Università degli studi di Pisa

Facoltà di Ingegneria
Corso di laurea specialistica in
Ingegneria delle Telecomunicazioni

Tesi di laurea:

Sviluppo di una Guida Elettronica ai Programmi per Sistemi DVB

Relatori:

Prof. Ing. Marco Luise

Ing. Daniele Sparano

Candidato:

Diego Strina

Anno Accademico 2006/2007

Prefazione:

"Il presente lavoro è basato su prodotti e soluzioni di proprietà MBI S.r.l. che ne detiene la proprietà intellettuale."

Indice:

Introduzione

La presente tesi è stata svolta nell'azienda MBI s.r.l di Pisa.Uno dei principali campi ove MBI opera è quello del Digital Video Broadcasting e rientra in tale contesto il suo prodotto di punta: Eracle.

Eracle è un sistema gateway IP-DVB che permette la trasmissione di flussi dati e flussi audio/video MPEG2 su un unico transport stream DVB di uscita.Il project-team entro il quale è stato inserito il lavoro di tesi è finalizzato allo sviluppo di una Guida Elettronica ai Programmi e più in generale di un sistema, denominato MBI-EPG, che permetta di creare e trasmettere le tabelle DVB di Service Information (SI) riservate dallo standard a tale scopo.

Di particolare interesse la simulazione finale effettuata in azienda.Si è proceduto infatti all'impostazione di un loop locale DVB-T dove, grazie all'uso congiunto di Eracle e del sistema MBI-EPG, è stato possibile visualizzare sul televisore, attraverso un Set-Top-Box standard, alcune informazioni di programmazione associate ad un canale TV di test.

Lo sviluppo del sistema MBI-EPG si è dimostrato essere una preziosissima occasione per poter operare con diversi strumenti informatici, primi fra tutti il linguaggio di programmazione C++ e il metalinguaggio XML.Quest'ultimo ha ricoperto un ruolo molto importante nell'intero progetto, basti pensare che MBI lo ha utilizzato, con il medesimo schema, sia per il trasporto dati che per la produzione del codice.

Durante lo svolgimento del progetto è stata approfondita l'intera famiglia degli standard DVB, riservando particolare cura alla trattazione degli argomenti più strettamente collegati con il trasporto delle informazioni di servizio: Program Service Information (PSI) MPEG2 e Service Infromation (SI).A questa prima fase più strettamente teorica ne è seguita una seconda (parallela allo sviluppo del progetto) di analisi concreta del segnale DVB.In particolare è stato preso in esame un transport stream DVB-T.Rientrano in questa analisi (ma si riveleranno utili anche in altre occasioni) i due codici C++ “Section_Choice” e “Section_Parser” descritti nel quarto capitolo.

1. MPEG2

1.1 Introduzione al video digitale

La digitalizzazione di un segnale video produce una grande quantità di dati da elaborare che supererebbe di gran lunga la capacità di molti sistemi di trasmissione, tra cui i transponder satellitari.E' quindi ovvio che risulti necessaria una procedura di compressione.Per convincersi di tale necessità e del relativo successo dello standard adottato per adempiervi, basterà fare qualche calcolo per capire quale sia la banda necessaria alla trasmissione di un segnale video una volta digitalizzato.Una sequenza video digitale è una successione di immagini catturate dalla videocamera ad intervalli discreti di tempo (figura1.1).

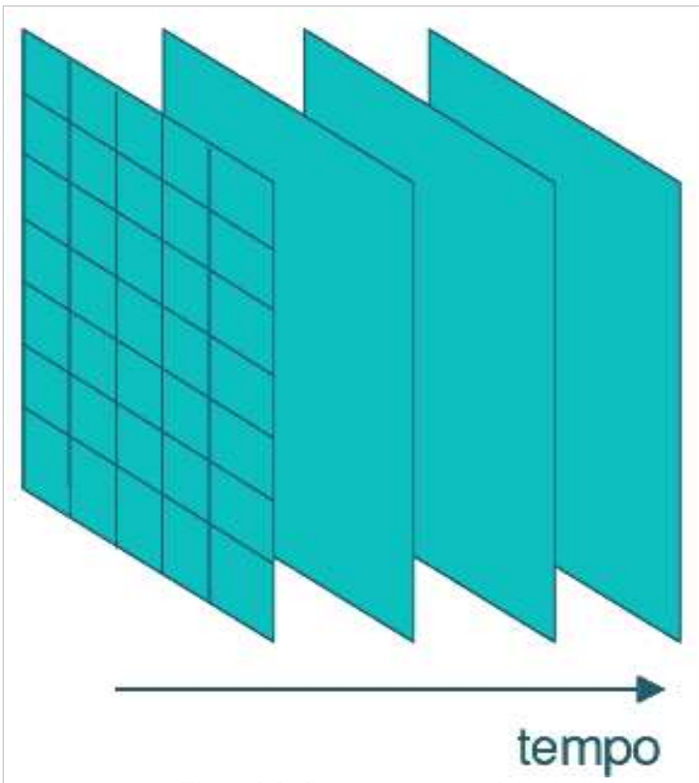

Figura1.1 (Sequenza temporale)

Ognuna di queste immagini può essere rappresentata con una matrice bidimensionale di pixel.Nel caso più semplice di immagine monocromatica, il valore assegnato al singolo pixel darà una misura dell'intensità di grigio in quel punto, viceversa con immagini a colori serviranno più valori per darne una descrizione completa.Visto che tutti i colori possono essere ottenuti da un'opportuna miscelazione di tre colori di base, si è scelto di adottare la

cosiddetta rappresntazione RGB (Red, Green, Blue).Ecco quindi che ad ogni pixel risulteranno associati tre valori, per l'appunto quelli relativi all'intensità del rosso, verde e blu.Un secondo metodo prevede la descrizione dei pixel mediante una cosiddetta componente di luminanza e due di crominanza.La luminanza è proporzionale alla percezione della luminosità, mentre le componenti di crominanza sono legate alla percezione della tonalità e della saturazione.
Questo è il metodo più utilizzato è viene solitamente indicato con l'abbreviazione YUV (Y=luminanza, U e V=crominanza).L'utilizzo di quest'ultimo è dovuto al fatto che l'occhio umano è più sensibile al verde che al rosso e più al rosso che al blu.Inoltre la capacità di apprezzare dettagli nello spazio è massima per il verde e minima per il blu.Proprio per ovviare a tali svantaggi si è scelto di passare alla rappresentazione YUV.L'RGB e l'YUV sono comunque legati fra di loro da una ben precisa relazione matematica qui di seguito riportata.

Y=(219/256)*(Ar*R+Ag*G+Ab*B)+16
U=(224/256)*1/(2*(1-Ab))*(B-Y)+128
V=(224/256)*1/(2*(1-Ar))*(R-Y)+128

Si è fatta l'ipotesi che i valori RGB e YUV siano rappresentati come interi senza segno su 8bit.I parametri Ar, Ag, Ab variano in funzione dello standard video adottato come mostrato dalla seguente tabella.

	A_r	A_g	A_b
ITU-R Rec. 709 (1990)	0,2125	0,7154	0,0721
Non specificato	0,299	0,587	0,114
Riservato	0,299	0,587	0,114
FCC	0,30	0,59	0,11
ITU-R Rec. 624-4 System B, G	0,299	0,587	0,114
SMPTE 170M	0,299	0,587	0,114
SMPTE 240M (1987)	0,212	0,701	0,087

Tabella 1.1 (Parametri Ar,Ag,Ab)

Si analizzano ora le variabili fondamentali che porteranno alla valutazione della banda necessaria alla trasmissione di un segnale video digitale.
L'operazione dovuta alla digitalizzazione di un segnale è la *quantizzazione*, a questa saranno quindi sottoposti anche i valori relativi alle tre componenti YUV.Solitamente si opera una quantizzazione su 8 bit.Con un rapido conto si vede che nella rappresentazione a colori sono necessari 24bit per pixel (8 per ogni componente).
Altro parametro importante è il *frame rate (fps)*, cioè il numero di immagini al secondo che vengono visualizzate.Con 12 al secondo si ottiene già l'impressione del movimento, ma i film di oggi ne usano 24, mentre lo standard PAL 25 e l'NTSC 30.Ovviamente nel computo totale influirà non poco la *dimensione* dell'immagine.Quest'ultima è data dalla larghezza e lunghezza della stessa in pixel.Nel caso di segnali video analogici un'immagine viene normalmente descritta come il risultato di una scansione operata da sinistra verso destra e dall'alto verso il basso (figura 1.2).Nello standard europeo ogni scansione completa è costituita da 625 linee.Di queste però solo 576 sono quelle strettamente legate all'immagine, infatti oltre alle informazioni relative al contenuto di crominanza e luminanza sono necessarie altre informazioni per la trasmissione delle quali è necessario un periodo di pausa pari al tempo di trasmissione di ben 49 linee.

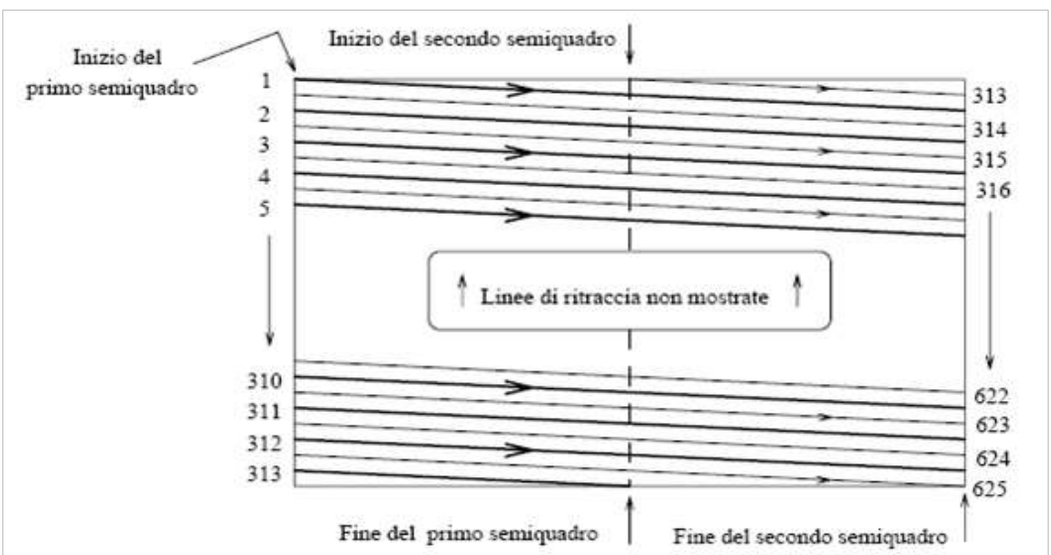

Figura1.2 (Principio di scansione interallacciata.Le righe sono numerate in sequenza temporale)

Come si vede dalla figura 1.3 ogni frame viene suddiviso in due semiquadri,uno contenente le linee dispari (Top field), l'altro quelle pari (Bottom field).

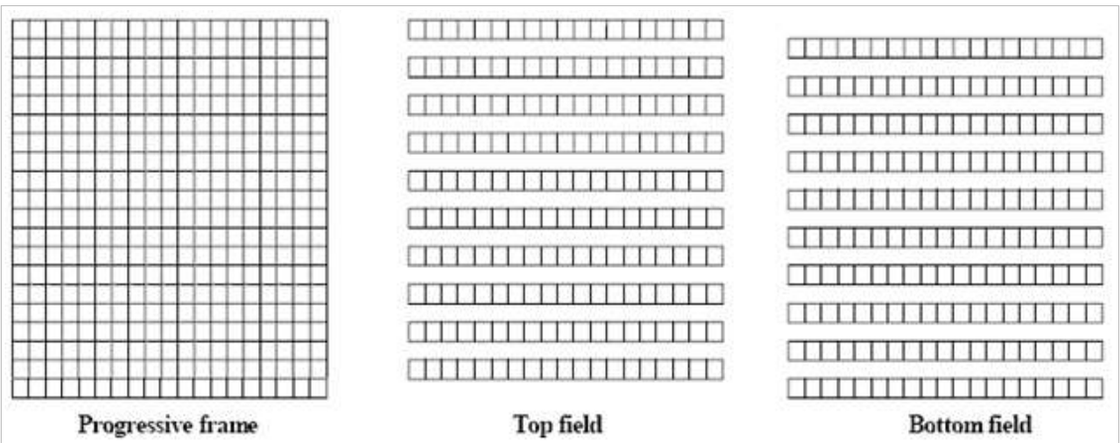

Figura1.3 (Frame scomposto in due fields)

I due semiquadri vengono visualizzati in successione, in particolare negli standard analogici interallacciati prima si visualizzerà il top field e poi il bottom field.Tale metodo venne utilizzato per rendere più stabile l'immagine sullo schermo televisivo evitando di dover aumentare il frame rate, cioè la banda.Come ben si nota il field rate è comunque il doppio di quest'ultimo.

Partendo dalla constatazione che il sistema visivo umano risulta essere più sensibile alla definizione della luminanza rispetto a quello della crominanza, già in ambito analogico questa caratteristica aveva indotto gli sperimentatori a limitare la banda disponibile per le informazioni di crominanza rispetto a quella disponibile per l'informazione di luminanza.In ambito numerico questa scelta si traduce nell'esecuzione di un filtraggio della sequenza di campioni di crominanza allo scopo di ridurne il numero rispetto a quello dei campioni di luminanza introducendo, in tal modo, una perdita di informazione che non pregiudica eccessivamente la qualità percepita dall'utente.Per questo motivo un singolo valore di crominanza si riferisce a più pixel e quindi a più campioni di luminanza.

Il sottocampionamento delle componenti di crominanza rispetto a quelle di luminanza dà origine ai cosiddetti *formati di crominanza*.Due formati di crominanza comunemente utilizzati in ambito televisivo sono il 4:2:0 ed il 4:2:2. Nel formato 4:2:0 le matrici di crominanza *Cr* e *Cb* associati a ciascun quadro hanno dimensioni pari a metà della corrispondente matrice *Y* di luminanza sia orizzontalmente che verticalmente, ossia le componenti di crominanza del

segnale analogico originario sono state campionate orizzontalmente e verticalmente con frequenze pari a metà di quella di luminanza; come si può osservare dalla figura 1.4, orizzontalmente i campioni di crominanza sono associati a campioni alterni di luminanza mentre verticalmente esse occupano righe alterne.Nel formato 4:2:2, le matrici *Cr* e *Cb* associate a ciascun quadro hanno dimensione pari a metà della corrispondente matrice *Y* di luminanza soltanto orizzontalmente mentre verticalmente hanno la stessa dimensione, ossia le componenti di crominanza del segnale analogico originario sono campionate a frequenza spaziale orizzontale pari a metà della frequenza della luminanza e a frequenza spaziale verticale pari a quella di luminanza; come si osserva dalla figura 1.5, infatti, i campioni di crominanza sono associati a campioni alterni di luminanza orizzontalmente mentre verticalmente non viene saltata nessuna riga.

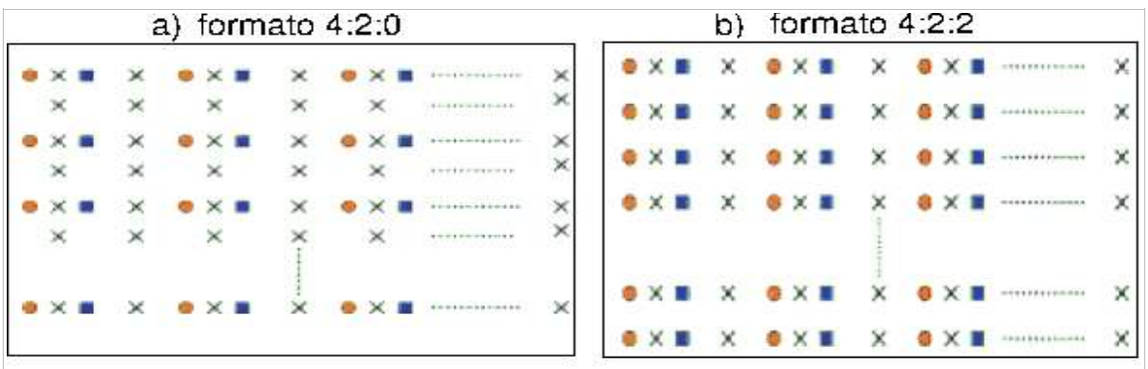

Figura1.4 (formati 4:2:0 4:2:2)

Nella tabella 1.2 si riportano i principali formati video con sottocampionamento.

Formato	Dimensioni in base a Y	Dimensioni Cb, Cr	Sottocampionamento verticale	Sottocampionamento orizzontale
4:4:4	720x480	720x480	No	No
4:2:2	720x480	360x480	No	2:1
4:1:1	720x480	360x240	2:1	2:1
4:2:0	720x480	720x120	No	4:1
4:1:0	720x480	180x120	4:1	4:1

Tabella1.2 (Sottocampionamento delle componenti di crominanza)

Fatta questa lunga, ma indispensabile premessa è possibile fare un rapido calcolo che porterà alla valutazione della banda necessaria alla trasmissione di un segnale video digitale non compresso.
A titolo di esempio si consideri un filmato PAL nel formato 4:2:2 con un frame rate di 25 fps.Ogni singola immagine conterrà allora 720*480 campioni di luminanza e 360*240 campioni per ciascuna componente di crominanza.
Quindi ogni immagine conterrà un numero totale di campioni pari a:

(720*480)+2*(360*480)=691.200

Associando un byte ad ogni campione si otterrà:

691.200*8=5.529.600 bit per immagine

Con un frame rate di 25 fps, un secondo di video digitalizzato necessiterebbe di:

691.200*25=17.280.000 byte ≈ 16,5 Mbyte

Per un intero film di due ore servirebbero:

16,5*7200=118.800 Mbyte ≈ 116 Gbyte

La banda necessaria è quindi pari a:

5.529.600*25=138,24 Mbps

Questo valore è assolutamente ingestibile, infatti supera di diversi ordini di grandezza la larghezza di banda usualmente disponibile su reti di trasmissione digitale, dell'ordine di 10Mbit/sec, e soprattutto, è molto superiore al data rate dei dispositivi di input e output (tipicamente CD, DAT o dischi magneto-ottici) attualmente dell'ordine di 1,5-2 Mbit/sec.
Sono allora inevitabilmente necessari efficienti metodi di compressione.

1.2 Codifica di sorgente MPEG2

1.2.1 Modello adottato

Allo scopo di definire un sistema standard per la codifica delle immagini in movimento, nel Gennaio del 1988 fu costituito l'Organismo MPEG (Motion Picture Expert Group) come Gruppo di esperti dell'ISO/IEC.

Il primo standard prodotto fu MPEG-1 la cui tipica applicazione era prevista in campo multimediale per la codifica video e audio e la memorizzazione su cdrom.Tra i principali limiti di questo standard esso non prevede la modalità interallacciata e tutti i flussi audio e video elementari condividono la stessa base dei tempi non consentendo così l'aggregazione di più programmi tra loro indipendenti.Inoltre non è previsto alcun supporto alla rivelazione di errori e alla perdita di informazioni che possono ad esempio avvenire su collegamenti radio.

Lo standard successivamente approvato in MPEG fu MPEG-2, identificato dalla sigla IS-13818, divenuto standard internazionale nel 1995.

L'obbiettivo di MPEG-2 era quello di definire una codifica adatta per le applicazioni diffusive (radiodiffusione e distribuzione via cavo).

Come si vedrà più nel dettaglio in seguito, tale obbiettivo è stato pienamente raggiunto e la rapida diffusione di tale standard ne è la dimostrazione.

Infatti già alla fine del 1996 MPEG-2 video fu scelto come base per il sistema televisivo digitale terrestre da introdurre in USA (ATV, advanced TV).Si optò per lo stesso, sempre in USA, anche per la diffusione digitale da satellite, nel progetto Direct TV.

In Europa, DVB(Digital Video Broadcasting), ne diventò il maggior utilizzatore mondiale.

Seguì il DVD Forum e infine anche in Giappone MPEG-2 fu introdotto per la diffusione digitale sia di immagini in definizione standard (SDTV) che in alta definizione (HDTV).

Il sistema MPEG-2 lo si può strutturalmente suddividere in otto parti, di cui le prime cinque sono in comune con MPEG-1.

I. ***System***: si occupa di combinare uno o più elemenatry stream (flussi elementari) video e audio così come altri dati in flussi singoli o molteplici i quali siano adatti ad essere scaricati o trasmessi.
II. ***Video***: descrive la sintassi (header ed elementi del bitstream) e la semantica (algoritmi di elaborazione) della sequenza video, suddivisa in

una serie di strati nidificati.Fornisce inoltre vari metodi differenti per assistere il decodificatore nella sincronizzazione, nell'accesso casuale e nella correzione degli errori.

III. ***Audio***: si occupa della codifica audio.Esistono tre classi di compressione, detti layer I, II, III, caratterizzati da una efficienza di compressione crescente.Il layer I è simile allo standard usato da Sony MiniDisk e da Philips Digital Compact Cassette (DCC), il layer II è usato nella telvisione via satellite mentre il layer III è usato su internet e su ISDN per il suo basso bitrate(mp3).

IV. ***Conformità***: precisa il modo in cui possono essere progettati i test di conformità sui bitstream e sui decodificatori.

V. ***Simulazione***: dà un completo software di attuazione delle prime tre parti dello standard MPEG-2.

VI. ***Digital Storage Medium Command and Control (DSMC-CC)***: fornisce una sintassi per controllare l'esecuzione e l'accesso casuale come avviene nei videoregistratori (riavvolgimento, avanti veloce, fermo immagine).

VII. ***Non-bckward compatible audio***: concerne nuove estensioni della codifica audio, che a differenze della parte 3, non sono compatibili con MPEG-1.

VIII. ***10bit-video extension***: era originariamente stata progettata per codifiche video nel caso in cui gli input samples superassero i 10 bit.Le ricerche su questa parte sono cessate per scarso interesse.

1.2.2 Codifica Audio

I codificatori MPEG, realizzati per qualsiasi tipo di segnale, sia esso musicale o vocale (il cd generic audio), sono di tipo percettuale e non del tipo chiamato waveform.In un codificatore di tipo percettuale il codice non ha l'obiettivo di mantenere il segnale identico dalla fase di codifica a quella di decodifica, bensì mira ad assicurare che il segnale d'arrivo appaia identico ad un orecchio umano.
Il nostro orecchio, infatti, non è uno strumento lineare, non percepisce tutti i suoni e soprattutto non li percepisce nello stesso modo.Da qui l'idea di eliminare tutte quelle componenti frequenziali che non possiamo udire.Questa è ovviamente una tecnica lossy, il suono compresso sarà diverso da quello originale ma i nostri sensi non riescono a percepirne la differenza.Si rende quindi necessario uno studio sul modello percettivo, cioè sulla percezione umana del suono.La banda di frequenze udibili va dai 16 Hz fino a 20 kHz, il primo taglio in frequenza viene effettuato direttamente da un banco di filtri, che elimina le frequenze troppo basse o troppo alte.Affinché un suono sia percepibile deve essere sufficientemente forte, deve superare un certo livello di pressione sonora. In più questo livello non è costante ma varia in funzione della frequenza.In figura 1.5 è riportato un grafico qualitativo che mostra quale deve essere la pressione minima che un suono deve avere per poter essere percepito, tutti quelli che si trovano nella zona grigia possono quindi essere eliminati.Nelle ascisse (in scala logaritmica) ci sono le frequenze mentre nelle ordinate i livelli di pressione sonora (in dB).

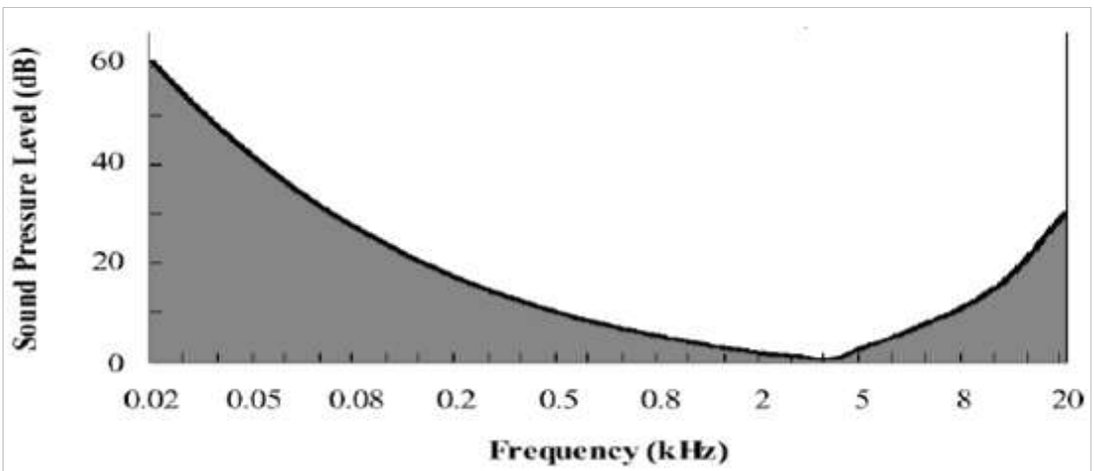

Figura1.5 (Andamento qualitativo della percezione dei suoni)

La percettività del suono inoltre non è costante nel tempo, varia in funzione di ciò che ascoltiamo.Nella figura 1.6 è riportata la variazione della caratteristica in presenza di un suono: mascheramento simultaneo.In pratica un tono forte copre i suoni d'intensità minore non solo ad una determinata frequenza ma anche in quelle vicine.

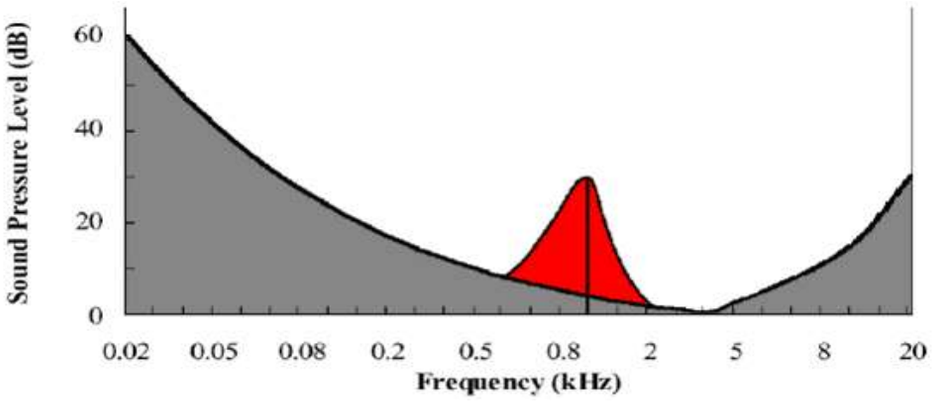

Figura1.6 (Mascheramento simultaneo)

Inoltre esiste un fenomeno di mascheramneto nel dominio del tempo, per il quale dei segnali forti possono coprire altri segnali deboli, anche se questi precedono il mascheramento.

Infatti, il sistema uditivo umano non ha tempi di reazione nulli, impiega cioè un certo tempo per adattarsi alle nuove condizioni e soprattutto impiega tempo a tornare in quiete dopo una sollecitazione.La figura 1.7 mostra come un suono modifichi la soglia d'udibilità nel tempo.Tale fenomeno viene chiamato post-mascheramento.

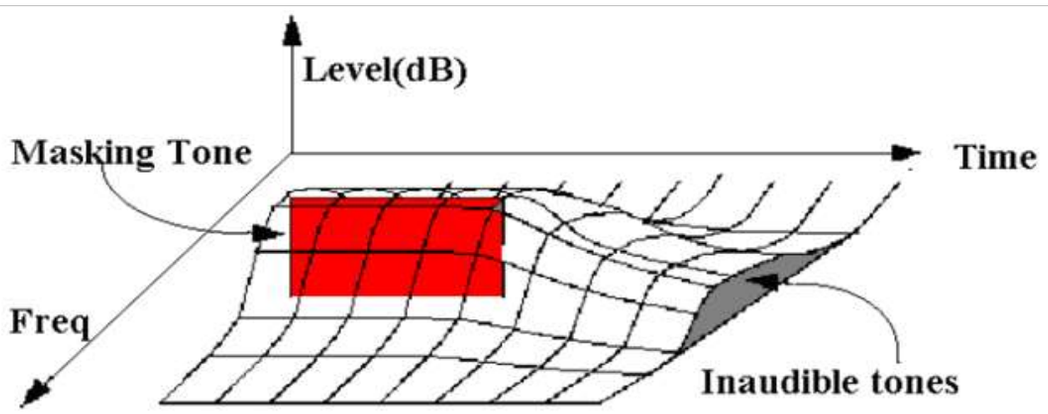

Figura1.7 (Post mascheramento)

Ultima nota è quella relativa al pre-mascheramento, un fenomeno analogo al precedente e si ha quando il volume passa da un livello basso a un livello alto: in tal caso gli ultimi millisecondi del livello basso non vengono percepiti in quanto il cervello trascura il livello basso per dedicarsi a quello alto.La durata del pre-mascheramento è generalmente minore dellla durata di post-mascheramento.
La maschera complessiva ottenuta sovrapponendo questi tre effetti, detta global Masking Threshold o soglia d'udibilità dinamica, rappresenta la parte dell'informazione ininfluente, che può esser eliminata senza alterare troppo il suono (vedi figura 1.8).

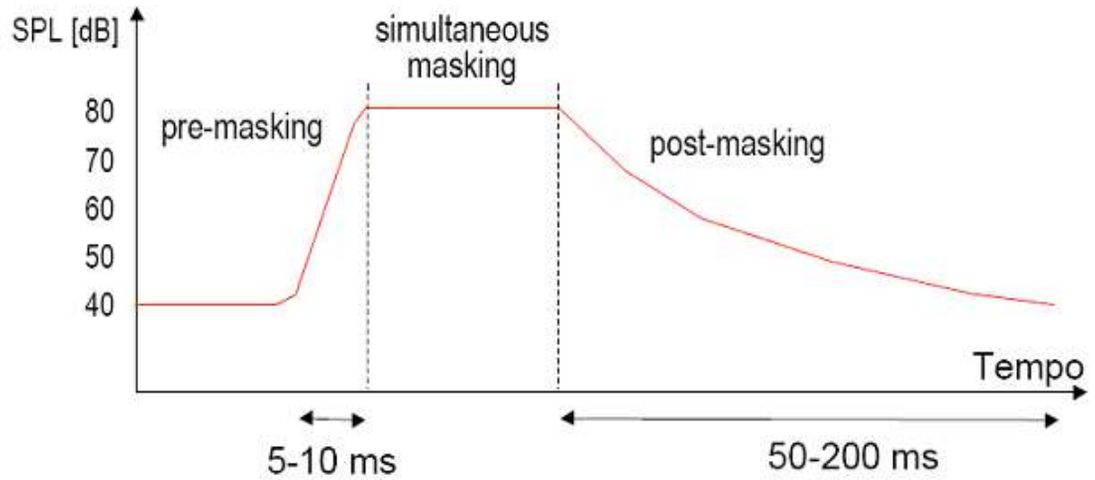

Figura1.8 (global Masking Threshold)

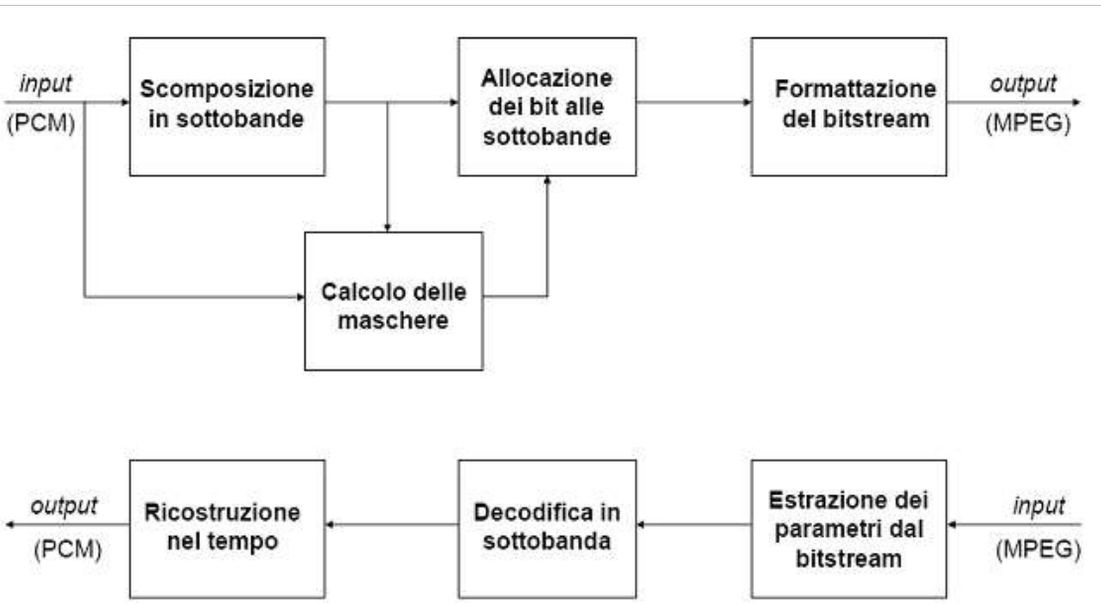

Figura1.9 (schema generale di codifica e decodifica audio MPEG)

Andando a fare una breve analisi del codificatore (figura 1.9), si vede come questo modello percettivo occupi un ruolo fondamentale nell'intero schema di codifica.Sarà infatti proprio tale blocco ad indicare quali componenti spettrali possono essere scartate.Il decodificatore non richiede, invece, un modello psicoacustico rendendolo quindi meno complesso.
Sia nell'MPEG-1 che nell'MPEG-2 troviamo tre differenti Layer, i quali rappresentano una famiglia di algoritmi codificanti.Adando dal layer I al III, la complessità cresce, ma anche il rapporto di compressione che si riesce a ottenere.In tutti i layers, la codifica è di tipo percettivo basata, come visto, sul modello psicoacustico del sistema uditivo.Viene quindi analizzato il segnale audio nel dominio della frequenza, in particolare suddividendolo tramite un banco di filtri in 32 sottobande, per poi andare a codificare solo quei contributi che il sistema uditivo è realmente in grado di percepire.
E' qui che interviene il modello piscoacustico, sarà lui a calcolare il livello di mascheratura in ogni banda causato dalle bande vicine.
Il blocco di quantizzazione decide quanti bit allocare ad ogni sottobanda in modo da mantenere il rumore di quantizzazione al di sotto del livello di mascheratura.
Il blocco finale formatta il flusso di bit in uscita.

- ***Layer I*** :

  l'algoritmo del Layer I codifica l'audio in trame di 384 campioni.Esso opera raggruppando insieme 12 campioni per ciascuna delle 32 sottobande (12*32=384).Si sottolinea ulteriormente, per non fare confusione, che sono campioni spettrali.Ogni gruppo di 12 campioni riceve un'allocazione dinamica dei bit di quantizzazione (per ogni campione), compresa tra 0 e 15 e un *fattore di scala.*Il fattore di scala è un termine moltiplicativo che serve a migliorare la quantizzazione e viene rappresentato su 6 bit.
  Tale metodo opererà in modo da ottenere la bit rate richiesta e da rispettare le imposizioni della Threshold Mask.
  Complessivamente il formato dei frame risulta essere (dimensioni in bit):

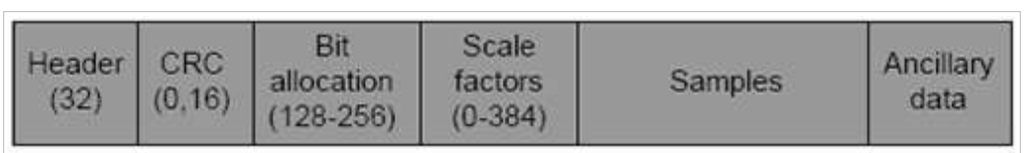

Figura1.10 (Frame)

E' importante sottolineare come sia nel layer I che nel II (a differenza del III) il modello psicoacustico non esegua alcuna operazione di pre/post mascheramento.Il decodificatore è molto più semplice del codificatore, lui si occuperà dell'operazione di dequantizzazione, tenendo conto del fattore di scala applicato al quantizzatore e dei bit allocati.
Vengono quindi ricostruite le sottobande e nel caso una di queste non abbia bit allocati viene trascurata.Per concludere si applica il blocco di sintesi e si ricostruisce il formato audio PCM.

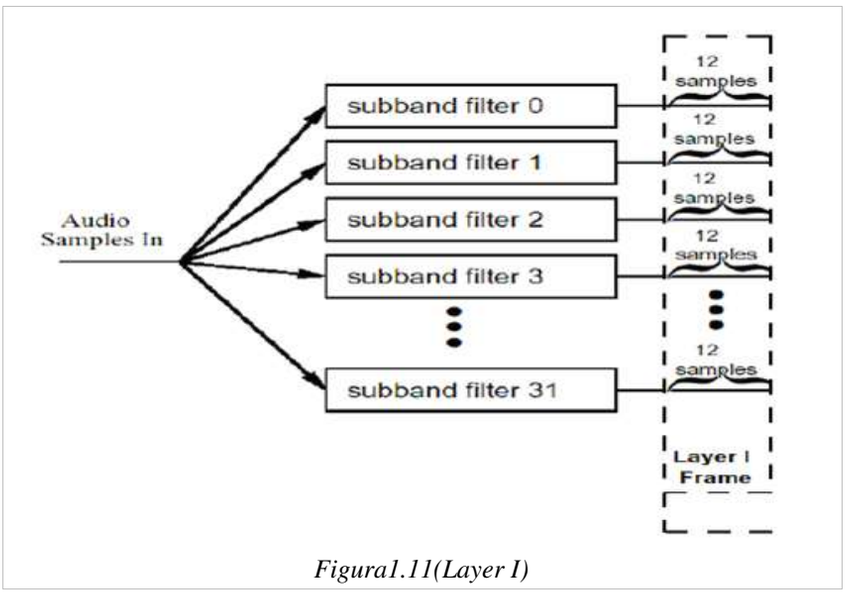

Figura1.11(Layer I)

- ***Layer II*** : l'algoritmo del Layer II è un arricchimento del Layer I.Esso codifica i dati audio in gruppi più grandi (finestramento ogni 36 campioni) ed impone alcune restrizioni sulla possibile allocazione di bit per i valori delle sottobande.Esso inoltre rappresenta l'allocazione dei bit, il fattore di scala ed i campioni quantizzati con un codice più compatto.Il layer II ottiene una migliore qualità audio risparmiando bit in queste aree per averne di più a disposizione per rappresentare i valori quantizzati delle sottobande.Il codificatore forma trame di 1152 campioni per canale audio e, a differenza del layer I, il Layer II codifica i dati in 3 gruppi di 12 campioni per ciascuna sottobanda.

