieme di immagine decodificabile indipendentemente dagli altri. In particolare si definiscono tre tipi si slice:

- slice I (Intra)
- slice P (Predictive)
- slice B (Bi-Predictive)

Nella codifica Intra si sfrutta la sola correlazione spaziale all'interno della stessa immagine. Per aumentare l'efficienza di codifica, vengono codificate le differenze tra i campioni del macroblocco e i campioni precedentemente codificati (tipicamente quelli posizionati sopra e a sinistra del pixel da codificare). Nella codifica di tipo Inter (P e B), si utilizza una predizione ottenuta sfruttando la correlazione temporale, da uno o due frame precedentemente codificati. La predizione può essere ottenuta mediante una stima ed una compensazione del movimento. La precisione per i vettori di movimento è di $\frac{1}{4}$ di pixel. Per la codifica di frame di tipo B, la correlazione temporale utilizzata è riferita sia ai frame precedenti che successivi e la predizione si basa quindi sulla media pesata di due distinti segnali di predizione ottenuti mediante compensazione del moto

Trasformata e Quantizzazione

Il tipo di trasformata utilizzato è basato su DCT (Discrete Cosine Transform) ed esistono 52 passi di quantizzazione, ognuno corrispondente ad un parametro

di quantizzazione, chiamato Q_p e legato al passo di quantizzazione Δ dalla seguente relazione:

$$\Delta \cong 0.64\ 2^{(Q_p/6)} \tag{5.1}$$

Codifica

Lo standard H.264/AVC specifica diversi tipi di codifica entropica. Una prima modalità denominata *Context Adaptive Variable Lenght Coding* (CAVLC), utilizza dei codici di lunghezza variabile le cui tabelle di codifica sono scelte in maniera adattiva a seconda dei dati da codificare. Una seconda modalità denominata *Context Adaptive Binary Arithmetic Coding* (CABAC) prevede la conversione degli elementi di sintassi in stringhe binarie di lunghezza variabile i cui bit vengono poi codificati da un codificatore binario aritmetico adattivo basato su diversi contesti di codifica.

5.4 Classificazione dei pacchetti mediante teoria dei giochi

Dopo essere stati codificati, i pacchetti delle quattro descrizioni devono essere opportunamente classificati. Il blocco di classificazione, ha il compito di distinguere i pacchetti per livello di importanza in modo da associare ognuno di essi ad una data classe di servizio. Ogni classe definisce un determinato livello di QoS in termini di probabilità di perdita e ritardo massimo consentito per l'arrivo del pacchetto a destinazione. Prima di definire i vari metodi utilizzati per la classificazione e basati su teoria dei giochi è necessario introdurre un parametro che consenta di definire il livello di importanza di ogni pacchetto in base alle sue caratteristiche. In [7] gli autori hanno legato la distorsione prodotta dalla perdita di un pacchetto con la percentuale ρ di coefficienti nulli all'uscita del blocco di trasformazione e quantizzazione in un codificatore H.264/AVC. Questo approccio sarà mantenuto anche in questo lavoro di tesi e sarà illustrato nel dettaglio nelle successive sezioni.

5.4.1 Misura della distorsione (δPSNR)

In questa sezione si definisce il concetto di distorsione introdotta dalla perdita di uno o più pacchetti al fine di definire il livello di importanza di ognuno di essi. Innanzitutto è necessario enunciare il concetto di *Peak Signal to Noise Ratio*(PSNR) per frame. Misurato in scala logaritmica, esso dipende dall'errore in media quadratica (MSE) tra il frame originale e quello ricevuto, moltiplicato

per un fattore costante, definito in base al numero n di bit per campione di immagine.

$$\mathrm{PSNR_{dB}} = 10 \log_{10} \frac{(2^{\mathrm{n}} - 1)^2}{\mathrm{MSE}} \tag{5.2}$$

Nell'ambito dei segnali considerati i campioni delle componenti Y, U, V del segnale originale vengono rappresentati con 8 bit ($n = 8$). Minore è la differenza MSE tra frame decodificato e frame originale, maggiore sarà il valore di PSNR. Più elevato è dunque il valore di PSNR maggiore risulterà la qualità dell'immagine ricostruita. Definito il PSNR è possibile introdurre il concetto di decremento di PSNR, chiamato $\delta\mathrm{PSNR}$ che rappresenta una stima della distorsione dovuta alla perdita di uno o più pacchetti. In particolare definiamo la distorsione introdotta dalla perdita del j-esimo pacchetto dell'i-esima descrizione nel seguente modo:

$$\delta\mathrm{PSNR_{i,j}} = \frac{\mathrm{PSNR_o} - \mathrm{PSNR_{i,j}}}{\mathrm{PSNR_o}} \quad \mathrm{i} = 1, \ldots, \mathrm{N_d} \tag{5.3}$$

dove $\mathrm{PSNR_o}$ e $\mathrm{PSNR_{i,j}}$ rappresentano i valori di PSNR relativi alla ricostruzione del frame contenente il j-esimo pacchetto, rispettivamente nel caso in cui non ci sia stata alcuna perdita di pacchetti e in cui il pacchetto j-esimo della descrizione i-esima sia perso. N_d rappresenta il numero di descrizioni ed in questo lavoro equivale a 4. Definiamo inoltre il decremento di PSNR relativo alla perdita simultanea del pacchetto j-esimo in una coppia di descrizioni (i, h) nel seguente modo:

$$\delta\mathrm{PSNR_{(i,h),j}} = \frac{\mathrm{PSNR_o} - \mathrm{PSNR_{(i,h),j}}}{\mathrm{PSNR_o}} \quad \mathrm{i} = 1, \ldots, \mathrm{N_d}\ ,\ \mathrm{h} = 1, \ldots, \mathrm{N_d} \tag{5.4}$$

dove $\mathrm{PSNR_{(i,h),j}}$ rappresenta il valore di PSNR relativo alla ricostruzione del frame contenente il j-esimo pacchetto in cui, però, tale pacchetto viene perso sia nella descrizione i-esima che in quella h-esima. L'obiettivo del blocco di classificazione è quindi quello di assegnare ad ogni pacchetto di ogni descrizione una classe di servizio in funzione dei valori di $\delta\mathrm{PSNR}$ e della probabilità di perdita associata ad ogni classe di servizio. In particolare, ogni descrizione sceglierà la classe, per il pacchetto j-esimo, in modo da minimizzare la seguente funzione.

$$D_i(\mathbf{c}_j) = \delta\mathrm{PSNR_{i,j}} \mathrm{p_{l,j}}(\mathrm{c_{i,j}}) \quad \mathrm{i} = 1, \ldots, \mathrm{N_d} \tag{5.5}$$

dove $\mathbf{c}_j$ rappresenta il vettore di classi per il pacchetto j-esimo di tutte le descrizioni, $c_{i,j}$ indica l'i-esimo elemento del vettore e $p_{l,j}(c)$ mappa una qualsiasi classe c con la relativa probabilità di perdita ad essa associata, considerando anche le classificazioni già effettuate negli istanti precedenti. Nella creazione dei

vari modelli di classificazione, si suppone di associare ad ogni classe di servizio una coda di dimensione B. Assumendo che il numero attuale di pacchetti all'interno della coda k-esima sia B_k, la probabilità di perdita $p_{l,j}(c_k)$ per il pacchetto j-esimo, classificato con la classe k è definita nel seguente modo:

$$p_{l,j}(c_k) = P^{late}_{j,k}(B_k) + [1 - P^{late}_{j,k}(B_k)]P_{chan,k} \tag{5.6}$$

dove $P^{late}_{j,k}(B_k)$ indica la probabilità, per la classe k che il j-esimo pacchetto arrivi a destinazione oltre il tempo limite consentito, mentre $P_{chan,k}$ indica la probabilità di perdita di pacchetti in funzione del livello attuale di congestione della rete. Più precisamente la probabilità $P^{late}_{j,k}(B_k)$ può essere così definita:

$$P^{late}_{j,k}(B_k) = P\left[\sum_{m=0}^{B_k-1} \Delta T_m > T_L\right] \tag{5.7}$$

dove ΔT_m è il tempo di attesa di un pacchetto in posizione m-esima della coda prima di essere trasmesso e T_L è il tempo limite entro il quale il pacchetto deve giungere a destinazione altrimenti diviene obsoleto e come tale sarà scartato. Nel nostro modello questo tempo è stato impostato a 0.2 secondi.

5.4.2 Classificazione tramite teoria dei giochi non cooperativi

Lavori precedenti hanno mostrato come sia possibile utilizzare la teoria dei giochi per la classificazione dei pacchetti all'interno di una rete [19]. In questa situazione, i nodi che contengono le descrizioni possono essere considerati come dei giocatori che competono tra loro per trasmettere, con successo, i pacchetti appartenenti alla propria descrizione. La strategia che ogni giocatore potrà adottare è rappresentata dalla classe di servizio da attribuire ad ogni pacchetto in loro possesso. La scelta della classificazione migliore deve necessariamente tenere conto delle scelte degli altri giocatori. Se infatti tutti i giocatori tentassero di trasmettere tutti i pacchetti con la classe a maggior priorità si perderebbero tutti i benefici dati dall'utilizzo di un modello DiffServ (Cap 4). Nella prima tipologia di gioco implementata, si suppone che i nodi contenenti le descrizioni non possano comunicare tra loro e scambiarsi informazioni: in questo modo, nessuno dei giocatori sarà a conoscenza delle strategie adottate dagli altri. Questa situazione potrà quindi essere rappresentata tramite un gioco non cooperativo a quattro giocatori, uno per ogni descrizione. Lo scopo del gioco, in un dato istante, è quindi quello di classificare i pacchetti j-esimi delle quattro descrizioni. Dopo aver definito i giocatori e le strategie da adottare, è necessario descrivere quali sono le funzioni costo per ogni pacchetto di ogni

descrizione, in un dato istante. L'obiettivo di ogni giocatore è quello di minimizzare la propria funzione costo in accordo con le possibili scelte fatte dagli altri giocatori. La funzione costo per il pacchetto j-esimo della descrizione i-esima può essere quindi definita ricorsivamente nel seguente modo:

$$f_i(\mathbf{c}_j) = \sum_{t=0}^{j-1} f_{i,t}(\mathbf{c}_t) + D_i(\mathbf{c}_j) \tag{5.8}$$

dove si fa riferimento alla funzione $D_i(\mathbf{c}_j)$ definita in 5.5 . Questa funzione suppone però di avere a disposizione tutti i valori di $\delta\mathrm{PSNR_{i,j}}$. Lavori precedenti [20] [21] hanno mostrato come esista una stretta relazione tra $\delta\mathrm{PSNR}$ e la percentuale di coefficienti nulli all'uscita dei blocchi di trasformazione e quantizzazione presenti in un codificatore H.264/AVC. All'ingresso del blocco di DCT, si trova l'errore di predizione definito come la differenza tra il segnale da codificare e la stima ottenuta utilizzando correlazioni spaziali e temporali. Se la percentuale di coefficienti nulli all'uscita del quantizzatore è elevata significa che il pacchetto in esame presenta un livello di dettaglio ridotto rispetto alla sua stima. Viceversa, se la complessità del pacchetto è elevata, l'errore di predizione aumenta di conseguenza e la percentuale di coefficienti nulli diminuisce. Questa percentuale è anche definita come percentuale di zeri ed è indicata con ρ. Analizzando varie sequenze è stato possibile definire, sperimentalmente, la relazione tra il numero di zeri nel pacchetto j-esimo della descrizione i-esima, indicato con $\rho_{i,j}$ ed il valore di $\delta\mathrm{PSNR_{i,j}}$. Per la codifica di ogni sequenza, inoltre, sono stati considerati i valori di $QP \in \{20, 22, 24, 26, 28, 30, 32\}$. Come

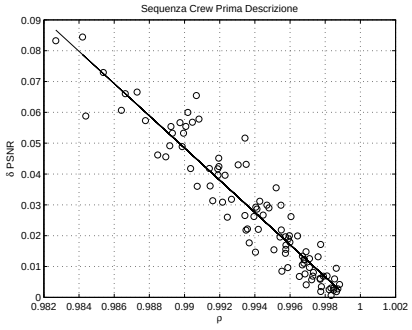


Figura 5.5. Caratteristica ρ e $\delta\mathrm{PSNR}$ relativa alla prima descrizione della sequenza Crew.

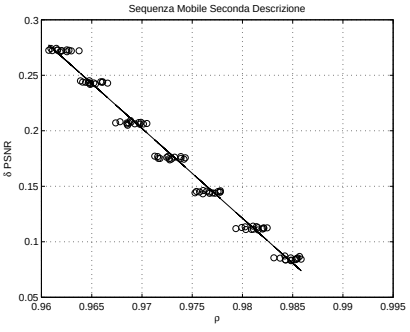


Figura 5.6. Caratteristica ρ e $\delta\mathrm{PSNR}$ relativa alla seconda descrizione della sequenza Mobile.

si può osservare dalle Figure 5.5 e 5.6, la relazione è di tipo lineare e $\delta\mathrm{PSNR_{i,j}}$ può essere così definito:

$$\delta\mathrm{PSNR_{i,j}} = \mathrm{k_{i,0}} + \mathrm{k_{i,1}}\rho_{\mathrm{i,j}} \tag{5.9}$$

Considerando che i valori di $\rho_{i,j}$ sono facilmente ottenibili, sia in fase di codifica sia a partire dalla sequenza già codificata, basterà avere a disposizione i coefficienti $k_{i,0}$ e $k_{i,1}$ che dipendono strettamente dalle caratteristiche della sequenza video. Analizzando un numero di sequenze campione con caratteristiche differenti, è quindi possibile ottenere un'ampia gamma di coefficienti per ogni tipologia di sequenza video da classificare. In particolare è possibile riutilizzare i coefficienti calcolati su sequenze diverse nella condizione in cui le sequenze codificate presentino le stesse caratteristiche modellate attraverso *activity* o altre metriche [22]. La funzione costo è ora definita per ogni pacchetto e, la razionalità individuale dei giocatori porta a raggiungere uno o più punti di equilibrio di Nash come soluzione del gioco. Come definito nel terzo capitolo, una configurazione di strategie è chiamata Equilibrio di Nash se:

$$f_d(c^*_{1,j}, \ldots, c^*_{d,j}, \ldots, c^*_{N,j}) \leq f_d(c^*_{1,j}, \ldots, c_{d,j}, \ldots, c^*_{N,j}) \qquad (5.10)$$
$$\forall c_{d,j} \neq c^*_{d,j}, \ \ \forall d$$

essa rappresenta una situazione in cui nessun giocatore ha interesse ad essere l'unico a cambiare la propria strategia. Il primo gioco, implementato in linguaggio di programmazione C, prevede, per ogni istante di tempo j, la valutazione della funzione $f_d(\mathbf{c}_j)$ per tutte le possibili configurazioni $\mathbf{c}_j$ ed identifica le configurazioni che sono di equilibrio. Finora j è sempre stato trattato come l'indice di pacchetto; a livello implementativo è inoltre stato creato un secondo gioco non cooperativo in cui j è l'indice di frame. In questo caso la classificazione dei pacchetti è definita frame per frame ed il valore di $\rho_{i,j}$ è ottenuto mediando i valori relativi a tutti i pacchetti del frame, al fine di calcolare il valore di $\delta\mathrm{PSNR_{i,j}}$. In questo caso tutti i pacchetti appartenenti al frame j-esimo della descrizione i-esima, avranno la stessa classificazione. Infine, se sono presenti più configurazioni di equilibrio $\mathbf{c}_j \in N_e$, una procedura di ottimizzazione sceglierà l'equilibrio che minimizza la distorsione complessiva, nel modo seguente:

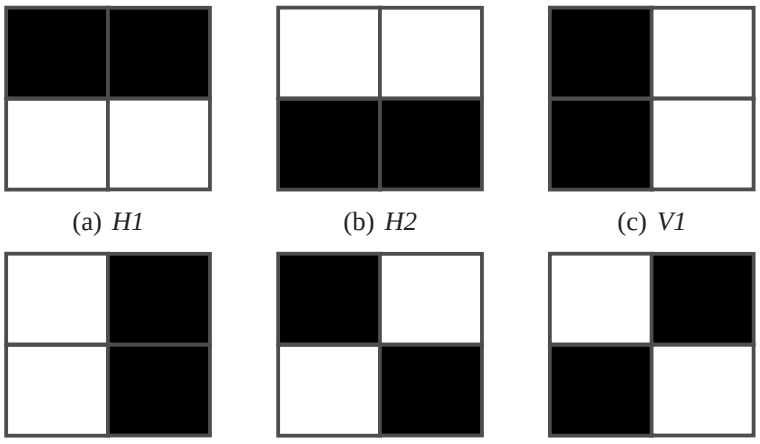
$$\mathbf{c}_j = arg \min_{\mathbf{c}_j \in N_e} \sum_{d=1}^{N} f_d(\mathbf{c}_j) \qquad (5.11)$$

5.4.3 Classificazione tramite teoria dei giochi cooperativi

Nel terzo capitolo, è stato possibile osservare come la comunicazione e la collaborazione di alcuni giocatori, possa permettere loro di migliorare il proprio guadagno. Un esempio analizzato è quello del Dilemma del prigioniero, in cui, i due giocatori, accordandosi, avrebbero potuto ottenere un *payoff* maggiore rispetto a quello ottenuto non conoscendo la strategia dell'altro giocatore. Tornando al problema di classificazione dei pacchetti appartenenti alle quattro

descrizioni di una sequenza video, ci si è chiesti se la possibilità da parte di due o più nodi della rete di comunicare e stabilire accordi tra loro, potesse portare ad un incremento delle prestazioni ottenute attraverso un gioco non cooperativo. In prima analisi, sono stati valutati i valori di PSNR e $\delta\mathrm{PSNR}$ relativi, non più alla perdita di pacchetti in un unica descrizione, ma in due descrizioni per tutte le possibili configurazioni. Sono state quindi definite 6 diverse configurazioni, chiamate *coalizioni*, ciascuna composta da una coppia di descrizioni. Ricordando il meccanismo di creazione delle 4 descrizioni, a partire da blocchi 4×4 pixel, definiamo le coalizioni come:

- Prima coalizione Orizzontale *H1* (Figura 5.7 (a))
- Seconda coalizione Orizzontale *H2* (Figura 5.7 (b))
- Prima coalizione Verticale *V1* (Figura 5.7 (c))
- Seconda coalizione Verticale *V2* (Figura 5.7 (d))
- Prima coalizione Obliqua *O1* (Figura 5.7 (e))
- Seconda coalizione Obliqua *O2*(Figura 5.7 (f))



(a) *H1* (b) *H2* (c) *V1*

(d) *V2* (e) *O1* (f) *O2*

Figura 5.7. Le 6 coalizioni.

Dopo aver definito le coalizioni, sono state simulate diverse situazioni di perdita di pacchetti sulle due descrizioni appartenenti ad ogni coalizione, per diverse sequenze video. In prima analisi, per ogni coalizione, è stata simulata la perdita di tutti i pacchetti di ciascuna delle due descrizioni coalizzate, mentre nelle restanti due descrizioni, non coalizzate, è stato supposto che nessun pacchetto

andasse perso. Successivamente, si è pensato di analizzare la situazione in cui solo una data percentuale di pacchetti appartenenti alle descrizioni coalizzate andasse persa. Attraverso il calcolo di PSNR e $\delta\mathrm{PSNR}$ si è potuto osservare che la perdita simultanea di pacchetti, in diverse configurazioni di descrizioni influisce diversamente sulle prestazioni del sistema. Per alcune sequenze risulta più "conveniente" perdere pacchetti sulle descrizioni appartenenti a una delle due coalizioni orizzontali $H1$ e $H2$. Per altre sequenze, invece, è preferibile perdere pacchetti in una delle due coalizioni verticali $V1$ e $V2$. Per altre ancora, infine, la perdita di pacchetti nelle descrizioni appartenenti ad una delle due coalizioni oblique $O1$ e $O2$ porta ad una distorsione minore. Nelle Figure 5.8, 5.9, 5.10, 5.11 sono illustrate le caratteristiche $QP - \delta\mathrm{PSNR}$ per diverse sequenze relative alla perdita di 15% dei pacchetti delle descrizioni coalizzate, per ogni configurazione possibile. Si è quindi deciso di investigare sulle pos-

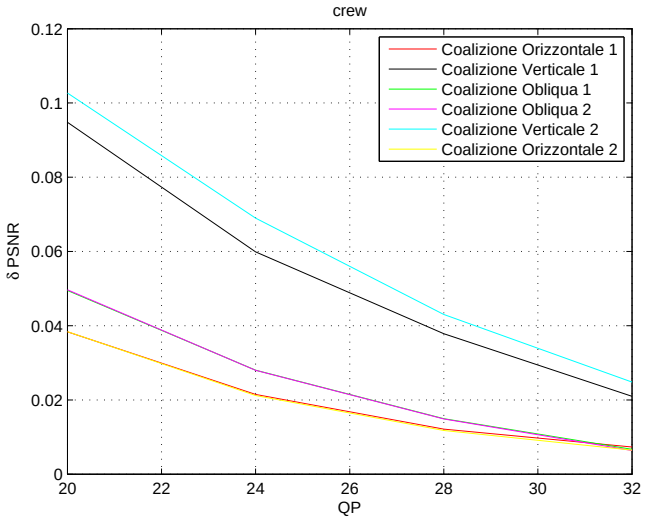


Figura 5.8. Relazione $QP - \delta\mathrm{PSNR}$ relativa alla perdita di 15% dei pacchetti sulle descrizioni di ogni coalizione per la sequenza Crew.

sibili motivazioni di questa differenza di prestazioni per diverse configurazioni di descrizioni, in una data sequenza. L'analisi, ha fatto notare come queste differenze fossero legate prevalentemente a caratteristiche differenti di correlazione spaziale nelle diverse sequenze. Nelle sequenze in cui, la correlazione verticale risulta maggiore di quella orizzontale è preferibile perdere pacchetti in una delle due coalizioni orizzontali (Figura 5.8). Viceversa, se la correlazione orizzontale è maggiore della verticale, le prestazioni migliorano perdendo pacchetti in una delle due coalizioni verticali (5.11). Nel caso infine in cui, correlazione verticale e orizzontale sono molto vicine, la perdita di pacchetti in

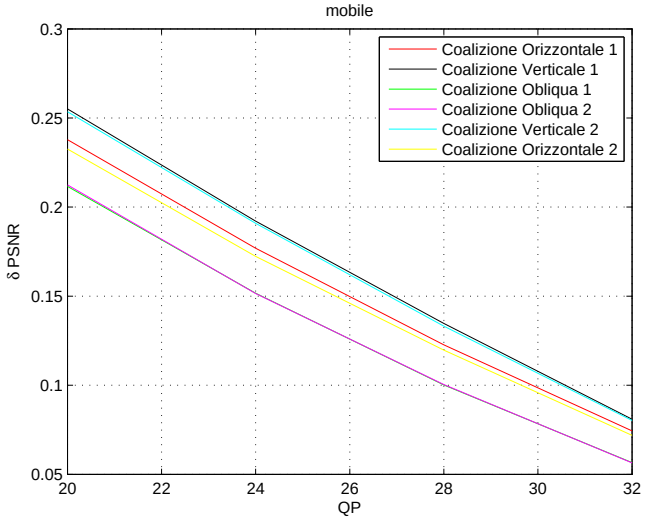


Figura 5.9. Relazione $QP - \delta\mathrm{PSNR}$ relativa alla perdita di 15% dei pacchetti sulle descrizioni di ogni coalizione per la sequenza Mobile.

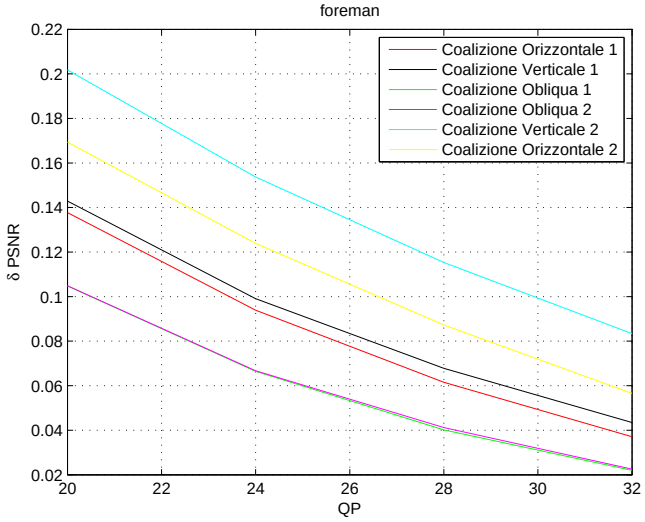


Figura 5.10. Relazione $QP - \delta\mathrm{PSNR}$ relativa alla perdita di 15% dei pacchetti sulle descrizioni di ogni coalizione per la sequenza Foreman.

una delle due coalizioni oblique comporta una distorsione minore (Figure 5.9 e 5.10). Le informazioni di correlazione possono essere ottenute sia in fase di analisi della sequenza originale, prima della creazione delle 4 descrizioni, sia attraverso dati ottenuti in fase di codifica relativi ai meccanismi di predizione

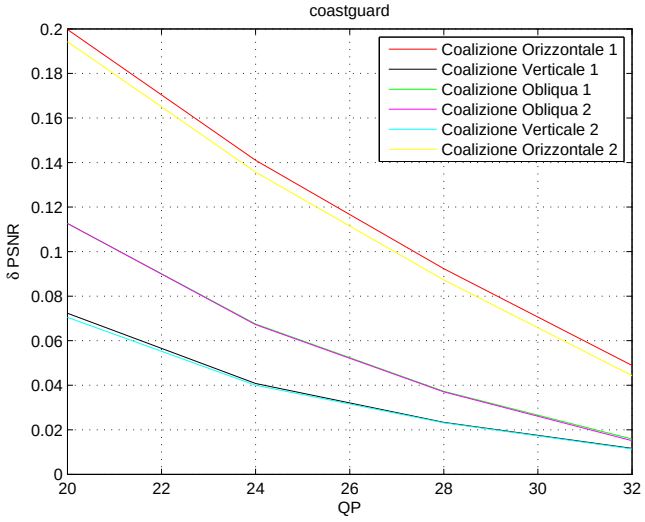


Figura 5.11. Relazione $QP - \delta\text{PSNR}$ relativa alla perdita di 15% dei pacchetti sulle descrizioni di ogni coalizione per la sequenza Coastguard.

spaziale utilizzati dal codificatore H.264/AVC. In Tabella 5.1 sono riportati i gradienti relativi alle sequenze video analizzate (Crew, Foreman Coastguard, Mobile). Questi gradienti rappresentano il valore medio, calcolato sull'intera sequenza, della differenza tra i valori di luminanza (8 bit) di pixel corrispondenti appartenenti a righe adiacenti (gradiente verticale) o colonne adiacenti (gradiente orizzontale). Una sequenza avrà quindi una maggior correlazione spaziale nella direzione dove il gradiente risulta minore.

Sequenza	Crew	Foreman	Coastguard	Mobile
Gradiente Verticale	2.9696	5.6109	12.5856	17.4481
Gradiente Orizzontale	4.9964	5.0803	6.2240	17.8329

Tabella 5.1. Gradienti relativi alle sequenze video Crew, Foreman, Coastguard e Mobile.

Relazione tra ρ e δPSNR

La differenza di prestazioni, provocata dalla perdita di pacchetti, applicata a diverse coalizioni ha dato l'idea per la realizzazione di un gioco cooperativo. In questo modello, 2 delle 4 descrizioni, rappresentanti i giocatori, potranno coalizzarsi, scambiarsi informazioni ed accordarsi sulle strategie da adottare per la classificazione dei pacchetti ad esse appartenenti. Analogamente all'analisi fatta per il gioco non cooperativo, in questo caso, è necessario misurare il

grado di distorsione $\delta\mathrm{PSNR}_{(\mathrm{i,h}),\mathrm{j}}$ dovuto alla perdita del pacchetto j-esimo per ogni coppia non ordinata di descrizioni (i, h). Anche in questo caso, dai dati sperimentali, è stati possibile notare, come esista una stretta relazione tra il valore di distorsione e la percentuale di coefficienti nulli ρ all'uscita dei blocchi di trasformazione e quantizzazione all'interno del codificatore H.264/AVC. La percentuale di zeri dev'essere però rappresentativa di entrambi i pacchetti delle descrizioni coalizzate. Per questo motivo è stati definito il parametro $\rho_{(i,h),j}$ nel seguente modo:

$$\rho_{(i,h),j} = \frac{\rho_{i,j} + \rho_{h,j}}{2} \tag{5.12}$$

esso rappresenta la media della percentuale di zeri per il pacchetto j esimo di ogni coppia di descrizioni (i, h) coalizzate. Nelle Figure 5.12, 5.13, 5.14, 5.15 si può osservare come la relazione tra $\delta\mathrm{PSNR}_{(\mathrm{i,h}),\mathrm{j}}$ e $\rho_{(i,h),j}$ sia ancora una volta lineare e potrà quindi essere definita nel modo seguente:

$$\delta\mathrm{PSNR}_{(\mathrm{i,h}),\mathrm{j}} = \mathrm{k}_{(\mathrm{i,h}),0} + \mathrm{k}_{(\mathrm{i,h}),1}\rho_{(\mathrm{i,h}),\mathrm{j}} \tag{5.13}$$

anche in questo caso, basterà, dunque, avere a disposizione i coefficienti $k_{(i,h),0}$ e $k_{(i,h),1}$ che dipendono dalle caratteristiche della sequenza video e dalla particolare coalizione esaminata. Per il calcolo dei coefficienti, sono inoltre stati considerati, per ogni coalizione, i valori di $QP \in \{20, 22, 24, 26, 28, 30, 32\}$.

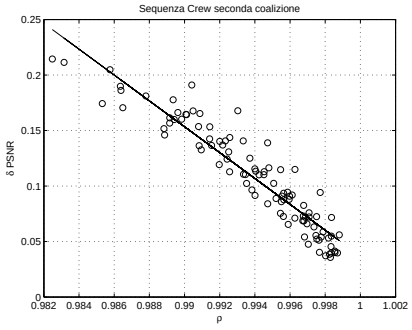


Figura 5.12. Caratteristica ρ e $\delta\mathrm{PSNR}$ relativa alla coalizione $V1$ della sequenza Crew.

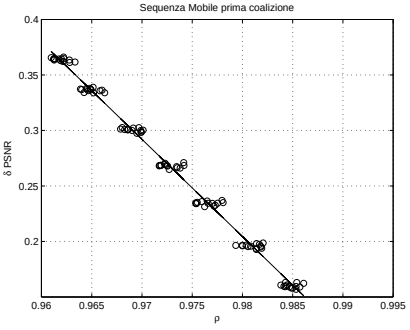


Figura 5.13. Caratteristica ρ e $\delta\mathrm{PSNR}$ relativa alla alla coalizione $H1$ della sequenza Mobile.

5.4.4 Implementazione dei giochi cooperativi per la classificazione

Utilizzando i parametri definiti nella precedente sezione, sono stati costruiti tre diversi modelli di giochi cooperativi. Per ognuno di essi sono state poi valutate

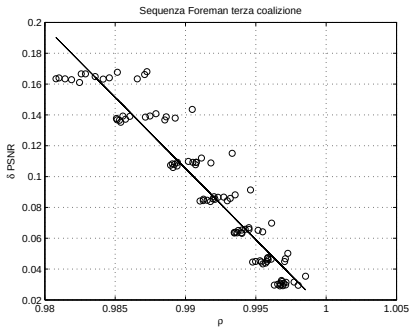


Figura 5.14. Caratteristica ρ e $\delta\mathrm{PSNR}$ relativa alla coalizione $O1$ della sequenza Foreman.

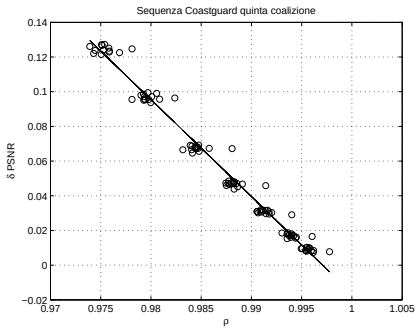


Figura 5.15. Caratteristica ρ e $\delta\mathrm{PSNR}$ relativa alla alla coalizione $V2$ della sequenza Coastguard.

le prestazioni attraverso diverse configurazioni di rete che saranno definite nel prossimo capitolo. Infine, per ogni gioco con coalizioni, i risultati sono stati confrontati con quelli ottenuti attraverso il corrispondente modello di gioco non cooperativo. In particolare sono stati implementati i seguenti modelli di gioco cooperativo:

1. Primo gioco: gioco a 4 giocatori con coalizione e classificazione frame per frame (in questo caso la strategia dei due giocatori coalizzati è la stessa).

2. Secondo gioco: gioco a 4 giocatori con coalizione e classificazione pacchetto per pacchetto.

3. Terzo gioco: gioco a quattro giocatori con coalizione e classificazione frame per frame (in questo caso la strategia dei giocatori coalizzati può variare)

Per la classificazione, si utilizzano le classi definite dal meccanismo DiffServ ed in particolare dal protocollo srTCM, entrambi definiti nel Capitolo 4. Si hanno quindi tre diversi livelli di priorità definiti da tre diversi colori di classificazione: *Verde* (V), che identifica la classe a maggior priorità, *Giallo* (G) che rappresenta la classe a priorità intermedia e *Rosso* (R), il quale indica la classe con il più basso livello di priorità.

Primo gioco

Il primo gioco implementato, consiste nel creare un gioco cooperativo utilizzando un gioco non cooperativo a tre giocatori. In particolare, date le 4 descrizioni della codifica MDC, corrispondenti ai 4 giocatori, due di esse saranno

unite a formare una coalizione, mentre le restanti due agiranno per conto proprio. Questo significa che i giocatori della coalizione possono accordarsi sulla strategia da adottare, mentre gli altri due giocatori sono all'oscuro delle loro strategie reciproche e della strategia adottata dai giocatori della coalizione. In questo caso, le descrizioni della coalizione si accordano per scegliere la stessa classificazione per i propri pacchetti in un dato istante di tempo. La coalizione può quindi essere interpretata come un unico giocatore che compete con gli altri due in un gioco non cooperativo. Inoltre, la classificazione avviene *frame per frame*, il che significa che tutti i pacchetti di una descrizione appartenenti allo stesso frame sono assegnati alla stessa classe di QoS. Sono stati quindi costruite 6 istanze diverse di questo gioco, uno per ogni coalizione possibile, al fine di analizzare quali fossero le migliori coalizioni per ogni sequenza video analizzata. Le funzioni costo da minimizzare in un dato istante j per i quattro giocatori sono state definite come espresso di seguito.

- Per la coalizione C, composta dai giocatori i ed h, data la funzione:

$$D_{(i,h)}(\mathbf{c}_j) = \delta\mathrm{PSNR}_{\mathrm{(i,h),j}}\ \mathrm{p_{l,j}}(\mathrm{c_{(i,h),j}}) \quad \mathrm{(i,h)} \in \mathrm{C} \tag{5.14}$$

  la funzione costo risulta così definita:

$$f_{i,h}(\mathbf{c}_j) = \sum_{t=0}^{j-1} f_{i,h}(\mathbf{c}_t) + D_{(i,h)}(\mathbf{c}_j). \tag{5.15}$$

- Per ognuno dei due giocatori $k \notin C$ invece, data la funzione:

$$D_k(\mathbf{c}_j) = \delta\mathrm{PSNR}_{\mathrm{k,j}}\ \mathrm{p_{l,j}}(\mathrm{c_{k,j}}) \quad \mathrm{k} \notin \mathrm{C} \tag{5.16}$$

  la funzione costo risulta:

$$f_k(\mathbf{c}_j) = \sum_{t=0}^{j-1} f_k(\mathbf{c}_t) + D_k(\mathbf{c}_j). \tag{5.17}$$

Come è possibile osservare, la funzione costo tiene conto anche della classificazione avvenuta negli istanti precedenti a quello di classificazione. Inoltre, tale istante, espresso per mezzo dell'indice j, rappresenta l'indice di frame in quanto la classificazione avviene frame per frame. Definite le funzioni costo per i tre giocatori, si passa al calcolo del punto di equilibrio di Nash. Questo calcolo, prevede, per ogni frame j, la valutazione delle funzioni $f_d(\mathbf{c}_j)$, $d = (i,h), k \notin C$ per tutte le possibili configurazioni $\mathbf{c}_j$ ed identifica le configurazioni che sono di equilibrio. Il valore $\delta\mathrm{PSNR}$ in un dato istante j è ottenuto

mediando i valori di distorsione per ogni pacchetto all'interno del frame. Infine, se sono presenti più configurazioni di equilibrio $\mathbf{c}_j \in N_e$, una procedura di ottimizzazione sceglierà l'equilibrio che minimizza la distorsione complessiva, nel modo seguente:

$$\mathbf{c}_j = arg \min_{\mathbf{c}_j \in N_e} \sum_{d=1}^{N-1} f_d(\mathbf{c}_j) \tag{5.18}$$

Secondo gioco

Il secondo gioco presenta la stessa struttura e le stesse funzioni costo del primo ma tra i due modelli c'è un'importante differenza di classificazione. In questo caso, la classificazione è effettuata *pacchetto per pacchetto*. Questo significa che l'indice j non identifica più il frame, bensì il pacchetto j-esimo di una data descrizione. Il valore di distorsione che precedentemente rappresentava il valore medio calcolato su tutti i pacchetti di un frame è ora identificativo della distorsione portata dalla perdita di un solo pacchetto per descrizione. La conseguenza più importante, però, è che i pacchetti di uno stesso stesso frame in una data descrizione possono ora appartenere diverse classi di servizio. Il meccanismo di classificazione rimane invece lo stesso definito per il primo modello di gioco e quindi, ai pacchetti delle due descrizioni coalizzate, è assegnata, in un dato istante, la stessa classe di QoS.

Terzo gioco

Questo modello di gioco è più complesso dei precedenti. Esso è formato da due diverse fasi di classificazione. La prima fase utilizza la teoria dei giochi per individuare il vettore delle strategie di equilibrio. Nella seconda fase, si applica un algoritmo che mira a migliorare la classificazione ottenuta precedentemente, attraverso una successiva analisi dei pacchetti delle descrizioni coalizzate. In questo caso, si crea un gioco cooperativo a partire da un gioco non cooperativo a quattro giocatori. Le descrizioni coalizzate non si accordano sulla strategia da adottare ma sono legate dalle funzioni costo ad esse relative. Nella definizione delle due funzioni, si utilizzano, parametri differenti rispetto a quelli utilizzati per le funzioni costo delle due descrizioni non coalizzate. In particolare, la funzione costo di ciascuna delle due descrizioni coalizzate, considera, come valore di distorsione dovuto alla perdita di un pacchetto, quello dovuto alla perdita anche del pacchetto corrispondente nella descrizione coalizzata. Le 4 funzioni costo risultano, quindi, così definite:

- Per il primo giocatore i appartenente alla coalizione C, data la funzione:

$$D_{(i)}(\mathbf{c}_j) = \delta \mathrm{PSNR}_{(\mathrm{i,h}),\mathrm{j}}\ \mathrm{p}_{\mathrm{l,j}}(\mathrm{c}_{\mathrm{i,j}}) \quad \mathrm{i} \in \mathrm{C} \tag{5.19}$$

la funzione costo risulta:

$$f_i(\mathbf{c}_j) = \sum_{t=0}^{j-1} f_i(\mathbf{c}_t) + D_{(i)}(\mathbf{c}_j). \tag{5.20}$$

- Per il secondo giocatore h appartenente alla coalizione C, data la funzione:

$$D_{(h)}(\mathbf{c}_j) = \delta\mathrm{PSNR}_{\mathrm{(i,h),j}}\ \mathrm{p_{l,j}(c_{h,j})} \quad \mathrm{h} \in \mathrm{C} \tag{5.21}$$

la funzione costo risulta:

$$f_h(\mathbf{c}_j) = \sum_{t=0}^{j-1} f_h(\mathbf{c}_t) + D_{(h)}(\mathbf{c}_j). \tag{5.22}$$

- Per ognuno dei due giocatori $k \notin C$ invece, data la funzione:

$$D_k(\mathbf{c}_j) = \delta\mathrm{PSNR}_{\mathrm{k,j}}\ \mathrm{p_{l,j}(c_{k,j})} \quad \mathrm{k} \notin \mathrm{C} \tag{5.23}$$

la funzione costo risulta:

$$f_k(\mathbf{c}_j) = \sum_{t=0}^{j-1} f_k(\mathbf{c}_t) + D_k(\mathbf{c}_j). \tag{5.24}$$

Analogamente al primo gioco, la classificazione avviene frame per frame e l'istante di classificazione j rappresenta l'indice di frame. I pacchetti di una descrizione, appartenenti allo stesso frame, avranno, quindi, la stessa classificazione. A differenza del primo gioco, però, i giocatori della coalizione non sono obbligati a scegliere la stessa strategia: i pacchetti j-esimi delle due descrizioni possono appartenere alle configurazioni di classi (R, R), (G G), (V, V), ma anche (G, R) e (V, G). Come si può notare, le funzioni costo relative alle due descrizioni coalizzate, sono legate dal parametro $\delta\mathrm{PSNR}_{\mathrm{(i,h),j}}$ che rappresenta la distorsione media dovuta alla perdita dei pacchetti del frame j-esimo delle due descrizioni. Definite le funzioni costo, la soluzione del gioco si ottiene esaminando tutte le configurazioni possibili $\mathbf{c}_j$ al fine i trovare le strategie di equilibrio, tali che:

$$f_d(c^*_{1,j}, \ldots, c^*_{d,j}, \ldots, c^*_{N,j}) \leq f_d(c^*_{1,j}, \ldots, c_{d,j}, \ldots, c^*_{N,j}) \tag{5.25}$$

Inoltre, anche in questo caso, se sono presenti più configurazioni di equilibrio $\mathbf{c}_j \in N_e$, una procedura di ottimizzazione sceglierà l'equilibrio che minimizza la distorsione complessiva, nel modo seguente:

$$\mathbf{c}_j = arg \min_{\mathbf{c}_j \in N_e} \sum_{d=1}^{N} f_d(\mathbf{c}_j) \tag{5.26}$$

La seconda fase della classificazione, si propone di migliorare quella ottenuta utilizzando il concetto di equilibrio di Nash, attraverso un'ulteriore analisi dei pacchetti appartenenti alle descrizioni coalizzate. Dati i pacchetti (9 per descrizione) appartenenti a frame j-esimo delle due descrizioni $(i, h) \in C$, e chiamate Cl_i e Cl_h le classi di appartenenza delle due sequenze di pacchetti, l'algoritmo è definito come segue:

1. Se i pacchetti delle due descrizioni presentano la stessa classe di servizio, $Cl_i = Cl_h$, non avviene alcuna modifica, altrimenti,

2. si considerano separatamente i pacchetti delle due descrizioni e si ordinano per ordine decrescente di ρ.