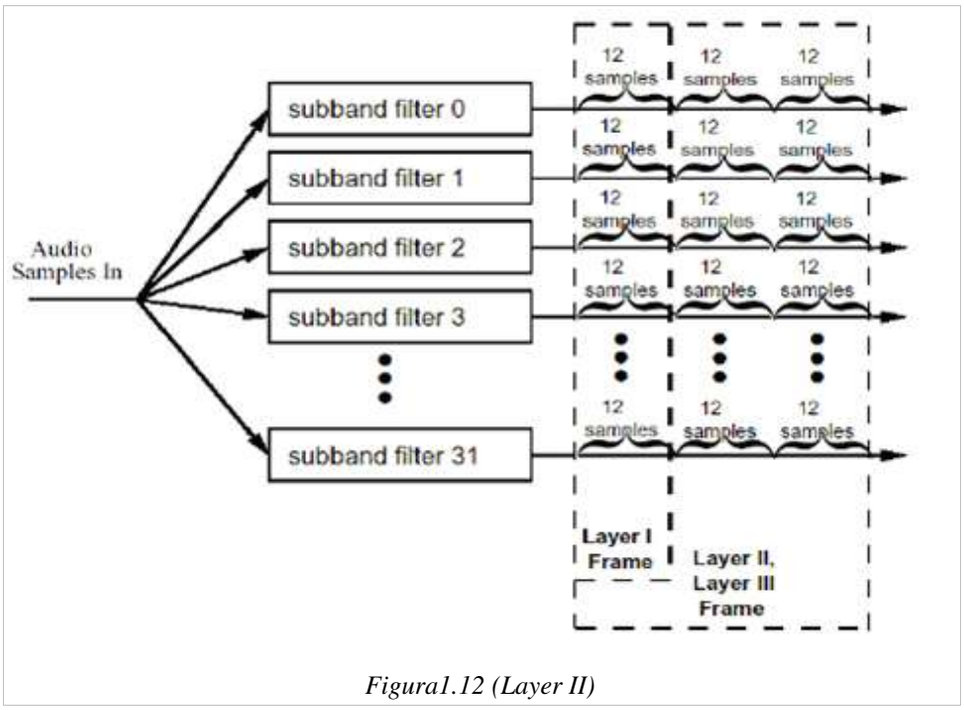

Figura1.12 (Layer II)

- ***Layer III*** :

Sebbene basato sullo stesso banco di filtri dei due layer inferiori, il Layer III compensa alcune deficienze del banco di filtri elaborando le uscite di questo con una Trasformata Coseno Discreta Modificata (MDCT).L`MDCT suddivide ulteriormente in frequenza le uscite delle sottobande per garantire migliore risoluzione spettrale.Inoltre, una volta che le componenti di sottobanda sono suddivise in frequenza, il codificatore può parzialmente cancellare parte dell'aliasing (sovrapposizione fra le sottobande) causato dal banco di filtri polifase.Varia dinamicamente la dimensione dei blocchi (12 o 36 campioni) a seconda che sia da privilegiare la risoluzione nel tempo (transitori) o in frequenza (segnali stazionari).Utilizza una quantizzazione più efficace (non uniforme) dei campioni e impiega una codifica più efficiente per i valori quantizzati, detta “codifica entropica”.Alloca il numero di bit ad ogni sottobanda con un algoritmo più sofisticato.Come già accenntao nella descrizione dei Layer I e II, il modello psicoacustico del terzo livello prevede anche una operazione di pre/post mascheramento.

Lo standard MPEG-1 produce un segnale audio con i seguenti tipi di canali: canale singolo, canale doppio o dual-channel (composto da due segnali indipendenti), stereo (composto da un segnale destro e uno sinistro portanti due segnali non indipendenti), joint stereo (composto come lo stereo ma con in aggiunta una elaborazione anti-ridondanza e anti-irrilevanza).
Lo standard MPEG-2 ha inserito la possibilità di avere segnali audio con 5 canali a banda intera: destro, sinistro, centrale, 2 canali surround (vedi figura 1.13) e opzionalmente un canale extra LFE (low frequenctìy enhancement) per i segnali di subwoofer, nel range da 15 a 120Hz.Inoltre una “Multilingual Extension” permette l’aggiunta di altri 7 canali audio per programma.

Figura1.13 (canali audio MPEG2)

MPEG-2 Audio prevede tanto algoritmi di codifica compatibili all’indietro con quelli definiti da MPEG-1, quanto algoritmi di codifica avanzati (Advanced Audio Coding).Tali algoritmi sono analoghi a quelli definiti in MPEG-1, ma con alcuni raffinamenti, principalmente riferibili ad un aumento della risoluzione in frequenza (MPEG-2 AAC opera una codifica a trasformata MDCT che conduce ad una risoluzione massima in frequenza di 1024 componenti spettrali), ad una

maggiore flessibilità di adattamento della finestra temporale alle caratteristiche del segnale ed ad una tecnica di adattatività temporale della quantizzazione (*Temporal Noise Shaping*).

Lo standard MPEG-2 Backward Compatible offre compatibilità all'indietro, nel senso che consente ad un codificatore MPEG-2 di generare un bit-stream decodificabile da un decodificatore MPEG-1, e lo standard Forward Compatible offre compatibilità in avanti, nel senso che consente ad un decodificatore MPEG-2 la decodifica di un bit-stream MPEG-1.

1.2.3 Codifica Video

I principi di codifica video sono sostanzialmente gli stessi adottati da MPEG-1, con qualche differenza relativa alle risoluzioni e ai rate massimi supportati e alcune rifiniture agli algoritmi di compressione (tali differenze veranno elencate in seguito).

Punto chiave dell'intero processo di codifica video MPEG è la distinzione delle immagini (frames) in tre tipi diversi (vedi fig 1.15):

- immagini codificate senza riferimento ad altre immagini (Intra-Coded Picture, **I);** forniscono punti di accesso alla sequenza codificata in corrispondenza dei quali può iniziare la decodifica; permettono inoltre la minimizzazione della propagazione di errori di trasmissione dovuti ad errori nei fotogrammi precedenti.Sono caratterizzate da un modesto rapporto di compressione.

- immagini codificate mediante motocompensazione del movimento da immagini precedenti, partendo da immagini di tipo I o P (*Predictive-Coded Picture,* **P**); utilizzate come riferimento per altre predizioni.

- immagini codificate mediante motocompensazone bidirezionale (*Bi-directional-Coded Picture*, **B**); offrono il maggior livello di compressione; non sono utilizzate come riferimento per ulteriori predizioni

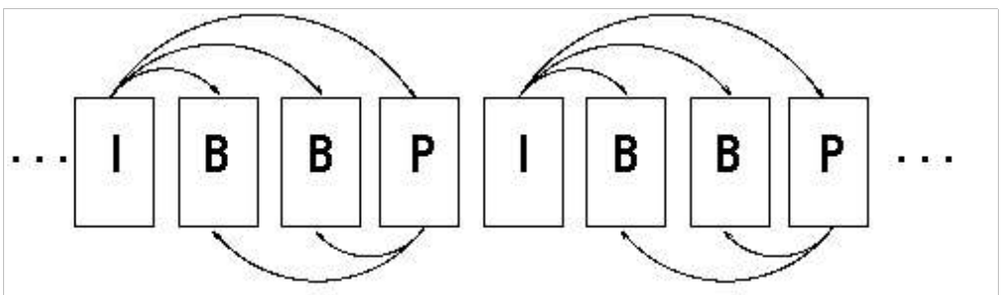

Figura1.14 (Iterazioni fra immagini di tipo I,P,B)

Prima di proseguire con l'analisi dello standard, risulta utile distinguere i vari elementi,detti *layers*,che andranno a formare l'intera sequenza video:

1. *Video Sequence Layer*: la vera e propria sequenza video codificata.

2. *Group of pictures (GOP) layer*: i sottoblocchi che vanno a formare la sequenza video.Ognuno di questi comprende un gruppo di immagini I,P,B consecutive, la prima immagine è comunque sempre di tipo I (vedi fig 1.15).Le distanze tra le stesse è configurabile in fase di codifica.Più è piccolo il GOP migliore è la risposta al movimento, ma più piccola la compressione (colpa delle immagini I).

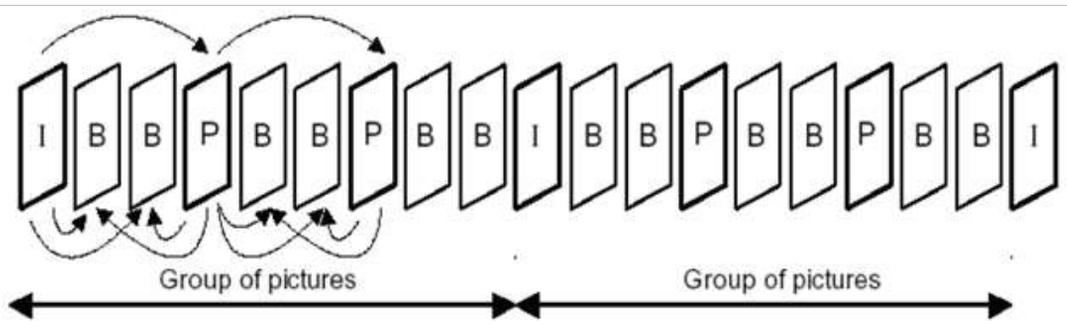

Figura1.15 (struttura GOP)

3. *Picture layer*: corrisponde ad una frame (fotogramma).

4. *Slice Layer*: le “fette” costituenti ogni singolo frame.A loro volta tali fette sono costituite da una sequenza contigua di macroblocchi disposti da sinistra verso destra e dall’alto verso il basso.

5. *Macroblock layer*: ad un macroblocco corrisponde un’area del frame di16x16 pixel.E’ a livello di macroblocco che viene fatta la compensazione di moto, per aumentare il fattore di compressione della sequenza finale.

6. *Block layer*: è un insieme di 8x8 pixel, sia di luminanza sia di crominanza; 4 blocchi formano un macroblocco.La trasformata DCT viene applicata a livello di blocco e si hanno quindi 64 coefficienti DCT da quantizzare per ogni blocco.

In figura 1.16 viene presentato un riassunto schematico dei vari layers.

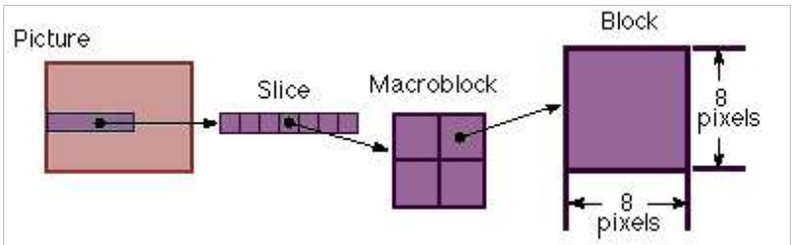

Figura1.16 (layers Mpeg)

Tutti i macroblocchi di un quadro I sono di tipo I, codificati senza predizione. Ciascun macroblocco di un quadro P può essere di tipo I, codificato senza predizione o di tipo P, predetto rispetto a un singolo macroblocco di un'immagine precedente.Ciascun macroblocco di un quadro B pu´o essere di tipo I, P o B.
In figura 1.17 sono rappresentati i principali passi di compressione a cui è sottoposto un macroblocco di tipo I.

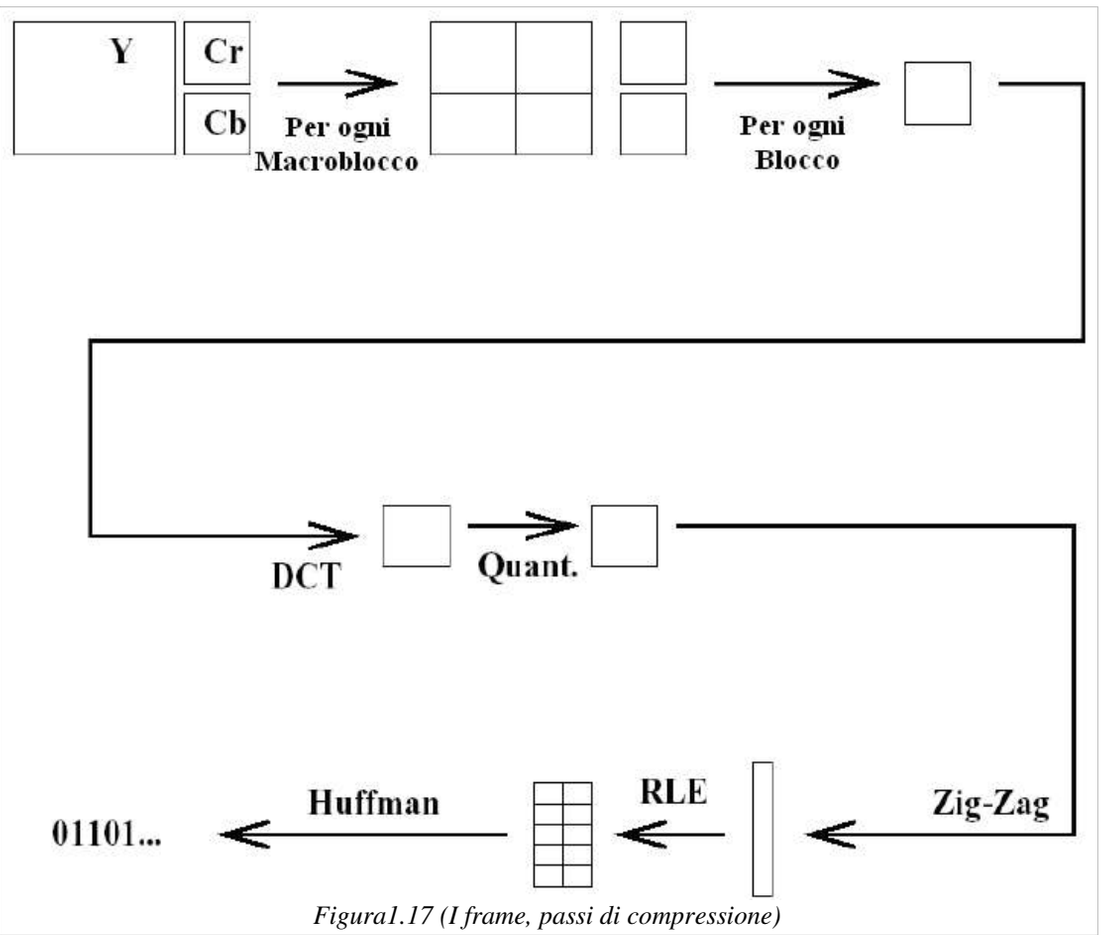

Figura1.17 (I frame, passi di compressione)

Come prima osservazione si fa notare che nella figura sovrastante, ad ogni macrobocco sono associati 4 blocchi 8x8 di luminanza Y e 2 blocchi 8x8 di crominanza Cr e Cb.L'esempio fa riferimento al formato 4:2:0, in cui i due blocchi di crominanza rappresentano sempre 16x16 pixel, ma sono sottocampionati.
Ogni singolo blocco viene processato mediante Trasformata Coseno Discreta (*DCT*).La seguente formula definisce la DCT fornendo l'espressione di un generico elemento *c(i,j)* della matrice 8×8 risultato della trasformazione di un blocco 8×8 composto dai campioni *b(x,y)* dell'immagine originale:

$$c(i,j) = \frac{K(i)}{2}\frac{K(j)}{2}\sum_{y=0}^{7}\sum_{x=0}^{7} b(x,y)\cos\left[(2x+1)\frac{i\pi}{16}\right]\cos\left[(2y+1)\frac{j\pi}{16}\right]$$

dove *i* e *j* sono gli indici delle frequenze orizzontale e verticale e le costanti *K(i)* e *K(j)* sono date da:

$$K(i) = \begin{cases} 1/\sqrt{2} & \text{se } i = 0 \\ 1 & \text{se } i > 0 \end{cases}$$

La formula inversa (chiamata *IDCT*) usata per la decodifica è:

$$b(x,y) = \sum_{i=0}^{7}\sum_{j=0}^{7}\frac{K(i)}{2}\frac{K(j)}{2}c(i,j)\cos\left[(2x+1)\frac{i\pi}{16}\right]\cos\left[(2y+1)\frac{j\pi}{16}\right]$$

dove le costanti *K(i)* e *K(j)* sono date da:

$$K(i) = \begin{cases} 1/\sqrt{2} & \text{se } i = 0 \\ 1 & \text{se } i > 0 \end{cases}$$

Fondamentalmente la DCT rappresenta il blocco dell'immagine nel dominio delle frequenze spaziali.Tale trasformata non fa altro che assegnare ad ognuna di queste frequenze un coeficiente, indicante il peso nel blocco di dati che è stato trasformato.Il coeficiente C(*0,0),* viene detto termine in continua o "DC value" a causa dell'analogia matematica con l'analisi in frequenza delle reti elettriche.Se ad esempio solo i coeficienti a bassa frequenza della DCT sono diversi da zero, ciò significa che i dati nel blocco variano poco con la posizione.Se invece sono

presenti anche coefficienti ad alte frequenze, i dati nel blocco variano bruscamente da una posizione all'altra.Nel caso di blocchi di pixel di un'immagine questo significa che, se sono presenti coefficienti ad alte frequenze spaziali, l'intensità dei pixel varia bruscamente tra pixel vicini, mentre se sono presenti solo coefficienti a basse frequenze spaziali i pixel vicini sono abbastanza simili tra loro.La DCT è stata scelta, tra tutte le possibili trasformate utilizzabili, per due motivi principali.Anzitutto, è stato dimostrato che i coefficienti della DCT sono relativamente scorrelati e questo rende possibile la costruzione di semplici algoritmi di compressione dati.In secondo luogo visto che la DCT offre una rappresentazione dei dati nel dominio delle frequenze spaziali, si può sfruttare la caratteristica dell'occhio umano, più sensibilie agli errori di ricostruzione dovuti alle basse frequenze spaziali che non a quelli dovuti a frequenze più elevate.Per tenere conto di questo fatto, i coefficienti della DCT vengono quantizzati diversamente a seconda della loro importanza percettiva per l'apparato visivo umano.Coefficienti poco importanti possono venire quantizzati molto di più senza introdurre una distorsione apprezzabile.MPEG usa quindi quella che viene chiamata quantizzazione pesata (*visually-weighted quantization*), definendo delle matrici di quantizzazione delle stesse dimensioni dei blocchi su cui si applica la DCT.Un coefficiente DCT viene quantizzato dividendolo per in un intero positivo e arrotondando all'intero più vicino il risultato (ciò per codifica Intra, mentre in caso di codifica Inter, l'arrotondamento è sempre all'intero inferiore).Maggiore è il valore di quantizzazione minore sarà la precisione del coefficiente quantizzato.La scelta delle matrici di quantizzazione è lasciata al codificatore.Tuttavia, a titolo di riferimento, lo standard suggerisce l'uso delle matrici QINTRA ,QINTER riportate in tabella1.3

INTRA								INTER							
8	16	19	22	26	27	29	34	16	16	16	16	16	16	16	16
16	16	22	24	27	29	34	37	16	16	16	16	16	16	16	16
19	22	26	27	29	34	34	38	16	16	16	16	16	16	16	16
22	22	26	27	29	34	37	40	16	16	16	16	16	16	16	16
22	26	27	29	32	35	40	48	16	16	16	16	16	16	16	16
26	27	29	32	35	40	48	58	16	16	16	16	16	16	16	16
46	27	29	34	38	46	56	69	16	16	16	16	16	16	16	16
27	29	35	38	46	56	69	83	16	16	16	16	16	16	16	16

Tabella1.3 (matrici di quantizzazione per frame INTRA e INTER)

Alla fine accadrà che buona parte dei coefficienti a più alta frequenza verranno azzerati.Come esempio pratico nelle seguenti figure viene riportato il processo di codifica applicato ad una immagine di esempio:

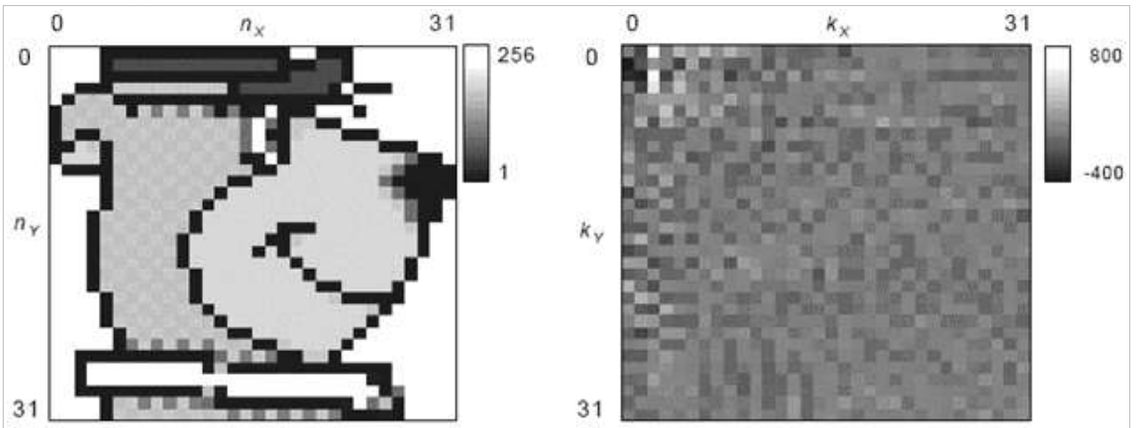

Figura1.18 (Destra:immagine da sottoporre a codifica.Sinistra:matrice dei coeficienti DCT)

La figura 1.18 di sinistra rappresenta l'immagine da sottoporre a codifica,mentre quella di destra la matrice dei coeficienti della DCT.

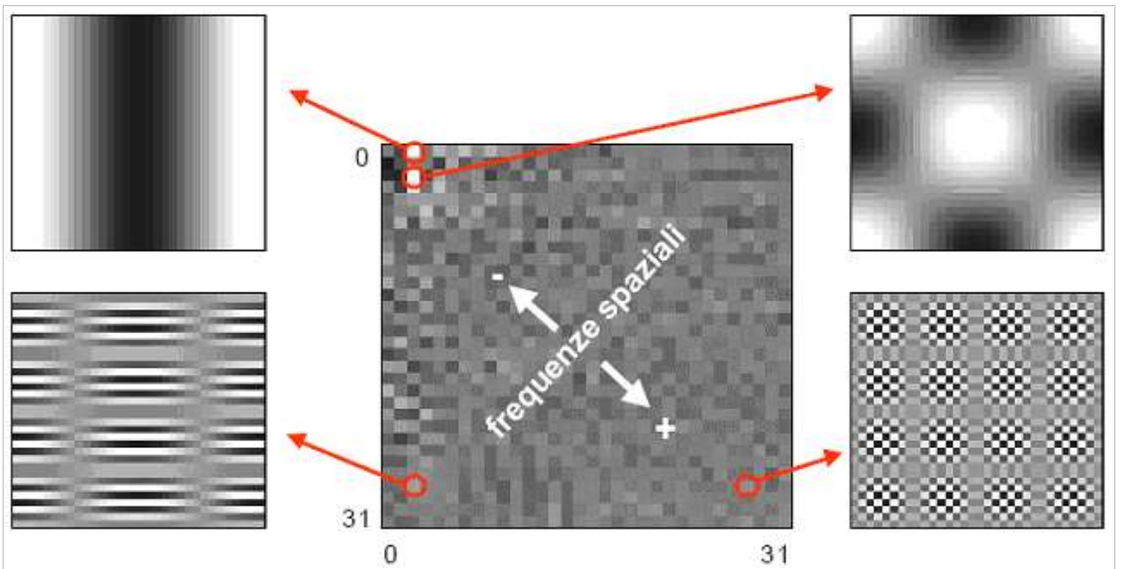

Figura1.19 (interpretazione dei coeficienti DCT)

Dalla figura 1.19 si può avere una più chiara comprensione del significato di tali coefficienti: come si nota i coefficienti a frequenze spaziali minori rappresentano funzioni con modesta variazione di intensità fra pixel vicini, viceversa per quelli a frequenze spaziali maggiori.Come già detto il processo di quantizzazione pesata sfrutta la caratteristica dell'occhio umano, meno sensibile ad eventuali errori di

ricostruzione sui coefficienti ad alta frequenza.In figura 1.20 si riporta il risultato finale ottenuto da applicazione della DCT con quantizzazioni a diverse pesature e trasformata inversa DCT su un'immagine di esempio.

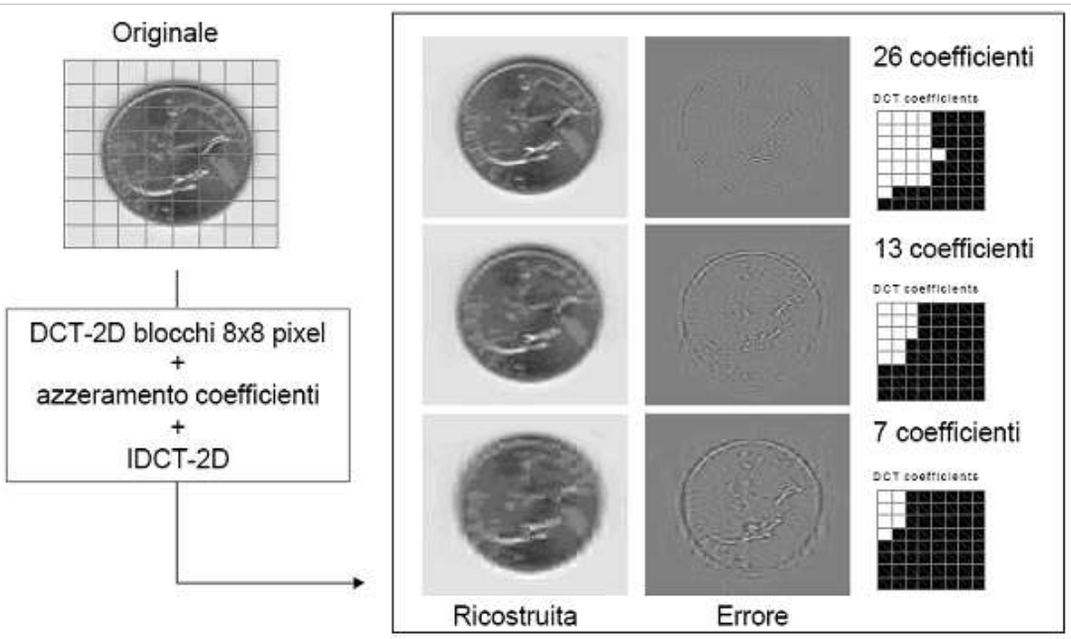

Figura1.20 (risultato codifica DCT con quantizzazione a diverse pesature)

Si nota come l'errore dovuto ad azzeramento (conseguenza della quantizzazione) dei 64-26 coefficienti DCT a più alta frequenza sia molto modesto e come comunque questo aumenti all'aumentare del numero di coefficienti azzerati.

I 64 coefficienti ottenuti dalla DCT, che sono in numero pari a quello dei punti del blocco, sono trasmessi nel bitstream in un ordine ben preciso, che tende ad anteporre i coefficienti di più bassa frequenza a quelli di frequenza via via più alta.Il procedimento, noto come "*zig-zag scan*", è mostrato in figura 1.21.Il punto di partenza è il coefficiente (0,0) della matrice.

Il percorso di sinistra è quello utilizzato nel caso di codifica di video in formato progressivo, quello di destra è invece utilizzato nel caso di video in formato interlacciato.

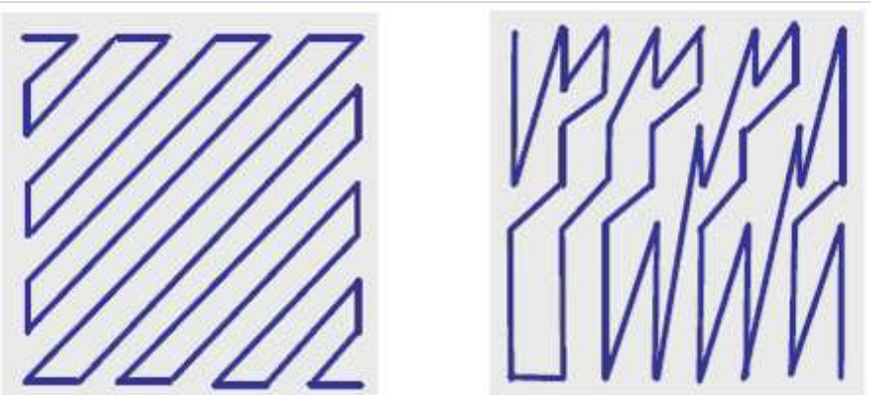

Figura1.21 (linearizzazione dei coefficienti DCT tramite "zig-zag scan")

Risulta a questo punto utile codificare in modo opportuno i coefficienti trovati, cercando cioè di sfruttare la presenza di lunghe sequenze di zeri.

La *RunLenghtEncoding* è la migliore tecnica di codifica per trattare tali tipi di dati. Essa prevede la rappresentazione del vettore tramite coppie (skip, value), dove skip è il numero di valori uguali a zero e value è il successivo valore diverso da zero.La coppia (0,0) viene considerata come segnale di fine sequenza (End Of Block, EOB).Da notare come la scansione a zig zag e la successiva codifica RLE sia utilizzata solo per i coeficienti AC, mentre per i termini in continua,vista la non marcata variazione fra quelli appartenenti a frame vicini, si applica la codifica differenziale (DPCM).Questo sistema modifica il coefficiente DC dal valore assoluto ad un valore relativo al coefficiente DC del blocco precedente.Ad esempio, se i coefficienti DC di due blocchi adiacenti sono 227 e 205, il secondo diventerà -22.I blocchi adiacenti solitamente hanno un elevato grado di correlazione, così la codifica del coefficiente DC come differenza dal coefficiente precedente tipicamente produce un numero piccolo.In figura 1.22 e 1.23 viene riportato un esempio pratico.

Il valore 168 è il coeficiente DC, quindi non c'è bisogno di codificarlo, mentre EOB è il simbolo indicante la fine del blocco,come definito dallo standard MPEG.

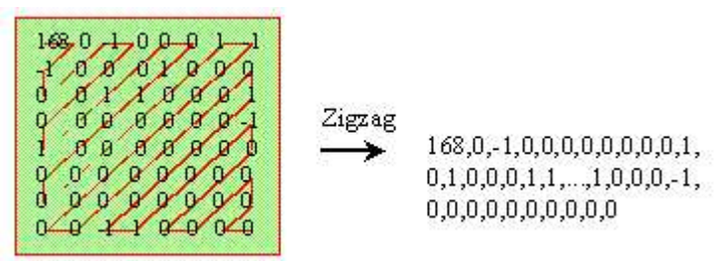

Figura1.22 (scansione a zig zag dei coeficienti DCT)

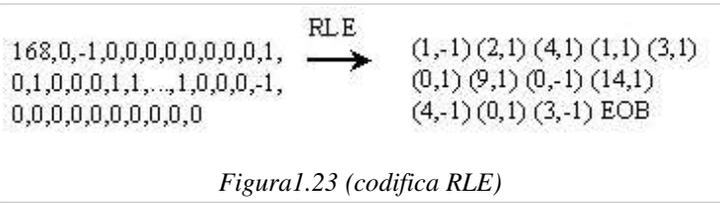

Figura1.23 (codifica RLE)

L'ultima codifica entropica applicata ai dati è la classica codifica a lunghezza di codice variabile.In pratica i dati vengono suddivisi in 'parole' (stringhe di bit), viene analizzata la frequenza statistica di ciascuna parola e ognuna viene ricodificata con un codice a lunghezza variabile in funzione della frequenza di apparizione.Un codice corto per le parole che appaiono frequentemente e via via codici più lunghi per quelle meno frequenti.Complessivamente il numero di bit necessari per rappresentare i dati si riduce consistentemente.
Per capire come funziona, utilizziamo un semplice esempio basato su un breve testo:

sopra_la_panca_la_capra_campa._sotto_la_panca_la_capra_crepa.

E' costituito da un totale di 61 caratteri in cui si individuano 13 simboli differenti (comprendendo fra i simboli anche lo spazio e il punto).Per codificare ciascun simbolo in binario si potrebbero utilizzare parole costituite da 4 bit (con quattro bit le configurazioni possibili sono 24 cioè 16) e quindi l'intera stringa sarebbe rappresentabile da 244 bit in totale.La codifica secondo Huffman permette di ridurre il numero di bit totali associando a ciascun simbolo una parola binaria di lunghezza non fissa, tale per cui ai simboli più probabili vengano associate parole più corte, ai simboli meno probabili parole più lunghe.L'algoritmo funziona in questo modo:

- Analizza il numero di ricorrenze di ciascun simbolo (vedi tabella 1.2, nell'esempio *a* ricorre 16 volte, lo spazio _ 11 volte, mentre *m* ed *e* compaiono una sola volta)
- Accomuna i due elementi meno frequenti in un sottoinsieme somma (nell'esempio *m* ed *e*) e li distingue associando, ad esempio, *0* ad *m* e *1* ad *e*.
- Ripete iterativamente il processo con i due sottoinsiemi meno frequenti, usando lo stesso procedimento e considerando un tutt'unico il sottoinsieme costituito da *m* ed *e*, caratterizzato da una probabilità di occorrenza pari alla

somma delle probabilità associate ai singoli *m* ed *e*.

- Si crea così un albero (figura 1.24) costituito da una serie di ramificazioni binarie, in cui le foglie costituite dai simboli più rari sono più lontane dalla radice e sono identificate da un codice binario più lungo.
- seguendo il percorso dalla radice fino alla foglia, si determina il codice assegnato a ciascun simbolo (tabella 1.2).

simbolo	numero di occorrenze	probabilità di occorrenza	parola di codice
a	16	0,262	10
_	11	0,180	00
p	7	0,115	010
c	6	0,098	1111
l	4	0,066	0111
r	4	0,066	0110
o	3	0,049	11101
n	2	0,033	11001
s	2	0,033	11000
t	2	0,033	11011
.	2	0,033	11100
e	1	0,016	110101
m	1	0,016	110100

Tabella1.2

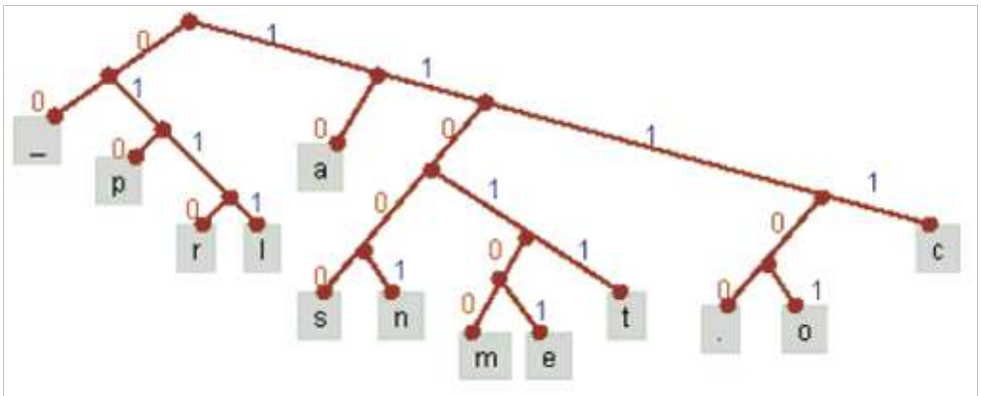

Figura1.24 (albero di Huffman)

Il decodificatore, seguendo il percorso indicato dai bit che esamina in sequenza

è in grado di individuare univocamente la foglia, ovvero il simbolo relativo a ciascuna parola, anche se le parole sono a lunghezza variabile (VLC, *Variable Lenght Code*).
Tornando all'esempio, la stringa di caratteri "*sopra*" diventa, codificata in binario 1000111010100110 10.Nel caso di questa stringa, un codice a lunghezza fissa, assegnando 4 bit a simbolo, avrebbe richiesto 20 bit, mentre con la codifica VLC ne sono sufficienti 19.Un guadagno significativo si ha nel caso della stringa "*_la_*", che viene codifi cata con 0001111000 ovvero con, mediamente, 2,5 bit per carattere.Nel complesso, la frase richiede 198 bit, anziché i 244 richiesti da una codifica a lunghezza fissa, con un risparmio dell'ordine del 19% e senza perdita di informazione (codifica *lossless*).

Le tecniche analizzate sino ad ora sono analoghe a quelle usate da JPEG e sono sostanzialmente tecniche psico-visive di riduzione della ridondanza spaziale, si eliminano cioè le informazioni non visibili o meno importanti per l'occhio umano che come visto sono quelle a più alta frequenza spaziale.
In una sequenza video c'è però anche molta ridondanza di informazione "temporale" dovuta al fatto che gli elementi visivi si trovano ripetuti (in quiete e/o in movimento) su fotogrammi successivi.E' quindi necessario ridurre questo tipo di ridondanza per comprimere ulteriormente il filmato.Il modo più semplice per recuperare informazione dal fotogramma precedente è quello di codificare solo il segnale differenza tra il macroblocco in analisi ed il corrispondente del fotogramma precedente (***motion compensation***).Siccome c'e' elevata probabilità che nel passaggio da un fotogramma ad un altro ci sia una certa coerenza locale tra le due immagini, il segnale differenza avrà certamente minore contenuto informativo del macroblocco considerato a se stante.Il segnale differenza può quindi essere successivamente compresso con la stessa tecnica usata per i blocchi Intra.Tale tecnica però perde di efficacia nel momento in cui si presentano, tra frame successivi, translazioni dei blocchi.Si intuisce comunque che nonostante la suddetta traslazione, parte dell'informazione da codificare per l'attuale fotogramma la si può già trovare in quello precedente anche se non con una esatta corrispondenza spaziale fra i vari blocchi.Mpeg provvede al suo recupero mediante l'utilizzo della ***motion estimation***.Invece di limitarsi al confronto del blocco in esame con quello corrispondente nel frame seguente,

viene effettuata una ricerca entro un'area prestabilita.Il vantaggio risulta subito evidente: se il blocco è traslato di poco, tramite il "vettore spostamento" si può mettere la posizione relativa, evitando di ricodificare l'intero blocco.La ricerca dei Motion Vector può estendersi massimo in una area da +15 a –15 pixel in orizzontale ed in verticale nel fotogramma precedente.Trovata l'area che minimizza il segnale differenza con il blocco in analisi, si trasmette al decodificatore solo il segnale differenza trasformato è quantizzato (ora però la matrice di quantizzazione usata da MPEG,riportata nella tabella1.3, ha tutti i coeficienti pari a 16, infatti si sta operando su blocchi ottenuti come differenza, i quali contengono dati già altamente scorrelati) e il vettore di Motion Estimation.Il Decoder è quindi in grado di ricostruire il blocco sapendo quale area andare a prendere del fotogramma precedente (gli viene indicata dal motion vector) ed aggiungendovi il segnale differenza trasmessogli dal codificatore.Le figure 1.25 e 1.26 riassumono schematicamente questo processo.

Resta da sottolineare come MPEG definisca solo la tecnica di decodifica non quella di codifica.Quindi la scelta dell'algoritmo per la misura del motion vector è lasciata al coder designer.Si capisce come questo sia un campo nel quale si possono trovare considerevoli differenze di prestazioni fra differenti algoritmi ed implementazioni.A prima vista sembrebbe utile avere un'area di ricerca la più ampia possibile per poter coprire qualsiasi traslazione fra frame successivi.Tuttavia, l'incremento dell'area di ricerca comporta un'appesantimento computazionale del processo di ricerca, servirà dunque mediare fra queste due necessità.