3. se $Cl_i > Cl_h$:

   - i 5 pacchetti con i valori maggiori di ρ appartenenti alla descrizione i ed i 4 pacchetti con i valori maggiori di ρ appartenenti alla descrizione h, saranno classificati con la classe Cl_i,
   - i 4 pacchetti con i valori minori di ρ appartenenti alla descrizione i ed i 5 pacchetti con i valori minori di ρ appartenenti alla descrizione h, saranno classificati con la classe Cl_h,

4. se $Cl_i < Cl_h$:

   - i 5 pacchetti con i valori maggiori di ρ appartenenti alla descrizione h ed i 4 pacchetti con i valori maggiori di ρ appartenenti alla descrizione i, saranno classificati con la classe Cl_h,
   - i 4 pacchetti con i valori minori di ρ appartenenti alla descrizione h ed i 5 pacchetti con i valori minori di ρ appartenenti alla descrizione i, saranno classificati con la classe Cl_i,

5. dopo la modifica di classificazione, i pacchetti sono riposizionati nel corretto ordine e si passa all'analisi del frame $j + 1$ tornando al punto 1.

In Figura 5.16 è illustrato un esempio di applicazione dell'algoritmo per i 9 pacchetti del frame j-esimo di due descrizioni $(i, h) \in C$, in cui, la classificazione mediante teoria dei giochi aveva stabilito $Cl_i = V$ e $Cl_h = G$. Ogni rettangolo rappresenta un pacchetto e il valore al sui interno la percentuale di zeri ρ corrispondente.

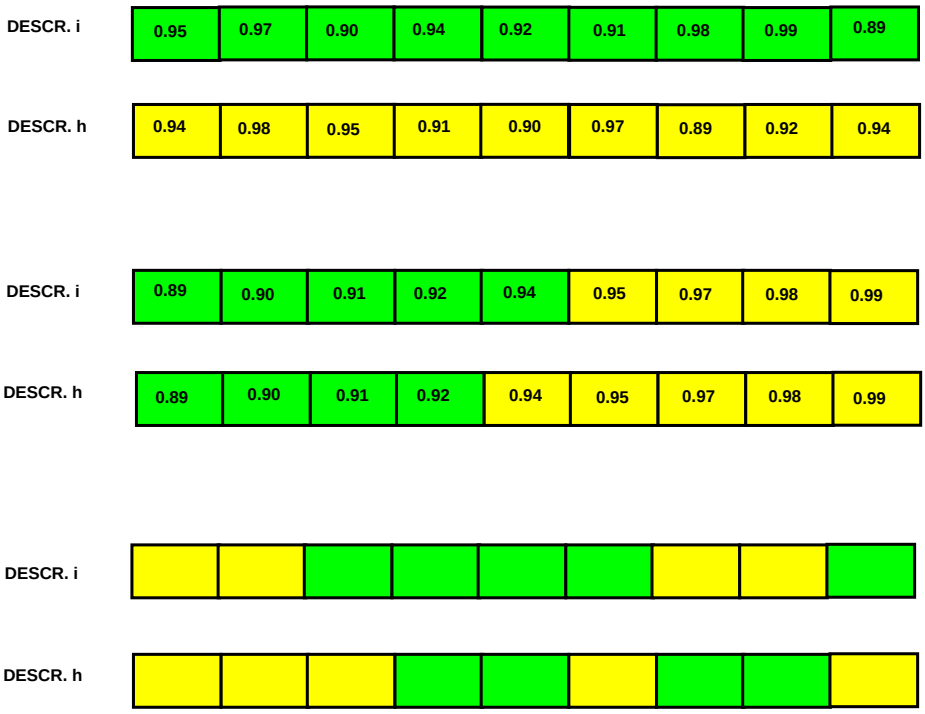


Figura 5.16. Algoritmo di analisi e riclassificazione dei pacchetti, successivo alla classificazione tramite equilibrio di Nash per il terzo modello di gioco.

5.5 Analisi delle prestazioni

Terminata la classificazione, tutti i modelli di gioco analizzati, salvano il risultato in 4 file di testo, corrispondenti alle 4 descrizioni. In corrispondenza di ogni pacchetto, sarà assegnato un valore rappresentativo della classe di servizio corrispondente. In particolare, il valore 0 rappresenta la classe $Verde$, il valore 1 la classe di colore $Giallo$, mentre il valore 2 indica che il pacchetto appartiene alla classe di colore $Rosso$, ovvero quella a minor priorità. Questi file di testo saranno poi utilizzati da diversi ambienti di simulazione per il calcolo delle prestazioni. I modelli di simulazione implementati saranno illustrati dettagliatamente nel prossimo capitolo e, per ognuno di essi, saranno presentati i risultati sperimentali corrispondenti.

Capitolo 6

Modelli di simulazione, risultati sperimentali e conclusioni

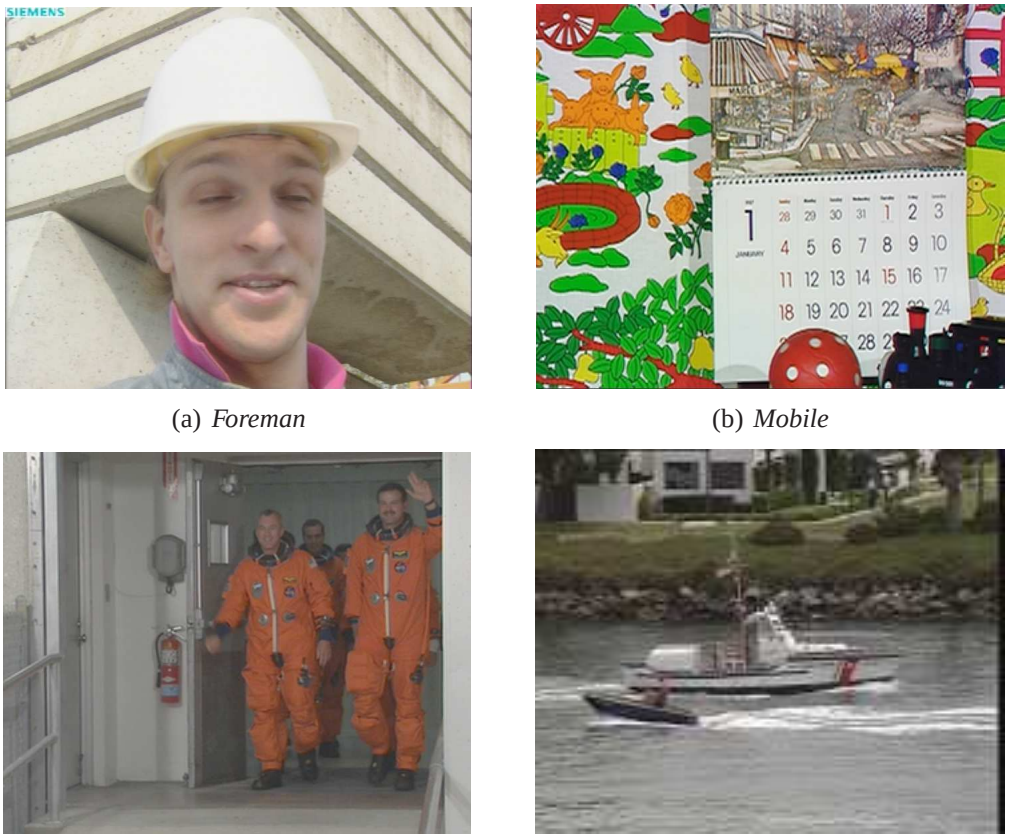
6.1 Introduzione

In questo capitolo, si definiscono i modelli di simulazione sviluppati, al fine di calcolare le prestazioni relative ai diversi modelli di classificazione illustrati nel capitolo 5. Per le prove, sono state utilizzate 4 diverse sequenze video CIF (352×288 pixel): *crew, foreman, mobile e coastguard*, illustrate in Figura 6.1. Ognuna di esse è poi stata suddivisa in 4 descrizioni di dimensioni (176×144 pixel). Ogni descrizione è composta da 9 macroblocchi in altezza e 11 in larghezza. Infine, ogni frame è codificato attraverso 36 pacchetti: 9 per ogni descrizione, corrispondenti ad una riga di macroblocchi (*slice*).

6.2 Modelli di simulazione

Nel capitolo 5 sono stati illustrati diversi meccanismi di classificazione, basati su teoria dei giochi. Ognuno di essi è stato realizzato attraverso un programma scritto in linguaggio di programmazione C che restituisce, come risultato finale un insieme di 4 sequenze di numeri, che saranno salvati in diversi file di testo. Questi numeri identificano la classe di QoS per ogni pacchetto appartenente ad una data descrizione e dovranno quindi essere utilizzati dai diversi meccanismi di simulazione. Gli ambienti di simulazione utilizzati per l'analisi delle prestazioni sono stati nell'ordine:

1. MATLAB: esso è un ambiente di programmazione ad alto livello, con il quale è stato sviluppato un primo, semplice, meccanismo di accesso



(a) *Foreman* (b) *Mobile*

(c) *Crew* (d) *Coastguard*

Figura 6.1. Sequenze utilizzate per le prove.

ad un mezzo trasmissivo comune a tutti i nodi della rete contenenti le 4 descrizioni.

2. Network Simulator 2 (NS2): esso è un ambiente di simulazione di reti molto più complesso, con il quale è stato possibile sviluppare diversi modelli di rete. Questi modelli mirano a rappresentare, nel modo più realistico possibile, la struttura ed il funzionamento di una rete P2P.

3. Georgia Tech - Internetwork Topology Models (GT-ITM): esso è un programma utilizzato per creare modelli di rete, con diverse topologie, molti nodi e di elevata complessità, a partire da una sequenza di parametri di configurazione. I modelli ottenuti, sono poi utilizzati all'interno dell'ambiente di simulazione NS2 per la definizione del funzionamento della rete.

6.2.1 Modello in MATLAB

Il primo modello sviluppato, è stato realizzato in ambiente di programmazione MATLAB e consiste nella creazione di un mezzo di comunicazione comune alle 4 descrizioni della codifica MDC. L'idea è basata sulla costruzione di un canale "slottizzato", in modo che in uno slot possa avvenire la trasmissione di al più un pacchetto. Nel caso in cui più pacchetti cerchino di accedere allo stesso slot, sarà trasmesso solamente il pacchetto con la classe di QoS a maggior priorità. Il funzionamento del modello può essere suddiviso in diverse fasi che sono di seguito illustrate:

1. **Riempimento delle code relative alle diverse classi di servizio:** in questa fase, sono create 3 code, associate alle tre diverse classi di QoS. Il pacchetto j-esimo di ogni descrizione viene posto nella coda relativa alla classe di appartenenza, in attesa di essere trasmesso (Figura 6.2). Per ogni pacchetto, al momento del suo inserimento nella coda, il modello memorizza, inoltre, l'istante di arrivo.

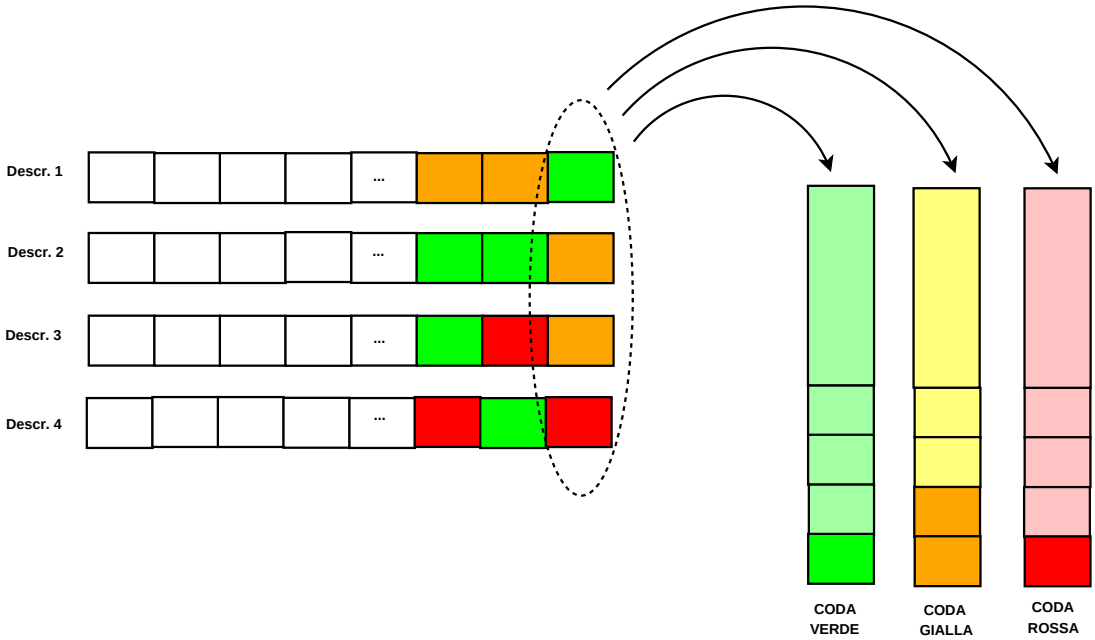


Figura 6.2. Riempimento delle code relative alle diverse classi di QoS.

2. **Accesso al canale:** in questa fase, i pacchetti contenuti nelle 3 code cercano di accedere al mezzo trasmissivo. In particolare, essi accedono al

canale nello slot relativo al loro tempo di arrivo, **in ordine di priorità**: dai pacchetti contenuti nella coda verde a quelli della coda rossa. Se un pacchetto trova lo slot occupato dalla trasmissione di un altro pacchetto a priorità maggiore attende un tempo di *backoff* aleatorio prima di ritentare la trasmissione. Inoltre, essendo il rate di codifica per descrizione di $R = 30\ frame/sec$ ed essendoci $9\ pacchetti/frame$; il tempo di interarrivo tra il pacchetto j-esimo e $(j+1)$-esimo di una stessa descrizione è di:

$$T_{int} = \frac{1}{R * 9} = \frac{1}{30 * 9} = 0.0037 sec \tag{6.1}$$

Il meccanismo di accesso al canale è illustrato in Figura 6.3.

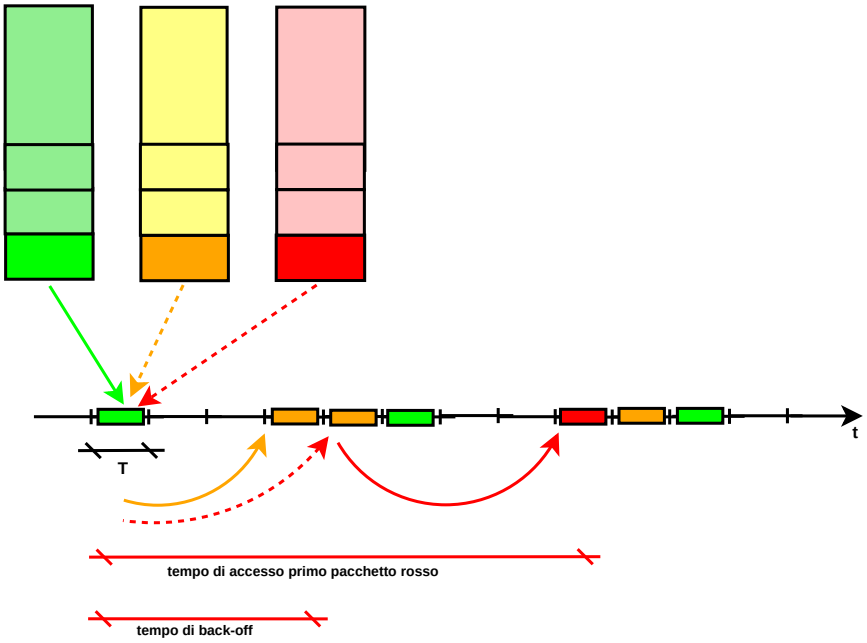


Figura 6.3. Accesso al canale per i pacchetti contenuti nelle 3 code.

3. **Creazione dei pattern d'errore:** In questa terza ed ultima fase, si simulano le perdite di pacchetti, dovute a corruzioni del mezzo trasmissivo e a ritardi di trasmissione, creando dei *pattern* d'errore relativi alle 4 descrizioni. Il tempo di trasmissione T di ogni pacchetto, equivale al tempo di uno slot ed è stato calcolato considerando la lunghezza media dei pacchetti delle diverse descrizioni al variare del tipo di sequenza e del valore di QP. Ponendo la velocità di trasmissione pari a $v_{tr} = 11 Mbit/s$ si

ottiene:

$$T = \left(\frac{1}{v_{tr}}\right) * \text{dim pack} \tag{6.2}$$

Per simulare la presenza di errori sul canale è stato realizzato un modello di canale con errori a *burst* (modello di Gilbert) con diverse probabilità di errore (Figura 6.4). In questo modo, ad ogni slot è associato un numero

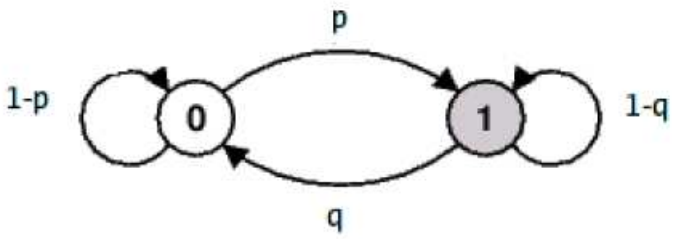


Figura 6.4. Modello di Gilbert.

appartenente all'insieme $\{0, 1\}$ tale che:

- 0 indica che il pacchetto è stato trasmetto correttamente
- 1 indica che il pacchetto è stato perso

Ad ogni pacchetto, di ogni descrizione, è quindi assegnato un valore che indica se tale pacchetto è stato o meno perso. I valori associati a tutti i pacchetti di una descrizione formano un pattern d'errore. Si costruiscono così i 4 pattern d'errore (Figura 6.5). Un altro motivo di perdita di un pacchetto è dovuto al ritardo di accesso al canale. Definito $T_{limit} = 0,2\ sec$ il tempo limite entro il quale è possibile accedere al canale, t_{arr} il tempo di arrivo del pacchetto all'interno della coda e t_{acc} il tempo di accesso del pacchetto al canale, se

$$t_{acc} - t_{arr} \geq T_{limit} \tag{6.3}$$

Il pacchetto sarà scartato riportando il valore 1 nella posizione ad esso relativa del pattern della descrizione a cui appartiene (Figura 6.6). Per ogni sequenza, fissato il valore di QP, sono state considerate 10 realizzazioni del canale per il calcolo di 10 diversi pattern d'errore per ogni descrizione. I pattern saranno poi utilizzati in fase di decodifica dal blocco di *concealment* per simulare la perdita dei pacchetti e calcolare le prestazioni del sistema.