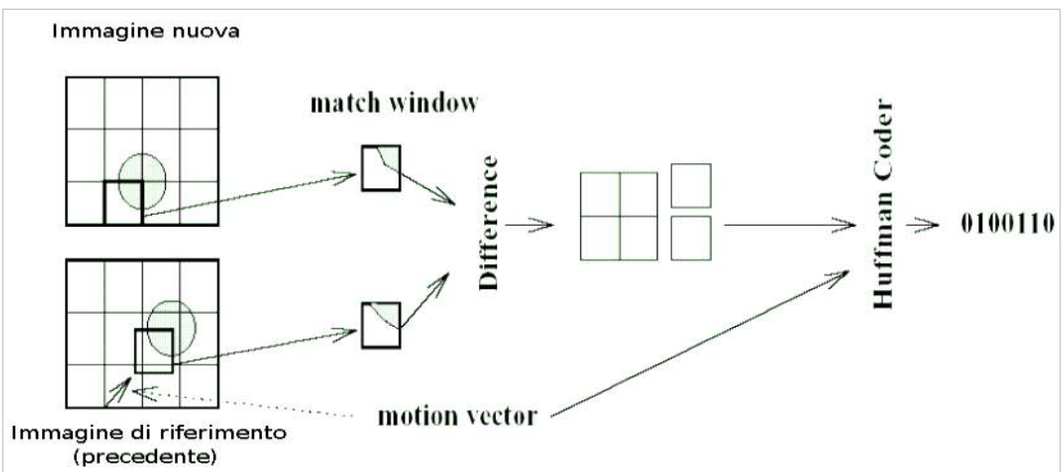

Figura1.25 (motion estimation P frame)

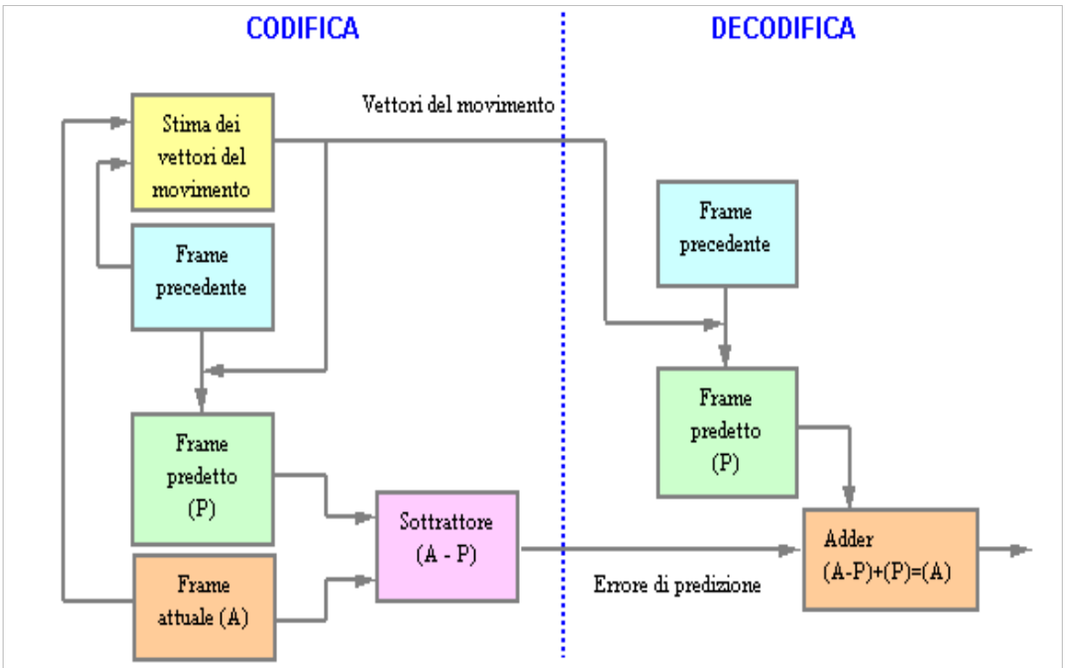

Figura1.26 (motion compensation codifica/decodifica)

L'ultimo passo è l'elaborazione dei B-Frame, compressi allo stesso modo dei P-Frame, verificando però le differenze con il frame precedente e quello successivo.Il livello di compressione dei B-Frame è superiore a quello dei P-Frame.

Si esaminano i macroblocchi dei fotogrammi compresi fra il frame I e quello P cercando per ogni blocco quello a lui piu' simile nel frame I (quindi indietro nel tempo), quello piu' simile nel frame P (quindi avanti nel tempo) oppure cercando di fare una media fra il blocco piu' simile nel frame I e quello piu' simile nel frame P e sottraendo a questa il blocco da codificare (vedi figura1.27).

Il codificatore, per evitare di fare una predizione all'indietro da un frame futuro (cioè non ancora ricevuto), riordina le immagini così che queste verranno trasmesse in un ordine diverso da quello effettivo di visualizzazione(vedi fig1.28).Si intuisce che questo processo come quello di riordino dei frames per correggere la sequenza di visualizzazione, introduce un considerevole end-to-end processing-delay,il quale potrebbe creare problemi ad alcune applicazioni.

In tal caso comunque MPEG definisce un profilo nel quale non è prevista l'uso di B frames.

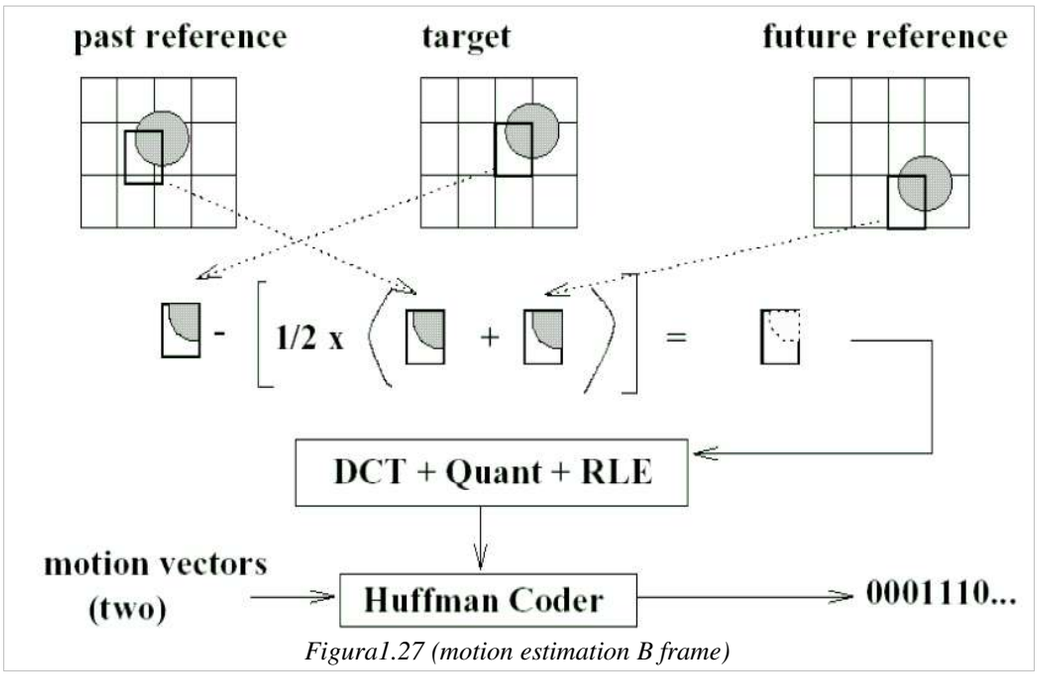

Figura1.27 (motion estimation B frame)

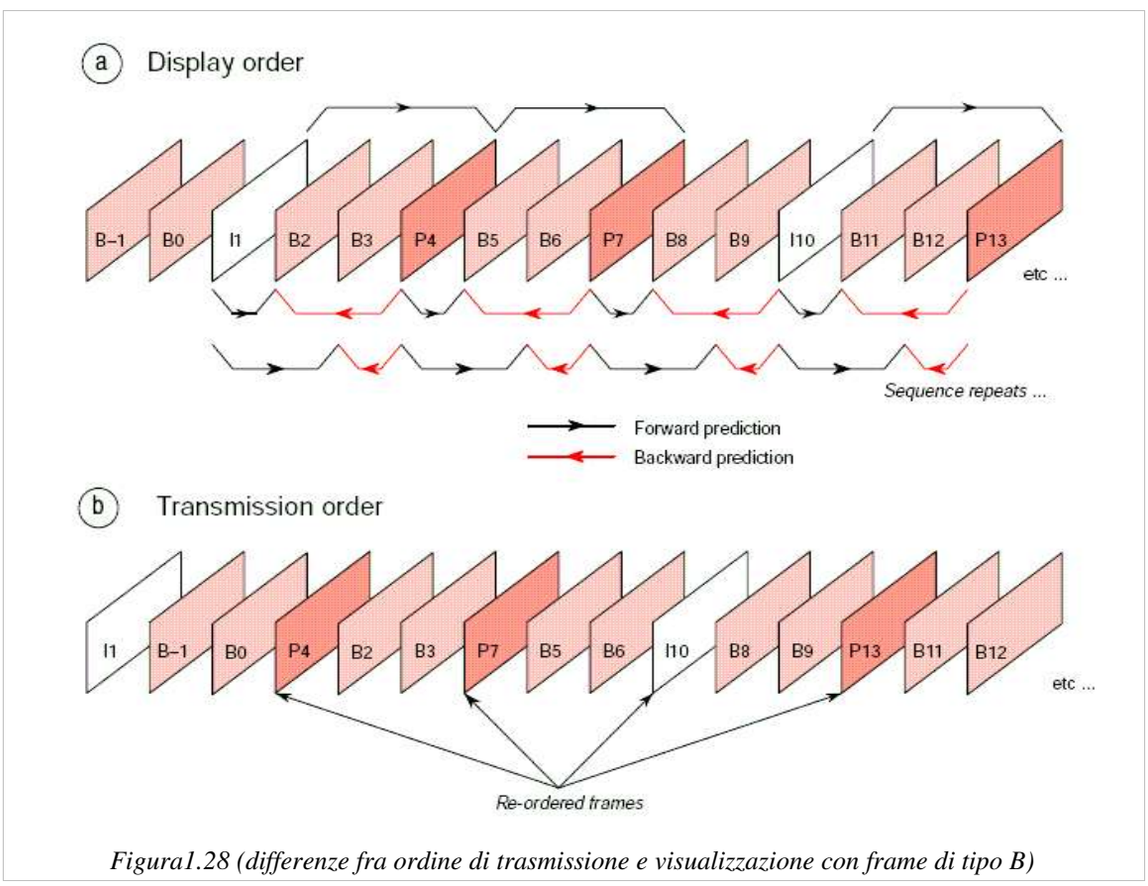

Figura1.28 (differenze fra ordine di trasmissione e visualizzazione con frame di tipo B)

Diversamente dall'MPEG-1, destinato alla compressione dati per supporti digitali, l'MPEG-2 è adattabile ad una vasta gamma di possibili applicazioni e perciò sono stati definiti nello standard cinque distinti “profili”, che pongono alcune restrizioni sulla sintassi della sequenza video.Come già accennato, uno di questi prevede ad esempio la non trasmissione di frames di tipo B.

I cinque profili sono: Simple (SP), Main (MP), SNR scalable (SNR), Spatially scalable (SPT), High (HP).

Nell'ambito di un singolo profile, possono essere poi definiti uno o più level (livelli di definizione dell'immagine) in termini di numero di pixel per linea, numero di linee per frame e numero di frame al secondo.

Sono stati definiti quattro livelli: Low (LL), Main (ML), High-1440 (H-14) e High (HL).Ogni profilo può essere combinato con uno o più di questi livelli.Ad esempio, la combinazione di Main Profile e Main Level è detta “Main profile at Main Level”' e si indica con MP@ML.

La tabella1.4 riporta uno schema riassuntivo di tutte le possibili combinazioni profile/level.

	Profile and maximum total bit-rate (Mbit/s)					
	Maximum sampling density *(Hor/Vert/Freq)*	Simple profile (SP)	Main profile (MP)	SNR profile (scalable)	Spacial profile (scalable)	High profile (HP)
Level	High level (HL) *(1920/1152/60)*	–	MP@HL *80 Mbit/s*	–	–	HP@HL *100 Mbit/s + lower layers*
	High–1440 *(1440/1152/60)*	–	MP@H–14 *60 Mbit/s*	–	Spt@H–14 *60 Mbit/s + lower layers*	HP@H–14 *80 Mbit/s + lower layers*
	Main level (ML) *(720/576/30)*	SP@ML *15 Mbit/s*	MP@ML *15 Mbit/s*	SNR@ML *15 Mbit/s + lower layers*	–	HP@ML *20 Mbit/s + lower layers*
	Low level (LL) *(352/280/30)*	–	MP@LL *4 Mbit/s*	SNR@LL *4 Mbit/s*	–	–
	ISO 11172 (MPEG-1) *1.856 Mbit/s*	–	–	–	–	–

Tabella1.4 (MPEG2,profiles/levels)

1.3 Flussi MPEG2

Sino ad ora sono state presentate le varie tecniche di codifica applicate a singoli stream audio e video, ma il flusso dati principale di MPEG2, può trasportare simultanemente molti programmi o servizi, con audio, video e dati tutti interallacciati fra di loro.Un decoder deve essere in grado di riconoscere quali sottoflussi appartengono ad un determinato programma o servizio per poterli poi riassemblare insieme.Inoltre deve sapere quando presentare il tutto all'utente finale.A tali compiti adempirà *MPEG2 system layer*.

Il System layer specifica la struttura dello stream di trasporto e il meccanismo di trasmissione dei dati compressi, inoltre provvede alla creazione delle *Program Specific Information Tables (PSI)***,** le quali rappresentano delle tabelle di contenuti, utili al decoder per poter rapidamente accedere ai vari dati da decodificare.

1.3.1 Transport Stream e gerarchia dati (ES, PES, PID Stream)

Il sistema transport è simile a quello IP, nel senso che lo stream MPEG2 viene anch'esso suddiviso in pacchetti di trasporto (transport packets), ognuno di questi provvisto di header e payload.

I dati prodotti dai codificatori audio e video sono organizzati in un flusso di unità di lunghezza variabile (pacchetti) denominato Elementary Stream (ES).A partire da ciascun ES, l'MPEG-2 System definisce la generazione del Packetized Elementary Stream (PES), ottenuto mappando ciascun pacchetto ES nel payload di un pacchetto PES e associandovi un PES Header contenente un prefisso, un numero identificativo dello stream (video, audio1, audio2, etc.) e dei dati opzionali di copyright, CRC, etc(vedi figura1.29).

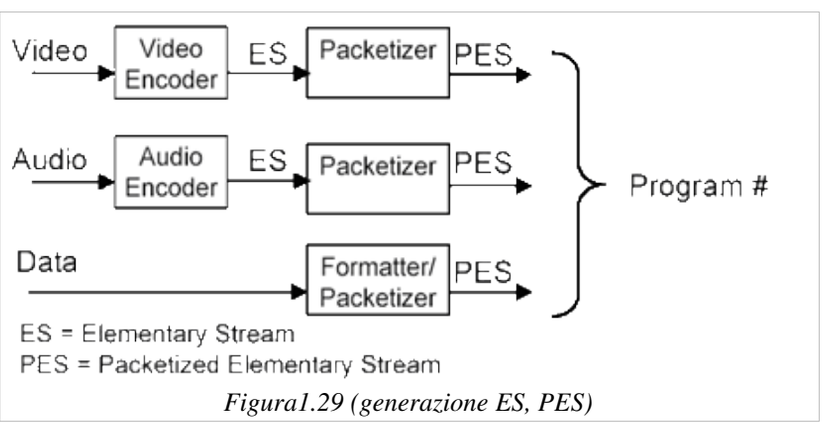

Figura1.29 (generazione ES, PES)

La sintassi del PES trasporta anche l'informazione necessaria per la sincronizzazione della decodifica e presentazione delle informazioni audio, video e dati, mediante indicatori temporali (Time Stamp), misurati rispetto ad un clock di riferimento a 90 Khz.La lunghezza di ogni pacchetto PES è variabile e può essere anche superiore a 64 kBytes.

MPEG2 prevede la creazione di due differenti stream di trasporto.

Il primo è il *Program Stream (PS)* che assume l'uso di un canale di trasmissione privo di errori ed è formato dai vari pacchetti PES.

Il secondo è il Transport Stream, ove i pacchetti PES vengono ulteriormente divisi in sottopacchetti di lunghezza fissa (188bytes), denominati *transport packets.*Il TS assume la presenza di un canale di trasmissione soggetto ad errori, questo il motivo per l'uso di pacchetti più corti e a lunghezza fissa.

Inizialmente si scelse la lunghezza di 188bytes per semplificare il mappaggio dei pacchetti MPEG2 sui canali di trasmissione ATM, i quali usano celle con payload di 47 bytes (47x4=188).

Ciascun transport packet contiene un campo header di lunghezza variabile, ma con valore minimo di 4 byte, ed un campo dati.

La figura1.30 riassume le iterazioni tra PS, TS e PES.

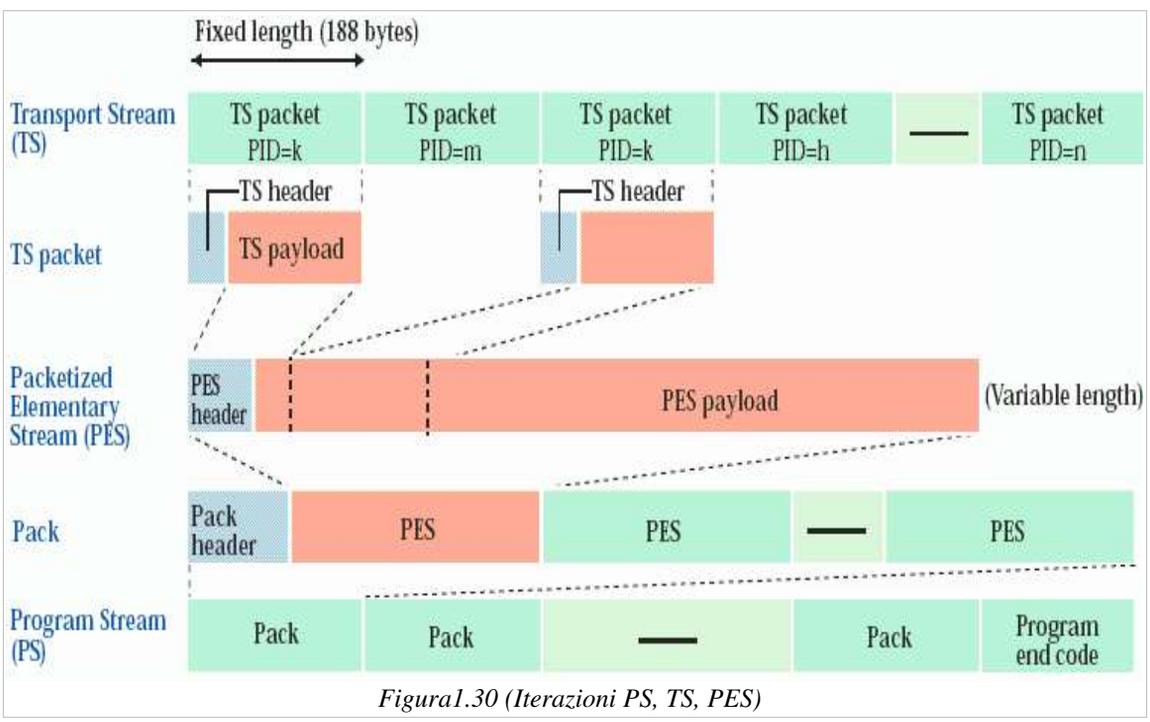

Figura1.30 (Iterazioni PS, TS, PES)

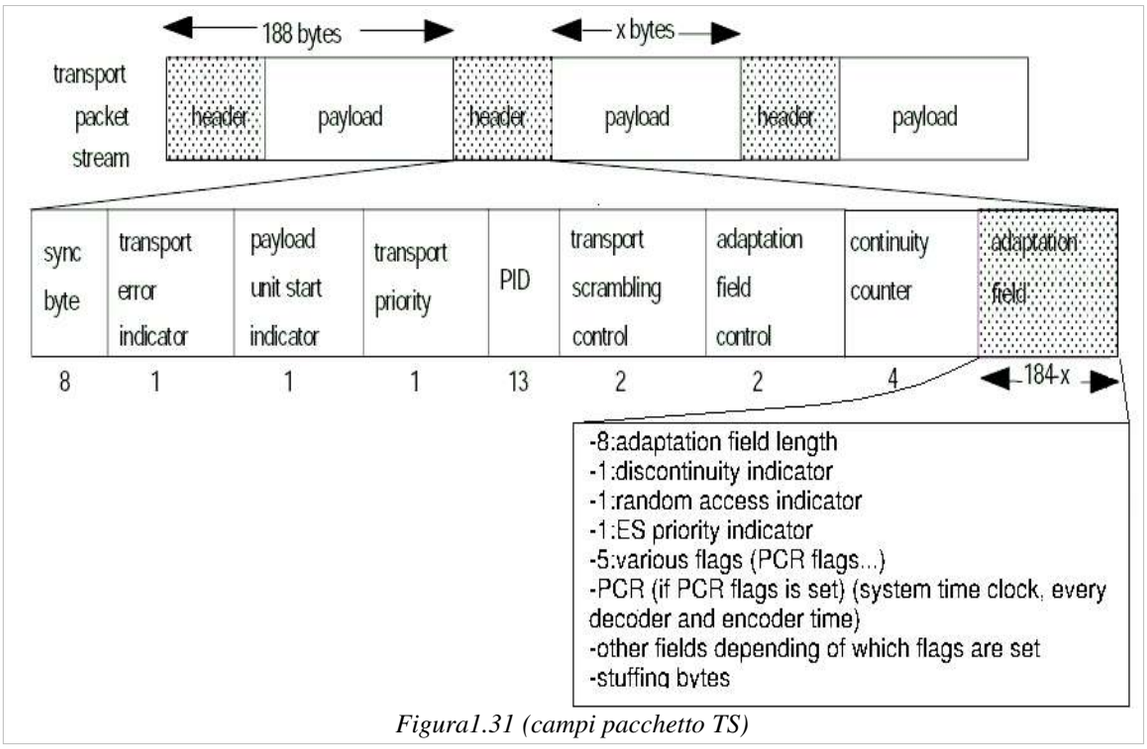

Figura1.31 (campi pacchetto TS)

Il TS Packet Header (figura1.31) contiene sempre un primo byte di sincronizzazione, un flag di errore dallo strato di trasporto, ed un campo di 13 bit detto Packet Identifier , o PID.Di seguito viene fornita una breve descrizione della funzione di ogni campo.

- *Sync byte*: 8 bit fissi all'esadecimale 47 (bin 1000 1111).Definisce l'inizio di un transport packet e permette la sincronizzazione della trasmissione.
- *Transport Error Indicator*: se settato a uno indica che il pacchetto è danneggiato.
- *Payload Unit Start Indicator*: settato a uno quando il pacchetto TS contiene l'inizio di un PES.
- *Transport Priority*: indicatore di priorità per dare priorità a certi pacchetti di un ES.
- *Transport Scrambling Control*: tipo di scrambling (cifratura).
- *Adaptation Field Control*: indica la presenza del campo opzionale "Adaptation Field" .

- *Continuity Counter*: contatore di continuità tra le sezioni PES.Questo contatore è un campo di 4 bit che viene incrementato in maniera ciclica ogni volta che un nuovo Transport Stream Packet dello stesso Elementary Stream viene generato; ciò dà al decoder un modo per rilevare se sono stati persi dei Transport Stream Packet.
- *PID (Packet Identifier):* 13 bit.E' un sistema di mappatura che differenzia i pacchetti e raggruppando quelli con lo stesso PID identifica quali trasportano un determinato flusso di informazioni, ad esempio il video di un certo programma , o l'audio in una certa lingua.A partire dal PID di un pacchetto è inoltre possibile ricostruire il corretto flusso di appartenenza, mediante un meccanismo di mappatura denominato Programm Specific Information, PSI.Il paragrafo successivo ne dà una dettagliata descrizione.
- *Adaptation field* : E' un estensione dell'header non sempre presente in ogni transport packet che contiene a sua volta diversi campi: tra cui l'indicatore di temporizzazione PCR (di seguito una breve spiegazione sul suo utilizzo) , un indicatore di discontinuità, i byte di riempimento (stuffing bytes) a volte necessari per completare i pacchetti ed altri flag.

Per quanto riguarda il PCR, acronimo di Program Clock Reference, la sua funzione è quella di fornire un riferimento per la rigenerazione del clock di sistema (System Time Clock a 27 Mhz) al ricevitore.
Il PCR svolge tale funzione in moda da permettere la ricostruzione di una base temporale coerente per i programmi.E' un campo di 42bit presente nell'Adaptation field di pacchetti TS dedicati, oppure in quelli di un PID video.
Il valore rappresentato da 9 bit del PCR è incrementato a 27Mhz, gli altri 33 a 90Khz.Al ricevitore un VCO (Voltage vontrolled oscillator) genera il clock locale a 27Mhz.Quando viene ricevuta la PCR, viene confrontata con un contatore locale pilotato dal VCO e la differenza viene usata per correggere la frequenza del VCO al fine di ottenere il lock con la PCR e quindi con il clock remoto(vedi figura 1.32).

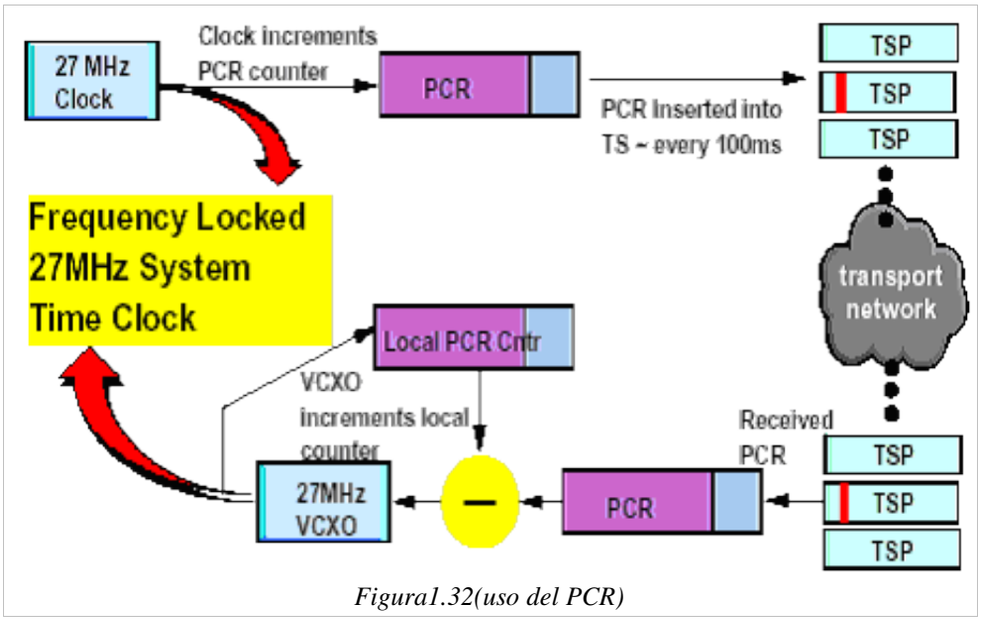

Figura1.32(uso del PCR)

1.3.2 Program Specific Information (PSI)

Come detto, le tabelle delle Program Specific Informations (PSI), operano come tabelle descrittive dei contenuti del transport stream.Contengono cioè le informazioni che servono per descrivere e associare attraverso i PID, i vari PES e di conseguenza i flussi elementari che vanno a costituire un programma. Strutturalmente sono delle tabelle di associazione (PSI Association Tables) che vengono inviate periodicamente dopo essere state suddivide in sezioni (PSI Sections) e pacchettizzate nel TS allo stesso modo dei PES.Per l'importanza dell'integrità delle informazioni delle tabelle, spesso le PSI Sections sono protette da CRC.

Il PSI è composto dalle seguenti tabelle:

- Program Association Table (PAT);
- Program Map Table (PMT);
- Conditional Access Table (CAT);
- Network Information Table (NIT);

▪ PAT:

La Program Association Table (PAT) (figure 1.32) è il primo passaggio obbligato del decoder quando cerca di localizzare un programma.

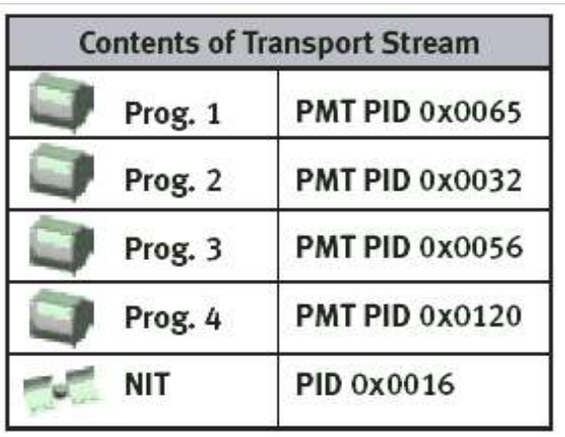

Figura1.32 (PAT)

La PAT, rapidamente rintracciabile perchè identificata dal PID 0x0000, comunica al decoder dove cercare la "mappa" per ognuno dei programmi contenuti nel

transport stream (identificati attraverso un numero detto SID, Service IDentification).Tale mappa è contenuta nel Program Map Table (PMT).
La PAT comunica al decoder il valore PID dei pacchetti contenenti la PMT del programma.
La PAT può anche fornire il PID dei pacchetti relativi alla Network Information Table (NIT), che da indicazioni sugli altri transport stream presenti nella rete.

■ PMT:

Ogni Program Map Table (PMT) (figura 1.33) mappa letteralmente tutte le informazioni necessarie alla decodifica di un determinato programma.Questa tabella lista infatti i PID dei pacchetti contenenti le componenti video, audio e dati relative al programma scelto.
Il decoder una volta consultata la PMT può quindi facilmente localizzare, decodificare e visualizzare i contenuti dei programmi.
La PMT indica anche il PID relativo al "*program's Entitlement Control Message* (ECM)".L'ECM fornisce al decoder le chiavi necessarie al "descrambling" del contenuto audio e video dei programmi sottoposti a criptatura.

Components of Program 1	
Video	PID 0x0131
Audio English	PID 0x0132
Audio German	PID 0x0133
ECM Prog. 1	PID 0x0150

Figura1.33 (PMT)

■ CAT:

Lo standard MPEG2 permette ai broadcaster di trasmettere nel transport stream delle informazioni di accesso condizionato proprietarie.Tali dati sono contenuti negli "Entitlement Management Messages (EMMs)".
Gli EMMs aggiornano periodicamente le opzioni di sottoscrizione o i diritti pay per view per ognuno dei clienti o gruppo di clienti registrati.La CAT (Conditional Access Table) (figura1.34) indica il PID dei pacchetti contenenti gli EMMs.La

CAT è sempre identificata dal PID 0x0001.Le figure 1.34 1.35 riportano uno schema riassuntivo dei concetti sopraesposti.

Location of EMMs	
EMM A	PID 0x0061
EMM B	PID 0x0076
EMM C	PID 0x0038
EMM D	PID 0x0109

Figura1.34 (CAT)

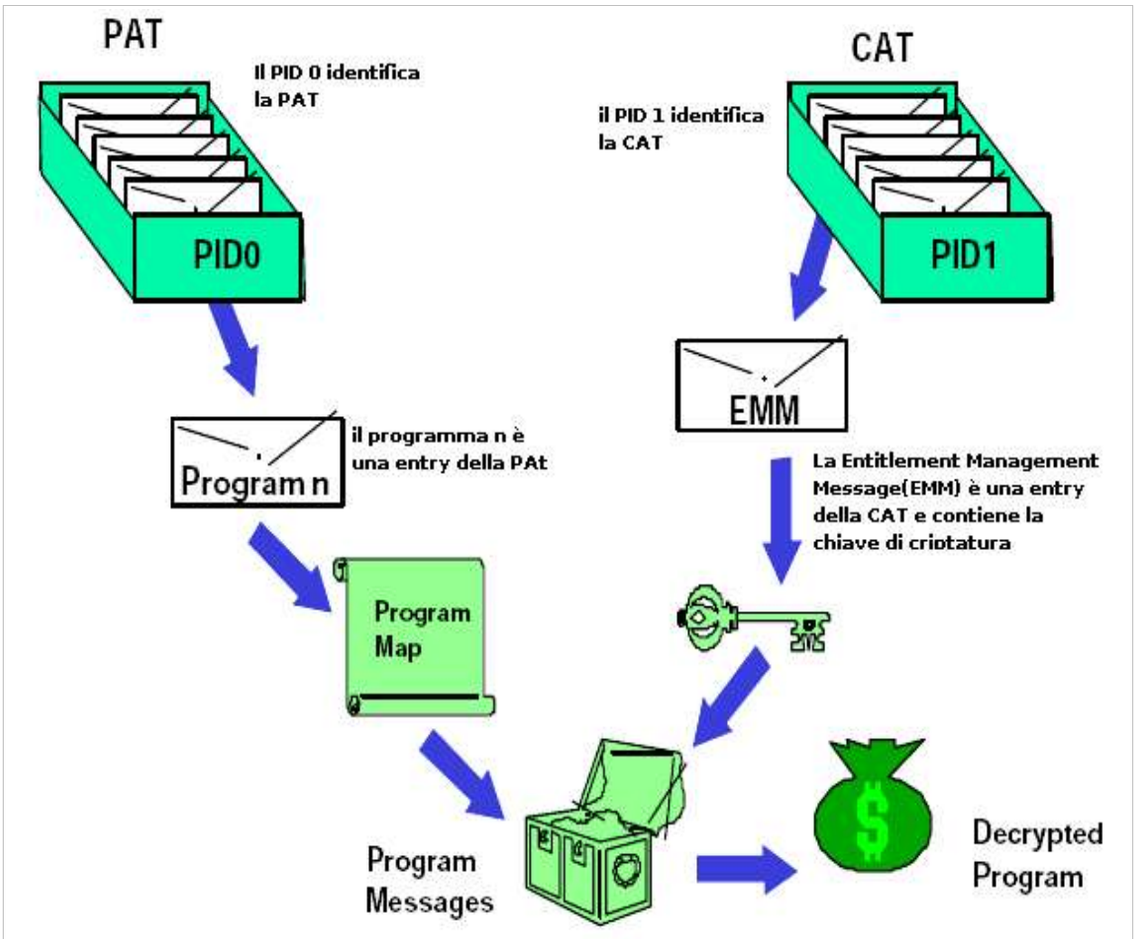

Figura1.35 (processo di decriptatura)

- **NIT:**

La Network Information Table (NIT) fornisce informazioni su di una rete ove risiedono più transport stream.Tale tabella e specificata, ma non definita da MPEG2.Viene comunque ripresa dal DVB è sarà discussa più dettagliatamente nel prossimo capitolo.

I seguenti passi danno un riassunto del processo seguito dal decoder per visualizzare un programma, che nello specifico è il programma numero 1:

1) Crea la PAT.Per fare ciò estrae il contenuto dei pacchetti con PID=0x0000.
2) Legge la PAT per identificare il PID dei pacchetti che trasportano la PMT del programma 1.Assumiamo che tale PID sia 0x0065.
3) Estrae il contenuto dei pacchetti con PID=0x0065 e costruisce la PMT.
4) Legge la PMT del programma numero 1 per cercare i PID identificanti i pacchetti audio e video e il PCR (Program Clock Reference).Si assume che la PMT assegni il contenuto video al PID 0x0131, l'audio italiano al PID 0x0132 e quello inglese al PID 0x0133.
5) Nella PMT il decoder trova anche il PID ECM (0x0150) del programma 1.
6) Localizza i pacchetti con tale PID ed estrae l'ECM.
7) Estrae il video dai pacchetti con PID 0x0132.
8) Se l'utente ha selezionato l'audio Italiano, localizza ed estrae l'audio dai pacchetti con PID 0x0132, viceversa per la lingua inglese dal PID 0x0133.
9) Mediante l'uso dell'ECM, localizzato dal PID 0x0150, decripta il video e l'audio del programma 1.
10) Assembla il video e l'audio in PES.
11) Usa il PTS(Presentation Time Stamp) e DTS(Decoding Time Stamp) contenuti nell'header di ogni pacchetto PES per determinare quando decodificare e presentare il loro contenuto all'utente.

E' importante sottolineare che mentre le tabelle PSI di MPEG2 permettono al decoder di decifrare i vari programmi contenuti nel transport stream, esse non forniscono le informazioni necessarie per gestire i numerosi programmi e servizi disponibili in una intera rete di transport steam.Nella terminologia del Digital Video Broadcasting per rete o “network” si intende una singola entità che

trasmette in broadcast uno o più transport stream.

Lo standard Digital Video Broadcasting(DVB) deifnisce un set di tabelle, chiamate Service Information (**SI**), che estendono le capacità del System Layer di MPEG2.

Riferimenti bibliografici:

- Generic Coding of Moving Pictures and associated audio: ISO/IEC 13818-1
- Technologies and Services on Digital Broadcasting (*www.nhk.or.jp/strl/publica/bt/en/le0011.pdf*)
- A. MacInnis, "The MPEG systems coding specification", Signal Processing: Image Communications,
- M. Barbero, M. Hofmann, N. D. Wells:"DCT Source Coding and Current Implementation of HDTV", EBU Tech. Review,No. 251, Spring 1992
- ETS 300 421, “Digital broadcasting systems for television, sound and data service; Framing structure channel coding and modulation for11/12 GHz satellite services”, Dicembre 1994.
- ETS 300 429, “Digital broadcasting systems for television, sound and data service; Framing structure channel coding and modulation for cable systems”, Dicembre 1994.
- MPEG-2 Digital Broadcast Pocket Guide (*www.mpeg.acterna.com*)
- MPEG-2 Fundamentals for Broadcast and Post-Production Engineers *(www.netmcmarine.co.uk/resources/mpeg2_broadcast.pdf)*
- MPEG video coding a simple introduction Dr. S.R. Ely (BBC) (*www.ebu.ch/trev_266-ely.pdf*)

2. DVB

2.1 Introduzione alla TV digitale

Il rapido progresso delle tecnologie digitali, già a partire dagli anni ’90, nei campi della produzione, distribuzione e diffusione televisiva e le nuove prospettive offerte agli operatori del settore e all’industria di consumo, hanno portato alla costituzione nel 1993 del Progetto europeo DVB (Digital Video Broadcasting).Il Progetto, che ora raccoglie oltre 300 Membri europei ed extraeuropei, ha l’obiettivo di armonizzare le strategie volte all’introduzione della televisione digitale e dei nuovi servizi multimediali e interattivi sui vari mezzi trasmissivi e definire le relative specifiche tecniche.Il primo significativo risultato è stato raggiunto con la definizione della specifica del sistema DVB-S per la diffusione diretta da satellite di TV multi-programma; subito dopo sono state definite le specifiche del sistema DVB-C per la distribuzione dei segnali televisivi attraverso le reti via cavo e, successivamente, una intera “famiglia” di specifiche (tra cui DVB-T per il digitale terrestre e DVB-H, che si presenta come una variante del DVB-T ottimizzata per la ricezione del segnale su dispositivi portatili in mobilità) che, partendo dal mondo della televisione, si sono progressivamente allargate a interessare lo scenario delle tecnologie emergenti e dei nuovi media.Le specifiche tecniche approvate dal DVB vengono ratificate dall’ETSI che ne attribuisce la veste di standard europei.Come già accennato nel precedente capitolo, lo standard DVB può essere visto come una estensione di quello MPEG2.

DVB prevede una differenzazione delle proprie specifiche in due distinti livelli: *logico* e *fisico*.

- Specifiche di livello logico:
  - Formati di compressione Audio/Video;
  - Costruzione dei flussi digitali (Transport Stream);
  - Formato dei pacchetti;
  - Segnalazione dei servizi;
  - Incapsulamento di dati generici non Audio/Video e di protocolli non DVB;
- Specifiche di livello fisico:
  - Specifiche sul segnale fisico (modulazione, codifica ecc...) nei vari mezzi trasmissivi (cavo, satellite, ripetitori terrestri).

Tra le specifiche di livello logico, oltre a sottolineare ulteriormente il ruolo fondamentale coperto da MPEG per quanto riguarda la compressione audio/video e la costruzione dei flussi digitali (TS), si fa notare l'estensione delle MPEG2-PSI (Program Service Information) mediante le SI(Service Information), tabelle caratteristiche dello standard DVB.Questa parte verrà approfondita nel terzo capitolo.
Le specifiche di livello fisico differenziano lo standard generico DVB in sottostandard a seconda del particolare canale trasmissivo utilizzato.Si definiscono infatti i vari parametri fisici della trasmissione, come ad esempio la frequenza, modulazione, codifiche etc...
Lo schema di una generica architettura DVB è riportato nella figura sottostante (figura 2.1).

figura2.1 (schema generico di una architettura DVB)

1) *Codifica di sorgente e multiplexing*: come già indicato la codifica di sorgente e la relativa multiplazione del segnale viene affidata allo standard MPEG2.
2) *Adattamento di canale*: questo blocco e finalizzato all'ottimizzazione della trasmissione dell'MPEG2 Transport Stream sullo specifico mezzo trasmissivo.Sarà infatti necessario implementare particolari tipi di modulazione e tecniche di codifica per assicurarsi che il segnale si adatti nel miglior modo

possibile al canale.

3) *Mezzo trasmissivo*: il canale trasmissivo,come accennato ad inizio capitolo, può essere basato sulla trasmissione via satellite, cavo o ripetitori terrestri.Ognuno di questi canali sarà caratterizzato da delle particolari proprietà fisiche, che sono state tenute in considerazione per lo sviluppo dei vari standard DVB.
4) *Demodulazione e decodifica*: tali funzioni sono implementate in ricezione per poter acquisire il segnale (MPEG-TS) in banda base.
5) *Presentazione*: è il blocco finale di tutta la catena che permette la definitiva visualizzazione dei contenuti del TS all'utente finale.

2.2 La famiglia di standard DVB

2.2.1 Tipologie di diffusione (DVB-S, DVB-C, DVB-T, DVB-H)

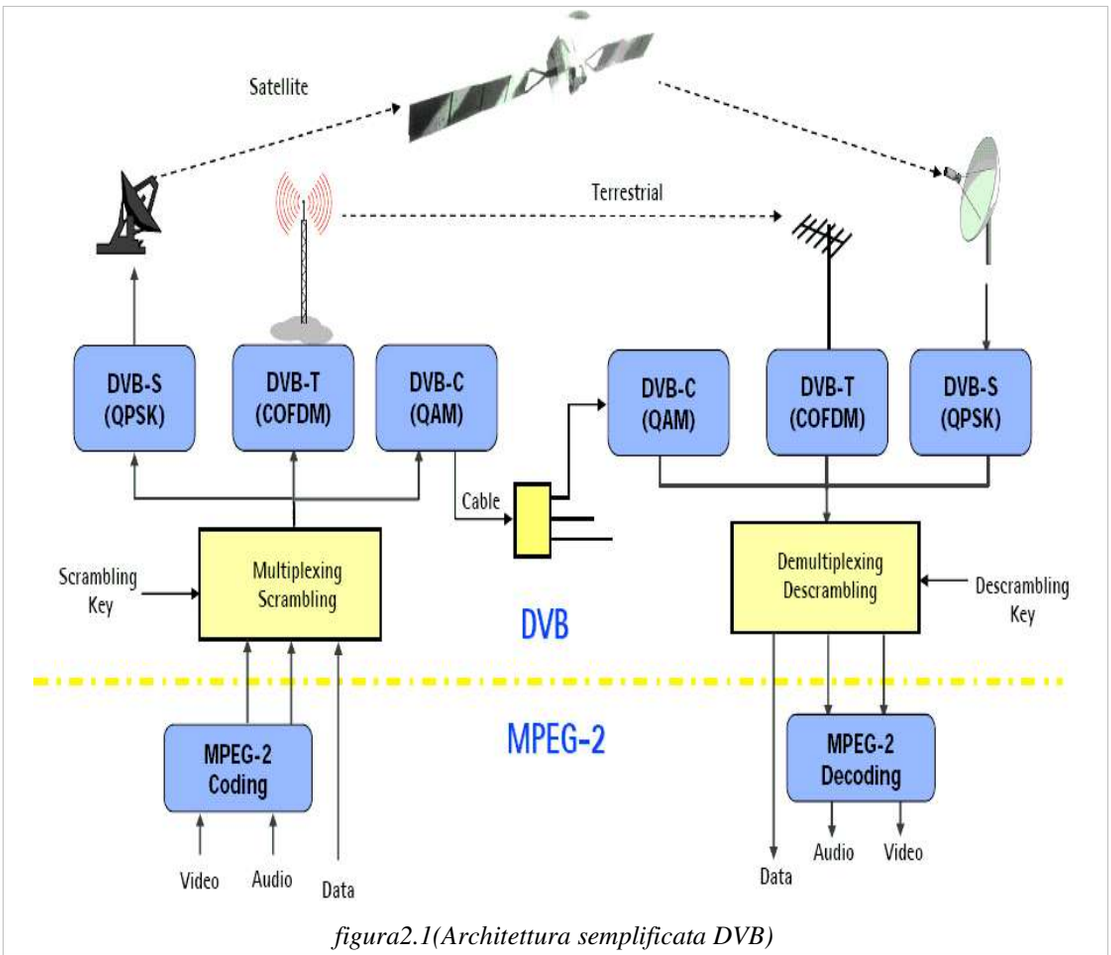

figura2.1(Architettura semplificata DVB)

In figura 2.1 è riportata l'architettura semplificata per un sistema di broadcasting televisivo DVB.
Si nota come il tutto poggi sulla codifica MPEG2 e come i vari sotto standard DVB si differenzino l'uno dall'altro solo per il particolare mezzo trasmissivo impiegato e le relative tecniche adottate per ottimizzarne l'utilizzo.
Il multiplex MPEG2 è flessibile e consente di convogliare in un singolo flusso numerico segnali relativi a un gran numero di programmi televisivi, ciascuno comprendente le relative informazioni video, audio e dati.
Canali di servizio aggiuntivi sono inoltre previsti per indicare i vari programmi inseriti all'interno del multiplex (Service Information,SI) , per attuare l'accesso condizionato (Conditional Access, CA) , per fornire una guida elettronica dei programmi (Electronic Program Guide, EPG).
Si da ora una breve presentazione delle varie tipologie di diffusione DVB.

- **DVB-S :**

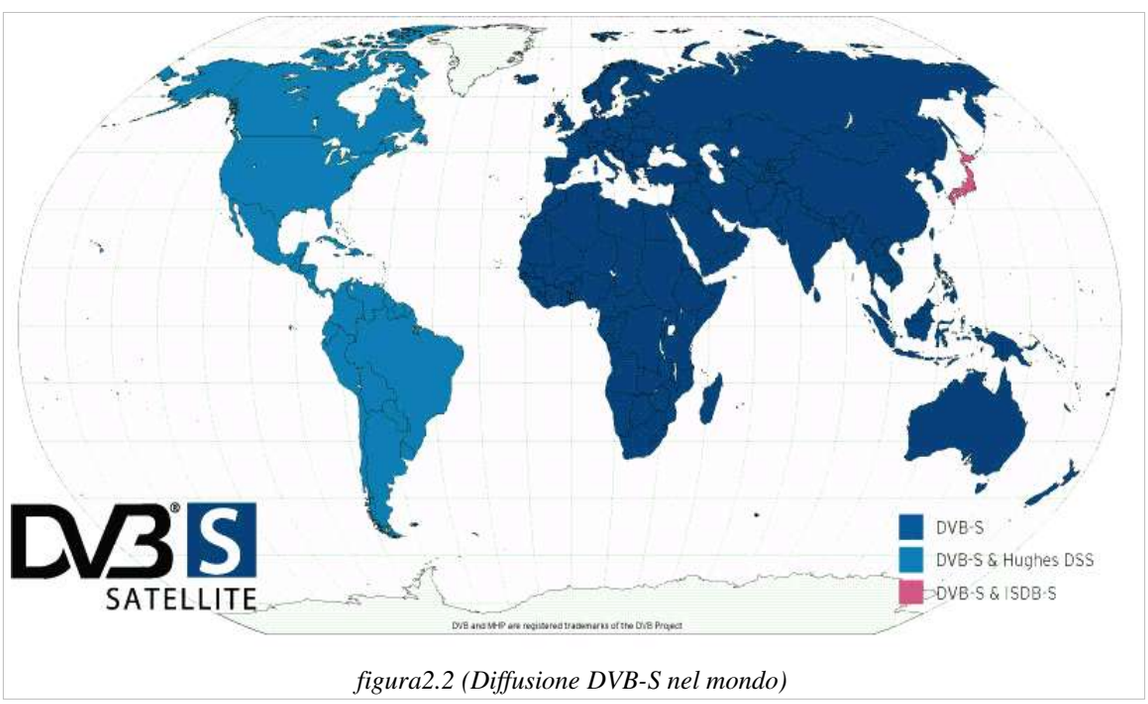

figura2.2 (Diffusione DVB-S nel mondo)

La progettazione degli standard trasmissivi deve partire da una profonda conoscenza delle caratteristiche dei canali fisici utilizzati.
Nel caso del DVB-S il canale è quello satellitare, caratterizzato da una banda

copresa tra i 24 e i 56 Mhz, da una potenza di segnale molto debole e dalla non linearità degli amplificatori di potenza (quelli sui satelliti) che lavorano molto vicino alla zona di saturazione.
Il sistema DVB-S, specificato nella norma ETS 300 421, utilizza gli standard MPEG per la codifica e la compressione del segnale di sorgente e per la multiplazione dei programmi, e si avvale di un adattatore di canale, appositamente progettato per ottimizzare le prestazioni del sistema sul collegamento satellitare.
La figura 2.3 schematizza lo schema a blocchi in trasmissione di tale standard.

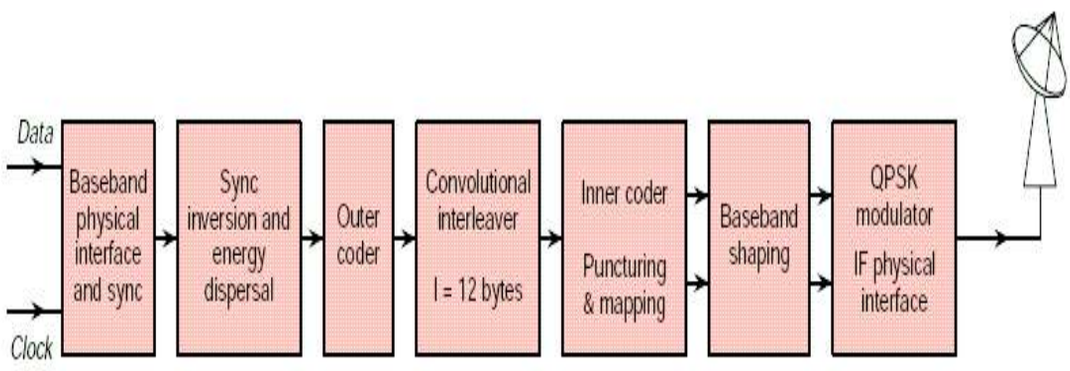

figure2.3 (Schema a blocchi di un sistema DVB-S)

- Il bit stream di ingresso deve essere organizzato in pacchetti di lunghezza fissa di 188Bytes, in accordo con il Transport Stream MPEG2.Con una tecnica chiamata dispersione d'energia (scrambling), tale sequenza di byte viene scorrelata.Questa operazione è eseguita mediante uno XOR bit a bit del byte stream di ingresso con una sequenza psudocasuale, preservando il primo byte che è costituito da una parola di sincronizzazione, pari a 47HEX .L'unica modifica della parola di sincronizzazione è rappresentata da un'inversione bit a bit, operata ogni 8 pacchetti TS, al fine di introdurre un sincronismo di trama.Il risultato finale sarà una più uniforme distribuzione degli zeri e uno del bit stream evitando così la presenza di sequenze critiche per il codice FEC.
- *Outer coder (codificatore esterno)*:un primo livello di protezione viene fornito ai dati trasmessi, usando un codice a blocchi non binario di tipo Reed-Solomon

RS(204,188), che permette la correzione di un massimo di 8 byte errati per ogni pacchetto di 188 byte.

- *Convolutional interleaver*:si utilizza una tecnica di interleaving convoluzionale per mescolare la sequenza di dati trasmessa, in modo da renderla più robusta in caso di lunghe sequenze di errori.
- *Inner coding (convoluzionale)*:un secondo livello di protezione è affidato all'uso di un codice convoluzionale binario con perforazione, che spesso viene indicato nei set top box con il termine FEC (Forward Error Correction).I valori di codifica ammessi sono cinque: 1/2, 2/3, 3/4, 5/6, 7/8.Nella configurazione FEC 1/2, ad ogni bit di informazione si aggiunge un bit di protezione, nella configurazione 2/3, a 2 bit di informazione, si aggiunge un bit di protezione, e così via.Di conseguenza le prestazioni del codice, in termini di efficienza di trasmissione, aumentano progressivamente, a scapito di una progressiva riduzione della capacità di correzione degli errori.
- *Baseband shaping (sagomatura in banda base)*:il segnale viene opportunamente filtrato con un filtro a coseno rialzato (roll-off 35%), che permette di diminuire le interferenze mutue del segnale in ricezione.
- *Modulation (modulazione)*:questo blocco effettua la modulazione in banda-base digitale della sequenza di bit, producendo una sequenza di simboli.Il metodo di modulazione utilizzato è QPSK.E' stata adottata la modulazione QPSK per la particolare robustezza contro il rumore, le interferenze e la non linearità dell'amplificatore di potenza a bordo del satellite, che opera normalmente vicino alla saturazione.

A seguito della recente standardizzazione del DVB-T, la versione per le trasmissioni terrestri che ora è stata introdotta nel nostro paese con il "digitale terrestre", anche il DVB-S subirà alcune modifiche per rimanere al passo con i tempi.Il DVB-S2 sarà il successore dell'attuale DVB-S rispetto al quale implementerà alcune migliorie (tra cui l'adozione dello standard MPEG-4 al posto di MPEG-2) destinate principalmente ad aumentare l'efficienza del sistema di utilizzo delle preziose risorse di trasmissione prese in affitto dalle varie emittenti.
Il DVB-S2 prevede un aumento del bitrate disponibile di circa il 30% e un miglioramento delle performance in ricezione (più robustezza anche in caso di bassa potenza di segnale o condizioni atmosferiche negative).Il DVB-S2, inoltre,

a differenza del DVB-S che era stato pensato esclusivamente per il broadcast (trasmissione simultanea a tutti gli utenti), è pensato anche per ottimizzare la trasmissione a livello di singolo utente.Per questo il DVB-S2 consente anche di offrire migliori servizi (Fast-Internet) ad un maggior numero di abbonati, abbattendo i costi per i service provider, il che dovrebbe tramutarsi in un beneficio economico anche per gli utenti finali.
Nella figura sottostante si riporta uno schema di riferimento che riassume le caratteristiche del DVB-S e evidenzia le principali differenze degli altri standard.

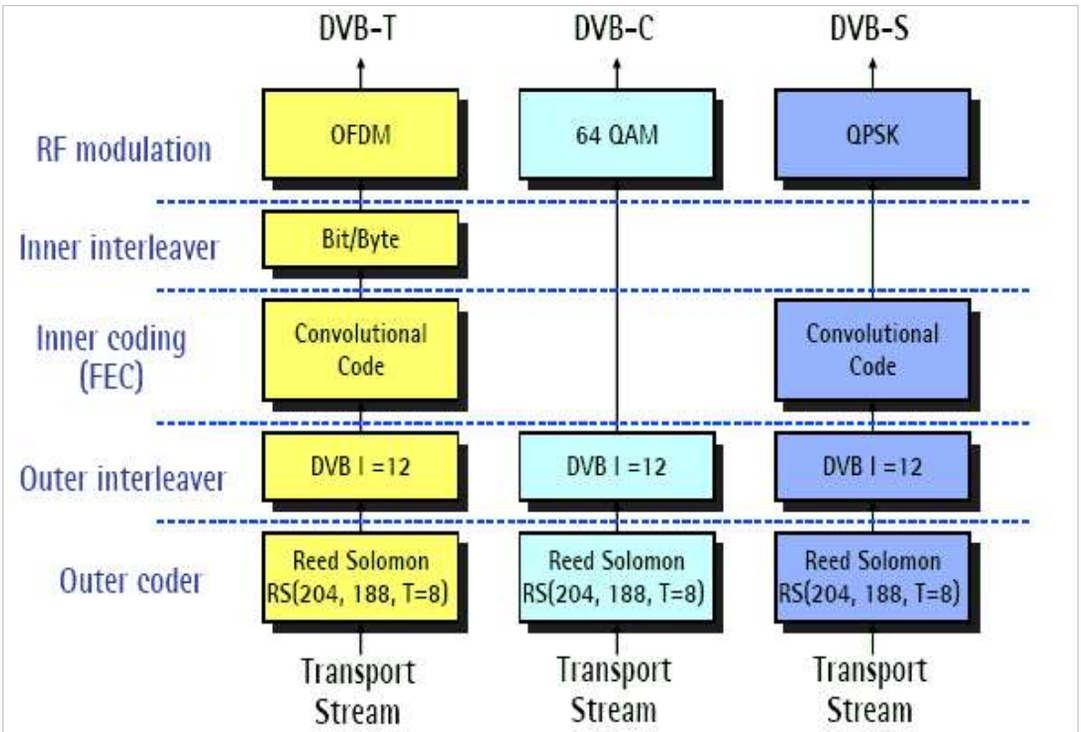

figura2.4(DVB-T,DVB-C,DVB-S principali differenze)

■ DVB-C :

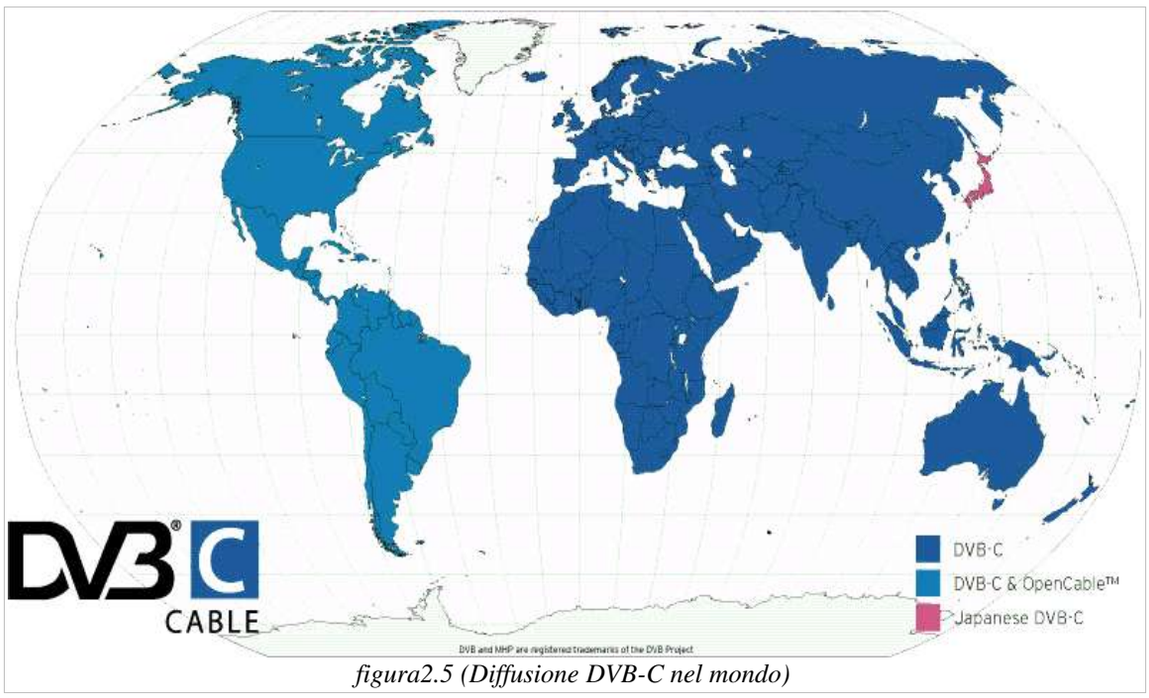

figura2.5 (Diffusione DVB-C nel mondo)

Il Digital Video Broadcasting - Cable (DVB-C) è lo standard del consorzio europeo DVB per una modalità di trasmissione televisiva digitale via cavo.
Il canale di trasmissione via cavo è un ambiente abbastanza protetto rispetto alla distorsione ed attenuazione del segnale.
Si possono raggiungere alti rapporti segnale rumore (SNR) e considernado poi che non vi sono effetti negativi dovuti al “Multipath” è possibile l'utilizzo di tecniche di modulazione di ordine più alto rispetto al DVB-S.
Basti pensare che in situazioni particolarmente favorevoli si possono utilizzare modulatori 1024QAM.Comunque solitamente si adotta una modulazione QAM a 64, 32 o 16 stati.
Vista la minore presenza di disturbi nella trasmissione via cavo, si adotta una più debole implementazione del FEC (Forward Error Correction), infatti ci si limita all'uso di un Reed Solomon encoder e di un Interleaver convoluzionale.
In ricezione è previsto l’impiego di un equalizzatore adattativo degli echi che possono essere presenti nella rete in cavo.
In Italia il DVB-C è prticamente inesistente e ciò a causa del fallimento del Progetto Socrate di Teecom Italia (prima metà degli anni 90), mentre è invece

molto usato in altri paesi (in particolare USA) sia per la difficoltà di propagazione del segnale elettromagnetico in città piene di grattacieli ma anche e sopratutto grazie alla possibilità di disporre di condotti sotterranei già esistenti.

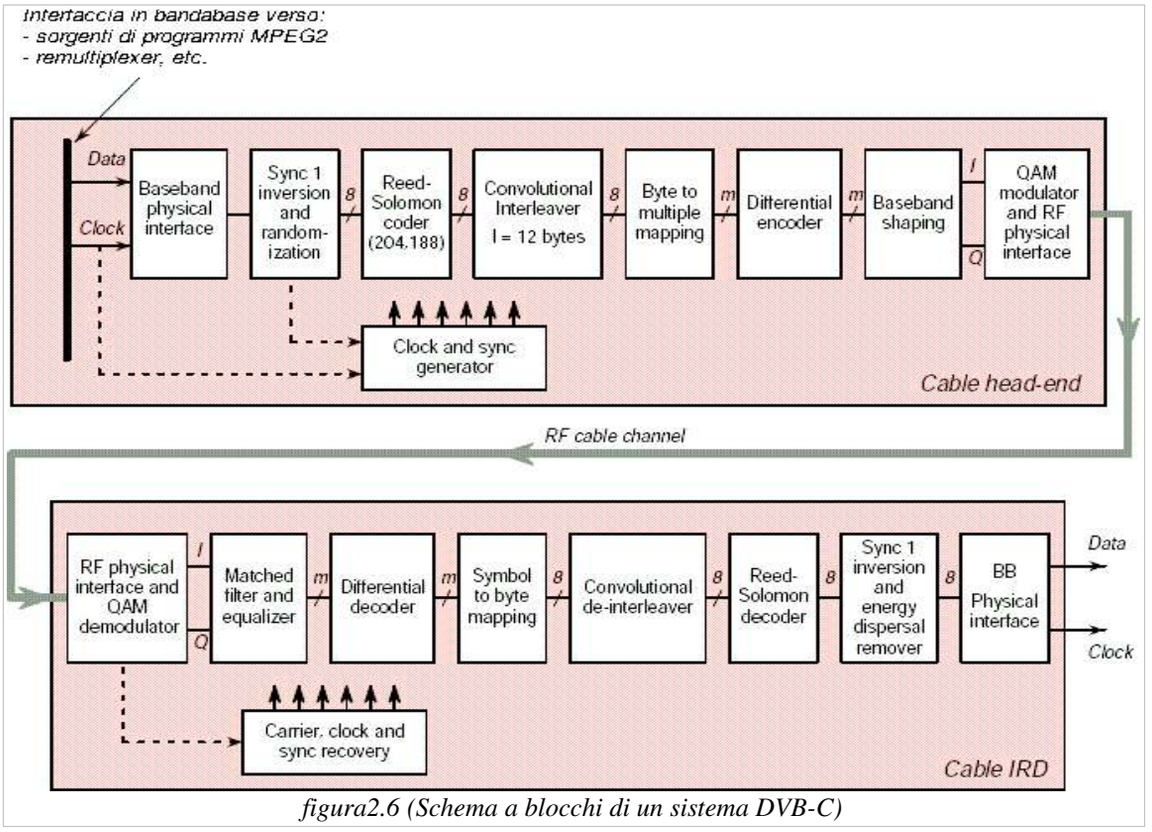

figura2.6 (Schema a blocchi di un sistema DVB-C)

In figura 2.6 è riportato lo schema a blocchi di un sistema DVB-C.
Si elencano le prinicipali caratteristiche di questo standard.

- Sostanziale uguaglianza in termini di codifica, multiplexing, framing, interleaving, outer code e variabilità del symbol rate rispetto al sistema DVB-S.
- Il principale ostacolo da superare nell'uso di un canale trasmissivo via cavo è la limitata larghezza di banda (8/7Mhz).Il sistema DVB-C è stato progettato per ottimizzarne al meglio l'uso e l'implementazione di una modulazione 16, 32 e 64 QAM su singola portante va proprio in questa direzione.
- Non è necessario l'uso di inner code convoluzionali e questo grazie all'alto SNR che caratterizza tale mezzo trasmissivo.

- E' stato specificato un fattore di roll-off pari al 15%, che è inferiore rispetto a quello utilizzato per il DVB-S (35%).

■ DVB-T :

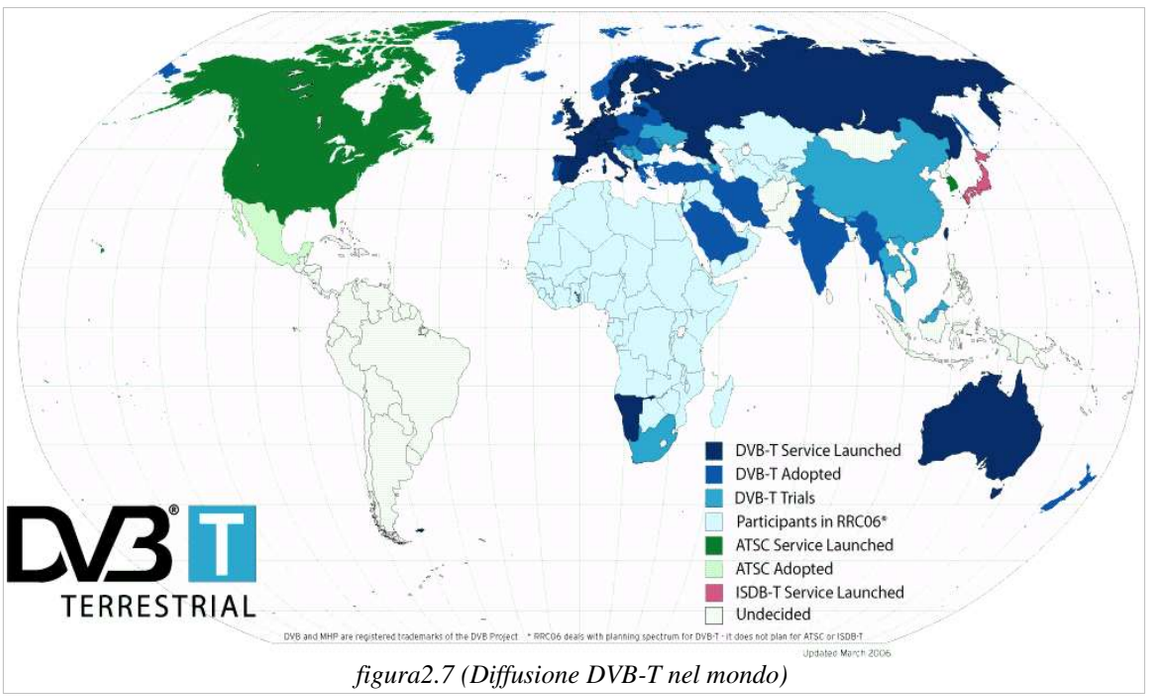

figura2.7 (Diffusione DVB-T nel mondo)

DVB-T è l'acronimo di “Digital Video Broadcasting – Terrestrial” e indica la ricezione digitale del segnale tv da ripetitori terrestri.
La principale caratteristca di questo canale trasmissivo è la propagazione multicammino (“multipath”) dovuta alle riflessioni che l'onda elettromagnetica subisce incontrando ad esempio edifici o altri tipi di ostacoli (vedi figura 2.7).Tale fenomeno può deteriorare pesantemente il segnale trasmesso.
La figura 2.9 rappresenta lo schema a blocchi adottato da tale standard.

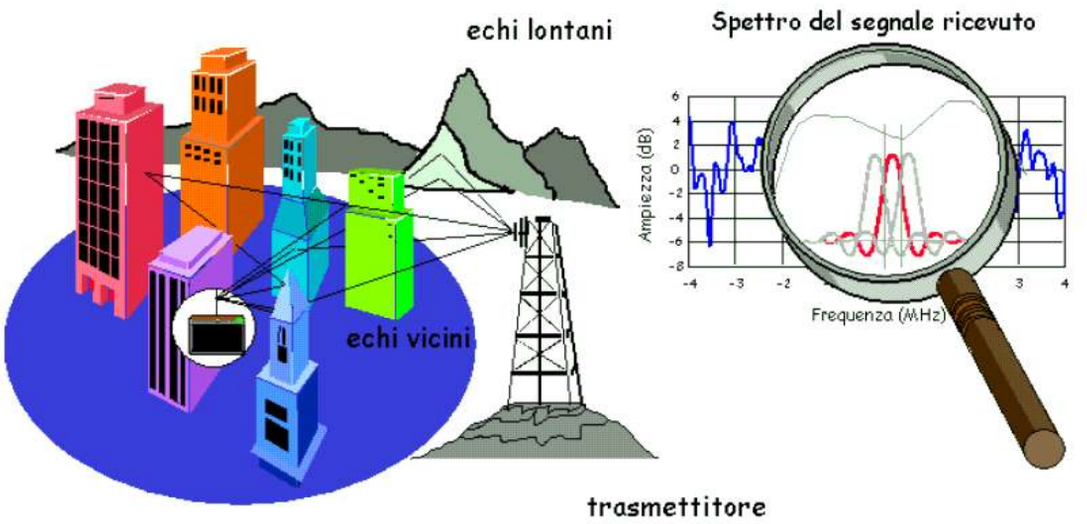

figura2.8 (modello di propagazione terrestre)

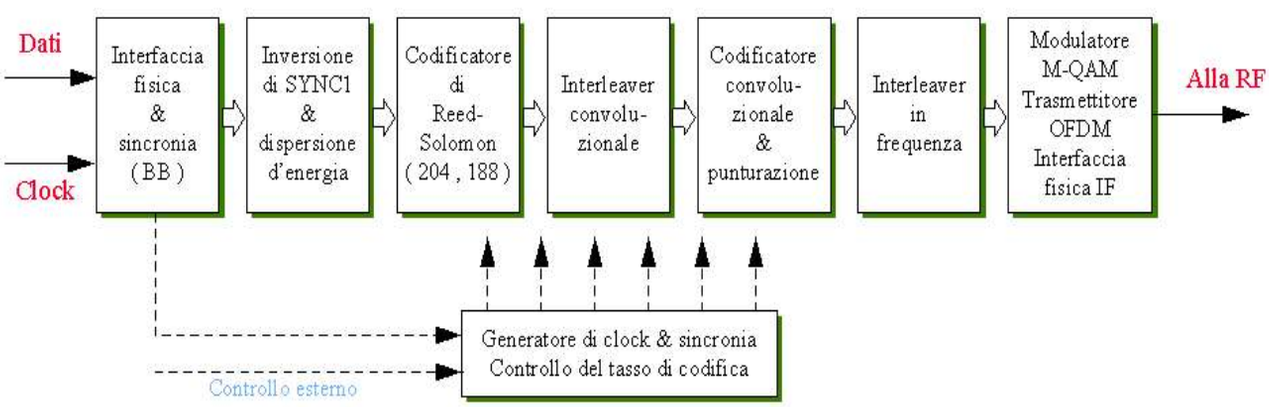

figura2.9 (Schema a blocchi sistema DVB-T)

Le uguaglianze con il DVB-S, sono ovviamente la struttura di trama, derivata dal multiplatore MPEG2, la dispersione dell'energia del segnale per uniformare la distribuzione spettrale all'interno del canale RF (“scrambling”) e la tecnica utilizzata per la protezione dagli errori, realizzata mediante la concatenazione di un codice esterno a blocco di tipo Reed-Solomon e di un codice interno convoluzionale (con possibilità di scegliere tra diversi rapporti di codifica, da 1/2a7/8).

Gli echi naturali dell'ordine di alcuni microsecondi e legati all'orografia del terreno, così come gli echi artificiali dell'ordine di centinaia di microsecondi dovuti ai segnali provenienti dai vari trasmettitori isofrequenziali, non possono essere trattati con tecniche di modulazione a portante singola, anche perchè richiederebbero l'impiego di equalizzatori molto lunghi e complessi.
Pertanto, sulla base di tali considerazioni e dei risultati di accurate valutazioni tecniche è stata scelta la modulazione multiportante CODFM (Coded Orthogonal Frequency Division Multiplexing).

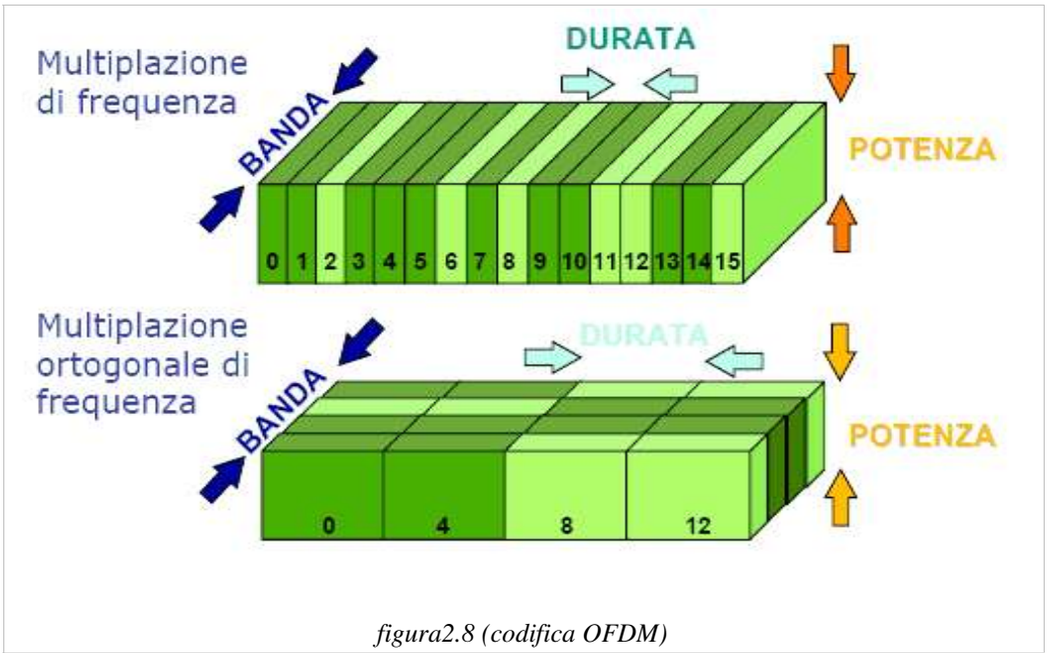

figura2.8 (codifica OFDM)

Il principio su cui si basa questa tecnica (vedi figura 2.8) consiste nell'utilizzare diverse frequenze portanti (a banda stretta, quindi a bassa velocità di trasmissione) fra loro ortogonali e poi suddividere l'informazione in altrettanti flussi di bit indipendenti.Ogni flusso di bit viene assegnato ad una diversa portante e poi alla ricezione i vari flussi vengono opportunamente ricombinati.
A ciascuna delle portanti viene solitamente applicata la modulazione digitale QPSK o QAM.
Si è sottolineato come le singole portanti siano a banda stretta, quindi a bassa velocità di trasmissione.Questo è il punto cardine, infatti l'effetto deleterio delle riflessioni multipath è direttamente proporzionale al bitrate (o symbol rate): più alto è più sarà importante l'effetto negativo di tali disturbi.
Si capisce quindi che più alto sarà il numero delle portanti utilizzate, più basso

sarà il bit rate utilizzato per ciascuna di esse, diminuendo l'effetto deleterio del multipath.Il numero di sottoportanti utilizzate da DVB-T sono 1705 nella modalità 2K e 6817 per quella 8K, inoltre devono essere separate l'una dall'altra in modo che non interferiscano, devono cioè essere ortogonali.
Il processo OFDM è attuato per mezzo di una I-FFT (Inverse Fast Fourier Transform).Si fa notare che tale modulazione, quando opportunamente utilizzata, presenta un'importante peculiarità: permette di sfruttare in modo costruttivo gli echi isofrequenziali che giungono all'antenna ricevente del televisore.

■ DVB-H :

Il DVB-H è il più recente protocollo definito dal consorzio DVB.Inizialmente nato come DVB-M(Mobile), successivamente rinominato DVB-X e quindi DVB-H (acronimo di Digital Video Broadcasting to Handheld, per sottolineare che si tratta di servizi rivolti non soltanto ad apparecchi installati in auto, ma più genericamente a terminali portatili "che stanno in una mano"), questo sistema si rivolge alla distribuzione di servizi multimediali a terminali mobili in modalità multicast e broadcast.
In realtà DVB-H non è uno standard a sè stante, alternativo agli altri presentati sino ad ora, ma è stato pensato come un'evoluzione del sistema DVB-T, con cui condivide la gamma di frequenze, aggiungendovi quelle funzionalità necessarie per garantire maggiore robustezza nella ricezione in movimento, un minor consumo delle batterie e una maggior sinergia con il mondo internet.
Il DVB-H è una tecnologia diffusiva caratterizzata dal fatto che lo stesso contenuto può essere ricevuto contemporaneamente da un numero elevatissimo di utenti, grazie all'uso del protocollo IP, che rende possibile la trsmissione simultanea sullo stesso canale di pacchetti video (stream DVB) e pacchetti dati sfruttabili da applicazioni presenti sul terminale ricevente (IP Datacast).
Il formato di codifica di canale è lo stesso specificato per il DVB-T, con poche significative differenze, legate alla circostanza che il terminale ricevente mobile soffre di limitazioni di alimentazione ed è affetto in maggiore misura da disturbi trasmissivi (cammini multipli, effetto Doppler) rispetto a quanto avviene nel caso DVB-T.Pertanto, il DVB-H prevede:

- un'ulteriore codifica di canale, detta MPE-FEC.
- un meccanismo di ricezione su brevi intervalli di tempo, alternati da periodi di disattivazione detto time slicing.
- l'opzione di modulazione OFDM a 4K portanti,compromesso fra le due soluzioni (2K e 8K) previste dal DVB-T, per migliorare la ricezione mobile.