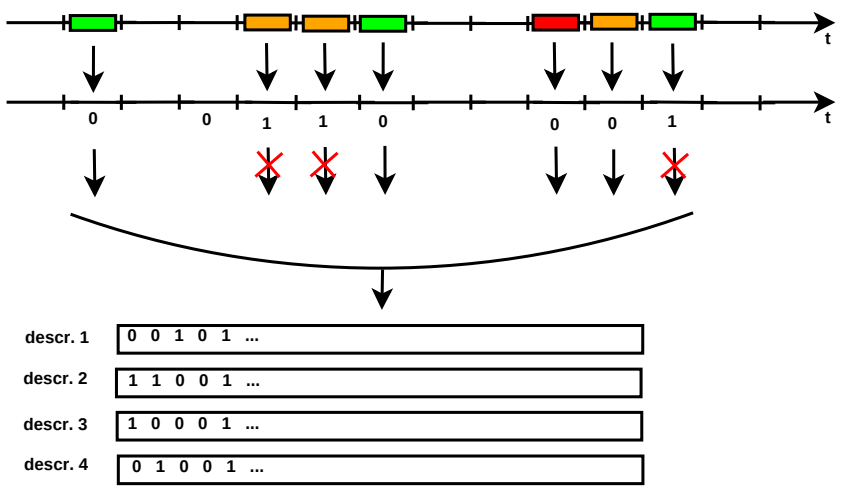


Figura 6.5. Creazione dei pattern d'errore.

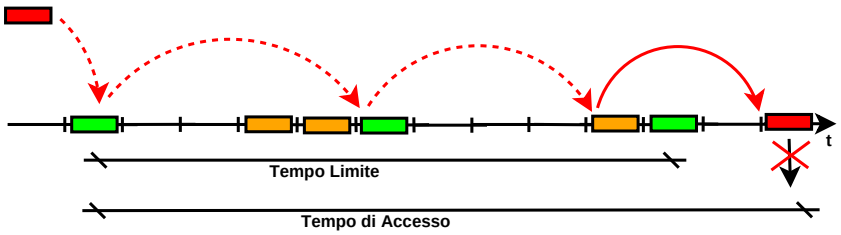


Figura 6.6. Situazione di accesso al canale fuori dal tempo limite.

6.2.2 Il simulatore NS2

Una insieme di modelli più complessi sono stati realizzati attraverso il simulatore *Network Simulator* versione 2 (NS2). NS2 è un simulatore di reti scritto in C++. Lo scenario di una simulazione è scritto in uno script OTcl, che rappresenta una versione del linguaggio Tcl (*Tool Command Language*), orientata agli oggetti. In particolare, utilizzando il linguaggio OTcl si gestiscono delle classi di oggetti, implementate in C++ ed organizzate in maniera gerarchica, che costituiscono la base del simulatore. NS è nato nel 1989, come variante del simulatore REAL, ma negli ultimi anni ha subito un notevole sviluppo, diventando lo strumento di simulazione più utilizzato in ambito accademico. Lo sviluppo di NS2 è affidato ai ricercatori del WINT Project, un progetto che vede la collaborazione di UC Berkeley, LBL, USC/ISI e Xerox PARC, e supportato da DARPA (*Defence Advanced Research Projects Agency*). A seconda di quanto specificato nello script Otcl di input, il simulatore NS2 può produrre diversi tipi di trace file. Essi contengono informazioni che riguardano la topologia della

rete (nodi e collegamenti), gli eventi della simulazione e i pacchetti che hanno viaggiato nella rete. NS2 supporta inoltre le seguenti tecnologie:

- Connessioni punto-punto, reti LAN, mobili e satellitari;
- IP, Mobile IP, multicast (DVMRP, PIM, etc.);
- routing unicast, multicast e gerarchico;
- TCP, UDP, ed altri protocolli di trasporto in fase di sviluppo;
- RTP/RTCP, SRM, e QoS (InterServ, DiffServ (Capitolo 5));
- diverse applicazioni (Telnet, FTP, WWW-like traffic, etc.).

6.2.3 L'applicazione GT-ITM

Georgia Tech-Internetworking Topology Models (GT-ITM) è un'applicazione che consente di creare diverse topologie di rete, utilizzando i parametri contenuti in un file di configurazione. L'architettura di rete realizzata tramite GT-ITM potrà poi essere convertita in un file scritto in linguaggio Tcl ed utilizzabile all'interno del simulatore NS2. GT-ITM si dimostra molto utile per la creazione di modelli di rete molto complessi, con un numero elevato di nodi e che richiedono una struttura ben precisa. In particolare, GT-ITM mette a disposizione tre diverse topologie, definite, di seguito, in ordine di complessità:

- *Flat Random* (FR): In questo modello, i nodi sono distribuiti in modo non gerarchico su un piano ed interconnessi tra loro.
- *N Levels* (NL): Questa topologia, parte dalla creazione di una struttura piana, tipica del modello flat random per poi sostituire uno o più nodi con altri modelli FR. L'operazione di sostituzione potrà poi essere ripetuta per i nuovi nodi creati, realizzando una struttura gerarchica a più livelli.
- *Transit Stub* (TS): Questo modello, permette di creare una struttura gerarchica attraverso l'interconnessione di domini di transito e domini di stub (Figura 6.7). Questa topologia è realizzata a partire da un modello flat random, in cui i nodi rappresentano interi domini di transito. Ogni nodo sarà quindi sostituito da un altro modello FR, che contiene i nodi di un dato dominio di transito. Ciascun nodo di transito, sarà poi collegato ad un certo numero di modelli FR rappresentanti i domini di stub associati ad un nodo di transito. Infine, nodi di *edge* addizionali sono aggiunti per collegare tra loro coppie di nodi appartenenti a diversi domini di stub o di transito.

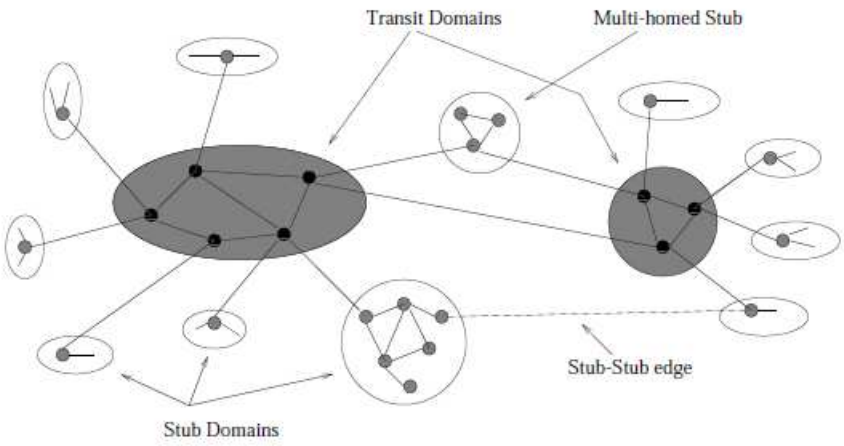


Figura 6.7. Struttura di un modello Transit Stub.

6.2.4 Descrizione dei modelli

Utilizzando NS2 sono stati sviluppati diversi modelli di reti, con diverse topologie, che riproducono, più fedelmente possibile, la struttura di una rete P2P. In ognuno di questo modelli, 4 nodi sono adibiti a contenere le diverse descrizioni della codifica MDC, mentre un nodo CBR (Constant Bit Rate) trasmette traffico ad un rate costante e funge da interferente. Un nodo destinazione, infine, scarica il traffico proveniente sia dai nodi contenenti le descrizioni che dal nodo interferente. Tutti i modelli realizzati utilizzano il meccanismo *DiffServ* ed il protocollo *srTCM*, entrambi definiti nel Capitolo 5 ed opportunamente modificati per assegnare la classe di QoS ai diversi pacchetti, utilizzando i file di testo ottenuti in fase di classificazione, tramite teoria dei giochi. In particolare sono state create 3 diverse tipologie di rete che si differenziano principalmente per struttura, numero di nodi e caratteristiche di congestione. Mentre i primi due modelli presentano un numero limitato di nodi, il terzo è più complesso ed stato realizzato con l'ausilio dell'applicazione GT-ITM descritta precedentemente.

Primo modello: rete a 9 nodi e congestione sul link comune a tutte le descrizioni

Il primo modello di rete realizzato presenta un numero limitato di nodi. In particolare, sono presenti i 4 nodi contenenti le descrizioni della codifica MDC, un nodo CBR ed un nodo di edge che rappresenta il nodo di accesso alla rete e al quale sono collegati i primi 5 nodi elencati. Il nodo di edge è poi collegato con un nodo rappresentante il cuore della rete che a sua volta è collegato ad un altro nodo di edge che rappresenta il nodo di uscita dalla rete. Il secondo nodo di edge è poi collegato al nodo di destinazione. L'intero modello è illustrato in Figura 6.8. Per creare un fenomeno di congestione che porti ad una perdita inevitabi-

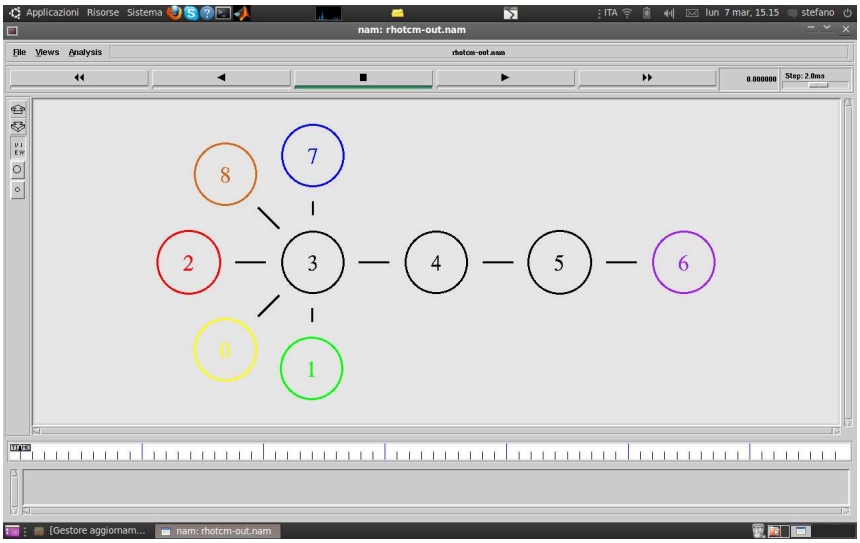


Figura 6.8. Primo modello di rete simulato in NS2.

le di pacchetti e dimostri l'efficacia dei meccanismi di classificazione adottati, si agisce riducendo la banda disponibile R_a sul link che congiunge il nodo di core (nodo 4 in Figura 6.8) con il secondo nodo di edge (nodo 5 in Figura 6.8). In particolare, è stata calcolata la dimensione media di ogni pacchetto delle 4 descrizioni e successivamente il rate medio di traffico per ogni descrizione al variare della sequenza e del valore di QP. Il nodo interferente trasmette ad un rate costante equivalente a quello di un nodo contenente una descrizione. La larghezza di banda del link soggetto a congestione, è stata quindi ridotta in percentuale rispetto alla banda necessaria a trasmettere il flusso proveniente dai 5 nodi. Più precisamente la banda disponibile R_a è stata ottenuta moltiplicando il rate complessivo dei 5 flussi per i seguenti valori $\{0.94, 0.97, 1, 1.03, 1.07, 1.1\}$. Per ognuno di questi valori, inoltre sono state realizzate 10 simulazioni variando in maniera aleatoria i parametri interni alla rete ed utilizzati da NS2. Per

ogni simulazione, il programma produce i 4 pattern d'errore, relativi alle quattro descrizioni ed utilizzati in fase di decodifica dal blocco di *concealment* per simulare la perdita dei pacchetti e calcolare le prestazioni del sistema. Infine, Per ogni valore di R_a è stata considerata la media dei risultati ottenuti sulle 10 simulazioni.

Secondo modello: rete a 15 nodi e congestione sul link reativo alla prima descrizione

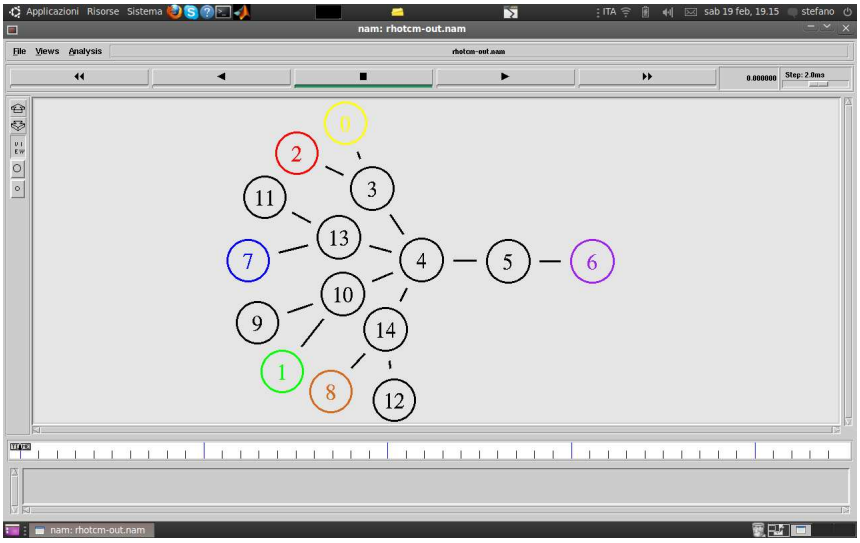


Figura 6.9. Secondo modello di rete simulato in NS2.

Questo modello è stato creato con l'idea di realizzare un fenomeno di congestione e quindi di possibile perdita dei pacchetti appartenenti ad una sola descrizione. In Figura 6.9 i nodi contenenti le descrizioni sono contrassegnati con i numeri 0, 1, 7, 8. Come si può notare, a differenza del primo modello, i nodi di accesso alla rete sono 4, uno per descrizione, ed il nodo interferente (nodo 2 in Figura 6.9) è collegato al nodo di edge relativo alla prima descrizione. La congestione avviene dunque sul link che collega il nodo 3 con il nodo 4 (nodo di core). I questo caso, la larghezza di benda del link soggetto a congestione è stata definita moltiplicando il rate medio di due flussi di traffico (rate prima descrizione e CBR dell'interferente) per la sequenza di valori utilizzata nel primo modello: $\{0.94, 0.97, 1, 1.03, 1.07, 1.1\}$. Anche in questo caso, per ogni valore, sono state realizzate 10 simulazioni e si è considerata la media dei risultati ottenuti per ognuna di esse.

Terzo modello: rete a 100 nodi realizzata attraverso GT-ITM

Il terzo e ultimo modello di rete è più complesso dei precedenti ed è stato realizzato con l'idea di creare una struttura che riproducesse fedelmente una reale rete P2P con molti utenti. Si è deciso quindi di utilizzare l'applicazione GT-ITM, per la creazione di una topologia di tipo Transit Stub contenente 100 nodi. Come già accennato precedentemente, l'applicazione GT-ITM ha bisogno di un file di configurazione per la creazione della rete. Il file di configurazione usato a tale scopo è stato chiamato ts100 ed è il seguente:

```
ts 1 47
3 0 0
1 20 3 1.0
4 20 3 0.6
8 10 3 0.42
```

Lanciando ora il comando:

```
itm ts100
```

GT-ITM crea una topologia di tipo transit stub come indicato nella prima riga del file di configurazione. Il numero 47 identifica il *seed* utilizzato per la definizione dei parametri aleatori della rete (es ritardo di percorrenza dei link, disposizione dei nodi, ecc). La seconda riga del file indica, invece, che la rete presenta 3 domini di stub per nodo di transito e nessun edge aggiuntivo che collega un nodo di stub con un nodo di un altro dominio di stub o di transito. La terza riga stabilisce che si ha un solo dominio di transito, mentre la quarta indica che il dominio di transito è composto, in media, da 4 nodi ed è presente un nodo di edge tra ogni coppia di nodi con probabilità 0.6. L'ultima riga, infine, indica che ogni dominio di stub possiede, in media 8 nodi e per ogni coppia di essi è presnte un nodo di edge con probabilità 0.42. I parametri successivi al numero di nodo per ogni riga, identificano la dimensione dello spazio in cui i nodi di ciascun dominio sono disposti.
Il file prodotto dal comando *itm ts100* produce un file chiamato *ts100-0.gb* che deve essere convertito in un file con estensione Tcl per poter essere utilizzato da NS2. Questa conversione avviene attraverso il comando *sgb2ns* che è così definito:

```
sgb2ns ts100-0.gb ts100.tcl
```

Il file *ts100.tcl* contiene soltanto la struttura della rete e lascia, come parametro da definire, la capacità dei vari link. Per la creazione del modello completo è stato quindi necessario creare diversi nodi "foglia", da collegare ai nodi appartenenti ai domini di stub. Questi nodi foglia rappresentano rispettivamente

le 4 descrizioni (nodi di colore verde, blu, giallo e marrone in Figura 6.10), il nodo interferente (nodo rosso) ed il nodo destinazione (nodo viola). In

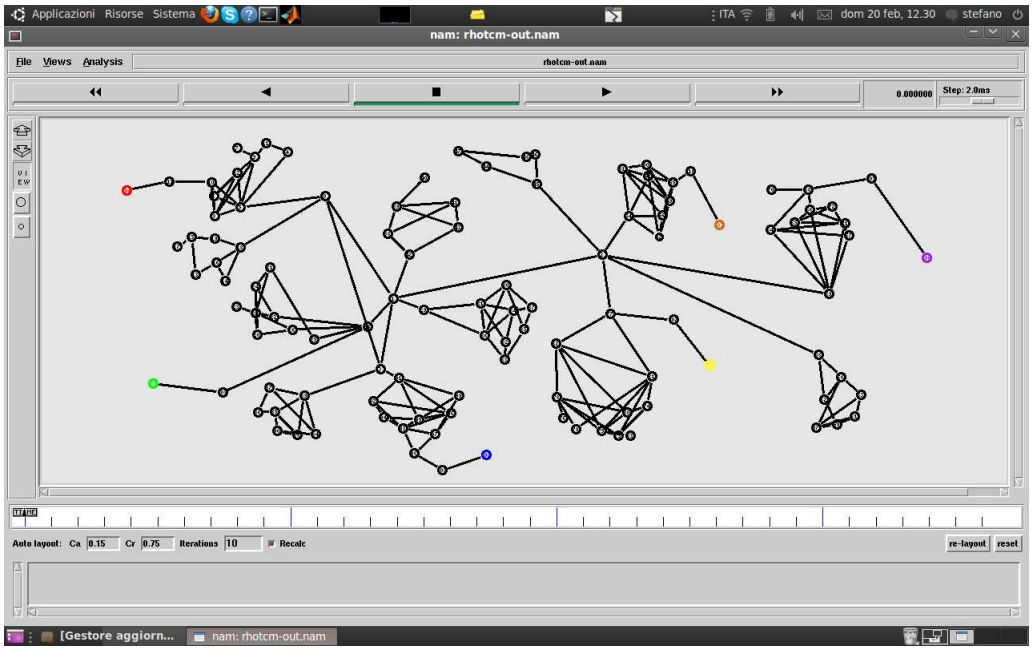


Figura 6.10. Terzo modello di rete simulato in NS2 e creato tramite GT-ITM.

aggiunta, tutti i nodi della rete sono stati configurati per supportare il meccanismo DiffServ ed il protocollo srTCM, modificato opportunamente per utilizzare la classificazione ottenuta tramite teoria dei giochi e contenuta all'interno dei 4 file di testo. Infine, le capacità di tutti i link sono state definite considerando il rate medio complessivo dei 5 flussi di traffico relativi alle descrizioni ed al traffico CBR. Inizialmente il rate del nodo interferente è stato posto uguale a quello di una singola descrizione. Successivamente, a differenza dei primi due modelli, per creare la congestione, si è deciso, di variare il rate del nodo interferente, moltiplicando quello di partenza per i seguenti valori $\{1.5, 1.3, 1.1, 0.9, 0.7, 0.5\}$. Anche in questo caso, come per i primi due modelli sono state realizzate 10 simulazioni e si è considerata la media dei risultati ottenuti per ognuna di esse. Inoltre, al fine di produrre altri modelli tramite GT-ITM, è stato realizzato un *script* in linux, chiamato *conf_gt_itm.sh*, con il compito di create l'intero modello di rete, a partire dal file di configurazione.