Lo schema a blocchi di principio del sistema DVB-H è riportato nella figura sottostante.

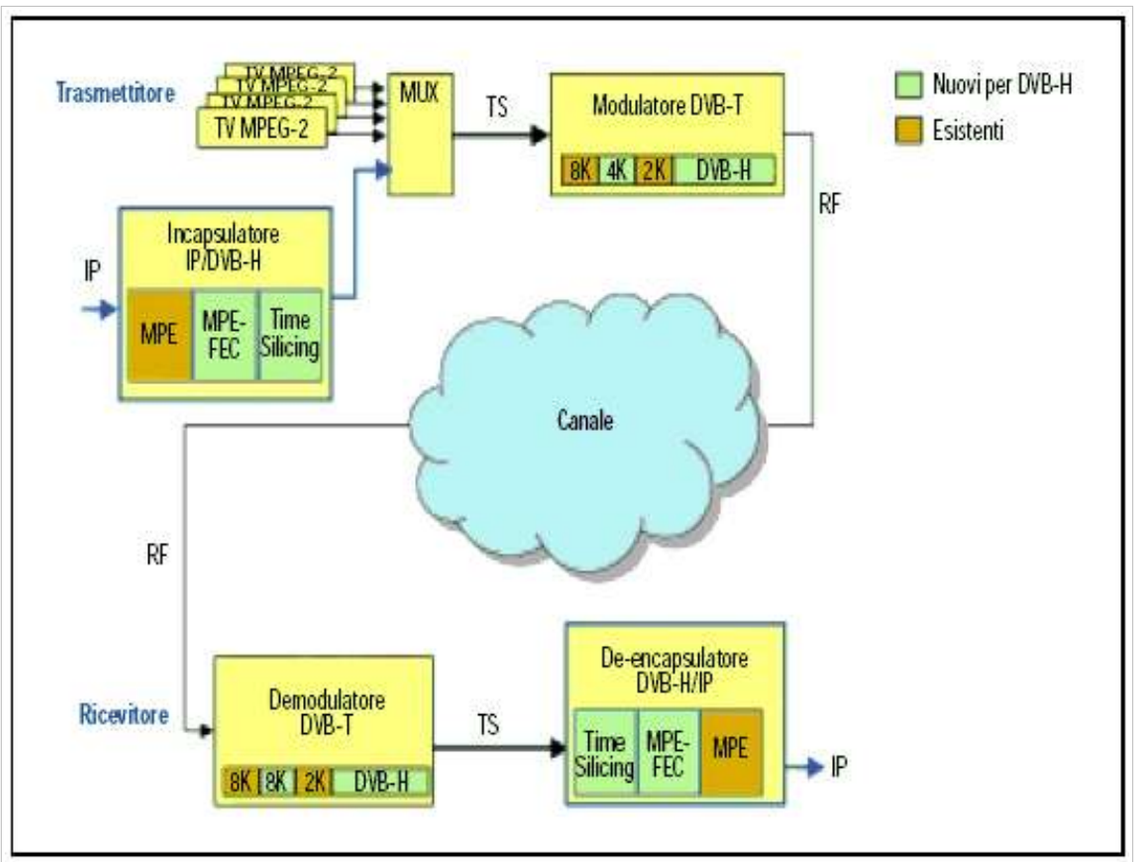

figura2.10(Schema a blocchi di principio del sistema DVB-H)

2.2.2 Conditional Access (CA)

Il sistema di accesso condizionato (conditional access, CA) è lo strumento usato per codificare i programmi prima della loro trasmissione, in modo da consentirne l'accesso ai soli utenti abilitati.

Le tecniche CA sono basate sulla definizione, da parte dello standard DVB, di un unico sistema di cifratura, denominato *Common Scrambling Algorithm*, le cui specifiche sono fornite dall'ETSI, con opportuna licenza, solo ad organizzazioni riconosciute.L'impiego congiunto di tale sistema con gli specifici campi dati e meccanismi di selezione definiti dallo standard MPEG (transport stream) consente di associare ad uno stesso programma cifrato, informazioni che consentono di decifrarlo con sistemi CA differenti.

Il sistema CA offre servizi come pay-per-view (PPV), video-on-demand (VOD), giochi, rende possibile la limitazione dell'accesso a certi contenuti (ad esempio canali sportivi o film) ed è in grado di inviare messaggi a specifici *set-top-box* (con questo termine si indica il terminale che ha il compito di ricevere, elaborare e trasformare segnali trasmessi con il sistema digitale, per renderli fruibili ai normali televisori), localizzati ad esempio in una particolare area geografica.

In DVB il sistema di Conditional Access è caratterizzato da tre funzioni principali:*scrambling / descrambling*, *entitlement checking* and *entitlement management.*

- *scrambling / descrambling*: la funzione di scrambling ha il compito di rendere il servizio incomprensibile, indecifrabile agli utenti non autorizzati.Il processo inverso di descrambling può essere attuato solo da quei ricevitori provvisti di un appropriato descrambler e che siano in possesso della parola segreta di controllo (Control Word, CW).
- *entitlement checking*: questa funzione si occupa della trasmissione di tutte quelle informazioni richieste per accedere ad un servizio, incluso il codice segreto (CW).Con tali dati i ricevitori autorizzati sono in grado di procedere al descrambling.Il tutto è trasmesso in appositi messaggi (tabelle DVB) chiamate Entitlement Control Messages (ECMs).
- *entitlement management*: quest'ultima funzione si occupa della distribuzione dei vari "entitlements" ai ricevitori.Vi sono diverse tipologie di entitlements che corrispondono a varie modalità di sottoscrizione ad un servizio:abbonamenti

per temi, livelli o classi, pay-per-programme, per service o per time.Tali informazioni sono trasmesse al'interno di messagi chiamati Entitlement Management Messages (EMMs).

Il controllo delle funzioni di gestione (ECM e EMM) richiede l'uso di misure di sicurezza ben più forti della Control Word (CW).
Quindi per criptare i messaggi di controllo e gestione sono utilizzate chiavi segrete e algoritmi di criptografia, come il Digital Encryption Standard (DES).

- ***Data Scrambling***: il sistema CA può operare lo scrambling dei programmi sia a livello del Packetized Elmentary Stream (PES) che a quello del Transport Stream MPEG2 (quello più usato).Se si fa uso del CA, l'architettura del sistema è modificata come indicato in figura 2.11.

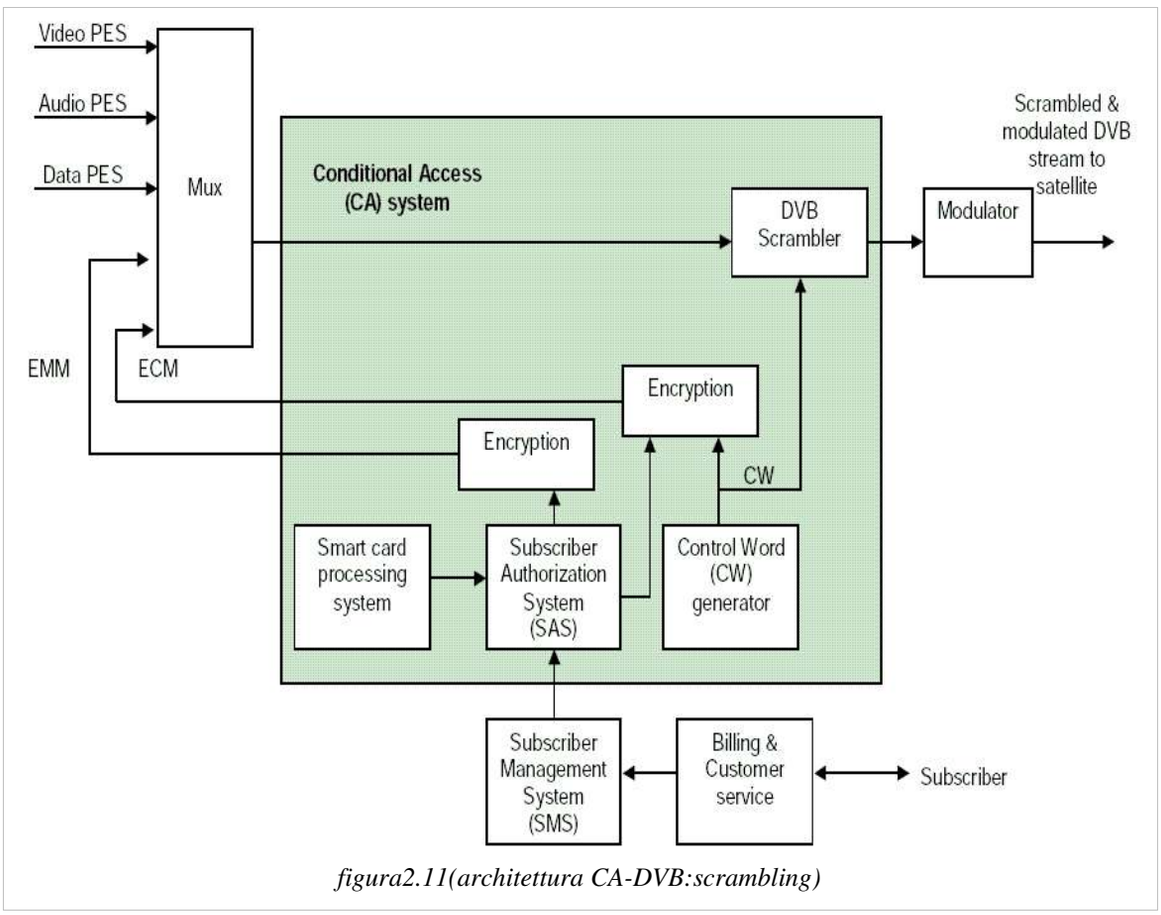

figura2.11(architettura CA-DVB:scrambling)

Il servizio di Conditional Access è posizionato tra il multiplexer e il modulatore.
Il multiplexer procederà alla combinazione degli ECM e EMM con i contenuti video, audio e dati, inserendo il tutto in un singolo transport stream DVB.Il modulatore riceve poi in ingresso il segnale già sottoposto a scrambling e lo modula per la trasmissione.
Il sistema CA è composto dai seguenti moduli:

- *Smart Card Processing System*: contiene le informazioni sui dati segreti della smart-card o del set-top-box del cliente, includendo in particolare la Secret Key di ogni Smart Card.Qualche volta tale modulo è integrato nel SAS.
- *Subscriber Authorization System (SAS)*: processa le differenti autorizzazioni di visualizzazione date al cliente e le usa per generare adeguati EMM e ECM.Si occupa anche della generazione della chiave segreta, trasmessa nell'EMM.
- *Control Word Generator*: crea la parola di controllo (CW).
- *Sistema di Encryption*: viene usato per cripatre la “service key”, trasmessa nell'EMM e la Control Word trasmessa nell'ECM.
- *Scrambler*: opera lo scrambling del payload dei pacchetti componenti il transport stream, usando la Control Word.Solitamente lo scrambling viene fatto sui pacchetti video e audio e alcune volte su quelli dati.I pacchetti contenenti i messaggi di EMM e ECM non vengono sottoposti a scrambling.Tipicamente lo scrambler è incluso nel multiplexer.

Inoltre, il Conditional Access interagisce con il Subscriber Management System (SMS) che contiene tutti i dettagli relativi agli abbonati.A sua volta l'SMS interagisce con il sistema di Billing e Customer Care.L'SMS comunica quali sono i programmi (o servizi) che i vari clienti sono autorizzati a vedere.

- ***Data descrambling***: dalla parte del ricevitore, il set-top box procederà al descrambling dello stream relativo allo specifico programma e decodificherà i dati MPEG2 per poterlo vedere.In figura 2.12 è riportato il relativo schema a blocchi.

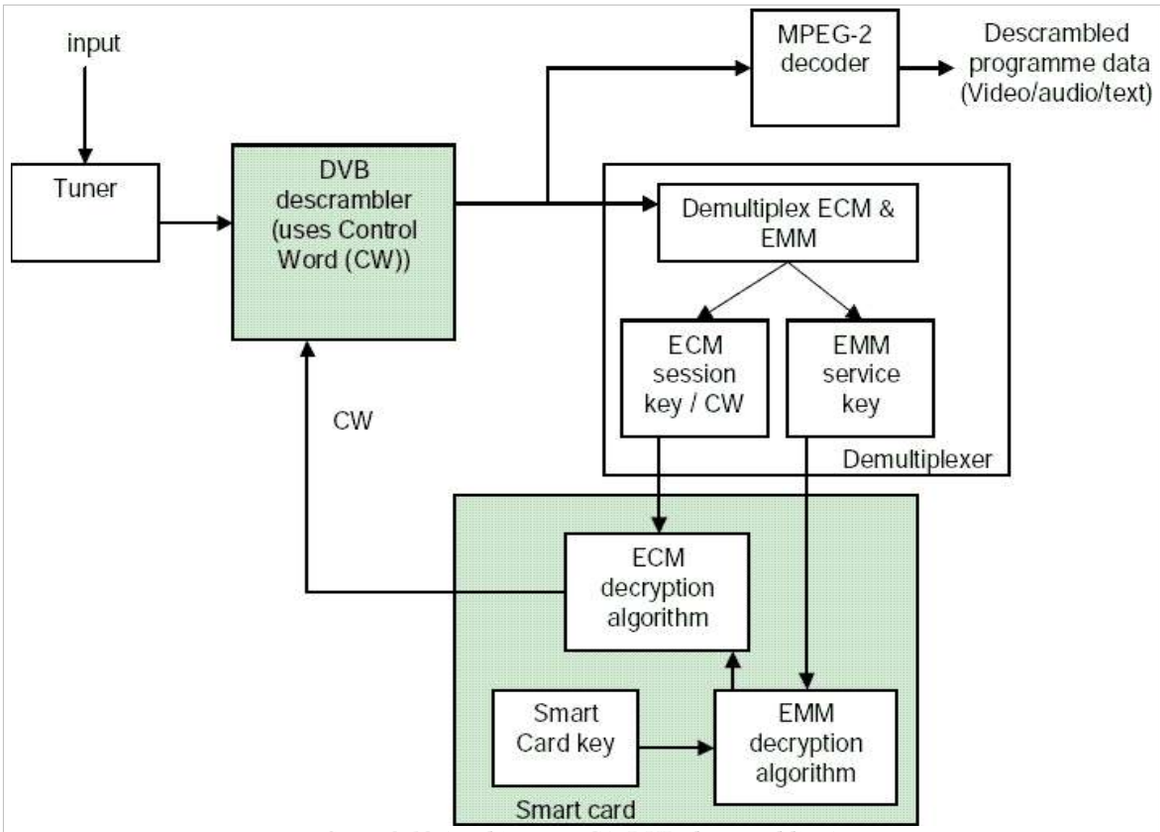

figura2.12 (architettura CA-DVB:descrambling)

Il "*tuner*" del set-top box riceve il segnale entrante, lo demodula e lo manda al "*transport stream generator*".Qui, come si può ben intuire dal nome, viene generato il transport stream che è successivamente passato al decoder MPEG2. Nella figura viene anche mostrato il sistema di descrambling DVB.Tuttavia, è importante sottolineare che lo standard DVB non specifica le caratteristiche della smart-card e quindi lo schema è da prendersi solo come un esempio.

La "service key" (trasmessa nell'EMM) viene madata alla smart-card, ove viene decriptata usando la sua (della smart-card) specifica chiave.La "service key" decriptata viene poi usata per decriptare la "session key" o "Control Word" (trasmessa nell'ECM).Questa sarà la chiave definitiva usata dal DVB transport stream descrambler.

2.2.3 Interattività MHP

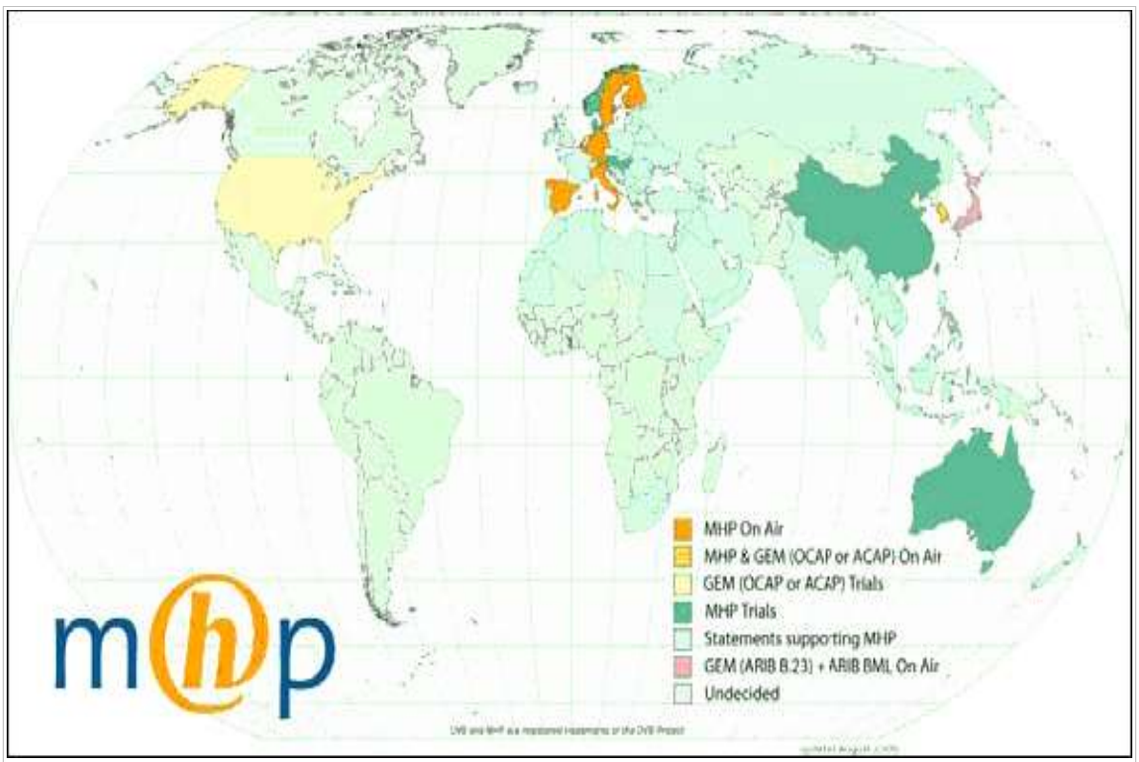

figura2.13 (Diffusione dell'MHP nel mondo)

Il DVB Project lavora sin dal 1997 alle specifiche relative ad uno standard aperto e flessibile per gestire applicazioni multimediali e interattive:MHP, Multimedia Home Platform.

Iniziano nel 1997 i lavori del progetto DVB-MHP, sviluppato in collaborazione con il Consorzio Europeo DVB (Digital Video Broadcasting) per la definizione di uno standard aperto e flessibile, basato su , Java™, che funga da un'interfaccia tra le applicazioni digitali interattive e i terminali di esecuzione (Set-Top Box, digital TV e PC multimediali) per la fruizione di servizi interattivi sullo schermo televisivo.Un fattore determinante la diffusione di applicazioni sul digitale è la compatibilità dello standard MHP con tutti e tre i sistemi di trasmissione del segnale digitale.

Lo scopo ultimo del progetto DVB-MHP è quello di standardizzare gli elementi della piattaforma interattiva (Set-Top Box, televisione digitale, ecc.), chiave del successo delle future applicazioni multimediali, per ridurre al massimo le incompatibilità tra le molteplici tecnologie e prodotti delle varie industrie e consorzi privati.La figura 2.13 mostra lo stato di diffusione dell'MHP nel mondo.

Come si può notare dalla cartina, Italia, Finlandia, Austria, Spagna, Germania,

Danimarca e Svezia mirano a sfruttare le potenzialità del Digitale terrestre e a fornire servizi interattivi on-air più o meno complessi.L'obbiettivo comune è comunque quello di distribuire un numero oneroso di decoder, raggiungere la piena standardizzazione dei terminali MHP e consolidare il mercato contingente partendo da un presupposto lungimirante che vede la piena convergenza tra radiodiffusione, telecomunicazioni e information technology.
Negli Stati Uniti si procede invece alla diffusione della televisione interattiva via cavo, proponendo in sostituzione allo standard MHP, l'equivalente OCAP (Open Cable Application Platform), ugualmente basato su Java e conforme allo standard GEM (Globally Executable MHP).
Quest'ultimo standard è stato creato per utilizzare applicazioni MHP anche in network non DVB, come quelli via cavo diffusi nel Nord America.
Attualmente, la Svezia sembra essere l'unico Paese che distribuisce offerte interattive MHP su tutte le piattaforme digitali (DVB-S, DVB-C, DVB-T).

2.2.3.1 Caratteristiche tecniche

Gli elementi essenziali per introdurre servizi interattivi/multimediali sono i seguenti:

- L'Electronic Programme Guide (EPG), la guida elettronica che indirizza l'utente alla scelta del programma all'interno del palinsesto.

- L'Application Programming Interface (API), il middleware del ricevitore che interagisce con il sistema operativo per interpretare e visualizzare le applicazioni.

Tali elementi, oggetto di soluzioni proprietarie, si sono in pratica tradotti in strumenti destinati a stabilire barriere tecnologiche alla libera scelta dei servizi pay a cui abbonarsi, favorendo l'instaurarsi di posizioni dominanti, oltre ad ostacolare lo sviluppo del mercato "consumer" dei ricevitori a causa del disorientamento generato nell'utenza.Sotto la spinta del nuovo mercato, caratterizzato dall'offerta di prodotti e servizi multimediali e interattivi, la scelta di una soluzione comune e "aperta" alla evoluzione tecnologica risulta una condizione essenziale per il successo del broadcasting digitale nel contesto

europeo.L'idea chiave dell'MHP è di definire una serie di caratteristiche minime che tutti gli apparecchi prodotti secondo le sue specifiche devono avere.
Il Multimedia Home Platform definisce una generica interfaccia tra le applicazioni e i terminali sui quali queste vengono eseguite (vedi figura 2.14).

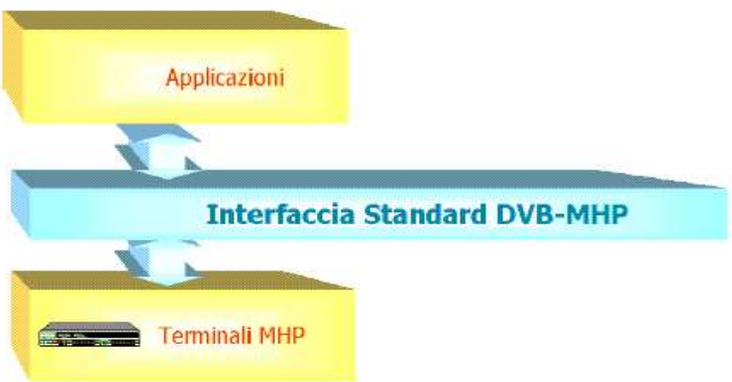

figura2.14(Interfaccia DVB-MHP)

Questa interfaccia divide le applicazioni dei diversi provider dalle specifiche hardware e software delle differenti implementazioni dei teminali MHP.Ciò permette ai fornitori di contenuti digitali di raggiungere coi loro servizi qualsiasi tipo di Set-Top Box (STB), di TV interattive integrate e PC multimediali.
DVB-MHP risiede in cima al sistema operativo di un STB, e rappresenta il traduttore locale che permette al STB di capire quali applicazioni e servizi devono essere attivati.
Durante la creazione di MHP, il DVB Project ha scelto di utilizzare Java per lo sviluppo del core del middleware.
Il modello architetturale di base di una piattaforma MHP è strutturato in tre parti (vedi figura 2.15):*resources*, *system software* e *applications*.

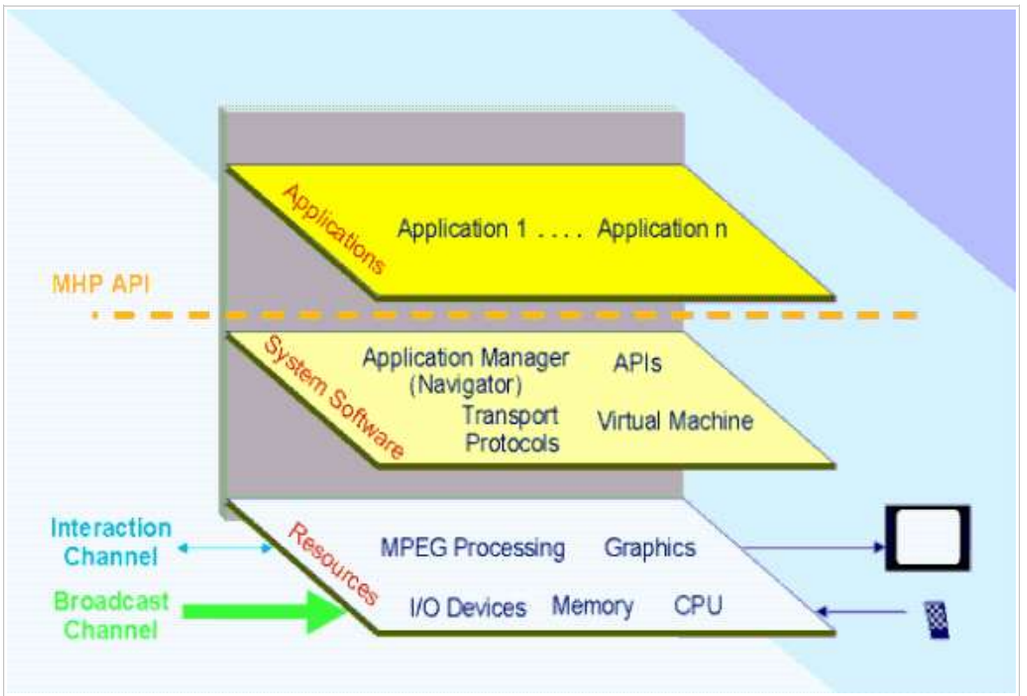

figura2.15 (strati architettura MHP)

- *Resources*: tipicamente le risorse hardware sono le seguenti: sintonizzatore, telecomando, demodulatore, hardware MPEG, modem, moduli di accesso condizionato, lettore, smart-card, processore e lettore DVD, Memorie(Ram Flash), Hard-disk.
- *System software*: Il software di sistema è dipendente dalla piattaforma ed è interamente sotto il controllo del relativo produttore.Esso ha accesso alle informazioni riguardanti uno stream audio-video in accordo con gli standard MPEG-2 e DVB.Inoltre è in grado di gestire tutte le risorse hardware (tuner, demultiplexer, accesso condizionato, CPU, RAM, hard disk, ecc.) e anche tutte le periferiche (modem, smart-card, lettore DVD, ecc.).Questo livello include una virtual machine e un numero di software APIs, basati su tecnologie Java, grazie ai quali le applicazioni MHP possono essere eseguite.Lo standard definisce anche gli strumenti per l'accesso condizionato (CA) includendo le API necessarie alla comunicazione con le CAM (Conditional Access Modules) che permetteno servizi come il pay-per-view o l' identificazione dell'utente.All'interno del software di sistema c'è l'Application Manager il quale è responsabile della gestione del ciclo di vita delle applicazioni: inizializzazione, esecuzione, pausa ed eliminazione di

un'applicazione.

- *Applications*: con il termine applicazione si fa riferimento ad una interfaccia tra un servizio interattivo, richiedente funzionalità specifiche proprie dell'hardware o del software del terminale e il display di visualizzazione dell'utente.La piattaforma MHP dispone di una gamma di applicazioni relative a ciascun profilo.

Il concetto di profilo è stato introdotto per venire incontro ad una situazione di utenza con esigenze molto differenziate e per proporzionare i costi degli apparati di utente.In particolare la piattaforma MHP definisce tre profili di ricevitore(vedi figura 2.16):

- il profilo *enhanced broadcasting*, che permette servizi di radiodiffusione avanzata per arricchire e completare i servizi televisivi di base con contenuti multimediali come brevi audio-video per notiziari, film, eventi sportivi, ecc. Inoltre tale profilo permette la trasmissione di servizi di EPG, teletext avanzato e giochi, memorizzandoli nella memoria del terminale d'utente;

- il profilo *interactive broadcasting* permette di aggiungere ai servizi del profilo enhanced broadcasting servizi di tipo interattivo con la possibilità per l'utente di interagire con un centro servizi attraverso un canale di ritorno.Sarà quindi possibile offrire servizi di tipo pay come la pubblicità interattiva, le transazioni (home-banking, commercio elettronico) ecc.In tale profilo la navigazione Internet è implicita, cioè codificata all'interno delle applicazioni e trasparente per l'utente;

- il terzo profilo è l'*Internet access Profile* che permetterà l'accesso Internet.Tale profilo offrirà la possibilità di accedere a servizi di tipo Internet come navigazione su siti Web e consentirà di effettuare transazioni commerciali del tipo e-commerce sfruttando i protocolli di sicurezza già sviluppati per Internet. L'esistenza di questo profilo rende lo standard MHP tecnicamente molto più potente delle piattaforme esistenti.

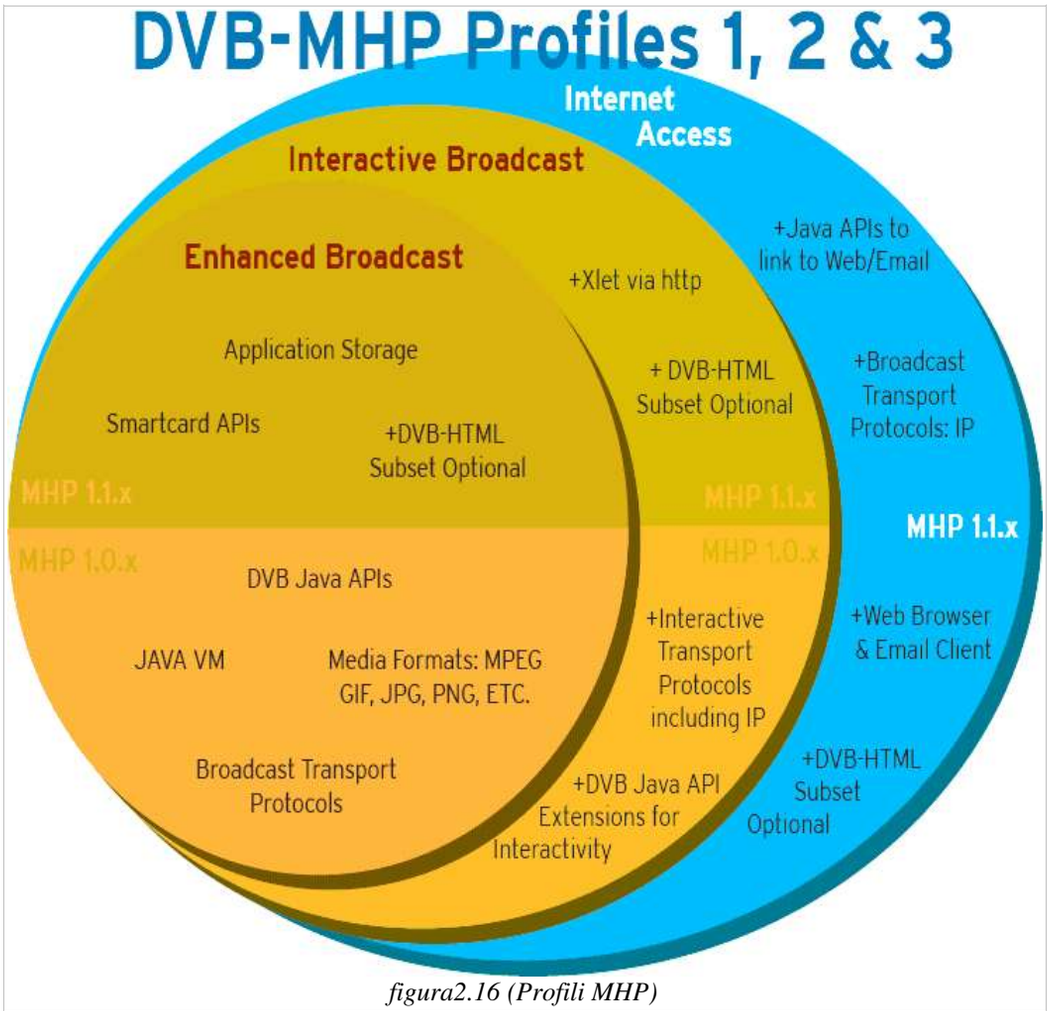

figura2.16 (Profili MHP)

Nello standard MHP le applicazioni interattive sono realizzate attraverso le Xlet. Una Xlet è una applicazione Java (simile ad una Applet) concepita per essere scaricata ed eseguita sui decoder interattivi MHP nei quali un software specifico, l'Application Manager, è in grado di controllarne il ciclo di vita.Mentre una Applet viene scaricata su “richiesta” dall’utente che carica la pagina HTML, ogni Xlet viene inviata in broadcast a tutti.L’utente riceve tutte le Xlet esistenti e tutti i dati associati (classi java, file dati, GIF, …).L’applicazione viene trasmessa ciclicamente e costantemente (broadcast carousel) e solitamente viene associata ad un particolare canale TV (tale associazione avviene nel Multiplexer).

La componente all’interno del decoder interattivo che si occupa effettivamente di caricare ed eseguire la Xlet è l’Application Manager o Navigatore.I suoi compiti

fondamentali sono:

- Segnalare all'utente la lista delle applicazioni MHP disponibili sul canale televisivo
- Caricare i componenti relativi all'applicazione

2.2.4 Dati su DVB (IP Data Broadcasting)

Sino ad ora sono state esposte principalmente le tematiche relative alla trasmissione di contenuti audio e video, codificati secondo lo standard MPEG2, via DVB.Tuttavia tale tecnologia rende possibile e desiderabile anche il trasporto di contenuti dati.Lo standard DVB-IP viene utilizzato per consentire la trasmissione di traffico TCP/IP all'interno dei pacchetti DVB.Perchè ciò sia possibile, i pacchetti TCP/IP dovranno essere incapsulati da un incapsulatore (detto anche Gateway) nel MPEG2 TS.I pacchetti di quest'ultimo hanno però una lunghezza fissa pari a 188bytes, ciò rende allora necessaria una procedura di frammentazione e riassemblaggio.Il primo passo di questo processo è il "framing" del datagramma IP mediante l'aggiunta di un header di incapsulamento e del CRC.Il datagramma può essere considerato come il "Protocol Data Unit" (PDU) e la relativa struttura dati comprensiva del nuovo header e CRC può essere chiamata "Sub Network Data Unit" (SNDU).L'ETSI propone quattro standard per trattare l'incapsulamento dei dati:

- Data Piping
- Asynchronous Data Streaming
- Synchronous Data Streaming
- Multi Protocol Encapsulation (MPE)

L'MPE è stato identificato come il metodo più idoneo da applicare ai datagrammi IP.In figura 2.17 è schematizzato lo stack dei protocolli DVB-IP.Come già accennato, nell'MPE i PDU sono sottoposti ad un processo di framing.Questa operazione comporta l'aggiunta di un "datagram section header" all'inizio della sezione (prima dell'header IP) e di un CRC alla fine, calcolato sull'intera sezione. Tale header contiene campi dichiaranti la lunghezza della sezione, lo stato di scrambling e l'indirizzo MAC del ricevitore.

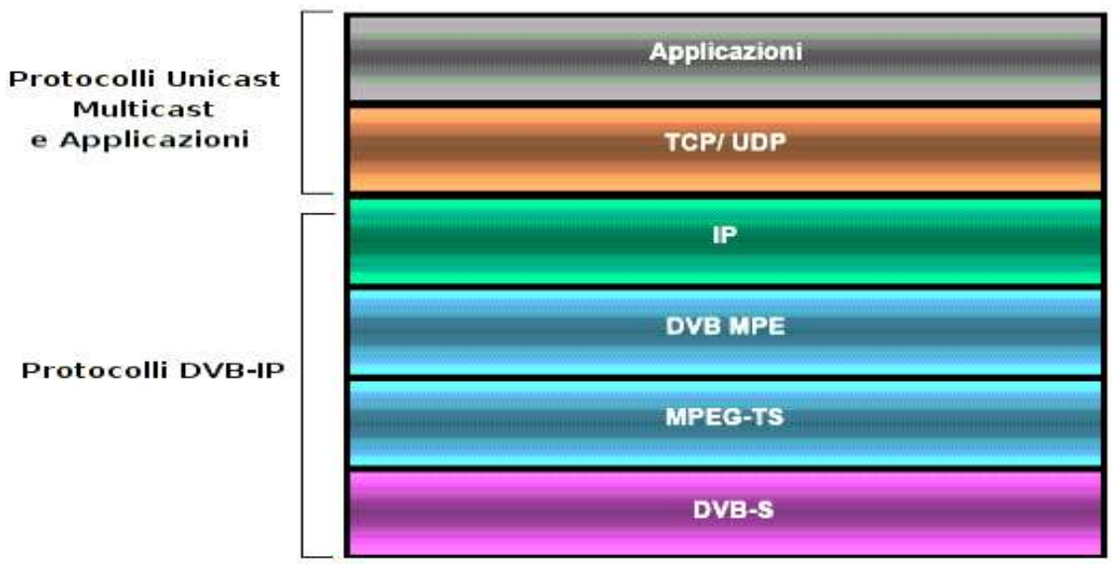

figura2.17(Stack dei protocolli DVB-IP)

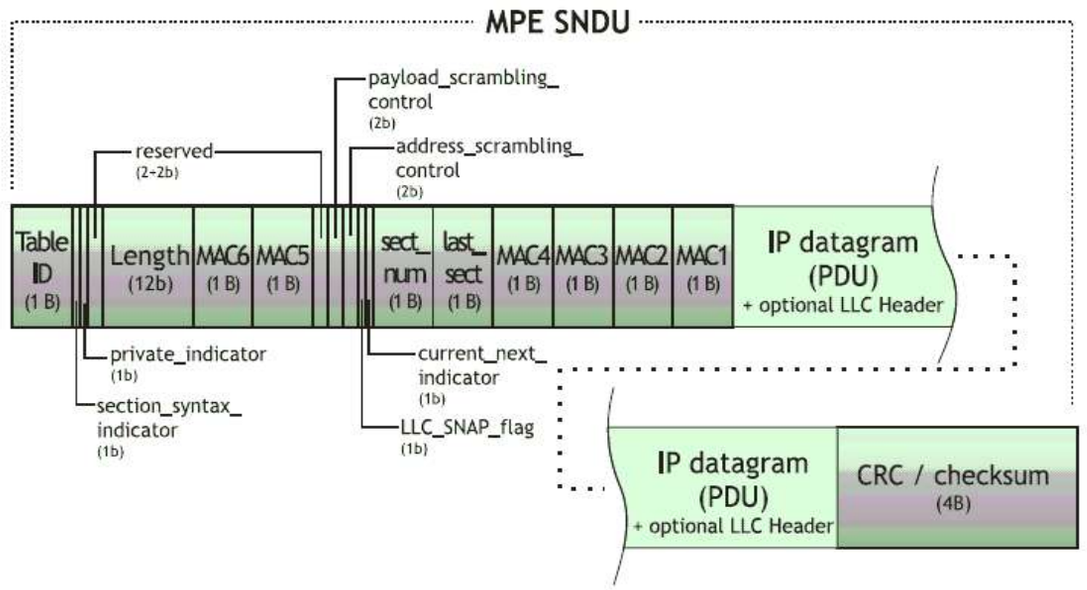

figura2.18(Struttura di un Datagram_Section, MPE SNDU)

Sono inoltre presenti molti campi definiti in MPEG (vedi figura 2.18) .

Una volta formata, la SNDU viene frammentata e inserita nel payload dei Transport Packet (TP) .Se la SNDU non ha una lunghezza pari ad un multiplo di 184 (il payload del TP), l'incapsulatore può operare ho un padding sulla parte rimanente del TP con bytes di stuffing o cominciarvi una nuova SNDU.

Il bit di “Payload Unit Start Indicator” (PUSI) , contenuto nell'header TP, dichiara

la presenza di un puntatore, localizzato subito dopo l'header, che punta l'inizio della nuova SNDU nel TP.

Il principale svantaggio dell'MPE è l'inclusione nell'header di alcuni specifici campi di MPEG, che potrebbero essere omessi.

La figura 2.19 riporta uno schema riassuntivo dell'intero processo.

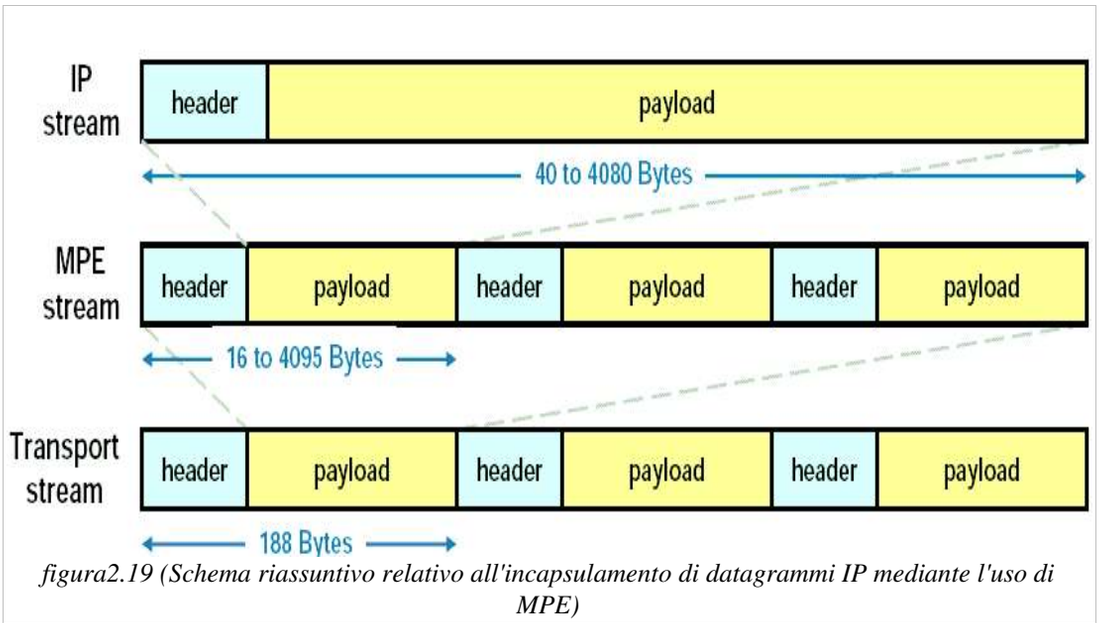

figura2.19 (Schema riassuntivo relativo all'incapsulamento di datagrammi IP mediante l'uso di MPE)

2.3 Componenti di un sistema di trasmissione DVB

2.3.1Centrale

La figura 2.20 mostra il diagramma generale del sistema DVB lato broadcaster.

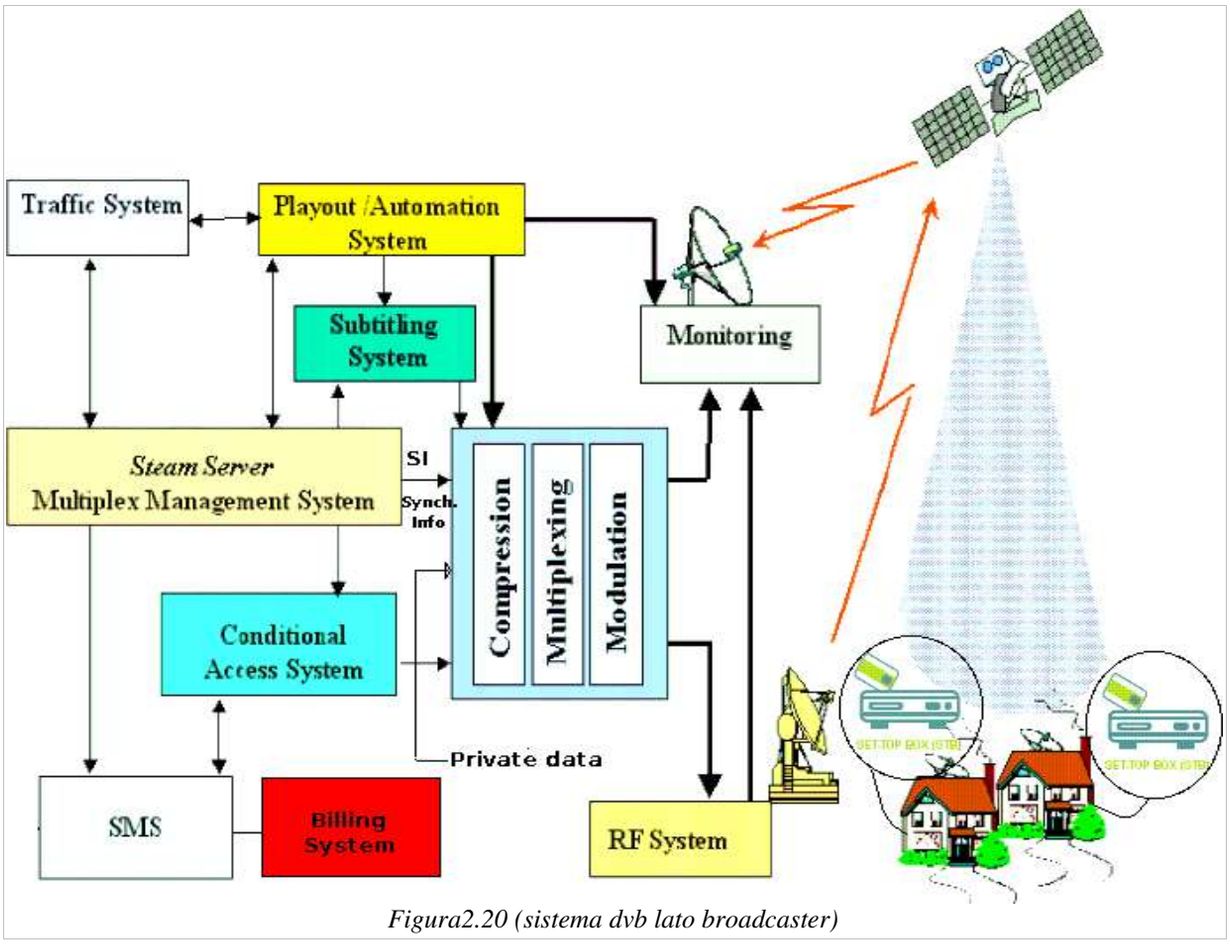

Figura2.20 (sistema dvb lato broadcaster)

Il blocco di *Automazione e Playout* comprende tutti i sistemi di immagazzinamento e recupero dei contenuti media (ad esempio hard disk magnetici, dvd etc.).I programmi e i dati da trasmettere sono recuperati dal centro di playout seguendo le indicazioni ricevute dal *Traffic System*, che rappresenta la principale guida e fonte per quanto riguarda le informazioni di programmazione.Il *Subtitling System* si occupa dell'inserimento dei sottotitoli nei vari programmi trasmessi.

I programmi e i dati da trasmettere sono poi passati al blocco di *Compressione*, che come facilmente intuibile dal nome si occuperà della codifica e compressione MPEG2 dei suoi canali di ingresso.

A valle del blocco di compressione si trova quello di *Multiplexing*, qui si procede

alla multiplazione e allo scrambling dei vari canali precedentemente codificati per formare il definitivo TS.

Come si nota dallo schema, anche le informazioni provenienti dal blocco di *Conditional Access System (CAS)* sono multiplate nel TS.Tali dati sono essenziali per garantire l'accesso ai programmi solo agli utenti autorizzati.Le autorizzazioni provengono al CAS dal *subscriber management system(SMS)*. Tipicamente questo blocco non è altro che un grande database, contenente le informazioni di Auhtentication-Authorization-Accounting (AAA) di tutti gli utenti.A sua volta l'SMS sarà direttamente collegato ad un *Billing System*, cioè al sistema che gestisce le fatturazioni per i programmi a pagamento.

Il TS è gestito e sincronizzato dal *Multiplex Management System (MMS)***.**Si nota come l'MMS sia direttamente collegato al Traffic System, che come detto in precedenza fornisce tutte le informazioni di programmazione dei vari eventi.

Nel TS vengono multiplati anche dei *Private Data*, che rappresentano uno stream di dati il cui contenuto non è specificato da MPEG e che può essere usato per trasportare ad esempio il Teletex, dati generici, informazioni di servizio addizionali specifiche di un particolare network o comandi finalizzati al controllo della modulazione e dei componenti di distribuzione della rete.

Per finire, il TS passa dal blocco *RF,* ove viene modulato e amplificato per la trasmissione finale.

Tutti gli elementi di tale sistema sono collegati insieme da connessioni ad alta velocità, come l'SDI (Serial Digital Interface) o l'ASI (Asynchronous Serial Interface), che sono degli standard nel settore televisivo.

Inoltre saranno connessi via ethernet ad un sistema di controllo e monitoraggio, il quale assicura che niente vada male e nel caso, fa in modo che l'utente non se ne accorga.Fondamentalmente per fare ciò si fa uso di blocchi che introducono ridondanza nel sistema, un esempio tipico è la presenza di più codificatori MPEG e multiplexer.

2.3.2 Set top box

I ricevitori DVB, indicati specialmente in ambito professionale con l'acronimo IRD (Integrated Reveiver Decoder), sono dispositivi alquanto complessi, essendo composti da una circuiteria analogica dedicata alla demodulazione del segnale ricevuto e da una parte digitale che può esser assimilata ad un vero e proprio computer.Infatti dal segnale ricevuto e demodulato si ottiene uno stream binario che viene a sua volta elaborato dalla circuiteria digitale.L'elaborazione del segnale digitale viene effettuata mediante un microprocessore e il software ad esso associato.Le funzionalità di base dei ricevitori DVB sono quindi:

- Sintonizzazione a RF
- Demodulazione digitale
- Demultiplexing MPEG-2
- Decodifica audio & Video MPEG-2
- Display e grafica su schermo
- Codifica PAL
- Applicazioni

Tutte queste funzionalità sono finalizzate alla conversione del segnale digitale DVB/MPEG2 in forma analogica.Qualsiasi servizio implementato dal ricevitore è controllato via software.Il STB può essere scomposto in tre livelli: la parte hardware che rappresenta il livello più basso, quello più alto relativo alle applicazioni e il livello intermedio di middleware.Il termine middleware è una caratterizzazione generica per tutti quei programmi che fanno da collante tra altri programmi gia esistenti, facendo in modo che le varie applicazioni possano comunicare.Una delle Application Program Interface (API) usata per l'interfacciamento con il livello più basso è l'Hardware Porting Kit (HPK), implementata anche dai ricevitori Sky.Fra le API usate per la seconda interfaccia, quella verso le applicazioni, si ricordano MHP e OpenTV.
Il mercato offre ricevitori di diverse tipologie, solitamente si incontrano box forniti di una memoria flash.L'intero stack software viene memorizzato in firmware.Questa è un'importante caratteristica dei STB che consente il completo aggiornamento del software anche da remoto, permettendo il miglioramento delle funzioni già realizzate e anche la realizzazione di funzioni non previste all'atto dell'immissione sul mercato.
I modelli più recenti sono provvisti anche di hard disk (hd), ne è un esempio il

nuovo STB della Sky *mySky,* fornito di un hd da 160Gb e in grado di registrare i programmi trasmessi via satellite.

2.3.2.1 Architettura hardware

Il diagramma di figura 2.21 riporta uno schema di massima dei compnenti hardware più importanti di un tipico STB.

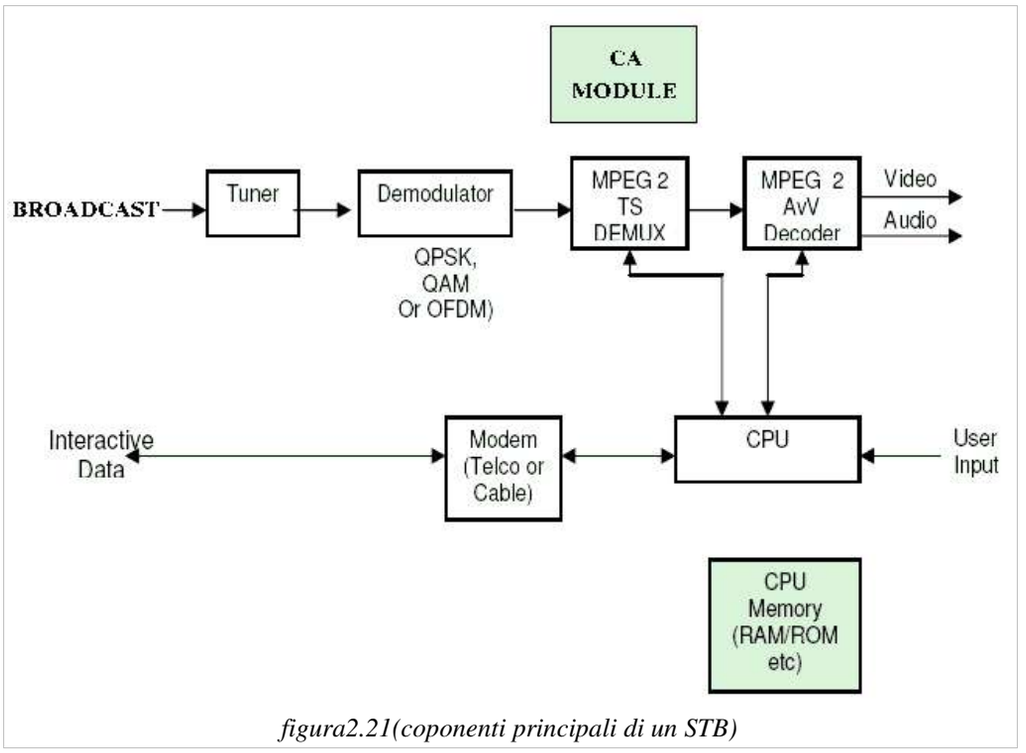

figura2.21(coponenti principali di un STB)

Il STB seleziona il canale TV di interesse mediante appropriata sintonizzazione su uno dei tanti canali di ingresso.Come visto in precedenza, il segnale è modulato usando una QPSK per la trasmissione via satellite, una QAM per quella via cavo e una OFDM per quella terrestre.Dal diagramma si deduce facilmente che il front end, contenente il sintonizzatore e il demodulatore, sarà diverso a seconda che la trasmissione avvenga via satellite, ripetitore terrestre o cavo.

L'informazione contenuta nel canale RF viene processata dal demodulatore per produrre un MPEG2-TS, contenente l'audio, il video e altre informazioni relative al programma TV selezionato.Questo passo prevede anche la demodulazione Reed Solomon e FEC.La circuteria del STB è spesso contenuta in un unico chip

integratio (soc).
Il STB di solito è anche in grado di mandare e ricevere dati interattivi.Ciò è reso possibile dalla presenza di un canale di ritorno che nei ricevitori satellitari e terrestri è tipicamente implementato da modem convenzionali, mentre per quelli relativi al DVB-C da cable modem.
In generale, la trasmissione di un programma TV può essere criptata.Il demultiplexer MPEG seleziona e decripta l'audio e il video compresso, usando le chiavi fornite dal conditional access sub system (CASS).
Successivamente il decoder MPEG, rispondente in pieno alle specifiche MPEG e ETR 154 DVB, decodifica i flussi audio e video del prgramma selezionato.

2.3.2.2 Architettura Software

La tipica architettura software di un STB è mostrata in figura 2.22.

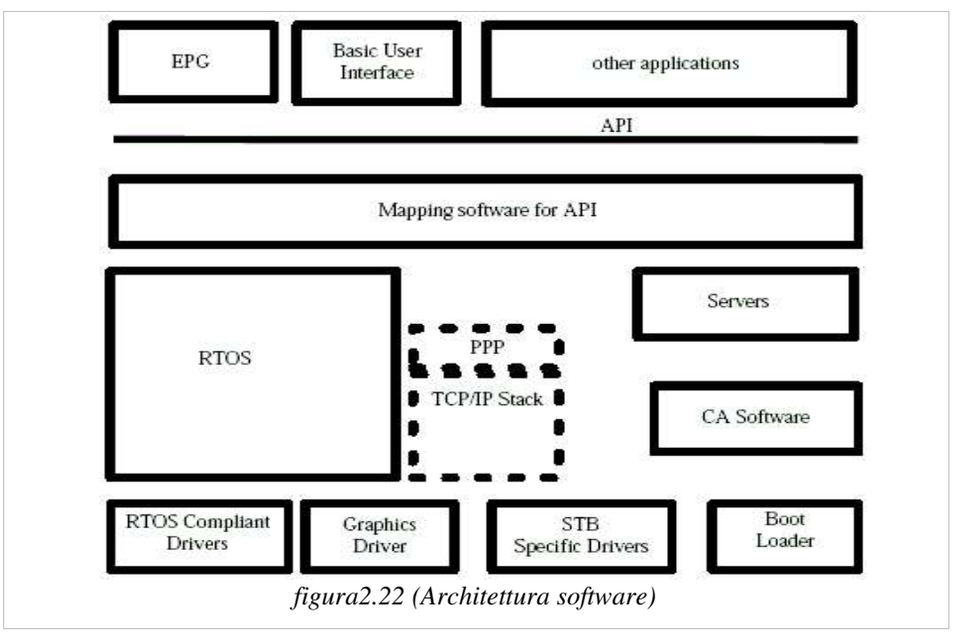

figura2.22 (Architettura software)

La struttura base del software può essere divisa in due livelli: il livello delle applicazioni, ove risiede l'EPG, posizionato sopra il livello composto da software che si interfacciano direttamente con l'hardware e il sistema operativo.Questa parte è formata da molte sezioni separate e per potervi accedere in maniera unificata e indipendente dall'hardware, viene fornito un Application Programming Interface (API).

Una minima forma di API è presente in ogni STB ed è solitamente sviluppata dalla casa madre.Ciò permette di costruire sopra di essa un EPG, che spesso rappresenta l'unico mezzo concesso all'utente per operare sul box.
I driver sono i moduli software dedicati al controllo diretto dell'hardware.
Vi possono essere elementi all'interno del STB sui quali i produttori stessi del box non hanno possibilità di controllo.L'esempio classico è costituito dal software dedicato al Conditional Access.
Tipicamente i produttori dei sistemi CA hanno il loro API proprietario per mezzo del quale si interfacciano con il software del STB.Sarà solo attraverso di esso che potrà avvenire l'accesso alle varie funzionalità supportate dal CA.
Altro elemento rilevante è il Boot Loader, responsabile dei controlli hardware durante il processo iniziale di boot.Inoltre installa la configurazione software del STB e costruisce un File Acces Table (FAT), cioè una mappa comprensiva di tutti gli oggetti di memoria presenti sul ricevitore.
Il sitema operativo è il cuore dell'architettura software e rappresenta le fondamenta su cui tutte le altre parti dei programmi vengono costruite.
Le necessità di memoria e la velocità del processore sono una grande restrizione per il STB.Per tali ragioni l'uso ipotetico di un pesante sistema operativo per PC non è appropiato.Migliore affidabilità rispetto a quella di un PC e performance real time sono inoltre necessarie (ad esempio il suo reboot nel caso di un errore del sistema operativo non è accettabile).Quindi è necessario un sistema operativo real time (RTOS).Quelli più comunemente usati sono VxWorks, eCos, RTLinux, DAVID OS-9, Nucleus e Windows CE.

Riferimenti bibliografici:

- *ETR 162: "Digital Video Broadcasting (DVB); Allocation of Service Information (SI) codes for DVB systems".*
- *EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".Known as (DVB-SI).*
- *Digital Interactive TV and Metadata (http://www.springer.com/0-387-20843-7)*
- *The DVB satellite, cable and SMATV systems: why the technical choices were made (J. Sesena Hispasat) (*www.ebu.ch/trev_266-sesena.pd*)*
- *Gli Standard DVB: dalla TV generalista ai servizi multimediali interattivi (Ing. Gianfranco Barbieri)* (www.crit.rai.it/eletel/2004-3/43-3.htm)
- *Simulation and Design of IP over DVB using* Multi-Protocol Encapsulation and Ultra Lightweight Encapsulation (nrg.cs.usm.my/~tcwan/Papers/NaCSPC05-SimulationULEvsMPE.pdf)
- *A new Approach to IP-based Transmission of Audio and Video Content via DVB-Networks (Heiko Follscher)*
- *Digital Entertainment Security Mechanisms DVB-Architecture* (www.ipvs.uni-stuttgart.de/.../seminare/WS0506/Reports/Muhammed_Awan_-_DVB_Architecture.pdf)
- *GEMINI IST-2001-33400 Scientific & Technology Framework (Mika Helsingius/ HUT)*
- *An extensible Set-Top-Box Architecture for interactive and broadcast Services offering sophisticated User Guidance (Frank Lonczewski, Rudolf Jaeger)*
- *Digital Set Top Box (STB) - Open Architecture/Interoperability Issues (B.Sundareshan)*

3. EPG (Guida Elettronica ai Programmi)

3.1 Service Information

Come già brevemente accennato alla fine del primo capitolo, la sezione di Service Information (SI) DVB, deifnisce un set di tabelle che estendono le capacità del System Layer di MPEG2.Queste tabelle sono aggiunte all'MPEG2 TS durante la fase di codifica e multiplexing.L'uso congiunto di quest'ultime con quelle PSI di MPEG2 (vedi figura 2.19) permettono al decoder di avere accesso a tutti i programmi disponbili su un'intera rete.Il documento tecnico di riferimento per il DVB-SI è l' “ETSI EN 300 468”.

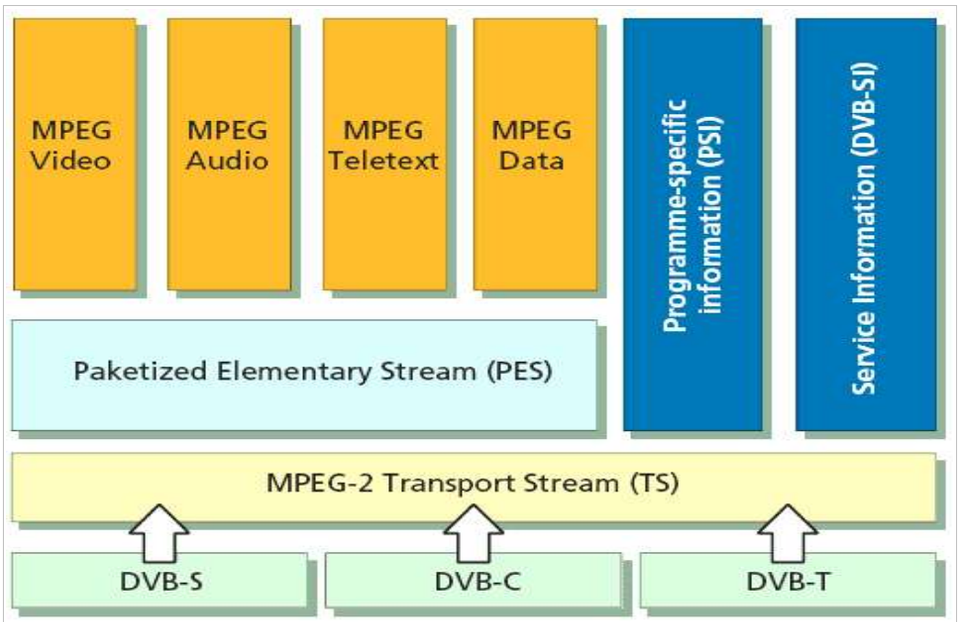

figure2.19 (Componenti sistem DVB/MPEG)

Le tabelle SI forniscono anche informazioni relative alla Guida Elettronica dei Programmi (Electronic Program Guide, EPG) , la quale mostra agli utenti una descrizione di tutti i programmi, inclusa la relativa ora di inizio e durata.Nella terminologia DVB si usa il termine “event” per indicare un programma.

Nel sistema DVB ogni tabella viene divisa in sezioni, ognuna delle quali fornita del proprio header e di alcuni loops di descrittori (il paragrafo successivo ne formisce un elenco dettagliato).Una tabella può quindi essere composta da una o più sezioni.Un apposito campo dell'header (section_number) è incrementato di uno ogni volta che viene trasmessa una nuova sezione.Considerando che tale campo è costituito da 8bit, se ne deduce che il numero massimo di sezioni è

256.La loro lunghezza per tutte le tabelle è di 1021byte, tranne che per la Event Information Table (EIT) che è 4096.
Come tutti gli altri pacchetti del TS, quelli contenenti le tabelle SI sono definiti da uno specifico PID.

3.1.2 Tabelle SI

Si riporta un elenco delle tabelle e rleativi PID, appartenenti al DVB SI.

TABELLE SI	**PID**
Network Information Table (NIT) - mostra l'organizzazione fisica della rete e le sue caratteristiche	0x0010
Time and Data Table (TDT) - fornisce l'attuale orario UTC	0x0014
Service Decsription Table (SDT) - descrive i servizi contenuti nella rete e il nome del service provider	0x0011
Event Information Table (EIT) - definisce tutti gli eventi nella rete, fornendone la descrizione, ora di inizio e durata.E' usata nella creazione dell'EPG	0x0012
Bouquet Association Table (BAT) - descrive i servizi disponibili in un detertminato *bouquet* (gruppo di servizi che possono essere venduti come un singolo prodotto)	0x0011
Running Status Table (RST) - aggiorna i riferimenti temporali degli eventi quando avvengono cambiamenti nella programmazione	0x0013
Time Offset Table (TOT) - contiene l'orario e data UTC e l'offset dell'ora locale	0x0014
Stuffing Table (ST) – provvede ad invalidare le rimanenti sezioni di una tabella quando una sezione è stata sovrascritta	da 0x0010 a 0x0014

Si procede ora ad una panoramica più dettagliata di queste tabelle, fornendone una rapida descrizione dei contenuti e delle funzionalità.

- **Time and Data Table (TDT)**:

Syntax	Number of bits	Identifier
time_date_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
UTC_time	40	bslbf
}		

fornisce l'ora e data attuale UTC, la quale può essere modificata in accordo con il fuso orario e presentata sullo schermo dell'utente.Il table_id ha valore 0x70.

- **Network Information Table (NIT)**:

Syntax	Number of bits	Identifier
network_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
network_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
network_descriptors_length	12	uimsbf
for(i=0;i<N;i++){ Network descriptor loop		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0;j<N;j++){ Transport stream loop		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

questa tabella contiene informazioni sulle caratteristiche della rete e mostra l'organizzazione fisica dei TS che trasporta.Ognuna delle reti descritte è identificata dal proprio *network_id*.Il campo *table_id* può avere due valori: 0x40 per le sezioni relative al network attualmente utilizzato e 0x41 per gli altri.Il bit di *current_next_indicator* consente alla NIT di segnalarsi come non ancora valida (se settato a 0).Il Network descriptor loop trasporta informazioni aggiuntive sulla rete.Tali dati sono contenuti in sotto tabelle chiamate descrittori.Ad esempio il *linkage_descriptor* fornisce collegamenti ad altri servizi che descrivono ulteriormente la rete.Il Transport Stream loop identifica ogni TS trasportato e la sua rete originaria (infatti l'uso congiunto dell'*original_network_id* e del *transport_stream_id* permette di identificare univocamente ogni TS).Un loop di Transport Stream più interno mostra inoltre i dettagli della rete di ogni TS.Ad esempio se il set top box è sintonizzato su un determinato canale a RF e l'utente richiede un servizio su un'altro canale, la relativa frequenza è contenuta in questo loop.

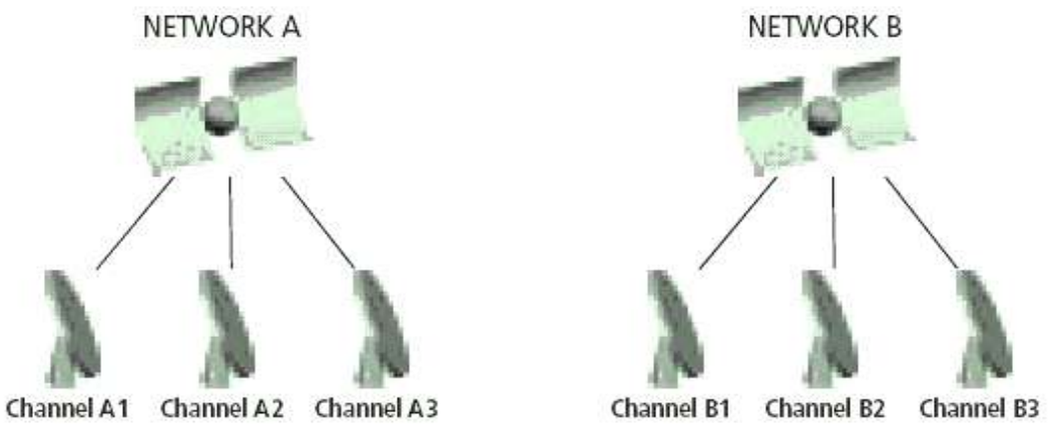

figure2.20 (NIT:organizzazione della rete e informazioni di sintonizzazione dei relativi canali)

- **Service Description Table (SDT)**:

Syntax	Number of bits	Identifier
service_description_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
original_network_id	16	uimsbf
reserved_future_use	8	bslbf
for (i=0;i<N;i++){		
service_id	16	uimsbf
reserved_future_use	6	bslbf
EIT_schedule_flag	1	bslbf
EIT_present_following_flag	1	bslbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for (j=0;j<N;j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Service loop

definisce i servizi disponibili sulla rete e il nome del service provider.Un servizio è una sequenza di eventi che possono essere trasmessi come parte di un programma.DVB richiede due tipi di STD, “Actual” (*table_id*=0x42) e “Other” (*table_id*=0x46) .La prima descrive i servizi fruibili dall'utente sull'attuale TS, mentre la seconda quelli disponibili sugli altri TS della rete.L'header include informazioni generali come l'id del TS e della rete originaria.Il Service loop identifica ogni servizio (*service_id) ,* indica se sono disponibili informazioni EIT (*EIT_schedule_flag*), se il dato servizio è partito o meno (*running_status*) etc.Il service descriptor loop più interno contiene informazioni più dettagliate su ogni servizio listato nel loop esterno.

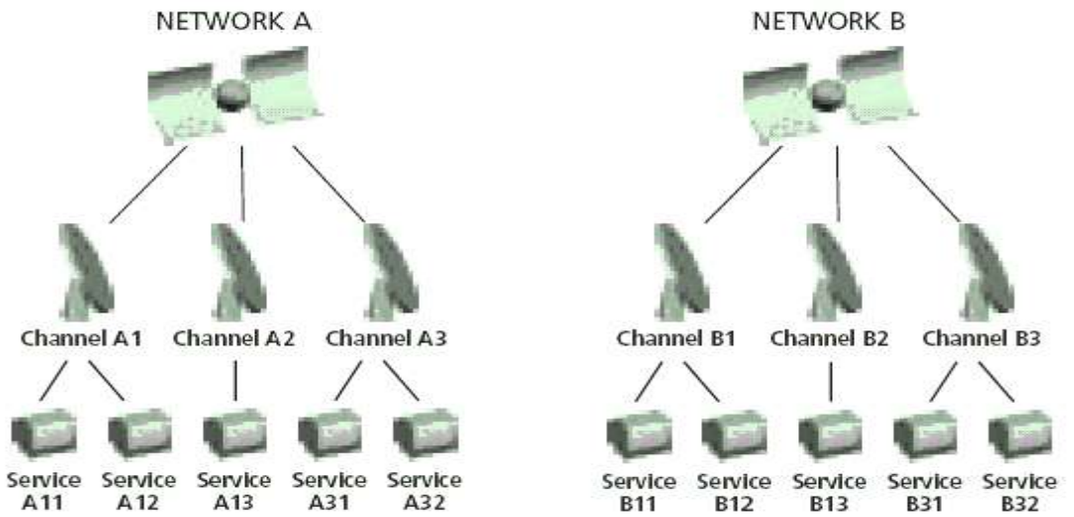

figura2.21 (SDT:servizi disponibili sulla rete)

- **Event Information Table (EIT)**:

Syntax	Number of bits	Identifier
event_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
service_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
segment_last_section_number	8	uimsbf
last_table_id	8	uimsbf
for(i=0;i<N;i++){		
event_id	16	uimsbf
start_time	40	bslbf
duration	24	uimsbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Event loop

elenca tutti gli eventi presenti nella rete, includendo la loro descrizione, ora di inizio e durata.In accordo con la definizione di MPEG un evento è un insieme di ES con una base dei tempi comune.Spesso si fa riferimento agli eventi con il termine programma TV.Possono essere trasmesse simultaneamente quattro tipi differenti di EIT: ”Actual Transport Stream present/following" (*table_id*=0x4E), “Other Transport Stream present/following” (*table_id*=0x4F), ”Actual Transport Stream schedule” (*table_id*=da 0x50 a 0x5F), “Other Transport Stream event schedule” (*table_id*=da 0x60 a 0x6F).

Le EIT present/following contengono informazioni sull'evento corrente e quello cronologicamente successivo, sull'attuale TS (Actual) o sugli altri (Other).

Le EIT schedule invece listano una più lunga programmazione degli eventi.La EIT schedule rappresenta la maggior fonte di informazioni per l'EPG.Le EIT present/following sono distribuite su due specifiche sezioni:quella con *section_number*=0x00 è riservata alla present mentre quella con *section_number*=0x01 alla following.Considerato il fatto che le schedule coprono invece diversi giorni, la loro organizzazione è più complessa.Le 256 sezioni sono divise in 32 segmenti formati da 8 sezioni.Ogni segmento contiene le informazioni sugli eventi per un periodo di tre ore, con il segmento 0 della table_id=0x50 che copre gli eventi dalle ore 24 alle 02:59:59 etc.Così che i 32 segmenti della tabella 0x50 coprono un periodo di 4 giorni.

Data la disponibilità di 16 table_id, la programmazione EIT può coprire un arco temporale di 64 giorni.Le table_id e le section_number sono ordinate cronologicamente.Il service_id, trasnsport_stream_id e l'original_network_id identificano rispettivamente il servizio, il TS e la rete a cui le sezione dell'EIT appartengono.L' event loop lista tutti gli eventi descritti da questa sezione, la loro ora di inizio, durata etc.Il loop di descrittori più interno, associato ad ogni evento, trasporta informazioni più dettagliate.

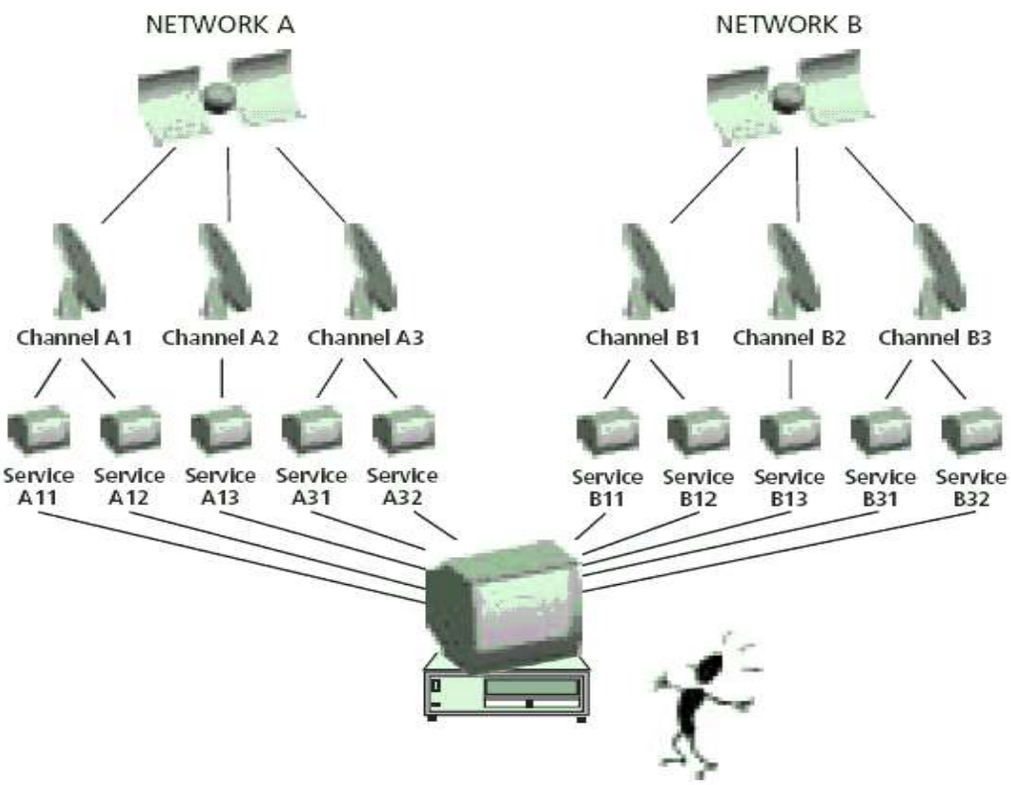

figura2.22 (EIT: informazioni sugli eventi usate per creare l'EPG)

- **Timing Offset Table (TOT)**:

Syntax	Number of bits	Identifier
time_offset_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
UTC_time	40	bslbf
reserved	4	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

la TOT (*table_id*=0x73) contiene la data e ora UTC e l'offset locale.L'offset indica le ore da aggiungere o sottrarre al riferimento UTC per allinearsi al fuso orario locale.

- **Bouquet Association Tables (BAT)**:

Syntax	Number of bits	Identifier
bouquet_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
bouquet_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
bouquet_descriptors_length	12	uimsbf
for(i=0;i<N;i++){ Bouquet descriptor loop		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0;j<N;j++){ Transport stream loop		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

un bouquet è un'offerta commerciale o un gruppo di servizi che possono essere venduti come un singolo prodotto.La BAT descrive i servizi disponibili in un dato bouquet.Il *table_id* è 0x4A.Il campo *bouquet_id* indica quale bouquet viene descritto.Nel bouquet descriptor loop vi sono poi i descrittori contenenti informazioni più specifiche.Il Transport Stream loop lista tutti i TS rilevanti per il dato bouquet.Nel Transport Stream loop più interno vengono listati tutti i servizi presenti nel TS.

La BAT permette di raggruppare nell'EPG i servizi di interesse comune.Ad esempio per un bouquet di canali sportivi, il Transport Stream loop indica tutti i TS contenenti sport, mentre il secondo loop lista tutti i servizi sportivi contenuti

in ogni TS.L'utente può selezionare l'icona di un determinato servizio sportivo, consultare la lista degli eventi e scieglierne uno.

- **Running Status Table (RST)**:

Syntax	Number of bits	Identifier
running_status_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
for (i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
service_id	16	uimsbf
event_id	16	uimsbf
reserved_future_use	5	bslbf
running_status	3	uimsbf
}		
}		

questa tabella trasporta le informazioni usate per aggiornare i riferimenti temporali nel sistema quando avvengono cambiamenti nelle programmazioni.Ciò fa risparmiare al broadcaster di ritrasmettere un'intera tabella se solo una sua porzione è stata modificata.Il *table_id* ha valore 0x71.