6.3 Risultati

Nelle simulazioni di tutti i modelli realizzati, sono stati utilizzati i primi 90 frame di ogni sequenza con $QP = 28$. Per il modello in Matlab, i risultati sono relativi all'utilizzo di un canale con una probabilità d'errore del 2%. Per ogni modello, sono state valutate le prestazioni relative alla classificazione dei pacchetti utilizzando i 3 diversi modelli di gioco cooperativo implementati, per tutte le 6 coalizioni possibili. Inoltre le stesse simulazioni sono state effettuate considerando la classificazione ottenuta tramite gioco non cooperativo al fine di confrontare le prestazioni con i corrispondenti modelli cooperativi. Nelle tabelle e nei grafici relativi ai modelli di rete realizzati in NS2, la sigla *RLC* identifica il rate del link dove si verifica la congestione per i primi due modelli, mentre *RI* indica il rate della sorgente interferente nel terzo modello. Dall'analisi dei risultati si può osservare come l'utilizzo delle coalizioni porti, in alcuni casi, ad un notevole incremento delle prestazioni. In particolare, nella maggior parte dei casi, le descrizioni della coalizione tendono a classificare i propri pacchetti con le classi a più basso livello di priorità possibile, in modo da semplificare la gestione della congestione alla rete ed incrementare il valore medio di PSNR sull'intera sequenza ricostruita. Inoltre è possibile notare come le coalizioni migliori sono quelle formate dalle descrizioni con il minor grado di correlazione. Supponendo di perdere le descrizioni della coalizione, il meccanismo di error concealment riuscirà a stimare le descrizioni perse sfruttando l'elevata correlazione spaziale tra i pacchetti delle descrizioni ricevute correttamente. Questo però non vale per le coalizioni verticali (V1 e V2). La maggior parte delle sequenze presenta infatti una correlazione orizzontale ridotta e, anche se maggiore di quella verticale, non è in genere sufficiente a permettere una valida ricostruzione delle descrizioni perse. In queste situazioni le configurazioni migliori si sono dimostrate quelle oblique O1 e O2. L'utilizzo delle coalizioni oblique porta inoltre ad un incremento delle prestazioni per ogni tipologia di sequenza: la perdita di pixel a quinconce, permette al meccanismo di error concealment di sfruttare sia la correlazione verticale sia quella orizzontale per la stima dei pacchetti persi, che quindi risulta molto efficiente. Analizzando le tre tipologie di gioco create, è possibile osservare come l'utilizzo delle coalizioni nel primo e nel terzo modello porti ad un sensibile incremento delle prestazioni rispetto alla classificazione ottenuta tramite approccio non cooperativo. Per la sequenza *Crew*, nel primo modello di rete realizzata in NS2 e per la prima tipologia di gioco, l'incremento di PSNR è di 2.41 dB per le coalizioni H1 e H2 (contenenti le descrizioni meno correlate) e scende di poco sotto i 2 dB per le coalizioni O1 e O2 (Tabella 6.4). Risultati analoghi si possono osservare per il terzo modello di rete creata tramite GT-ITM: per elevati livelli di congestione, l'incremento di PSNR ottenuto utilizzando le coalizioni raggiunge i 2.88 dB

(Tabella 6.18). I migliori risultati di guadagno di PSNR tramite classificazione con giochi cooperativi si hanno per la sequenza *Foreman*. Per il primo e terzo gioco, l'incremento di PSNR utilizzando le coalizioni O1 e O2 supera i 4 dB sia nel primo che nel terzo modello di rete (Tabelle 6.9, 6.21 e 6.23). L'incremento di PSNR è più contenuto per la sequenza mobile ma in ogni caso supera 1 dB per il primo modello di rete (Tabelle 6.13 e 6.15) e arriva a 1.77 dB per il terzo modello di rete (Tabella 6.27). La sequenza *Coastguard* presenta una correlazione orizzontale maggiore di quella verticale ma, come accennato precedentemente, non sufficiente a permettere una ricostruzione efficiente delle colonne di pixel perse. In questo caso, le coalizioni migliori sono quelle oblique O1 e O2, attraverso le quali si ha un guadagno di PSNR di 2.04 dB per il primo modello di rete (Tabella 6.12) e di 2.14 dB per il terzo modello di rete (Tabella 6.24). Per il secondo modello di gioco, il miglioramento ottenuto attraverso l'utilizzo delle coalizioni è più contenuto. Questo è dovuto principalmente al fatto che, utilizzando una classificazione pacchetto per pacchetto, il gioco non ha la possibilità di utilizzare pienamente le caratteristiche di correlazione spaziale di un frame, cosa che invece avviene nel primo e terzo gioco, in cui la classificazione è fatta frame per frame. Ad esempio, per la sequenza Crew l'incremento di PSNR non supera 0.55 dB per il primo modello di rete (Tabella 6.5) e 0.51 per il terzo modello (Tabella 6.19) mentre per la sequenza Foreman rimane di poco inferiore ad 1 dB per entrambi i modelli di rete (Tabella 6.14 e 6.28). Infine, nel secondo modello di rete in NS2, le prestazioni ottenute tramite giochi cooperativi e non cooperativi sono analoghe. Questo è dovuto al fatto che nel secondo modello, la perdita di pacchetti coinvolge una sola descrizione che attraversa il link soggetto a congestione. Per questo motivo, nessuna delle altre tre descrizioni trae vantaggio coalizzandosi con essa (Tabella 6.16).

COALIZ.	PSNR(dB)
NO	26.7254
H1	29.2222
V1	30.2834
O1	28.4166
O2	28.1064
V2	30.2885
H2	28.3547

(a) *Coastguard*

COALIZ.	PSNR(dB)
NO	31.4990
H1	33.0266
V1	31.7173
O1	31.7001
O2	30.6426
V2	31.6423
H2	33.0888

(b) *Crew*

Tabella 6.1. Risultati della simulazione in Matlab del primo modello di gioco.

COALIZ.	PSNR(dB)
NO	29.9303
H1	30.3861
V1	30.0370
O1	29.2166
O2	29.7800
V2	30.3306
H2	30.5367

(a) *Coastguard*

COALIZ.	PSNR(dB)
NO	33.1468
H1	33.3170
V1	33.6463
O1	33.0434
O2	33.6139
V2	32.2797
H2	33.8295

(b) *Crew*

Tabella 6.2. Risultati della simulazione in Matlab del secondo modello di gioco.

COALIZ.	PSNR(dB)
NO	26.7254
H1	28.3169
V1	29.6870
O1	27.2267
O2	28.2401
V2	29.5445
H2	28.2455

(a) *Coastguard*

COALIZ.	PSNR(dB)
NO	31.4990
H1	33.4919
V1	31.7058
O1	29.8503
O2	29.4929
V2	32.3281
H2	32.0822

(b) *Crew*

Tabella 6.3. Risultati della simulazione in Matlab del terzo modello di gioco.

RLC(kbit/s)	2186	2256	2325	2395	2488	2558
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	31.77	32.61	33.42	33.78	34.10	34.29
H1	34.18	34.21	34.26	34.33	34.47	34.51
V1	32.45	33.15	33.31	33.70	34.03	34.17
O1	34.11	34.23	34.28	34.35	34.49	34.52
O2	33.62	34.21	34.27	34.36	34.50	34.52
V2	32.52	32.99	33.25	33.71	34.04	34.20
H2	34.11	34.15	34.19	34.26	34.45	34.49

Tabella 6.4. Risultati relativi al primo modello di rete in NS2 e primo modello di gioco per la sequenza Crew.

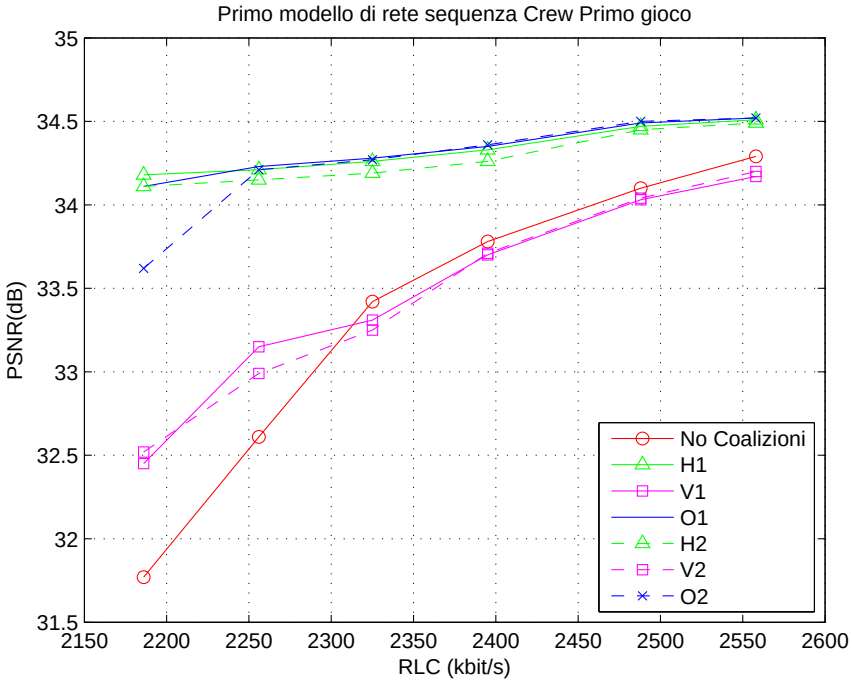


Figura 6.11. Grafico relativo al primo modello di rete in NS2 e primo modello di gioco per la sequenza Crew.

RLC(kbit/s)	2186	2256	2325	2395	2488	2558
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	33.39	33.59	33.73	33.96	34.23	34.31
H1	33.39	33.90	34.02	34.23	34.43	34.47
V1	30.46	31.37	32.37	32.89	33.45	34.22
O1	33.94	33.97	34.07	34.27	34.44	34.49
O2	33.91	34.10	34.18	34.29	34.47	34.49
V2	30.64	31.63	32.90	33.24	33.60	34.38
H2	33.91	34.02	34.14	34.25	34.44	34.49

Tabella 6.5. Risultati relativi al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Crew.

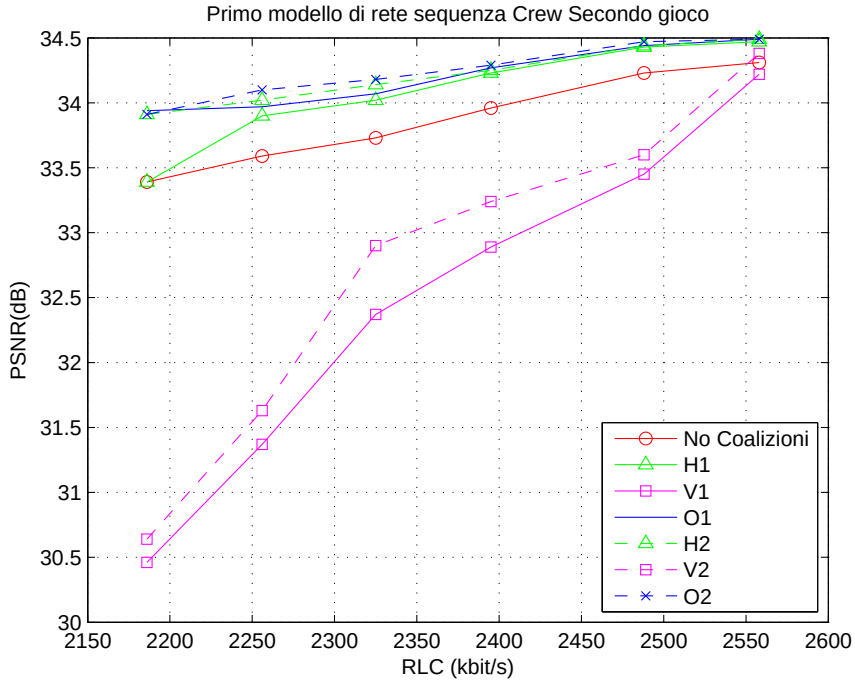


Figura 6.12. Grafico relativo al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Crew.

RLC(kbit/s)	2186	2256	2325	2395	2488	2558
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	31.77	32.61	33.42	33.78	34.10	34.29
H1	34.45	34.18	34.21	34.26	34.44	34.48
V1	29.43	30.51	31.34	32.36	33.15	33.81
O1	34.19	34.24	34.27	34.31	34.46	34.51
O2	34.19	34.23	34.27	34.32	34.46	34.51
V2	29.19	30.36	31.08	32.40	33.33	33.75
H2	32.85	32.73	33.04	33.80	34.26	34.25

Tabella 6.6. Risultati relativi al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Crew.

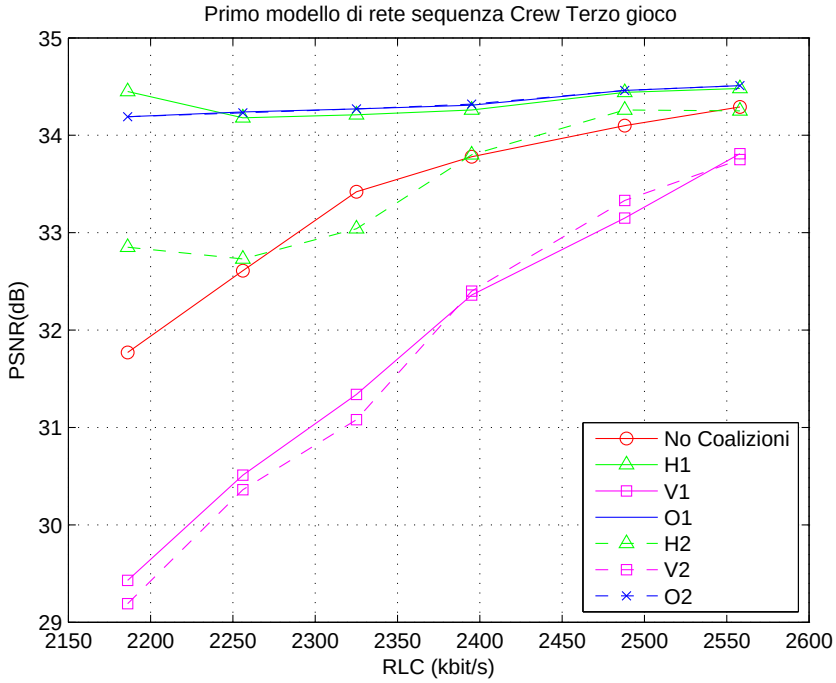


Figura 6.13. Grafico relativo al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Crew.

RLC(kbit/s)	1770	1827	1883	1940	2015	2071
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	26.46	27.22	29.50	30.72	31.49	32.22
H1	27.96	29.14	30.70	31.43	31.92	32.44
V1	28.34	28.88	30.15	30.60	31.53	32.11
O1	30.41	30.81	31.90	32.63	32.92	33.03
O2	30.39	30.71	32.15	32.72	32.96	33.05
V2	28.02	28.59	29.87	30.48	31.25	31.82
H2	26.80	27.93	29.94	30.84	31.66	32.32

Tabella 6.7. Risultati relativi al primo modello di rete in NS2 e primo modello di gioco per la sequenza Foreman.

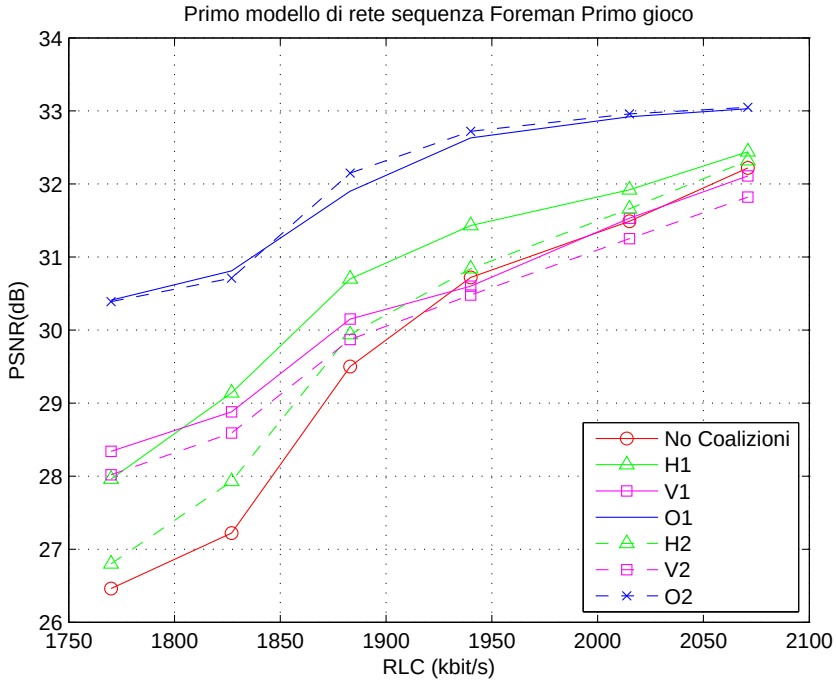


Figura 6.14. Grafico relativo al primo modello di rete in NS2 e primo modello di gioco per la sequenza Foreman.

RLC(kbit/s)	1770	1827	1883	1940	2015	2071
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	26.89	28.43	29.76	30.59	31.54	32.33
H1	27.23	28.65	29.68	31.08	31.68	32.46
V1	25.67	26.84	28.66	30.73	32.11	32.57
O1	27.64	29.03	30.32	31.35	31.86	32.40
O2	27.49	28.39	30.48	31.32	32.30	32.47
V2	23.88	26.12	29.45	30.80	31.54	32.05
H2	28.40	29.49	30.90	31.66	32.20	32.78

Tabella 6.8. Risultati relativi al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Foreman.

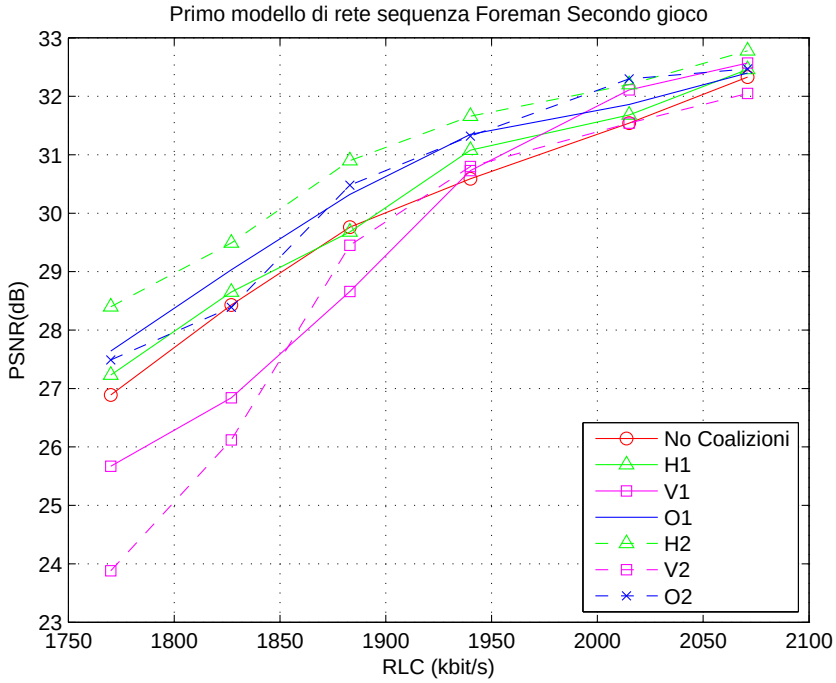


Figura 6.15. Grafico relativo al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Foreman.

RLC(kbit/s)	1770	1827	1883	1940	2015	2071
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	26.46	27.22	29.50	30.72	31.49	32.22
H1	29.08	29.67	30.62	31.44	31.87	32.16
V1	24.22	25.93	27.65	29.82	31.03	31.97
O1	30.47	30.81	31.91	32.76	32.94	33.01
O2	30.49	30.80	32.14	32.81	32.90	33.00
V2	24.41	26.06	28.56	29.99	31.30	32.22
H2	29.91	30.14	31.63	32.06	32.41	32.61

Tabella 6.9. Risultati relativi al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Foreman.