- **Stuffing Table (ST)**:

Syntax	Number of bits	Identifier
stuffing_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
for (i=0;i<N;i++){		
data_byte	8	uimsbf
}		
}		

la ST è usata per invalidare le rimanenti sezioni in una tabella una volta che la sezione stessa è stata sovrascritta.Ciò permette di mantenere l'integrità del

campo section_number.

Lo schema riassuntivo di figura 2.23 riporta un esempio dei contenuti informativi sino ad ora trattati di un tipico TS.

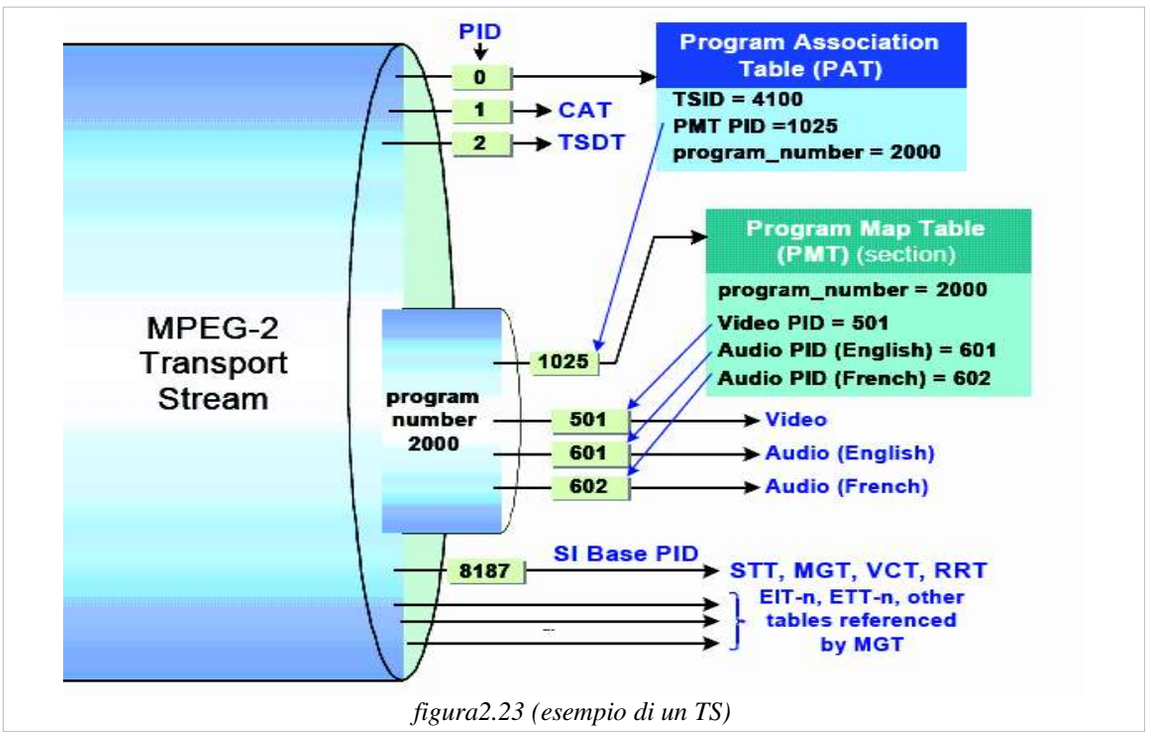

figura2.23 (esempio di un TS)

3.1.3 Descrittori

Le varie tabelle SI hanno una struttura molto simile.

Tutte sono formate da un header seguito da uno o più loop.Ognuno di loro include una serie di descrittori che, come già intuibile dal nome, provvedono a fornire ulteriori informazioni relative all'oggetto cui sono legati.

Ogni descrittore viene identificato da uno specifico campo di 8 bit, il *descriptor_tag*, mentre il campo *descriptor_length*, sempre di 8 bit, specifica il numero totale di di byte della porzione dati immediatamente seguenti il byte descriptor_length.La tabella sottostante riassume la struttura generale di un descrittore.

nome del campo	**numero di bit**
descriptor_tag	8
descriptor_length	8
for (i=0, i<descriptor_length; i++) {	
data byte	8*N
}	

Nella tabella di pagina seguente si riporta un elenco completo dei descrittori definiti dalla DVB-SI.

Per ognuno di essi viene indicato il valore del descriptor_tag e il collocamento nelle tabelle SI.Per una dettagliata descrizione di ogni descrittore si rimanda al documento ETSI EN 300 468.

Descriptor	Tag value	NIT	BAT	SDT	EIT	TOT	PMT	SIT*
network_name_descriptor	0x40	●	-	-	-	-	-	-
service_list_descriptor	0x41	●	●	-	-	-	-	-
stuffing_descriptor	0x42	●	●	●	●	-	-	●
satellite_delivery_system_descriptor	0x43	●	-	-	-	-	-	-
cable_delivery_system_descriptor	0x44	●	-	-	-	-	-	-
reserved for future use	0x45	-	-	-	-	-	-	-
reserved for future use	0x46	-	-	-	-	-	-	-
bouquet_name_descriptor	0x47	-	●	●	-	-	-	●
service_descriptor	0x48	-	-	●	-	-	-	●
country_availability_descriptor	0x49	-	●	●	-	-	-	●
linkage_descriptor	0x4A	●	●	●	●	-	-	●
NVOD_reference_descriptor	0x4B	-	-	●	-	-	-	●
time_shifted_service_descriptor	0x4C	-	-	●	-	-	-	●
short_event_descriptor	0x4D	-	-	-	●	-	-	●
extended_event_descriptor	0x4E	-	-	-	●	-	-	●
time_shifted_event_descriptor	0x4F	-	-	-	●	-	-	●
component_descriptor	0x50	-	-	-	●	-	-	●
mosaic_descriptor	0x51	-	-	●	-	-	●	●
stream_identifier_descriptor	0x52	-	-	-	-	-	●	-
CA_identifier_descriptor	0x53	-	●	●	●	-	-	●
content_descriptor	0x54	-	-	-	●	-	-	●
parental_rating_descriptor	0x55	-	-	-	●	-	-	●
teletext_descriptor	0x56	-	-	-	-	-	●	-
telephone_descriptor	0x57	-	-	●	●	-	-	●
local_time_offset_descriptor	0x58	-	-	-	-	●	-	-
subtitling_descriptor	0x59	-	-	-	-	-	●	-
terrestrial_delivery_system_descriptor	0x5A	●	-	-	-	-	-	-
multilingual_network_name_descriptor	0x5B	●	-	-	-	-	-	-
multilingual_bouquet_name_descriptor	0x5C	-	●	-	-	-	-	-
multilingual_service_name_descriptor	0x5D	-	-	●	-	-	-	●
multilingual_component_descriptor	0x5E	-	-	-	●	-	-	●
private_data_specifier_descriptor	0x5F	●	●	●	●	-	●	●
service_move_descriptor	0x60	-	-	-	-	-	●	-
short_smoothing_buffer_descriptor	0x61	-	-	-	●	-	-	●
frequency_list_descriptor	0x62	●	-	-	-	-	-	-
partial_transport_stream_descriptor	0x63	-	-	-	-	-	-	●
data_broadcast_descriptor	0x64	-	-	●	●	-	-	●
CA_system_descriptor †	0x65	-	-	-	-	-	●	-
data_broadcast_id_descriptor	0x66	-	-	-	-	-	●	-
Reserved for future use	0x67 to 0x7F							
user defined	0x80 to 0xFE							
Forbidden	0xFF							

3.2 Uso delle PSI/SI

L'EPG (Electronic Programme Guide) è una funzione che conferisce reale valore aggiunto al servizio di televisione digitale rispetto all'analogico. L'EPG offre infatti all'utente una guida aggiornata in tempo reale dei palinsesti.Esso permette di avviare la ricezione del programma scelto, navigando all'interno del "bouquet" e di ottenere informazioni aggiuntive sull'evento (nome del regista, attori, trama, ecc.) direttamente sullo schermo utilizzando il telecomando.Tramite l'EPG l'utente può anche conoscere e selezionare eventi a pagamento (pay-per-view), o eventi in genere soggetti a controllo d'accesso.Nel DVB, come visto nel precedente capitolo, è il sistema stesso a trasmettere tutte queste informazioni di servizio (tabelle PSI/SI).Dal lato utente tali dati saranno memorizzati nel ricevitore e usati per implementare l'EPG.Ogni STB propone la propria particolare interfaccia per la presentazione di queste informazioni.

E' di particolare interesse sottolineare il caso di Sky, un classico esempio di sistema a struttura verticale, proprietario.Sky infatti ha definito delle proprie tabelle di EIT schedule, differenti da quelle DVB, mentre le EIT P/F sono le stesse.Ovviamente questa scelta ha portato allo sviluppo di un apposito STB Sky, fornito all'utente al momento della stipulazione dell'abbonamento.

Dell'EPG sono previste due presentazioni:

- la prima, essenzialmente testuale, denominata "*banner*" o *"zapper",* utilizza il protocollo DVB-SI, e costituisce l'interfaccia utente base per il STB, semplice, essenziale e con minimi requisiti di memoria; la descrizione del palinsesto fornisce essenzialmente informazioni sul programma in onda e su quello successivo (N o w / N e x t).Solitamente il banner appare in sovraimpressione non appena ci si sintonizza su di un canale ;
- la seconda, di tipo multimediale, si basa sulla piattaforma interattiva(MH P, openTV etc) e offre all'utente un servizio più evoluto sia per l'interfaccia grafica di presentazione sia per la modalità di gestione dei contenuti (foto, animazioni, preview, ecc.). Facilita inoltre l'accesso alla programmazione televisiva, su base giornaliera o periodica, consentendo all'utente di personalizzare le modalità di fruizione dei servizi secondo i propri gusti.L'EPG, nella versione multimediale "aperta" basata sulla piattaforma interattiva, include il Navigatore e costituisce lo strumento più adatto per introdurre e

gestire l'intera famiglia di nuovi servizi che la tecnologia digitale rende disponibili, lasciando all'editore la massima libertà operativa e garantendo all'utente l'accesso all'EPG fornito dai vari gestori.

L'EPG fa riferimento principalmente alle informazioni contenute nelle tabelle EIT, SDT e BAT.Per meglio comprendere come l'EPG faccia uso delle PSI/SI, si riporta un semplice esempio nel quale l'utente decide di visualizzare un particolare evento sportivo.Con riferimento alla figura 3.1, si parte da una schermata EPG che mostra una serie di icone di eventi appartenenti a diverse categorie (sport, film, etc.).

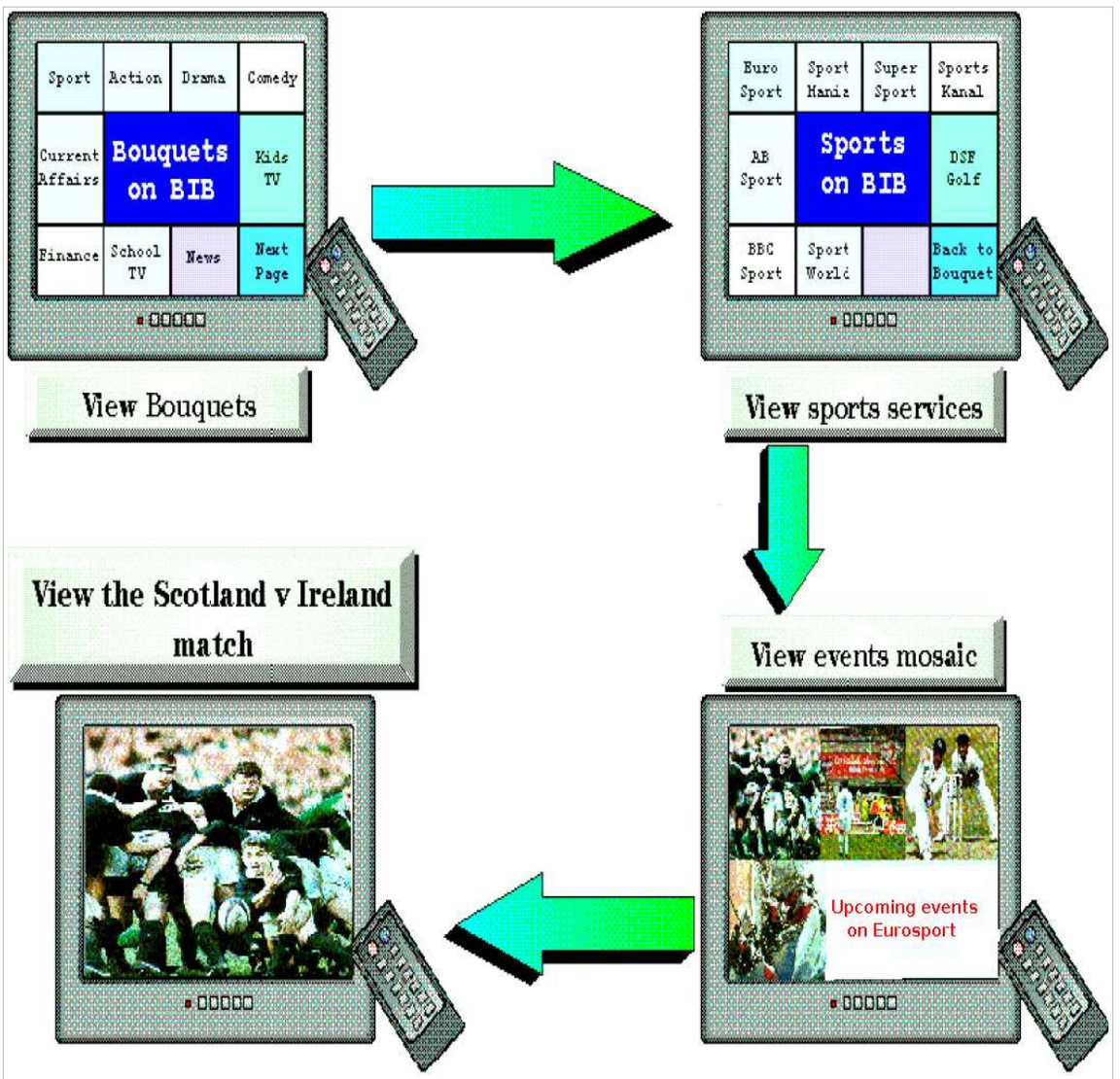

figura3.1 (esempio EPG)

Per la creazione di tali icone si è fatto uso dei *Bouquet_name_descriptors* estratti dalle BAT (tale descrittore è contenuto nel primo loop della BAT).Si assume che venga scelto un bouquet della categoria sport.

- **Uso della BAT per vedere se "Eurosport" è disponibile**:

Una volta che si è selezionata una BAT, si passa alla seconda schermata. Quest'ultima fa uso del *service_list_descriptor* (vedi tabella sottostante) per identificare tutti i servizi disponibili nel bouquet mediante un *service_id* e *service_type*.

Syntax	Number of bits	Identifier
service_list_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i=0;i<N;I++){		
service_id	16	uimsbf
service_type	8	uimsbf
}		
}		

Tabella3.1 (service_list_descriptor)

Il *service_list_descriptor* si trova nel secondo loop della BAT e come detto lista i servizi che appartengono al bouquet scelto per ogni TS.Si suppone ora che l'utente decida di vedere Eurosport.L'EPG recupera il transport_stream_id relativo (contenuto nel transport_stream_loop della BAT) e accede alla NIT.
Ora l'EPG cerca nel transport_stream loop il transport_stream_id identificato in precedenza, poi scorre il transport_stream_descriptor_loop per estrarvi il frequency_descriptor.Grazie alle informazioni contenute in questo descrittore l'STB può sintonizzarsi sul corretto canale rf.

- **Uso della SDT per vedere cosa offre "Eurosport"**:

A questo punto, una volta che l'STB si è sintonizzato sul canale contenente il TS cercato, mediante l'uso del service_id, estratto a sua volta dal service_list_descriptor (vedi tabella3.1) della BAT di Eurosport, l'EPG cerca i pacchetti con PID=0x11 e Table_id=0x42: SDT attuale, che, come indicato nel precedente capitolo, descrive tutti i servizi disponibili sul corrente TS.
Nel service loop identifica il corretto service_id.L'EPG, se riscontra il campo a un bit *EIT_schedule_flag* settato a uno, comunica all'utente che è disponibile una

Event schedule (programmazione).

Inoltre dal controllo del campo a 3 bit *running_status* ne verifica lo stato.Supponendo che sia attivo (*running_status*=4), l'EPG lo rende noto all'utente.Dal service descriptor loop, decodifica il *Mosaic_descriptor* (tabella 3.1) e crea una schermata multi cella che mostra una selezione degli eventi su Eurosport (vedi terza schermata di figura3.1).Il *Mosaic_descriptor* organizza le informazioni reltive ai vari eventi in modo tale che, quando visualizzate, appaiano su delle apposite aree dello schermo, dette celle.Questo descrittore provvede al partizionamento della componente video in celle elementari, all'allocazione di quet'ultime con celle logiche e per finire assegna un link fra il loro contenuto e le informazioni corrispondenti (ad esempio bouquet, servizio, eventi etc.).

Syntax	Number of bits	Identifier
mosaic_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
mosaic_entry_point	1	bslbf
number_of_horizontal_elementary_cells	3	uimsbf
reserved_future_use	1	bslbf
number_of_vertical_elementary_cells	3	uimsbf
for (i=0;i<N; i++) {		
logical_cell_id	6	uimsbf
reserved_future_use	7	bslbf
logical_cell_presentation_info	3	uimsbf
elementary_cell_field_length	8	uimsbf
for (i=0;j<elementary_cell_field_length;j++) {		
reserved_future_use	2	bslbf
elementary_cell_id	6	uimsbf
}		
cell_linkage_info	8	uimsbf
If (cell_linkage_info ==0x01){		
bouquet_id	16	uimsbf
}		
If (cell_linkage_info ==0x02){		
original_network_id	16	uimsbf
transport_stream_id	16	uimsbf
service_id	16	uimsbf
}		
If (cell_linkage_info ==0x03){		
original_network_id	16	uimsbf
transport_stream_id	16	uimsbf
service_id	16	uimsbf
}		
If (cell_linkage_info ==0x04){		
original_network_id	16	uimsbf
transport_stream_id	16	uimsbf
service_id	16	uimsbf
event_id	16	uimsbf
}		
}		
}		

Tabella3.1(mosaic_descriptor)

- **Uso della EIT per sciegliere un evento su "Eurosport"**:

Supponiamo ora che l'utente decide di vedere la partita internazionale di Rugby Scozia-Irlanda.L'EPG estrae l'event_id dal mosaic_descriptor e cerca i pacchetti con PID=0x12 e Table_id=0x4E: EIT, programmazione present/following del TS corrente.L'EPG fa uno scansione degli header delle sezioni EIT sino a quando non trova corrispondenza con il service_id già memorizzato.Conoscendo l'event_id, entra nell'event loop corretto e lista le informazioni reltive all'evento: ora di inizio, durata, descrizione della partita etc.).

- **Visione dell'evento e pagamento**:

Una volta che l'utente decide per questo evento, il service_id (identico al program_number della PAT) è usato per cercare la PMT di Eurosport.Per fare ciò, l'EPG consulta la PAT (PID=0x0000) e ricava il PID della PMT relativa al service_id (program_number) di interesse.La fase di decodifica e visualizzazione può ora iniziare.
Usando poi l'original_network_id (recuparata dalla BAT), l'EPG può automaticamente connettersi con l'ente originale di broadcasting (per fare ciò si usano le informazioni del *telephone_descriptor* nell'event_descriptor loop) e provvedere al pagamento.

Lo scenario appena riportato è solo uno dei tanti che il realizzatore dell'EPG dovrà tenere in considerazione al momento della sua progettazione.Si è scelto questo perchè fa uso dei concetti di Bouquet e Mosaic.La cosa importante da sottolineare è la predominanza delle tabelle SI.Sono loro a dire al decoder cosa decodificare.Le tabelle PSI, nel contesto EPG, spesso ricoprono un ruolo secondario rispetto alle SI.

3.3 Architettura di un sistema EPG

Di seguito si riporta uno schema generale di un architettura EPG.

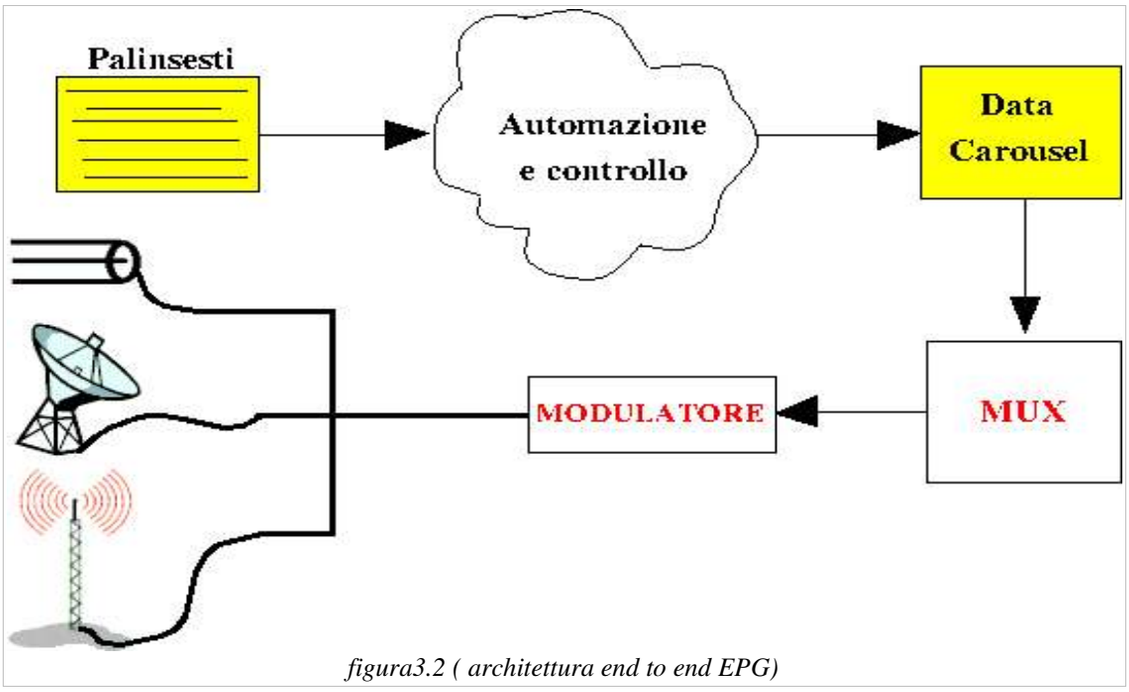

figura3.2 (architettura end to end EPG)

Con il termine palinsesto si indica l'insieme delle trasmissioni programmate da una emittente per un certo periodo di tempo(un giorno, una settimana, un mese...).Il broadcaster può acquisire tali informazioni esternamente, nel caso si occupi della trasmissione di canali appartenti ad altri enti, oppure, se è lui stesso il produttore dei contenuti, può generarle localmente.

Il blocco di “Automazione e Controllo”, a partire da tali dati, provvede alla creazione delle relative tabelle di EIT.Ovviamente con lo scorrere dei programmi tali tabelle dovranno essere aggiornate.In particolar modo per la present/following EIT, dovrà essere previsto un sistema di controllo che, memorizzata la durata dell'evento attualmente in onda, faccia scattare un timer, il quale raggiunta tale durata, avvii la procedura di aggiornamento della EIT.L'aggiornamento non farà altro che listare come evento present quello che in precedenza era indicato come following e assegnare tale etichetta a quello immediatamente sucessivo.

La figura di pagina seguente riporta uno schema utile a comprendere tale meccanismo.

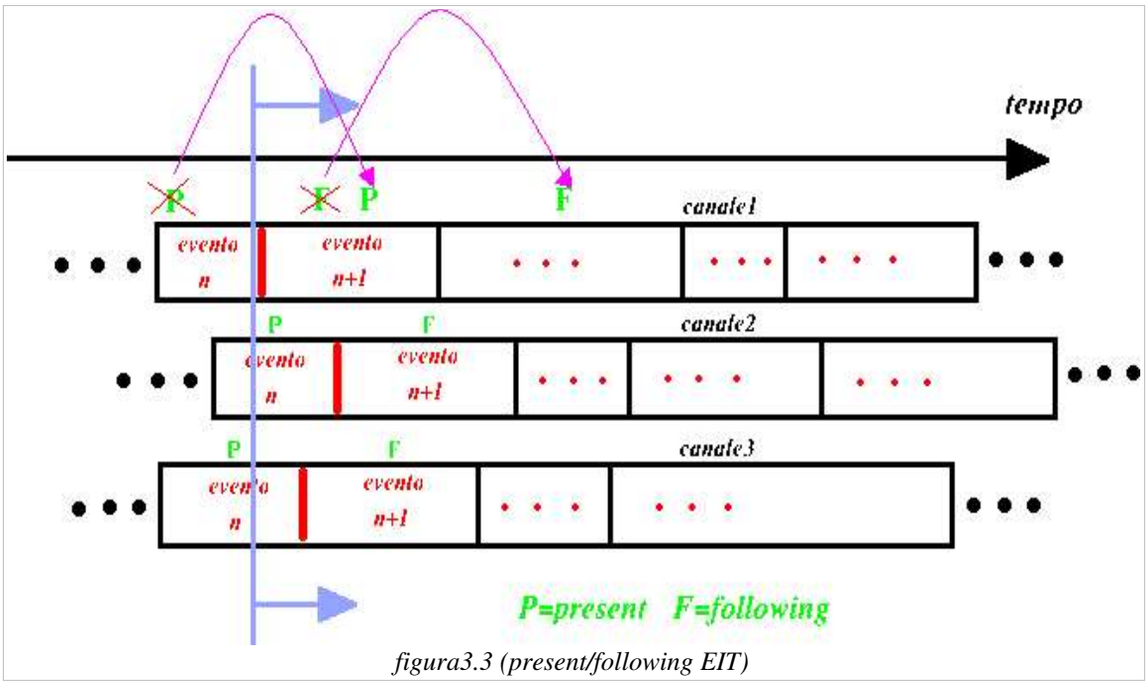

figura3.3 (present/following EIT)

Sul canale 2 e 3, gli eventi presenti sono ancora in fase di trasmissione e non sono ancora conclusi, mentre sul canale 1 l'evento presente è appena terminato.Ciò fa scattare la procedura di aggiornamento della EIT P/F (present/following EIT) .

Oltre a tale tabella esiste anche la schedule EIT.Come già visto nel capitolo 2, qui vengono riportate le informazione di programmazione a più larga scadenza.La EIT schedule permette lo scheduling dei programmi sino ad un arco temporale massimo di 64 giorni.Tale intervallo varia da un ente televisivo all'altro, ma mediamente tende a coprire 7 giorni.Solitamente l'aggiornamento della EIT schedule viene fatto su base giornaliera e come riferimento temporale viene presa la mezzanotte del primo giorno di programmazione.

Una volta che le tabelle EIT sono state create vengono passate al blocco “Data Carousel”.La sua funzione è quella di inserire periodicamente nel TS tali informazioni.Se un ricevitore vuole accedere ad un particolare modulo del data carousel, aspetterà semplicemente la prossima volta che tali dati saranno ritrasmessi.Questo sistema risulta molto utile anche per la protezione contro gli errori, infatti se il modulo ricevuto ne è affetto si può sempre aspettare la successiva trasmissione.Creato il TS questo verrà poi presentato al blocco di

modulazione per poi essere trasmesso dall'apposito mezzo trasmissivo.
Ritornando al data carousel, si fa notare che più l'intervallo di ritrasmissione è basso, più basso sarà il tempo medio di attesa per la ricezione delle tabelle EIT, ma si intuisce come ciò porti ad un uso maggiore di banda.
Sarà quindi necessario raggiungere un compromesso fra qualità del servizio (QOS) e banda disponibile.La banda richiesta dipende dalla quantità di dati da inviare e ovviamente dall'intervallo di trasmissione.
Nel documento etsi "*ETR 211*" si possono trovare alcune indicazioni al riguardo.
Si riportano quelle relative al sistema di trasmissione dvb-t.

- *tutte le sezioni della NIT devono essere trasmesse almeno ogni 10 secondi;*
- *tutte le sezioni della BAT devono essere trasmesse almeno ogni 10 secondi, se presenti;*
- *tutte le sezioni della SDT per il TS attuale devono essere trasmesse almeno ogni 2 secondi;*
- *tutte le sezioni della SDT per gli altri TS devono essere trasmesse almeno ogni 10 secondi, se presenti;*
- *tutte le sezioni della EIT P/F per il TS attuale devono essere trasmesse almeno ogni 2 secondi;*
- *tutte le sezioni della EIT P/F per gli altri TS devono essere trasmesse almeno ogni 20 secondi, se presenti.*

Il rate di ripetizione per le schedule EIT dipende dal numero di servizi contenuti nel mux e dalla quantità delle relative informazioni SI.
I seguenti intervalli di trasmissione dovrebbero essere seguiti se praticabili ma devono aumentare nel caso aumenti la mole di dati delle EIT.In seguito si riporterà un esempio pratico per avere dei riferimenti numerici più precisi.

- *tutte le sezioni della schedule EIT relative all'intero primo giorno del TS attuale, devono essere trasmesse almeno ogni 10 secondi, se presenti;*
- *tutte le sezioni della schedule EIT relative all'intero primo giorno degli altri TS, devono essere trasmesse almeno ogni 60 secondi, se presenti;*
- *tutte le sezioni della schedule EIT per il TS attuale, devono essere trasmesse almeno ogni 30 secondi, se presenti;*
- *tutte le sezioni della schedule EIT per gli altri TS, devono essere trasmesse*

almeno ogni 300 secondi, se presenti;

- *le TDT e TOT devono essere trasmesse almeno ogni 30 secondi.*

Si procede ora ad un rapido calcolo per quantificare le reali necessità di banda. Prima di iniziare è però necessario fare alcune assunzioni preliminari, come la quantità di dati contenuti nelle tabelle.Infatti, nonostante la loro struttura sia ben definita, il numero e la dimensione dei descrittori che esse includono può variare. In questo esempio si assume che venga usato solo il seguente descrittore:

- Short Event Descriptor

Si assume inoltre:

- TS attuale composto da 8 canali TV
- Informazioni EIT di altri 5 TS composti anch'essi da 8 canali TV
- 48 eventi per servizio TV in ogni periodo di 24 ore
- il nome dell'evento ha una lunghezza media di 20 caratteri
- la descrizione testuale media di ogni evento è lunga 200 caratteri

La EIT P/F è formata da due sezioni, la 0x00 per la Present e la 0x01 per la Following.
L'header della EIT ha una lunghezza pari a 18 Byte, mentre per ogni evento la lunghezza complessiva dei relativi dati nel loop EIT, comprensiva del descrittore, è di 242 Byte.
Quindi il numero totale di byte della EIT P/F di un singlolo canale è pari a

(18+242)*2=520 Byte

Considerato poi che il TS contiene 8 canali i Byte totali sono

520*8=4160 Byte

Ora questi dati devono essere ripetuti ogni 2 secondi, portando quindi ad una occupazione di banda per le EIT P/F del TS attuale di circa 16.6 Kbit/sec.

4160*8=33280 bit
33280/2=16640 bit/sec ~16.6 Kb/sec

Le EIT P/F degli altri 5 TS vengono trasmesse ogni 20 secondi, la banda da loro occupata è pari a

33280*5/20=8320 bit~8.3 Kb/sec

Quindi la banda totale per le informazioni di EIT P/F è 24.9 Kbit/sec.
Si considerano ora le EIT schedule assumendo che la programmazione copra un periodo di 7 giorni.Ogni tabella EIT schedule può essere formata da un massimo di 256 sezioni, a loro volta queste sono divise in 32 segmenti formati da 8 sezioni ciascuno.Ogni segmento copre uno slot temporale di tre ore.Si assume una durata media per ogni evento di circa 30 minuti.Ciò porta alla presenza di 6 eventi per ogni slot temporale.Vengono inseriti due eventi per sezione, risultando così in un impiego di 3 sezioni per la descrizione di un periodo di tre ore, di 24 per un giorno e di 168 per l'intero arco temporale di 7 giorni coperto dalla schedule.

18+(242*2)=502 Byte *Dimensione della sezione composta da due eventi*
502*3*8*7=84336 Byte *Dimensione totale delle sezioni per la copertura dei 7 giorni*
84336*8=674688 Byte *Dimensione totale comprensiva di tutti gli 8 canali contenuti nel TS*

Secondo specifiche le sezioni della EIT schedule relativa all'intero primo giorno del TS attuale devono essere trasmesse almeno ogni 10 secondi.Quindi la banda necessaria per coprire il primo giorno è

502*3*8*8=96384 Byte *Dimensione totale delle sezioni che coprono il primo giorno*
96384*8/10=77107.2 bit/sec~77.1 Kbit/sec *banda per il primo giorno del TS attuale*
96384*8*5/60=64256 bit/sec~64.2 Kbit/sec *banda per il primo giorno degli altri TS*

per i restanti 6 giorni, considerando che le relative sezioni devono essere trasmesse almeno ogni 30 secondi, la banda necessaria è invece

502*3*8*8*6=578304 Byte *Dimensione totale delle sezioni che coprono i successivi 6 giorni*
72288*8/30=154214.4 bit/sec~154.2 Kbit/sec *banda per i 6 giorni successivi del TS attuale*
72288*8*5/300=9638.4 bit/sec~9.6 Kbit/sec *banda per i 6 giorni successivi degli altri TS*

La banda totale (EIT P/F actual/othrer + EIT schedule actual/other) risulta essere quindi

24.9+77.1+64.2+154.2+9.6=330 Kbit/sec=41.25 Kbyte/sec

3.4 Analisi di uno stream DVB

In questo paragrafo si procederà all'analisi di uno stream DVB per trovare riscontri pratici alla teoria sino ad ora esposta.Di seguito viene data una breve presentazione dei tools utilizzati.

3.4.1 Strumenti utilizzati

Per le prove effettuate si è fatto uso di un laptop Fujitsu Siemens Amilo L1310G, del ricevitore TV-USB Hauppauge HVR-900 (vedi figura 3.4) e della sua antenna mobile.

L'HVR-900 è un ricevitore ibrido, può cioè ricevere e decodificare il segnale DVB-T, ma anche il segnale TV analogico.Per effettuare dei test aggiuntivi su DVB-S, si è proceduto anche all'installazione di un Kit, composto da parabola di diametro 60cm (puntata su hotbird13E) e dal ricevitore TV-USB SkyStar (vedi figura 3.5).In questo paragrafo viene comunque riportata l'analisi di un TS ricevuto in ambiente DVB-T, quindi per mezzo dell'HVR-900.

figura3.4 (Hauppauge HVR-900)

figura3.5 (SkyStar USB)

Le foto sottostanti (figura 3.6-3.7) riprendono l'intera postazione di lavoro comprendente tutti gli apparati appena elencati.

figura3.6 (postazione di lavoro)

figura3.7 (Dettaglio laptop e ricevitori DVB)

Il software fornito insieme agli apparati di ricezione non consente di fare un'analisi dettagliata dell'intero TS, si è quindi proceduto all'installazione sul laptop di un efficacie e utilissimo programma che permettesse di fare ciò:TsReader.Le sue funzionalità e caratteristiche saranno chiarite durante la presentazione dei test effettuati.

3.4.2 Analisi di un Transport Stream DVB-T

Una volta settato il corretto ricevitore (HVR-900) nel programma TsReader si è proceduto allo scanning frequenziale per acquisire tutti i TS disponibili.Per la presenza di numerosi canali (tra i quali alcuni trasmettenti eit-schedule in chiaro), si è scelto di procedere all'analisi del TS alla frequenza di 642000Khz.

Una volta sintonizzati su tale TS, la prima schermata offerta da TsReader è quella di figura 3.8.Come si può notare, la colonna di sinistra elenca tutte le tabelle PSI/SI ricevute con i relativi PID.Se ne viene selezionata una in particolare, la parte alta della colonna centrale visualizza tutte le informazioni da essa contenute.In questo caso è stata selezionata la PMT del programma 1.
Nella parte centrale della stessa colonna si trovano elencati tutti i PID attivi,la percentuale di utilizzo della banda del TS e il relativo bitrate.
La parte più bassa mostra i dettagli di trasmissione delle tabelle più importanti, in particolare il numero di sezioni ricevute e gli errori riscontrati.Inoltre viene

indicato anche il bitrate del Mux.Come già chiarito in precedenza ci si può riferire al TS anche con il termine Mux.Per concludere la panoramica sulla schermata principale di TsReader, si fa notare come l'ultima colonna visualizzi l'immagine dei vari programmi appartenenti al TS anailizzato.Tale immagine risulta essere il primo I frame ricevuto subito dopo la sintonizzazione.