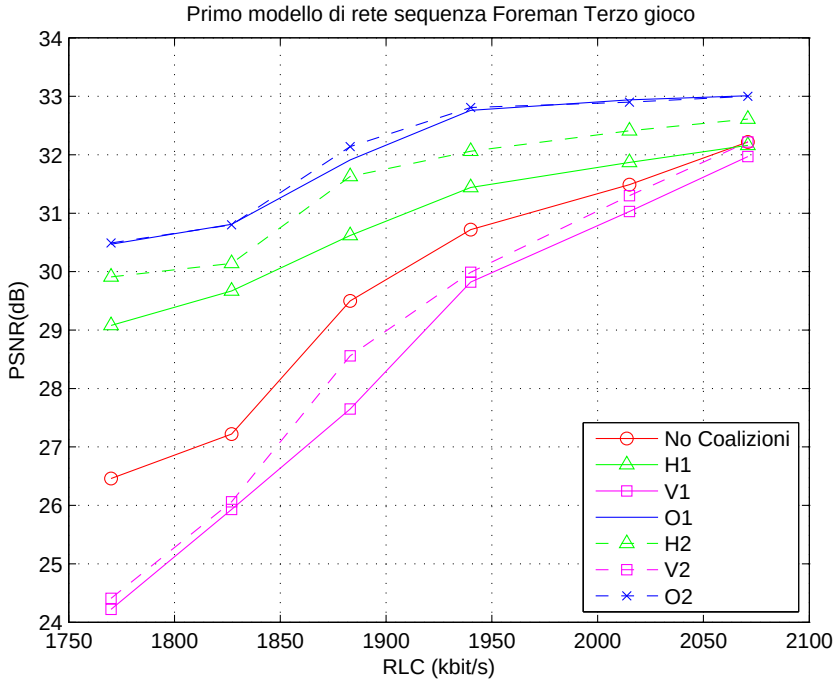


Figura 6.16. Grafico relativo al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Foreman.

RLC(kbit/s)	2822	2913	3003	3093	3213	3303
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	29.37	29.96	30.43	30.80	31.28	31.55
H1	28.89	29.65	30.21	30.71	31.19	31.49
V1	27.95	29.06	29.24	29.91	30.56	31.38
O1	31.22	31.50	31.77	31.97	32.19	32.26
O2	31.34	31.59	31.68	31.97	32.16	32.25
V2	27.96	28.65	29.33	29.59	29.78	30.94
H2	29.28	30.03	30.55	31.00	31.44	31.65

Tabella 6.10. Risultati relativi al primo modello di rete in NS2 e primo modello di gioco per la sequenza Coastguard.

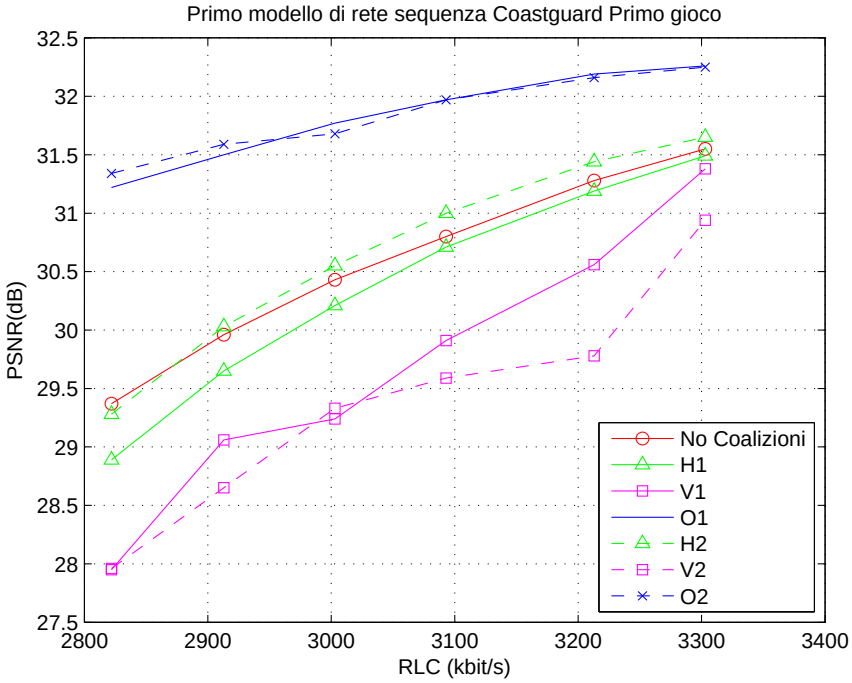


Figura 6.17. Grafico relativo al primo modello di rete in NS2 e primo modello di gioco per la sequenza Coastguard.

RLC(kbit/s)	2822	2913	3003	3093	3213	3303
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	29.20	29.81	30.48	30.79	31.14	31.44
H1	29.20	29.69	30.62	30.98	31.15	31.54
V1	29.68	29.87	30.56	30.99	31.31	31.74
O1	29.41	29.90	30.78	31.04	31.41	31.60
O2	29.60	30.04	30.69	31.01	31.26	31.43
V2	29.16	29.55	29.81	30.70	31.09	31.48
H2	29.01	29.62	30.39	30.73	31.16	31.42

Tabella 6.11. Risultati relativi al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Coastguard.

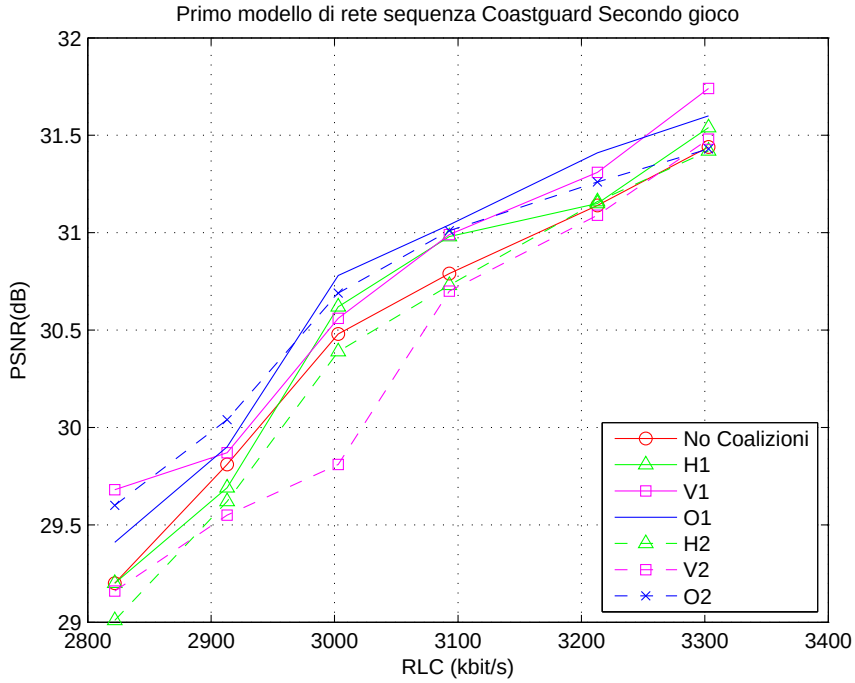


Figura 6.18. Grafico relativo al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Coastguard.

RLC(kbit/s)	2822	2913	3003	3093	3213	3303
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	29.37	29.96	30.43	30.80	31.28	31.55
H1	30.25	30.73	31.06	31.42	31.75	31.90
V1	28.07	28.72	29.20	29.59	30.01	30.28
O1	31.41	31.61	31.76	31.99	32.18	32.27
O2	31.41	31.59	31.77	31.97	32.18	32.26
V2	27.69	28.58	29.21	29.62	29.91	30.09
H2	29.99	30.47	30.82	31.24	31.62	31.80

Tabella 6.12. Risultati relativi al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Coastguard.

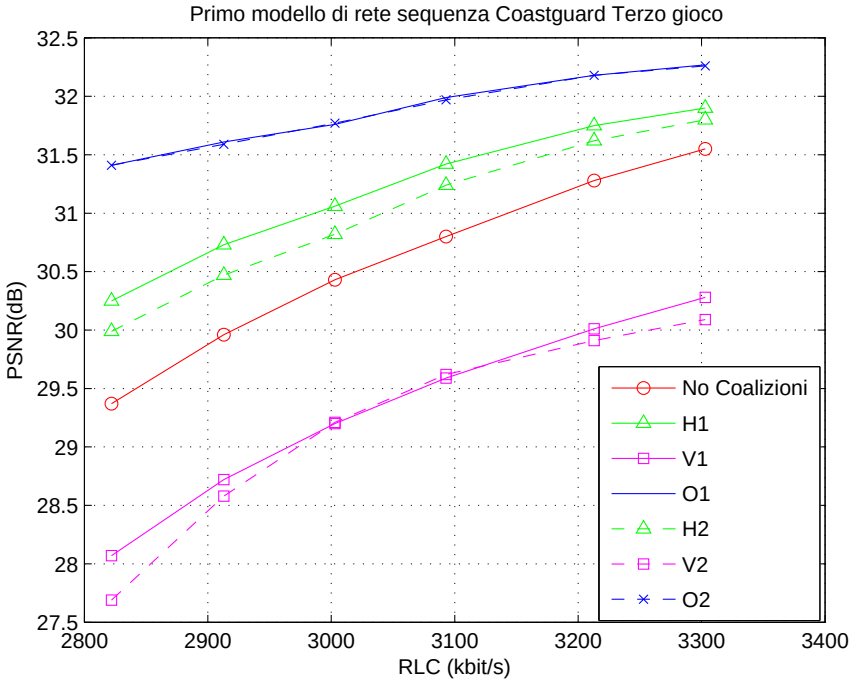


Figura 6.19. Grafico relativo al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Coastguard.

RLC(kbit/s)	6971	7194	7416	7639	7935	8158
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	27.93	28.86	29.70	30.56	32.31	32.49
H1	28.42	29.23	29.91	30.87	32.31	32.49
V1	28.07	29.05	29.91	30.83	32.33	32.49
O1	28.99	29.64	30.23	30.89	32.32	32.49
O2	27.92	28.85	29.70	30.61	32.31	32.49
V2	27.20	28.33	29.61	31.06	32.32	32.49
H2	28.60	29.28	29.99	30.85	32.32	32.49

Tabella 6.13. Risultati relativi al primo modello di rete in NS2 e primo modello di gioco per la sequenza Mobile.

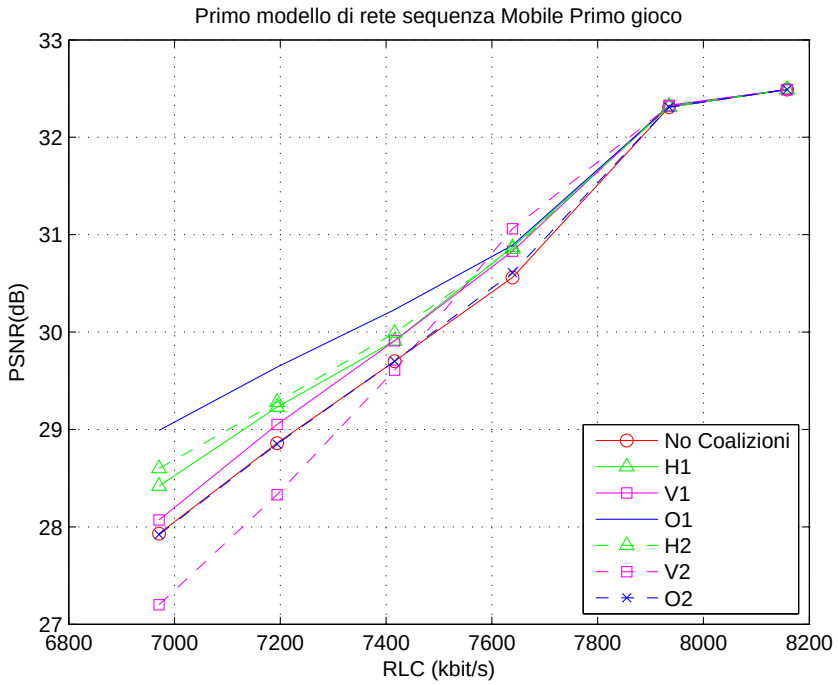


Figura 6.20. Grafico relativo al primo modello di rete in NS2 e primo modello di gioco per la sequenza Mobile.

RLC(kbit/s)	6971	7194	7416	7639	7935	8158
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	26.82	27.42	29.51	30.81	32.30	32.49
H1	28.56	29.32	30.06	31.02	32.32	32.49
V1	26.56	28.07	29.88	30.99	32.35	32.49
O1	29.00	29.65	30.24	31.04	32.30	32.49
O2	28.92	29.54	30.21	31.06	32.31	32.49
V2	26.33	27.96	30.22	31.30	32.40	32.49
H2	28.52	29.20	30.01	30.93	32.33	32.49

Tabella 6.14. Risultati relativi al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Mobile.

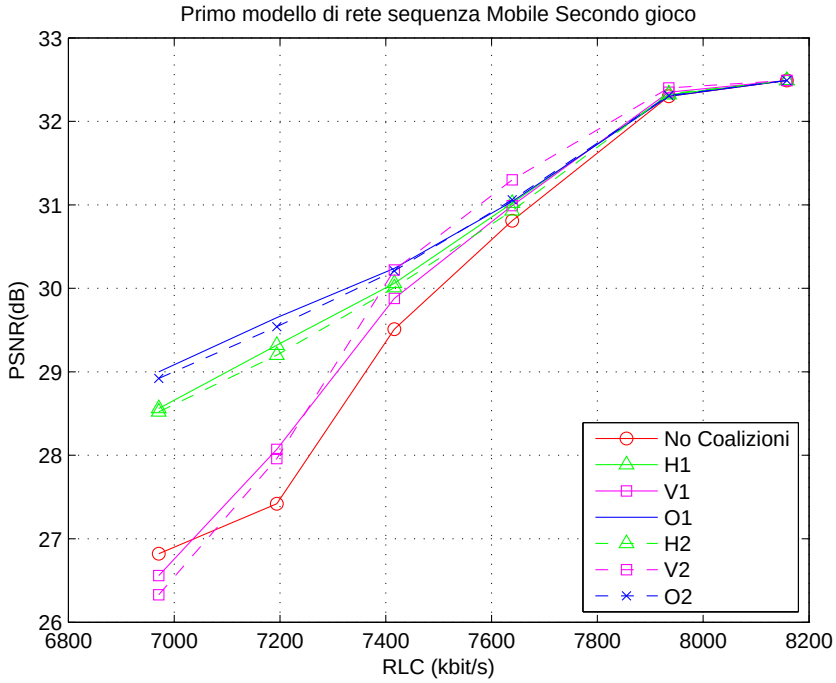


Figura 6.21. Grafico relativo al primo modello di rete in NS2 e secondo modello di gioco per la sequenza Mobile.

RLC(kbit/s)	6971	7194	7416	7639	7935	8158
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	27.93	28.86	29.70	30.56	32.31	32.49
H1	28.58	29.30	29.98	30.75	32.32	32.49
V1	26.70	28.30	29.35	30.98	32.34	32.49
O1	28.97	29.60	30.19	30.90	32.29	32.49
O2	28.99	29.64	30.24	30.89	32.32	32.49
V2	26.87	28.46	29.56	30.98	32.40	32.49
H2	28.45	29.24	29.94	30.75	32.29	32.49

Tabella 6.15. Risultati relativi al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Mobile.

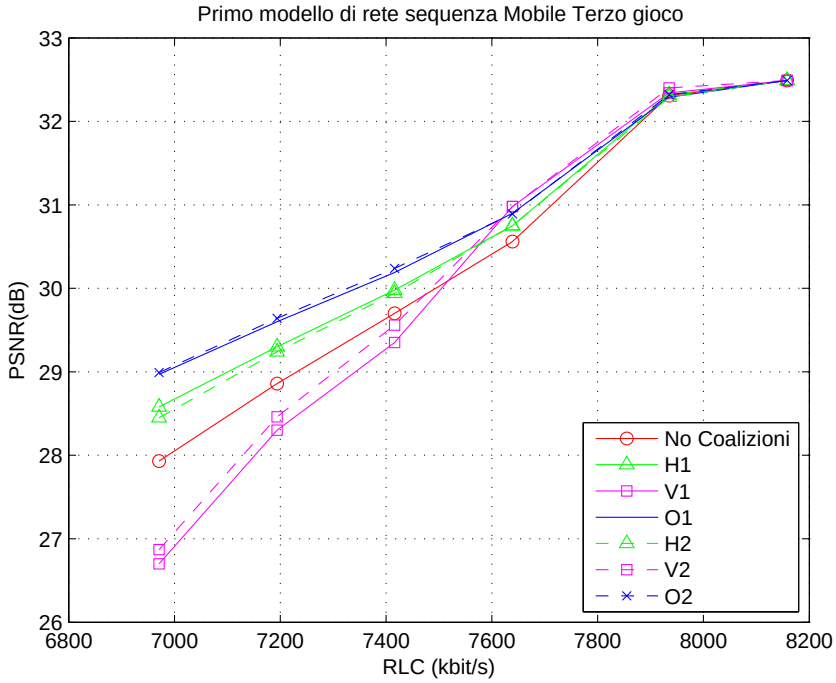


Figura 6.22. Grafico relativo al primo modello di rete in NS2 e terzo modello di gioco per la sequenza Mobile.

RLC(kbit/s)	708	730	753	776	806	828
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	33.72	33.76	33.83	33.90	34.11	34.18
H1	33.68	33.75	33.82	33.94	34.15	34.18
V1	33.69	33.73	33.80	33.92	34.12	34.18
O1	33.75	33.77	33.82	33.90	34.11	34.18
O2	33.84	33.85	33.97	34.13	34.18	34.18
V2	33.70	33.75	33.80	33.92	34.16	34.18
H2	33.65	33.79	33.81	33.91	34.12	34.18

Tabella 6.16. Risultati relativi al secondo modello di rete in NS2 e primo modello di gioco per la sequenza Foreman.

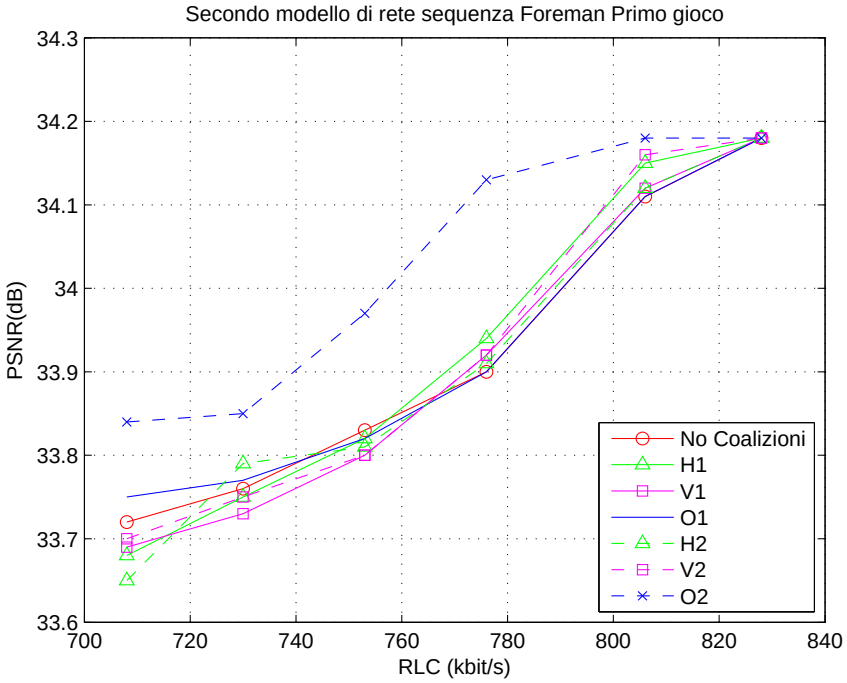


Figura 6.23. Grafico relativo al secondo modello di rete in NS2 e primo modello di gioco per la sequenza Foreman.

RLC(kbit/s)	2788	2877	2966	3055	3174	3263
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	31.33	31.39	31.68	32.23	32.49	32.49
H1	31.39	31.58	31.99	32.49	32.49	32.49
V1	31.39	31.58	31.99	32.49	32.49	32.49
O1	31.39	31.58	31.99	32.49	32.49	32.49
O2	31.12	31.23	31.61	32.13	32.49	32.49
V2	31.12	31.23	31.61	32.13	32.49	32.49
H2	31.12	31.23	31.61	32.13	32.49	32.49

Tabella 6.17. Risultati relativi al secondo modello di rete in NS2 e terzo modello di gioco per la sequenza Mobile.

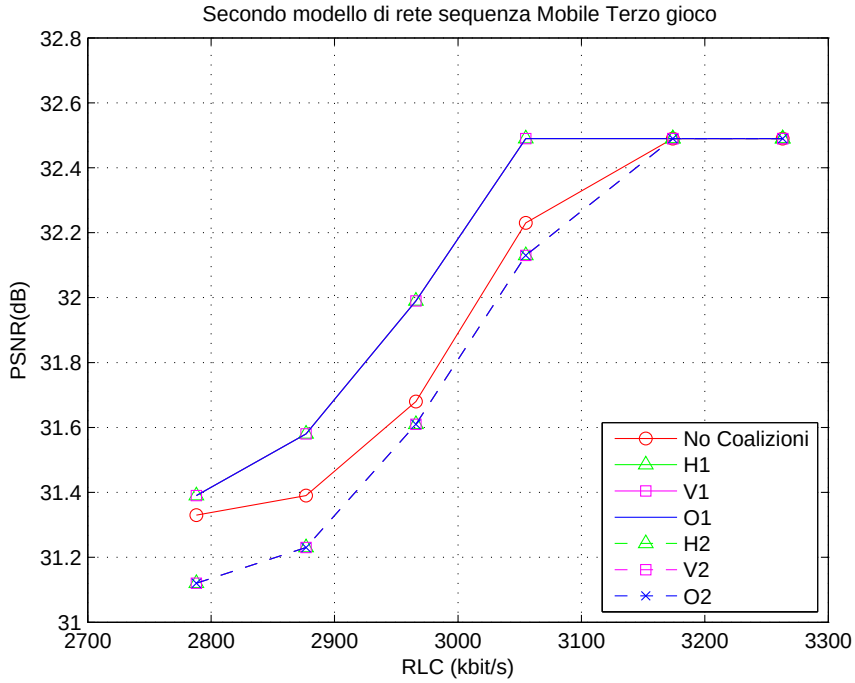


Figura 6.24. Grafico relativo al secondo modello di rete in NS2 e terzo modello di gioco per la sequenza Mobile.

RI(kbit/s)	697	604	511	418	325	232
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	31.39	31.67	33.29	33.77	34.12	34.33
H1	34.27	34.32	34.39	34.45	34.51	34.54
V1	33.18	33.36	33.70	33.95	34.12	34.30
O1	34.27	34.34	34.41	34.46	34.51	34.55
O2	33.89	34.24	34.30	34.38	34.43	34.50
V2	30.35	31.55	32.95	33.69	34.00	34.29
H2	34.10	34.15	34.22	34.33	34.40	34.47

Tabella 6.18. Risultati relativi al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Crew.

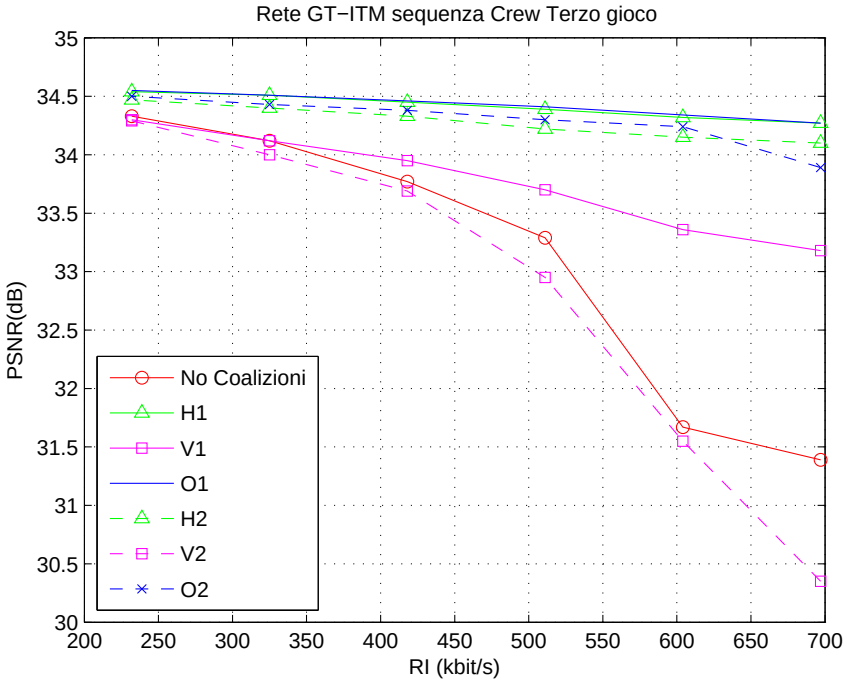


Figura 6.25. Grafico relativo al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Crew

RI(kbit/s)	697	604	511	418	325	232
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	33.40	33.61	33.85	34.08	34.23	34.37
H1	33.90	33.99	34.10	34.30	34.40	34.44
V1	32.29	32.58	33.27	33.38	33.77	34.47
O1	33.91	34.05	34.15	34.17	34.35	34.47
O2	34.12	34.14	34.27	34.41	34.47	34.54
V2	30.99	32.15	32.97	33.44	34.04	34.45
H2	33.96	34.14	34.34	34.39	34.48	34.55

Tabella 6.19. Risultati relativi al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Crew.

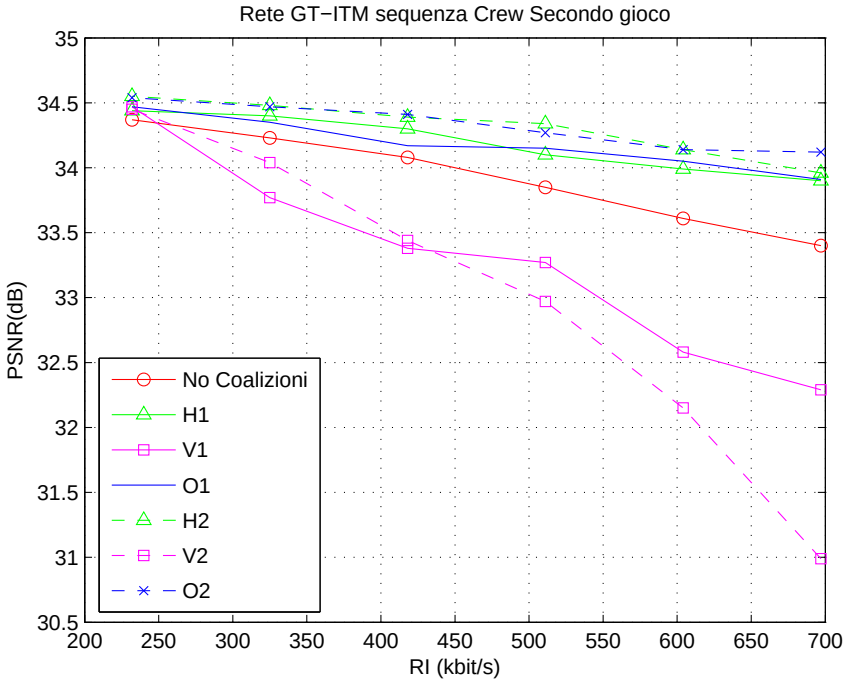


Figura 6.26. Grafico relativo al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Crew.

RI(kbit/s)	697	604	511	418	325	232
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	31.39	31.67	33.29	33.77	34.12	34.33
H1	34.10	34.15	34.22	34.32	34.40	34.46
V1	29.35	30.92	31.67	32.39	33.31	33.67
O1	34.19	34.25	34.31	34.34	34.44	34.49
O2	32.77	32.54	33.73	34.26	34.49	34.55
V2	29.58	30.46	31.62	32.75	33.48	34.38
H2	32.92	32.98	33.67	34.40	34.45	34.16

Tabella 6.20. Risultati relativi al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Crew.

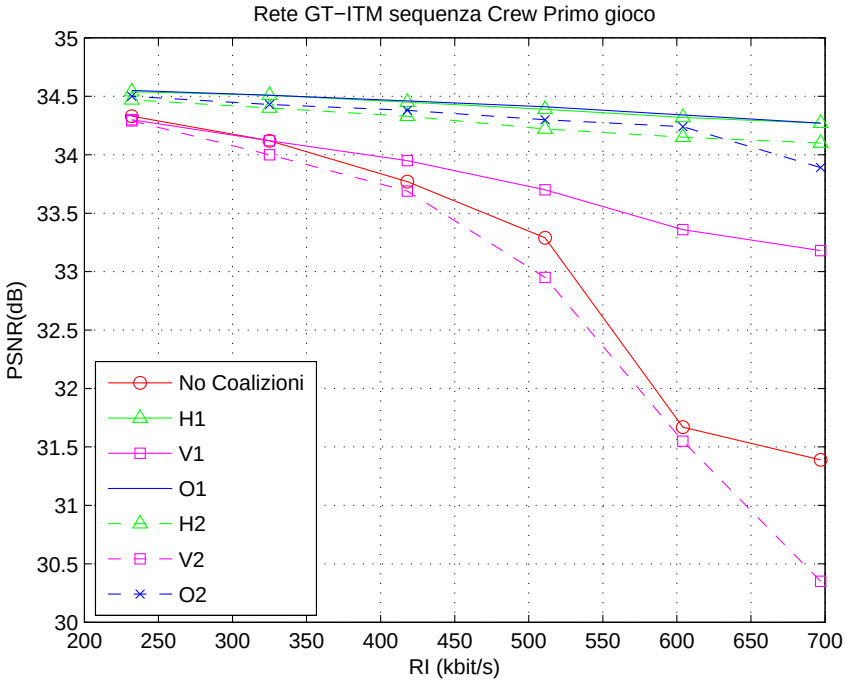


Figura 6.27. Grafico relativo al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Crew.

RI(kbit/s)	565	489	414	339	263	188
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	28.26	29.60	30.26	31.14	32.23	32.87
H1	29.52	30.10	30.80	31.87	32.50	33.13
V1	30.63	30.83	31.58	31.98	32.32	33.01
O1	32.54	33.01	33.13	33.22	33.33	33.50
O2	29.85	31.61	32.64	33.00	33.14	33.38
V2	27.68	29.13	30.58	31.21	32.13	32.94
H2	28.95	29.74	30.77	31.88	32.47	33.10

Tabella 6.21. Risultati relativi al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Foreman.

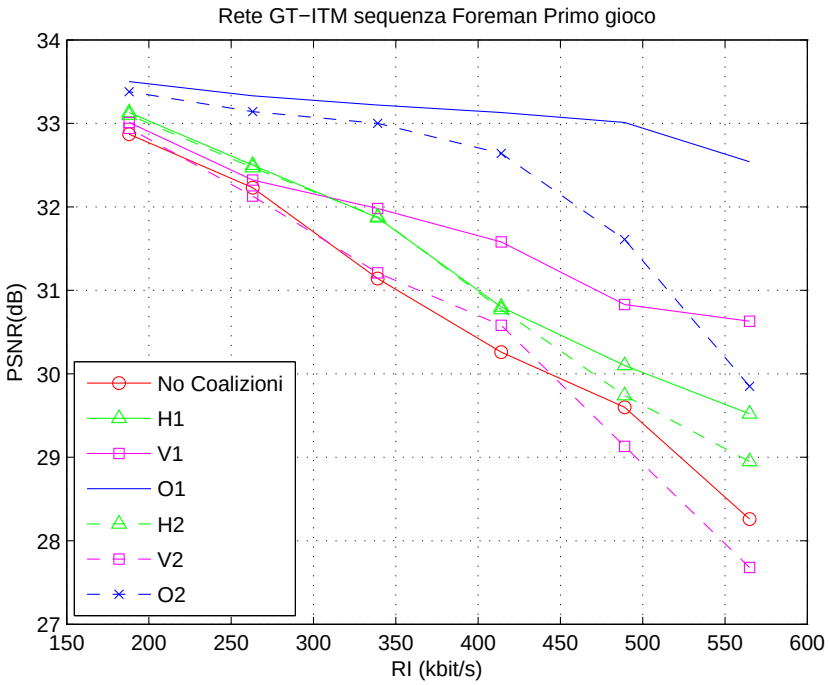


Figura 6.28. Grafico relativo al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Foreman.

RI(kbit/s)	565	489	414	339	263	188
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	29.95	31.11	31.82	32.10	32.56	33.12
H1	28.89	30.21	30.79	32.08	32.87	33.20
V1	29.01	29.20	30.36	31.56	32.50	33.26
O1	30.01	31.12	31.80	32.00	32.72	33.28
O2	30.52	31.16	31.84	32.41	32.51	33.17
V2	27.15	28.17	29.88	31.15	32.42	32.95
H2	29.56	30.01	30.75	32.32	32.92	33.31

Tabella 6.22. Risultati relativi al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Foreman.

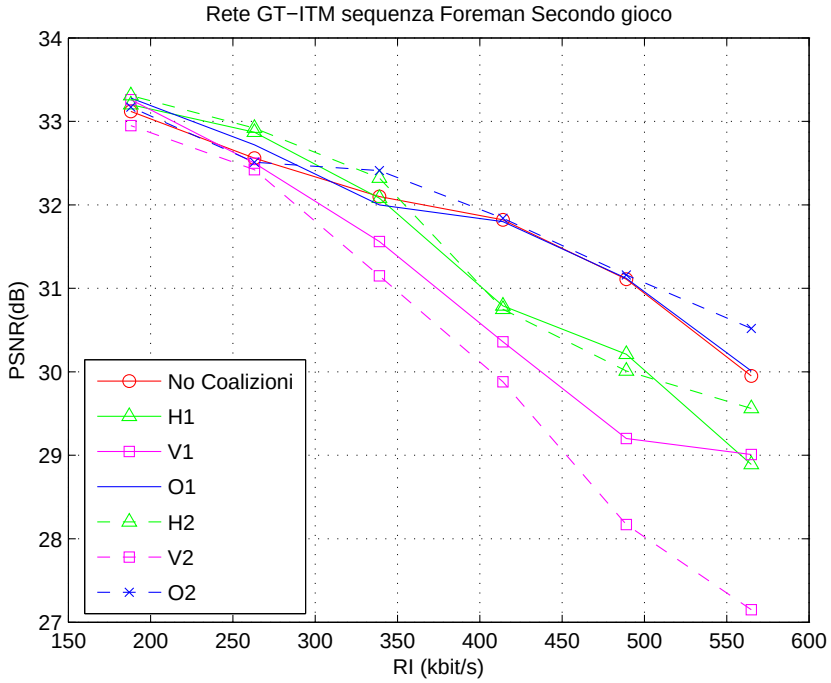


Figura 6.29. Grafico relativo al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Foreman.

RI(kbit/s)	565	489	414	339	263	188
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	28.26	29.60	30.26	31.14	32.23	32.87
H1	31.13	31.53	31.75	31.98	32.50	33.05
V1	25.85	27.32	28.76	30.07	31.66	32.74
O1	32.58	32.81	32.93	33.02	33.14	33.40
O2	31.27	31.67	31.93	32.72	33.30	33.52
V2	26.41	27.71	28.63	30.31	31.76	33.11
H2	31.25	31.56	32.02	32.54	33.00	33.35

Tabella 6.23. Risultati relativi al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Foreman.

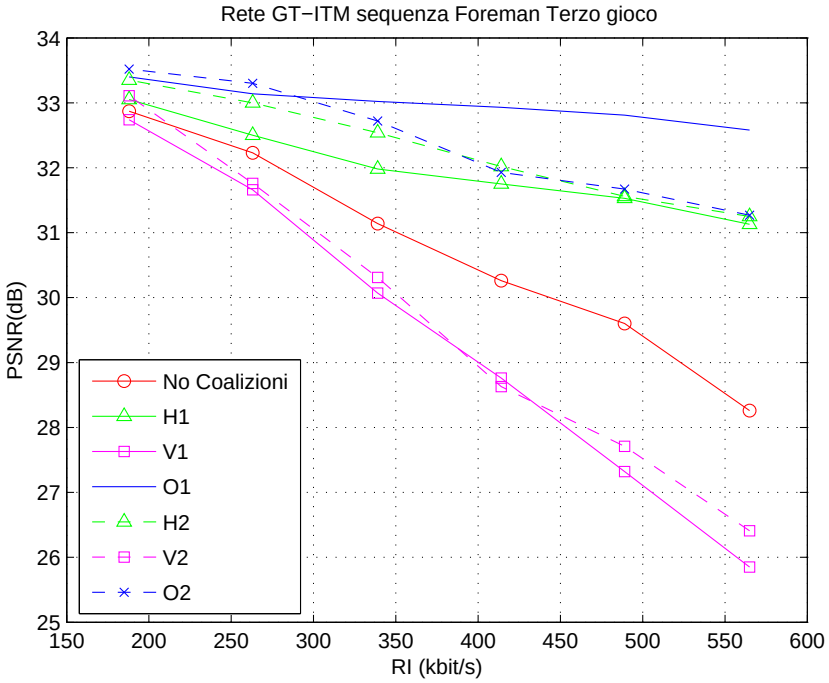


Figura 6.30. Grafico relativo al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Foreman.

RI(kbit/s)	900	780	660	540	420	300
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	29.51	29.86	30.56	31.45	31.74	31.92
H1	29.14	29.87	30.39	31.21	31.66	31.79
V1	28.15	29.10	29.79	30.98	31.72	32.20
O1	31.65	31.78	31.90	32.19	32.30	32.35
O2	31.13	31.43	31.68	31.96	32.13	32.22
V2	27.77	28.92	30.04	31.12	31.80	32.06
H2	29.43	30.09	30.62	31.45	31.85	32.03

Tabella 6.24. Risultati relativi al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Coastguard.

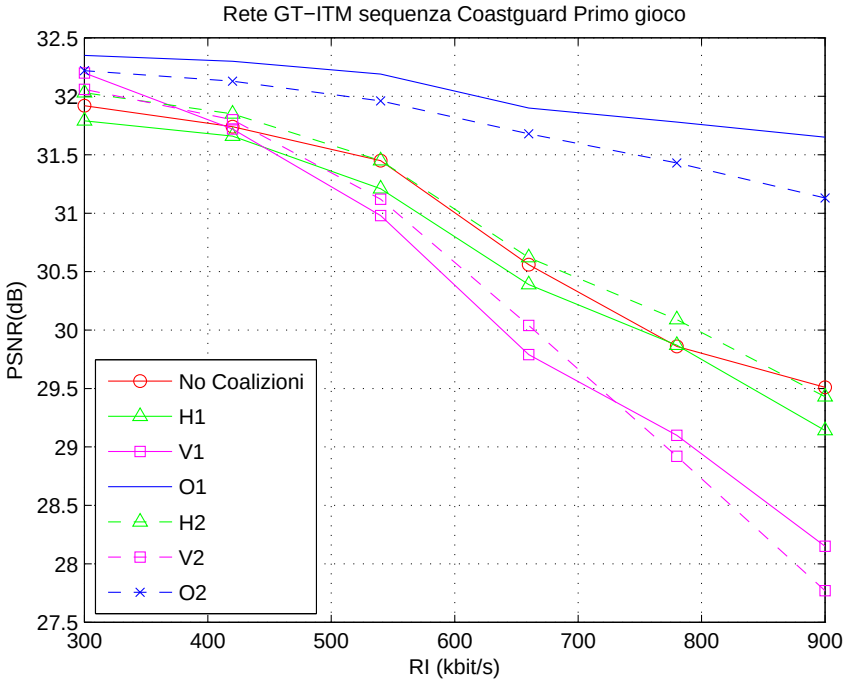


Figura 6.31. Grafico relativo al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Coastguard.

RI(kbit/s)	900	780	660	540	420	300
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	28.73	29.19	30.55	31.20	31.70	31.94
H1	28.63	29.75	30.60	31.45	31.79	31.99
V1	29.88	30.33	30.93	31.22	31.89	32.16
O1	29.57	30.10	31.05	31.47	31.76	32.06
O2	29.68	30.24	30.88	31.12	31.54	32.10
V2	28.65	29.54	30.22	31.23	31.61	32.04
H2	28.58	29.89	30.62	30.98	31.48	32.07

Tabella 6.25. Risultati relativi al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Coastguard.

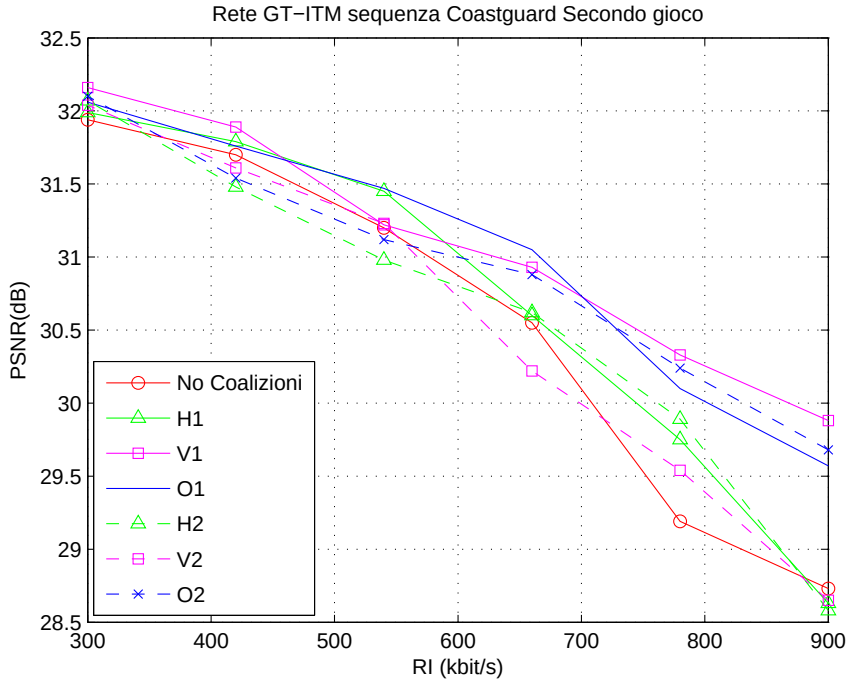


Figura 6.32. Grafico relativo al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Coastguard.

RI(kbit/s)	900	780	660	540	420	300
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	29.51	29.86	30.56	31.45	31.74	31.92
H1	30.33	30.56	30.96	31.66	32.02	32.21
V1	28.08	28.35	29.98	30.81	31.74	32.13
O1	31.42	31.50	31.66	31.93	32.10	32.20
O2	30.47	30.86	31.87	32.20	32.30	32.33
V2	28.06	28.73	29.62	30.98	31.41	31.76
H2	29.92	30.42	31.13	31.71	31.91	32.05

Tabella 6.26. Risultati relativi al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Coastguard.

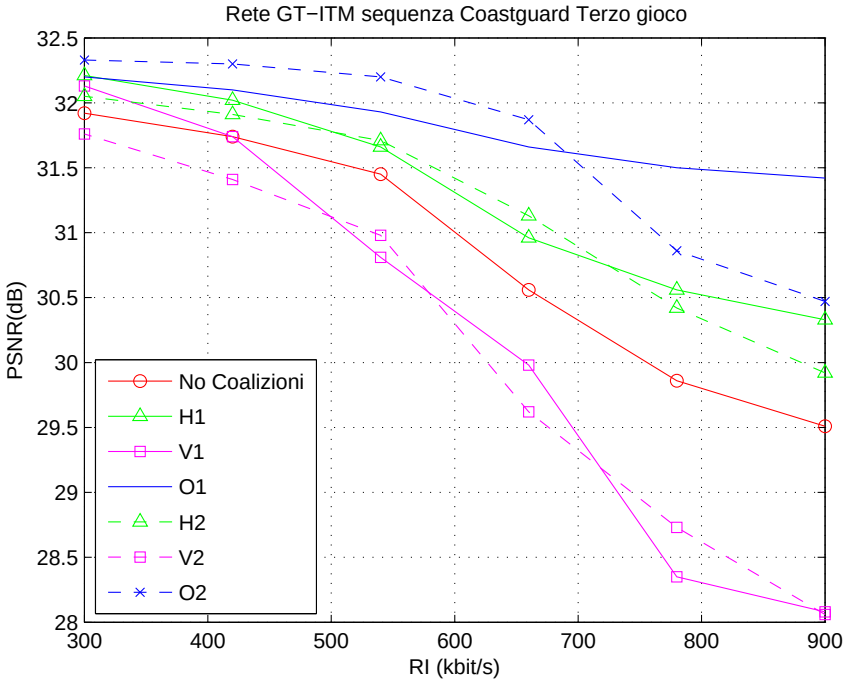


Figura 6.33. Grafico relativo al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Coastguard.

RI(kbit/s)	2225	1928	1631	1335	1038	741
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	27.91	28.59	29.47	30.98	32.49	32.49
H1	29.28	29.72	30.46	31.53	32.49	32.49
V1	27.90	28.95	30.08	31.26	32.49	32.49
O1	29.68	30.07	30.59	31.48	32.49	32.49
O2	29.67	29.89	30.07	31.17	32.49	32.49
V2	26.50	27.74	29.07	31.37	32.49	32.49
H2	28.48	29.11	29.91	31.16	32.49	32.49

Tabella 6.27. Risultati relativi al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Mobile.

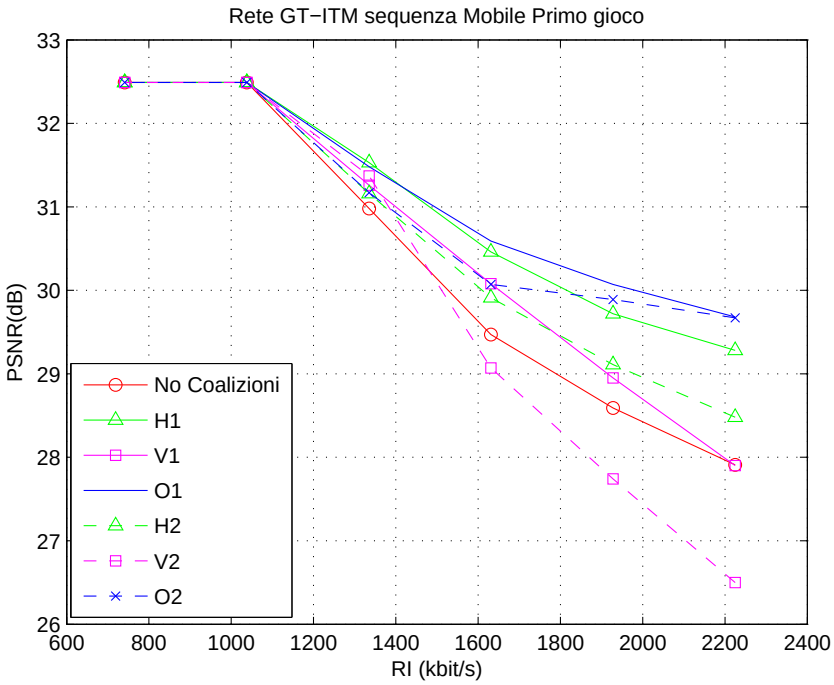


Figura 6.34. Grafico relativo al terzo modello di rete in NS2 e primo modello di gioco per la sequenza Mobile.

RI(kbit/s)	2225	1928	1631	1335	1038	741
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	27.96	28.73	29.69	31.17	32.49	32.49
H1	28.60	29.25	29.98	31.28	32.49	32.49
V1	27.83	28.93	30.32	31.56	32.49	32.49
O1	29.00	29.78	30.22	31.36	32.49	32.49
O2	29.19	29.86	30.49	31.59	32.49	32.49
V2	26.32	28.39	30.21	31.77	32.49	32.49
H2	29.12	29.77	30.47	31.48	32.49	32.49

Tabella 6.28. Risultati relativi al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Mobile.

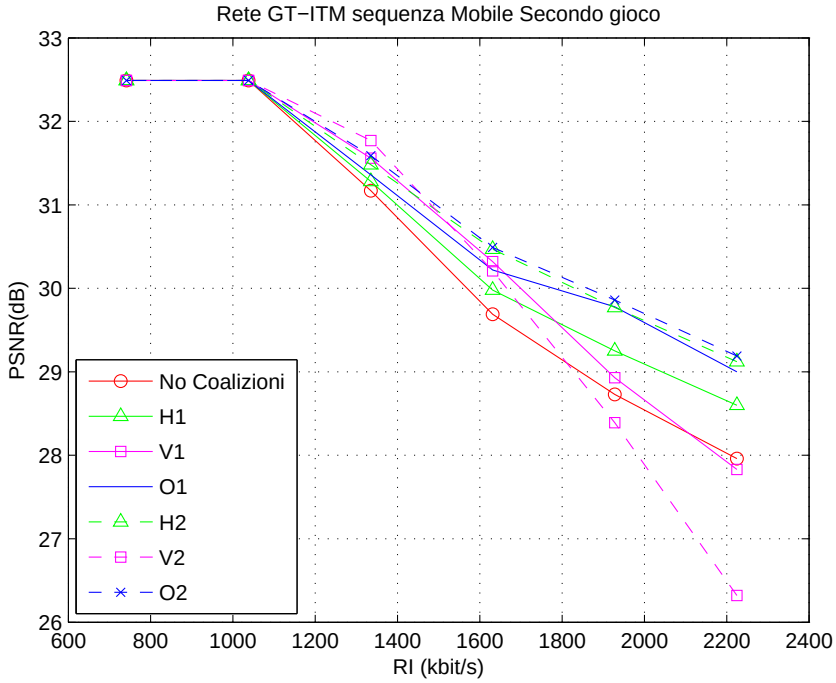


Figura 6.35. Grafico relativo al terzo modello di rete in NS2 e secondo modello di gioco per la sequenza Mobile.

RI(kbit/s)	2225	1928	1631	1335	1038	741
COALIZ.	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
NO	27.91	28.59	29.47	30.98	32.49	32.49
H1	28.50	29.09	29.89	31.10	32.49	32.49
V1	26.94	28.19	29.55	31.34	32.49	32.49
O1	28.89	29.41	30.03	31.14	32.49	32.49
O2	29.05	29.70	30.53	31.48	32.49	32.49
V2	26.11	27.82	29.63	31.28	32.49	32.49
H2	28.86	29.51	30.38	31.44	32.49	32.49

Tabella 6.29. Risultati relativi al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Mobile.

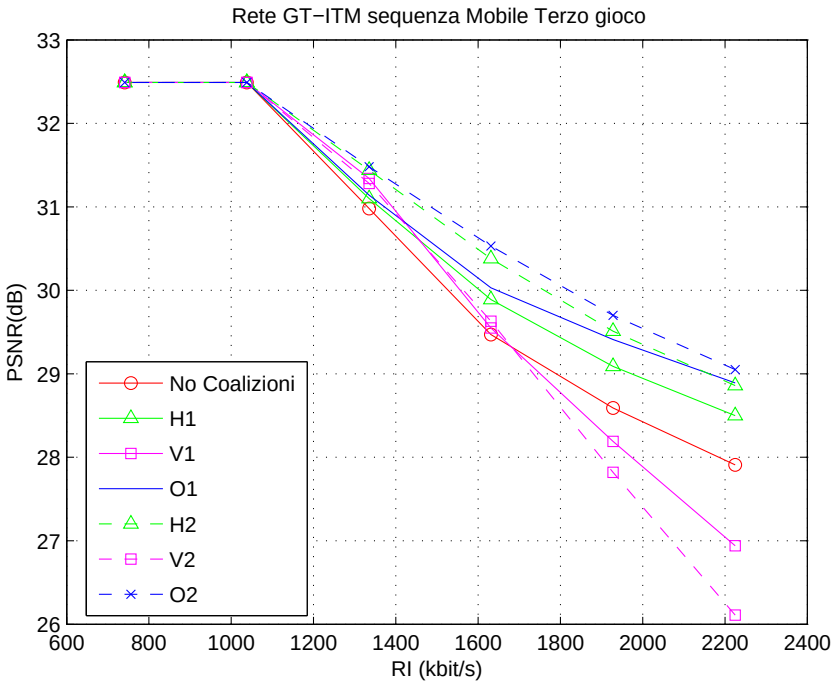


Figura 6.36. Grafico relativo al terzo modello di rete in NS2 e terzo modello di gioco per la sequenza Mobile.

6.4 Conclusioni

Dall'analisi dei risultati è possibile osservare come l'utilizzo di giochi cooperativi, per la classificazione dei pacchetti di sequenze MDC, porti a notevoli vantaggi. In particolare, si sono rivelate più efficaci le coalizioni fra descrizioni che presentano una minor correlazione tra loro. In ogni caso, le due coalizioni oblique (O1 e O2) portano ad un sensibile incremento delle prestazioni per ogni tipo di sequenza. Questo è dovuto principalmente al meccanismo di *concealment* utilizzato. Per stimare un pixel perso, esso utilizza un'interpolazione bilineare tra quelli lo che circondano. I pixel appartenenti ad una delle due coalizioni oblique forniscono, quasi sempre, una buona approssimazione del pixel mancante, in quanto permettono di usufruire sia della correlazione orizzontale sia di quella verticale delle immagini della sequenza. Entrando nel dettaglio delle diverse tipologie di gioco considerate, si può osservare come, la classificazione tramite il primo e terzo modello gioco porti ad un sensibile incremento delle prestazioni ottenute utilizzando il corrispondente gioco non cooperativo. In particolare, per il primo modello di rete in NS2, con il primo gioco, si ha un incremento che arriva oltre i 2.4 dB per le sequenze *Crew* e *Foreman*, e 2 dB per la sequenza *Mobile*. Analogamente, l'incremento delle prestazioni per il terzo modello di gioco è della stessa entità del primo e, in alcuni casi addirittura superiore di qualche frazione di decibel, ad eccezione della sequenza *Mobile*, in cui l'incremento delle prestazioni non va oltre 1 dB. Per il secondo modello di gioco, il miglioramento ottenuto attraverso l'utilizzo delle coalizioni è più contenuto, rimanendo compreso entro 1 dB. Questo è dovuto principalmente al fatto che, utilizzando una classificazione pacchetto per pacchetto, il gioco non ha la possibilità di utilizzare pienamente le caratteristiche di correlazione spaziale di un frame, cosa che invece avviene nel primo e terzo gioco, in cui la classificazione è fatta frame per frame. Il terzo modello di rete, creato attraverso GT-ITM è il più indicativo dell'efficacia del meccanismo di classificazione in quanto rappresenta una struttura di rete che mira a riprodurre l'architettura di una rete P2P con molti nodi. Anche per questo modello, i vantaggi sono notevoli e quantitativamente, l'incremento di PSNR raggiunge i valori del primo modello di rete analizzato. Per il secondo modello di rete in NS2, invece, le prestazioni ottenute attraverso l'utilizzo dei giochi cooperativi e non cooperativi sono analoghe. In questo caso, infatti, la perdita di pacchetti investe solo la prima descrizione, e nessuna delle altre 3 ha vantaggi a coalizzarsi con essa. Per concludere, il meccanismo di classificazione attraverso l'utilizzo di giochi cooperativi, si è dimostrato molto efficace nel contesto delle reti P2P e potrebbe essere applicato, oltre alla trasmissione di video *streaming* o *real-time*, anche ad altre applicazioni, come IPTV e confrontando le prestazioni con altre tecniche in fase di sviluppo. Un'eventuale evoluzione del meccanismo di classificazione

implementato in questo lavoro di tesi può inoltre riguardare uno studio più accurato delle caratteristiche della sequenza video, in modo da poter creare altre tipologie di giochi cooperativi che utilizzano informazioni sempre più precise ed accurate. Infine, un altro meccanismo di classificazione tramite teoria dei giochi potrebbe essere realizzato attraverso l'utilizzo di giochi a strategie miste (capitolo 3). L'idea potrebbe essere sviluppata implementando, dapprima una versione non cooperativa del gioco, per poi introdurre la possibilità, per le descrizioni, di formare coalizioni. Un ulteriore passo potrebbe infine interessare il numero di descrizioni utilizzate: un incremento del numero di descrizioni consentirebbe la formazione di un maggior numero di coalizioni, che potrebbero utilizzare meglio le caratteristiche della sequenza video, a discapito di una maggior complessità realizzativa.

Bibliografia

[1] N. Magharei, R. Rejaie, and Y. Guo, “Mesh or multiple-tree: A comparative study of live p2p streaming approaches.” in Proceedings of IEEE INFOCOM, 2007.

[2] D. E. Comer, “Internetworking with tcp/ip.” Prentice Hall, 2005.

[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications (RFC1889),” in *Network Working Group*, Jan. 1996.

[4] ISO/IEC JTC1, “Coding of Audio-Visual Objects - Part 2: Visual.” ISO/IEC 14 496-2 (MPEG-4 Visual version 1), Apr. 1999; Amendment 1 (version 2), Feb. 2000; Amendment 4 (streaming profile), Jan. 2001, Jan. 2001.

[5] T. Wiegand, “Version 3 of H.264/AVC,” in , 12^{th} *Meeting*, (Redmond, WA, USA), July 17 – 23, 2004.

[6] B. Girod and N. Färber, “Feedback-based error control for mobile video transmission,” *Proc. of the IEEE*, vol. 87, pp. 1707–1723, Oct. 1999.

[7] V. K. Goyal, “Multiple description coding: Compression meets the network.” IEEE Signal Process. Mag., vol. 8, no.5, pp. 74-93, Sept. 2001.

[8] D. Reudink, “The channel splitting problem with interpolative coders.” Bell Lab. Tech Rep., 1980.

[9] V. Vaishampayan and A. Siddiqui, “Speech predictor design for diversity communication systems.” IEEE Workshop Speech Coding for Telecommunications, Annapolis, Oct. 1979.

[10] A. Gersho, “The channel splitting problem and modulo-pcm coding.” Bell Labs. Memo for Record, 1999.

[11] V. Vaishampayan, “Design of multiple description scalar quantizers.” IEEE Trans. Inform. Theory, vol. 39, pp 821-834, May 1983.

[12] D. Taubman and M. W. Marcellin, *JPEG2000 Image Compression: Fundamentals, Standards and Practice*. Boston, MA, USA: Kluwer, 2002.

[13] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An Architecture for Differentiated Service.” IETF, Tech. Rep., RFC 2475, Dec. 1998. Internet Draft.

[14] S. Milani, G. Calvagno, R. Bernardini, and R. Rinaldo, “A low-complexity packet classification algoritm for multiple description video streaming over ieee 802.11e networks.” Proc. of the 2008 IEEE International Conference on Image Processing, 2008.

[15] M. J. Osborne, *An Introduction to Game Theory*. Oxford University Press, 2004.

[16] V. F. Fragnelli, *Teoria dei Giochi*. Universitá del Piemonte, dipartimento di scienze e tecnologie avanzate, 2007-2008.

[17] I. V. Bajić, “The Effects of Channel Correlation on the Performance of Some Multiple Description Schemes,” in *Proc. of* 10^{th} *Canadian Workshop on Information Theory (CWIT 2007)*, (Edmonton, Alberta, Canada), pp. 93 – 96, June6–8, 2007.

[18] S. Milani, G. Calvagno, R. Bernardini, and P. Zontone, “Cross-Layer Joint Optimization of FEC Channel Codes and Multiple Description Coding for Video Delivery over IEEE 802.11e Links,” in *Proc. of IEEE FMN 2008*, (Cardiff, Wales, GB), pp. 472 – 478, Sept.17 – 18, 2008.

[19] S. Milani and G. Calvagno, “Improving quality-of-experience for multiple description video transmission in peer-to-peer networks.” in Proc. of ACM Workshop on advanced video streaming thecniques for peer-to-peer networks and social networking, Florence, Italy pp 65-69, Oct. 29, 2009.

[20] S. Milani, G. A. Mian, and L. Celetto, “Joint optimization of source-channel video coding using the h.264 encoder and fec codes.” in Proc. of The 13th European Signal Processing, Antalya, Turkey, Sept. 2005.

[21] Z. He, Y. K. Kim, and S. K. Mitra, “Low-delay rate control for DCT video coding via ρ-domain source modeling,” vol. 11, pp. 928–940, Aug. 2001.

[22] S. Milani, L. Celetto, and G. Mian, "A rate control algorithm for the H.264 encoder," in *Proc. of the Sixth Baiona Workshop on Signal Processing in Communications*, (Baiona, Spain), Sept. 8–10, 2003.

[23] E. Akyol, A. M. Tekalp, and M. R. Civanlar, "A flexible multiple description coding framework for adaptive peer-to-peer video streaming." IEEE J. Select. Topics Signal Processing, vol. 1, no. 2, pp. 231-245, Aug. 2007.

[24] S. Milani and G. Calvagno, "A game theory based classification for distributed downloading of multiple description coded videos." in Proc. of thr IEEE ICIP 2009, Cairo, Egypt pp. 3077-3080, Nov. 24-28 2009.

[25] F. Zhai, C. E. Luna, Y. Eisenberg, T. N. Pappas, R. Berry, and A. K. Katsaggelos, "Joint Source Coding and Packet Classification for Real-Time Video Transmission Over Differentiated Service Networks," vol. 7, pp. 716–726, Aug. 2005.

[26] R. Bernardini, M. Durigon, R. Rinaldo, P. Zontone, and A. Vitali, "Real-Time Multiple Description Video Streaming Over QoS-Based Wireless Networks," in *Proc. of International Conference on Image Processing (ICIP 2007)*, (San Antonio, TX, USA), pp. 245 – 248, Sept. 2007.

[27] Z. He and S. K. Mitra, "A Unified Rate-Distortion Analysis Framework for Transform coding," vol. 11, pp. 1221–1236, Dec. 2001.

[28] Z. He and S. K. Mitra, "Optimum bit allocation and accurate rate control for video coding via ρ-domain source modeling," vol. 12, pp. 840–848, Oct. 2002.

[29] http://www.isi.edu/nsnam/ns, *The Network Simulator NS2*.

[30] http://www.cc.gatech.edu/projects/gtitm, *Modeling Topology of Large Internetworks: GT-ITM Software*.