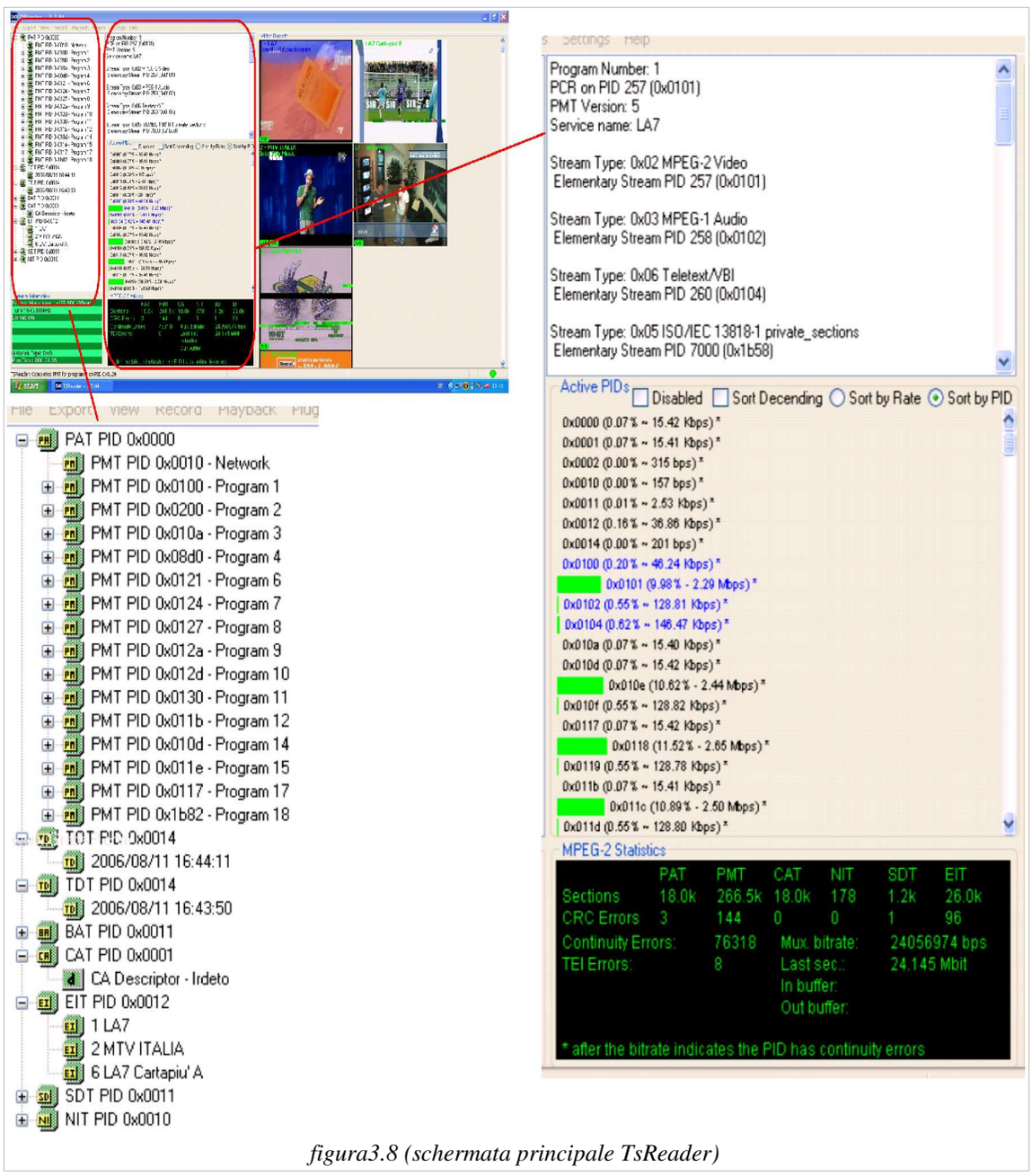

figura3.8 (schermata principale TsReader)

Grazie alla possibilità di accedere a tutte queste informazioni, è possibile ripercorrere i passi compiuti dal decodificatore per poter visualizzare un determinato programma.
Si rende qui necessaria una precisazione: nelle tabelle PSI/SI, con il termine programma non si fa riferimento al classico significato assegnatogli nell'ambiente del broadcasting.Infatti un programma è solitamente inteso come un'insieme di elementary stream con una base dei tempi comune e con una comune ora d'inizio e fine.Una serie di programmi di tal tipo possono essere trasmessi sequenzialmente in un TS usando lo stesso *program_number* per andare a fromare un canale TV convenzionale.
Nel caso particolare si suppone che l'utente sia interessato al programma numero 1, cioè al canale LA7.
Il software fornito dall'HVR-900 (SmartTV) prevede un'utile funzione di Scanning frequenziale.Una volta avviata questa operazione, per ogni TS rilevato, vengono memorizzati i canali da esso trasportato.Alla fine, il menù fornirà una lista completa di tutti i canali contenuti nei vari TS incontrati durante lo scanning (vedi figura 3.9).
Dalla schermata riportata si nota che l'operazione di ricerca è stata fatta esclusivamente per canali DVB-T (infatti in figura è selezionata l'opzione “DVB-T Source”).Come detto ad inizio paragrafo è però prevista anche la possibilità di procedere alla più tradizionale ricerca dei canali trasmessi in analogico (basta selezionare l'opzione “Analog Source”).
Per i fini di questo lavoro si pone comunque l'attenzione solo su quelli digitali.
Per poter listare i nomi dei vari programmi a disposizione, il ricevitore, per ogni TS trovato, si ricostruisce la PAT (facilmente rintracciabile dal PID fisso ad essa assegnato:0x0000).
La PAT contiene un elenco dei vari programmi trasportati dal TS e il PID delle relative PMT.Si ricorda che la PMT indica quali sono i PID che contraddistinguono i vari elementary stream del programma in questione.Oltre a queste fondamentali informazioni, se ne possono trovare altre che vanno a descrivere più nel dettaglio il programma e fra queste si trova anche il *service_name.*Il nome che viene associato al programma è per l'appunto questo.

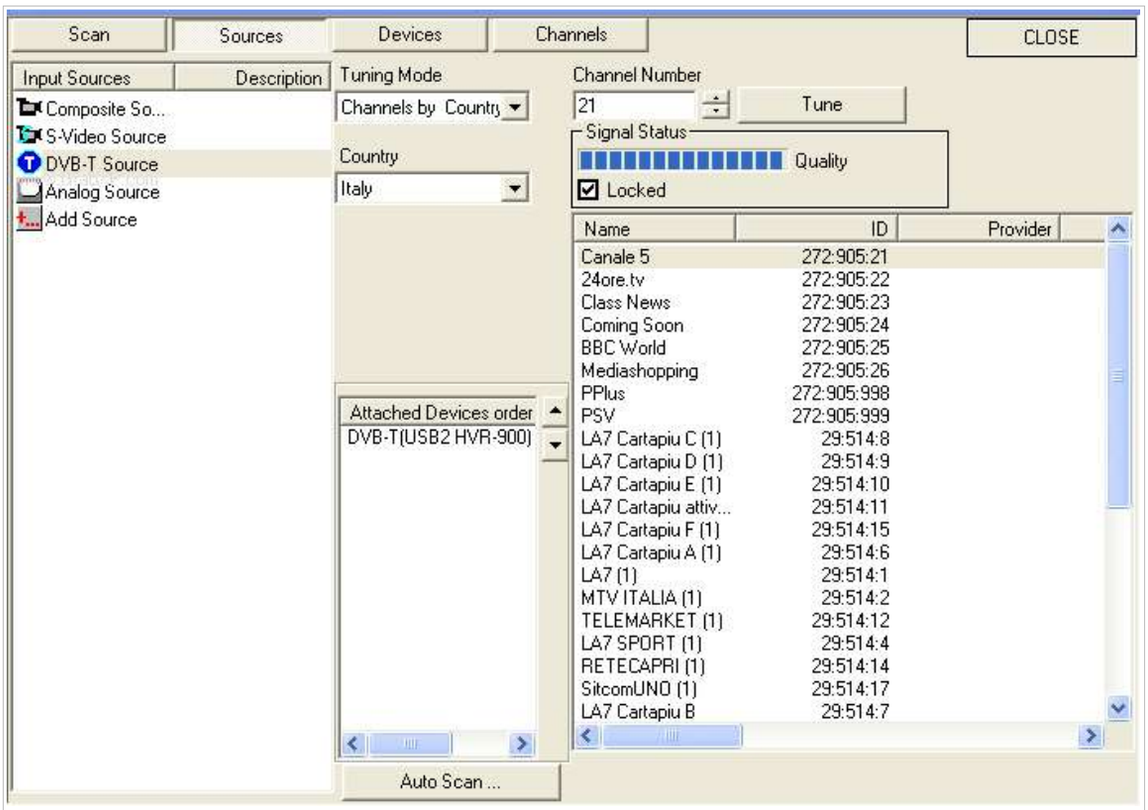

figura3.9 (Funzione di scanning del programma SmartTV)

Quando dal menù si seleziona LA7, le operazioni compiute dal ricevitore per permetterne la visualizzazione sono le seguenti:

- come detto in precedenza ricostruisce la PAT (figura 3.10) assemblando opportunamente tutti i pacchetti identificati dal PID 0x0000.Qui cerca la entry relativa al programma numero 1 (LA7) per sapere a quale PID è associato la PMT.Il PID è il 0x0100.

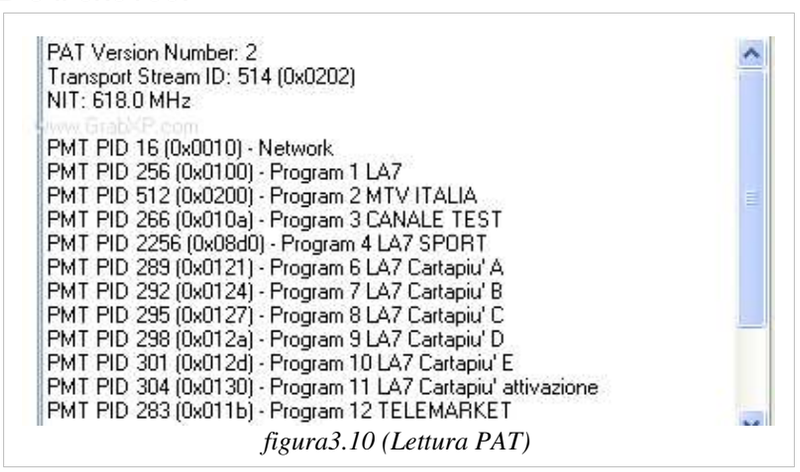

figura3.10 (Lettura PAT)

- Ora vengono ricevuti tutti i pacchetti con PID 0x0100 per costruire la PMT

(vedi figura 3.11). Qui si nota il campo poc'anzi menzionato *service_name* (LA7) e le informazioni su dove andare a trovare gli elementary stream componenti il programma.Il contenuto video è nei pacchetti contrassegnati dal PID 0x0101 , l'audio in quelli con PID 0x0102 e il Teletext in quelli con PID 0x0104.Viene inoltre indicato dove andare a cercare il PCR, cioè il program clock reference, che come gia descritto in precedenza serve per sincronizzare il clock di sistema del ricevitore con quello del trasmettitore.

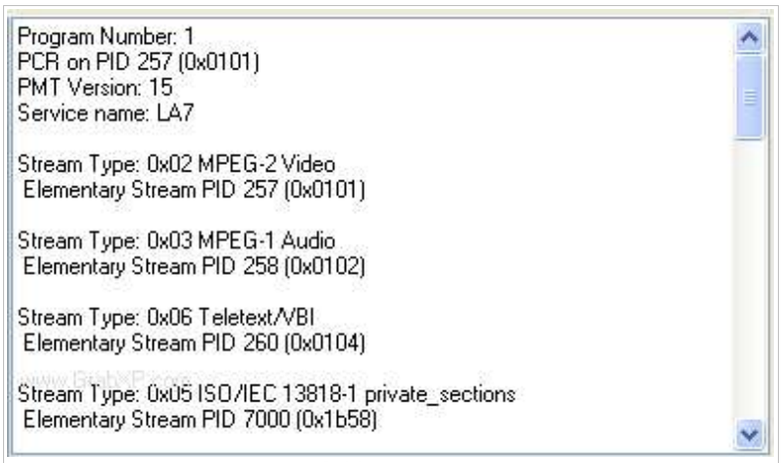

figura3.11 (Lettura PMT)

- il ricevitore può ora cominciare a ricevere e decodificare i contenuti del canale LA7 ed avviarne la visualizzazione (vedi figura 3.12).

figura3.12 (visualizzazione del canale LA7)

Ritornando alle informazioni contenute nella PAT, e più precisamente alla prima entry, si vede che viene segnalato anche il PID su cui trovare le sezioni della NIT: 0x0010.La figura 3.13 ne mostra il contenuto.

figura3.13 (Lettura NIT)

Come già detto nel capitolo 2, la NIT descrive i parametri del sistema di trasmissione utilizzato per il TS.Queste informazioni possono essere relative al Network che sta attualmente trasmettendo il TS su cui si è sintonizzati, oppure ad altri.In questo caso il campo *Current_Network* settato a True segnala che si fa riferimento al Network corrente.

La tripletta Network ID, Transport Stream ID e Original Network ID permette di identificare univocamente il TS.In questo caso la NIT tratta solo l'attuale TS e nel suo loop sono elencati una serie di Service List Descriptor, i quali provvedono a fornire una descrizione dei servizi mediante i campi *service_id* e *service_type*.Il primo campo corrisponde al program_number utilizzato dalla PAT, mentre il secondo da indicazioni sul tipo di servizio, nel caso specifico *digital television service*.E' stata scelta l'analisi del canale LA7 soprattutto per la presenza di una EIT particolarmente ricca.Infatti oltre alla trasmissione obbligatoria della present/following si ha a disposizione anche la EIT schedule.Quest'ultima copre un arco temporale di 7 giorni.Si ricorda che la struttura di questa tabella prevede un event loop interno che lista tutti gli eventi inseriti nella programmazione, comprensivi della data e ora d'inizio, durata e di un'eventuale loop di descrittori.Nel caso specifico è stato inserito solo lo *short_event_descriptor*, da cui è stato prelevato il nome dell'evento.Viene di seguito riportato il contenuto della EIT di LA7.

“Channel 1
Service Name: LA7
Provider Name: MBone
Transport Stream ID: 514 (0x0202) 618.0 MHz

Starts: 17/08/2006 11.30.00
Length: 01:00:00
EIT Source: actual TS, present/following
Name: Leggende Della Terra
Alla scoperta di Nuovi Mondi
Description Source: DVB Short Event

Starts: 17/08/2006 12.30.00
Length: 01:00:00
EIT Source: actual TS, present/following
Name: Matlock
Nuovi ed avvincenti casi legali per l'avvocato Benjamin
Matlock (Andy Griffith)che lavora ad ...
Description Source: DVB Short Event

Starts: 17/08/2006 13.30.00
Length: 00:30:00
EIT Source: actual TS, event schedule
Name: Tg La7
Informazione a cura della redazione del TG LA7
Description Source: DVB Short Event

▪

▪

▪

Starts: 24/08/2006 0.30.00
Length: 00:30:00
EIT Source: actual TS, event schedule
Name: Sex And The City
Serie tv che ha fatto storia e tendenza. All'insegna della trasgressione è diventato un ...
Description Source: DVB Short Event

Starts: 24/08/2006 1.00.00
Length: 01:00:00
EIT Source: actual TS, event schedule
Name: Practice - Professione Avvocati
Un team d'avvocati alle prese con la risoluzione di nuovi ed avvincenti casi legali
Description Source: DVB Short Event ”

Come già ampiamente spiegato nel presente capitolo, la EIT è la fonte da cui vengono letti i dati per la creazione dell'EPG.La figura sottostante (figura 3.14) riporta la funzione EPG del programma TsReader.

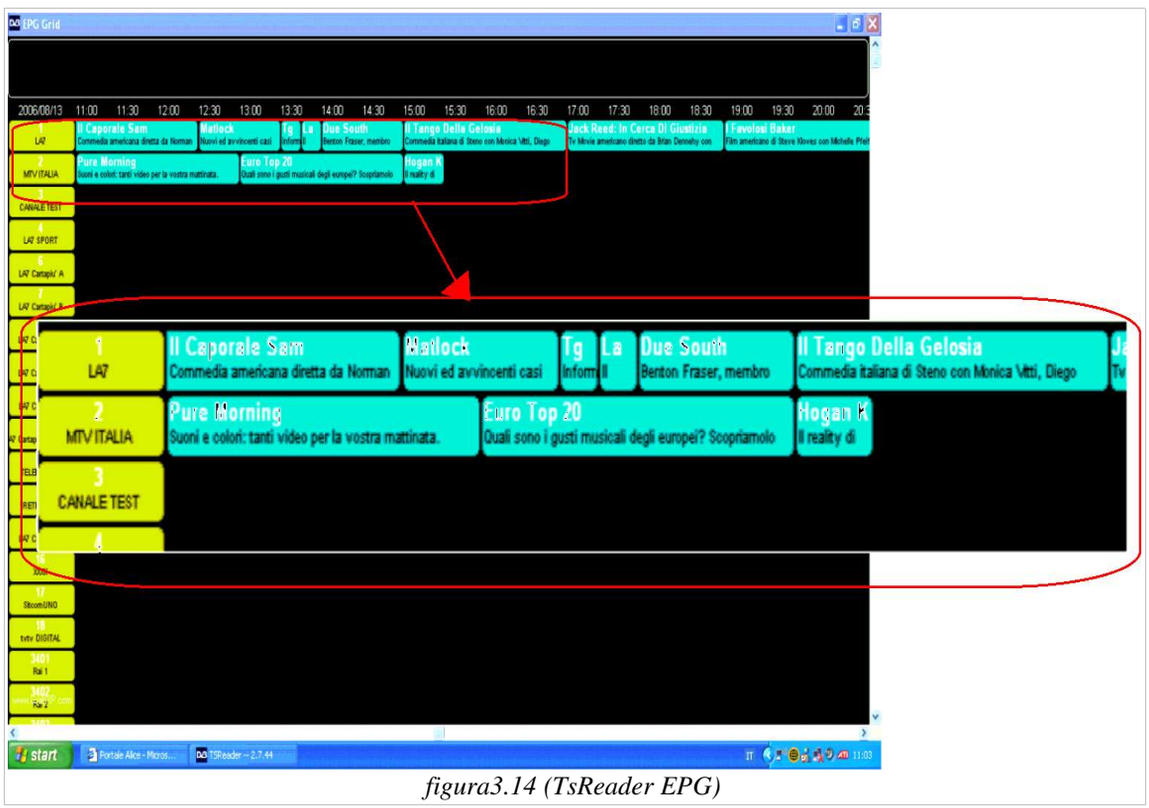

figura3.14 (TsReader EPG)

Risulta subito evidente che gli unici canali per i quali è fornito il servizio EPG sono LA7 ed MTV ITALIA.Infatti nel TS corrente le uniche EIT schedule presenti sono quelle relative a questi due canali.

Ritornando ai calcoli fatti in precedenza riguardo l'occupazione di banda di un tipico ES EIT è ovvio aspettarsi che in questo caso il bitrate sia molto più basso (si ricorda che il risultato ottenuto era pari a 330 Kbit/sec).Infatti nel TS analizzato solo due canali trasportano sezioni EIT schedule e di questi solo LA7 copre un arco temporale di 7 giorni (MTV ITALIA non supera le 4 ore).

Con la figura 3.15 si ripresenta nuovamente la schermata iniziale di TsReader ove è stata zoommata la parte relativa all'occupazione di banda del PID 0x0012.

l PID 0x0012 identifica la tabella EIT e si nota che il bitrate ad essa associato è di circa 36.86Kbps, corrispondente allo 0.16% della banda totale assegnata al TS.

E' bene sottolineare ulteriormente che non si pretende di arrivare ad una esatta previsione del bitrate effettivamente registrato da TsReader.Basta infatti avere almeno un'ordine di grandezza utile a verificare la consistenza del dato letto.

figura3.15 (EIT bitrate)

Interessante il grafico fornito da Tsreader (figura 3.16), sempre relativo all'utilizzo delle risorse di banda.Risulta evidente che i contenuti più pesanti sono quelli video, subito a seguire vi sono gli ES ISO/IEC 13818-6 Type B.Questi stream contengono dati usati da applicazioni MHP, si possono quindi identificare come *application resource data*.

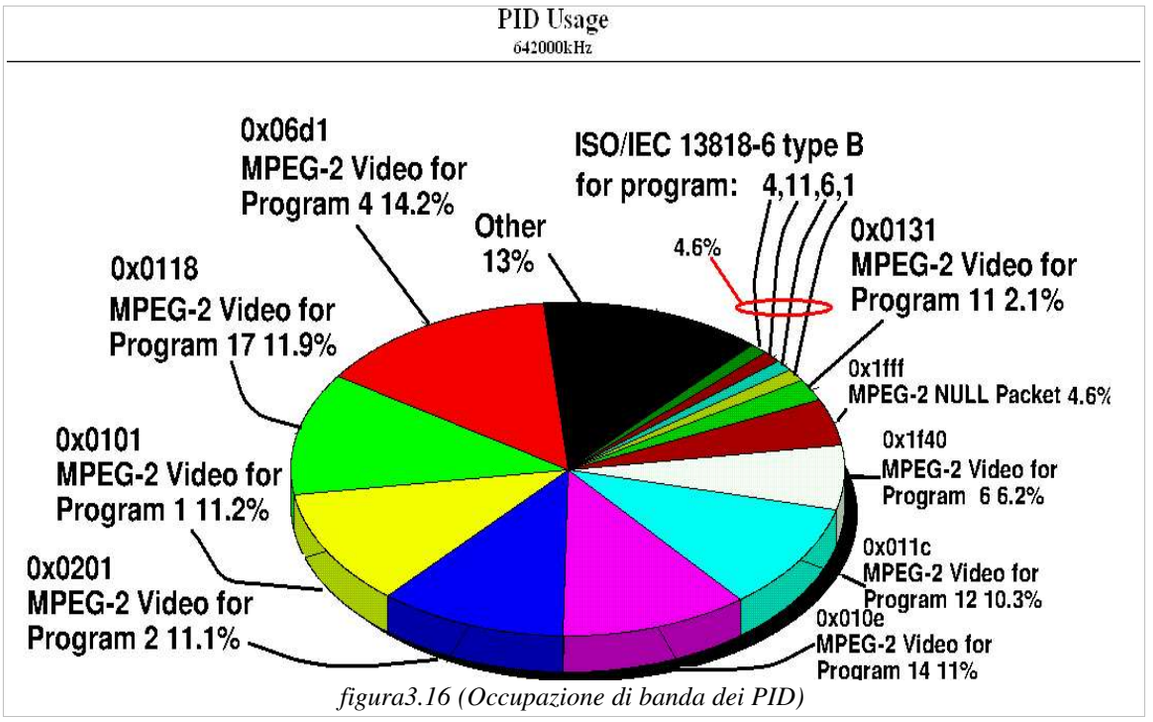

figura3.16 (Occupazione di banda dei PID)

L'ultima figura (figura 3.17) riporta invece l'occupazione totale di banda di ogni singolo canale trasportato nel TS.

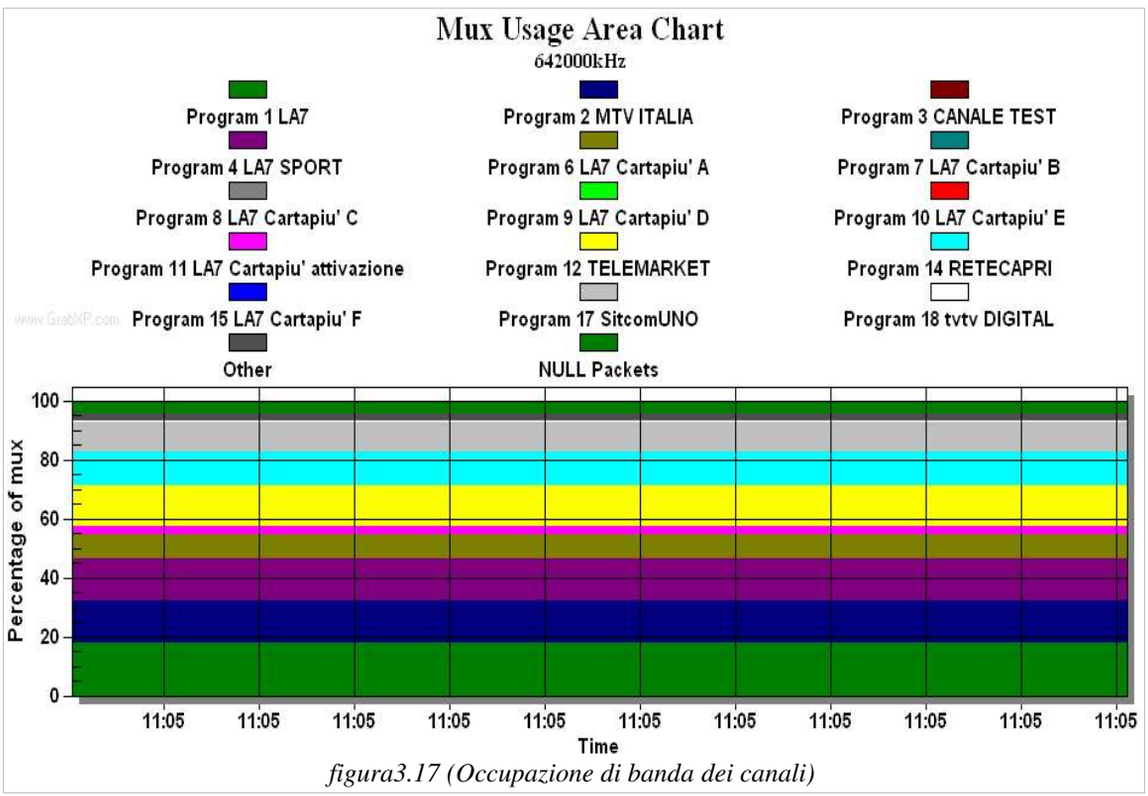

figura3.17 (Occupazione di banda dei canali)

Riferimenti bibliografici:

- *Guidelines on implementation and usage of Service Information DVB Document A005 (ieeexplore.ieee.org/iel3/5179/14018/00642905.pdf)*
- *A digital television Navigator (Chengyuan Peng, Petri Vuorimaa) (users.tkk.fi/~pcy/jnavigator_final)*
- *Using SI Tables to Create Electronic Program Guide (ieeexplore.ieee.org/iel5/6974/18858/00871031.pdf)*
- *Improving DVB-T network efficiency (Christoph Lindner & Lars Lastein*)
- (*www.scientificatlanta.com/products/customers/white-papers/Improving_DVB_T_network_efficiency.pdf*)
- *Data Management in Set-Top Box* Electronic Programming Guides (*www.mcobject.com/downloadfiles/epgwhitepaper.pdf*)
- *ETSI TR 101 211 V1.7.1 (2006-02) Technical Report Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)*
- *EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".Known as (DVB-SI).*
- *Carouseled EPG data delivery (www.bci.eu.com/library/articles/delivery.pdf)*
- *Data Broadcasting Service (Yukio Yokoyama and Heikan Izumi) (* global.mitsubishielectric.com/pdf/advance/vol85/85tr10.pdf *)*

4. Software e sistemi DVB presso MBI

4.1 XML

Il progetto MBI-EPG ha fatto largo uso del linguaggio XML, si ritiene quindi utile darne una breve presentazione.

L'Extensible Markup Language (XML) è un metalinguaggio, ovvero un insieme di regole base utilizzate per creare altri linguaggi.Dalla sua nascita avvenuta nel 1998, ha assunto un ruolo centrale per quanto riguarda lo sviluppo di nuove tecnologie in ambito Web e non solo.XML permette di creare nuovi linguaggi per rappresentare informazione strutturata in formato testuale.Il suo successo è dovuto a molti motivi e tra i principali si può sicuramente includere la sua semplicità, espandibilità e portabilità.La specifica di XML, definita dal World Wide Web Consortium (W3C), descrive un insieme di regole per definire linguaggi basati su tag (marcatori) all'interno dei quali inserire in maniera strutturata il contenuto informativo da rappresentare.

I tag devono essere inseriti correttamente l'uno dentro l'altro, ad ogni tag di apertura deve corrispondere un tag di chiusura e gli attributi dei tag devono essere racchiusi tra apici.Utilizzando queste semplici ed essenziali regole l'utente ha la possibilità di creare un nuovo linguaggio definendo i tag e gli attributi più appropriati a memorizzare l'informazione che si vuole trattare.Inoltre XML favorisce l'interoperabilità in quanto è un formato testuale, quindi facilmente trasferibile ed elaborabile su differenti piattaforme hardware e software.Una caratteristica fondamentale di XML è quella di occuparsi del contenuto dell'informazione e non della sua rappresentazione.La modalità di rappresentazione dell'informazione può essere scelta in un secondo momento e, partendo dallo stesso file XML, si può rappresentare l'informazione contenuta al suo interno in vari modi (HTML, XHTML etc.).

Come esempio si procede alla memorizzazione in formato XML delle informazioni relative ad una rubrica.Tipicamente in una rubrica, sono conservati alcuni dati (quali ad esempio nome, cognome, indirizzo, numero di telefono) riguardanti una determinata persona e tutto questo in XML può essere espresso nel seguente modo:

```
<?xml version="1.0"?>
<rubrica>
  <persona>
    <nome>Mario</nome>
    <cognome>Rossi</cognome>
    <indirizzo>
      <via>via bianchi 1</via>
      <cap>00000</cap>
      <citta>Roma</citta>
    </indirizzo>
    <telefono>
      <telefono_fisso>123456</telefono_fisso>
      <cellulare>987656412</cellulare>
    </telefono>
  </persona>
</rubrica>
```

Proprio a causa delle caratteristiche descritte, XML ha incontrato molto successo come formato per lo scambio di dati tra applicazioni differenti.Affinché due applicazioni possano scambiarsi dei dati XML è però necessario che queste conoscano come viene strutturata l'informazione all'interno del file, ovvero quali sono e come sono chiamati i tag e gli attributi che lo costituiscono.Per questo scopo sono state sviluppate alcune tecnologie, tra le quali XML Schema.XML Schema è un linguaggio basato su XML nato per definire la struttura di un documento XML.La struttura dell'esempio, espressa con un XML Schema, diventa:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="rubrica" type="tipo_rubrica"/>
 <xsd:complexType name="tipo_rubrica">
  <xsd:sequence>
   <xsd:element name="persona" type="tipo_persona"/>
   </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="tipo_persona">
  <xsd:sequence>
   <xsd:element name="nome" type="xsd:string"/>
   <xsd:element name="cognome" type="xsd:string"/>
```

```
    <xsd:element name="indirizzo" type="tipo_indirizzo"/>
    <xsd:element name="telefono" type="tipo_telefono"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="tipo_indirizzo">
    <xsd:sequence>
    <xsd:element name="via" type="xsd:string"/>
    <xsd:element name="cap" type="xsd:decimal"/>
    <xsd:element name="citta" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="tipo_telefono">
    <xsd:sequence>
    <xsd:element name="telefono_fisso" type="xsd:decimal"/>
    <xsd:element name="cellulare" type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Come prima cosa viene indicato il “namespace” per XML-Schema (XML Namespace nasce con lo scopo di evitare possibili ambiguità tra i nomi degli elementi che fanno parte di un linguaggio basato su XML, visto che utilizzando i Namespace si è in grado di identificare univocamente i nomi dei tag e degli attributi).

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Poi viene inserita la dichiarazione dell’elemento (tag) *Rubrica* (di tipo *Tipo_Rubrica*):

```
<xsd:element name="rubrica" type="tipo_rubrica"/>
```

E di seguito come saranno composti gli elementi *Rubrica*, attraverso la definizione del tipo complesso *Tipo_Rubrica* costituito dall'unico elemento di tipo (complesso) *persona.*

```
<xsd:element name="persona" type="tipo_persona"/>
```

Si procede alla descrizione di quest'ultimo tipo (*tipo_persona),* definito come sequenza di elementi: gli elementi *nome* e *cognome* di tipo (semplice,predefinito) *string*, l'elemento *indirizzo* di tipo (complesso) *tipo_indirizzo* e *telefono* di tipo (complesso) *tipo_telefono.*

```
<xsd:element name="nome" type="xsd:string"/>
<xsd:element name="cognome" type="xsd:string"/>
<xsd:element name="indirizzo" type="tipo_indirizzo"/>
<xsd:element name="telefono" type="tipo_telefono"/>
```

E' interessante notare che ogni volta che si definisce un tipo complesso significa che in seguito verrà specificata la sua composizione fino ad arrivare ad una definizione che non contiene altro che tipi semplici.
Dopo aver definito l' XML- schema, si scriverà il documento XML con i dati che contiene e naturalmente anche con il riferimento all'XML-schema attraverso la seguente istruzione:

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

Di solito i documenti XML-Schema vengono rappresentati mediante una intuitiva struttura ad albero.La figura sottostante (figura 4.1), tratta dal programma XMLSpy, riporta quella relativa all'esempio trattato.

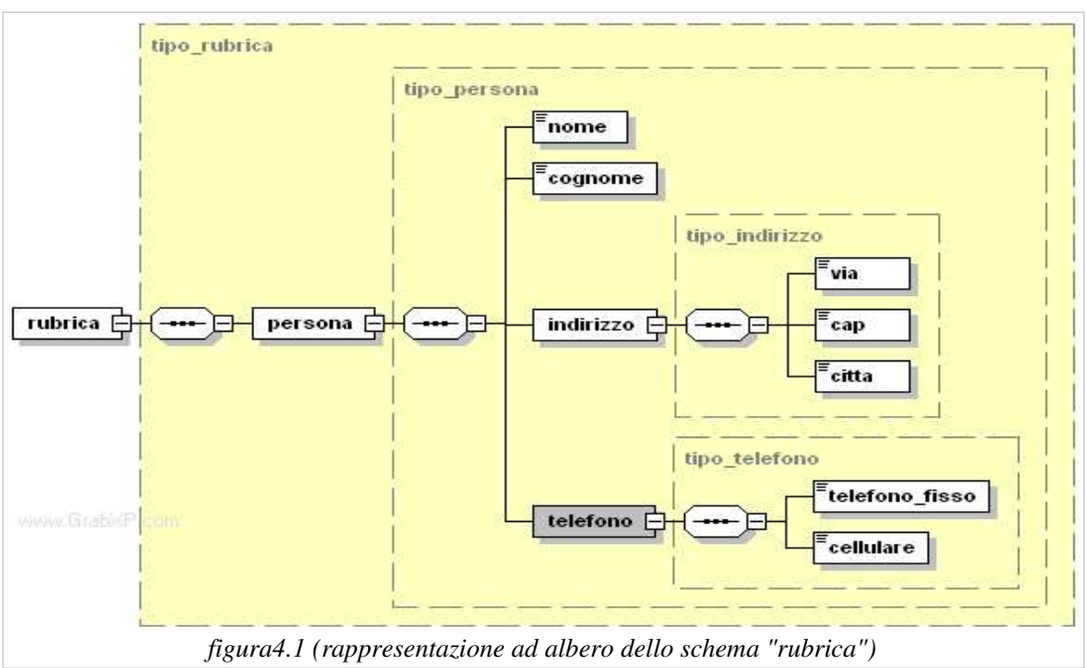

figura4.1 (rappresentazione ad albero dello schema "rubrica")

Al centro dell'architettura di un sistema che utilizza documenti XML vi è sempre un parser XML.Il parser XML può essere un componente esterno, una libreria o un insieme di classi e ha il compito di caricare il documento XML e di renderlo disponibile all'applicazione.
Vi sono due categorie di parser, validanti e non validanti.
I parser validanti permettono di verificare il contenuto del documento attraverso un file esterno che contiene la descrizione della struttura (DTD o XML Schema).
I parser non validanti non permettono di controllare la struttura del documento, delegando questo compito all'applicazione.
I parser a loro volta si suddividono in due grosse categorie:

• SAX (Simple API for XML)
• DOM (Document Object Model)

I Simple API (Application Program Interface) per XML (in breve SAX) fanno l'analisi sintattica del documento XML basandosi sul concetto di "evento".Ogni singolo elemento di un documento XML è caratterizzato da un tag di inizio, un contenuto e un tag di chiusura.Usando un parser SAX verrà segnalato, durante il parsing (cioè il controllo del documento), quando inizia un nuovo tag, le informazioni contenute e quando si chiude.Dal momento che un parser è spesso invocato dall'interno di un programma (ad esempio Java), tale segnalazione consiste, tecnicamente, nell'invio di un "evento" al programma chiamante (figura 4.2).In altre parole, un parser SAX scorre il documento XML, segnalando l'inizio e la fine di ogni tag (parser "event-based").

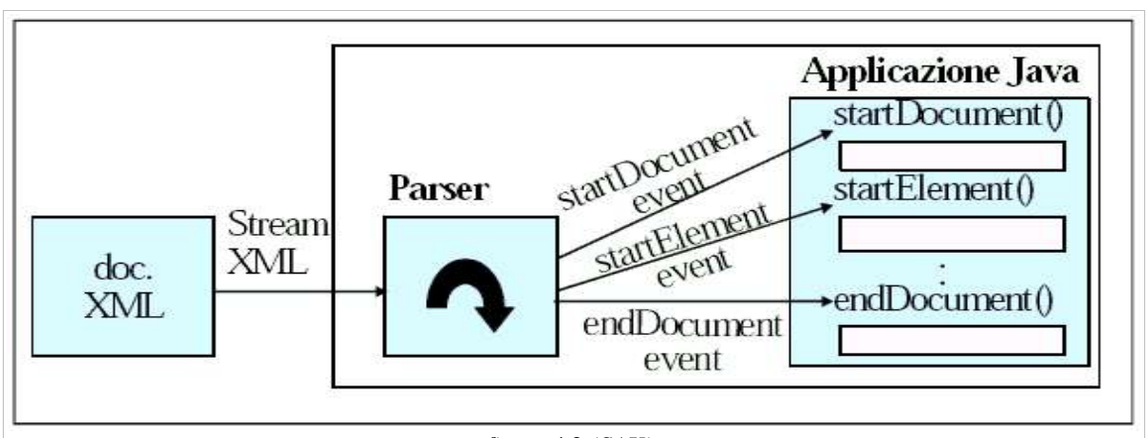

figura4.2 (SAX)

Tuttavia i parser SAX possono essere utilizzati anche per estrarre delle informazioni dai file XML, o in altre parole per leggerne il contenuto (eventualmente in modo selettivo).Infatti, scorrendo sequenzialmente il file XML e analizzandolo tag per tag, è possibile approfittare dell'invio degli eventi corrispondenti per selezionare delle parti di interesse e utilizzarle per successive elaborazioni.
I parser che si basano sul concetto di Document Object Model (DOM), invece che limitarsi a scorrere sequenzialmente il documento XML, sfruttano la già precedentemente accennata struttura ad albero, in cui gli elementi rappresentano i nodi dell'albero.In questo modo la lettura del documento XML e l'estrazione di informazioni specifiche è ancora più semplice ed intuitiva che con i parser SAX. Per esempio, per leggere tutti i nomi contenuti nella rubrica, dal programma si potrebbe semplicemente chiedere la lista dei nodi persona (figli del nodo rubricai) e, per ogni elemento di tale lista chiedere il contenuto dei suoi due nodi figli, nome e cognome.

4.1.1 XSL

Si conclude questa panoramica introducendo lo strumento utilizzato per visualizzare i dati contenuti in un documento xml: i fogli di stile XSL.
Il W3C ha definito uno standard per la scrittura di fogli di stile, chiamato eXtensible Stylesheet Language (XSL).I fogli di stile XSL rappresentano un ulteriore passo nella direzione della separazione tra il contenuto e l'aspetto con il quale i dati verranno presentati.L'XSL permette di scrivere delle regole per "trasformare" gli elementi contenuti nel documento XML in qualcos'altro, per esempio in elementi HTML all'interno di una pagina Web, che potrà in seguito essere visualizzata in un Browser.
Come si è sottolineato sin dall'inizio, la principale caratteristica di XML è la separazione del contenuto del documento dal suo aspetto (grafico).In particolare, con l'introduzione dei fogli di stile XSL si avrà che:

- il documento XML rappresenta unicamente il contenuto informativo che si intende pubblicare
- l'aspetto finale del documento è prodotto per mezzo di fogli di stile scritti in XSL

In questo modo è possibile visualizzare un singolo documento XML su diverse piattaforme, definendo fogli dei stile XSL specifici, senza dover apportare modifiche al contenuto (documento XML).
Lo strumento che rende possibile questa operazione prende il nome di processore xslt (figura 4.3).

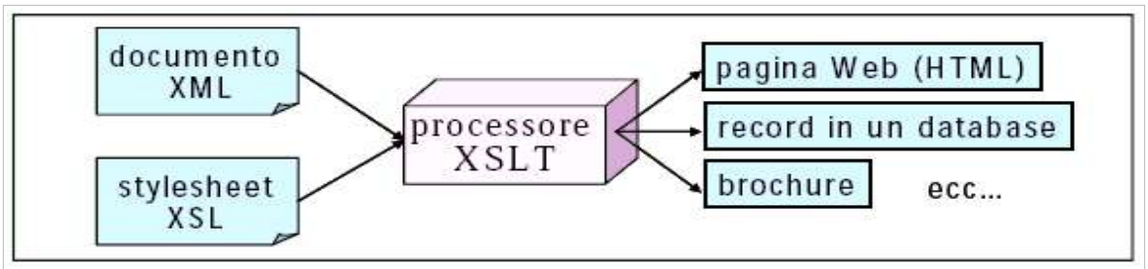

figura4.3 (funzione del processore xslt)

4.2 XML per le PSI/SI

Come già anticipato ad inizio capitolo nel sistema MBI-EPG si è fatto largo uso del linguaggio XML.In particolare, data la sua semplicità, si è provveduto a definire la grammatica delle varie tabelle PSI/SI mediante documenti XML, i quali sono stati usati anche per veicolare i dati (il progetto sarà descritto più nel dettaglio nel capitolo successivo).

Si riporta a titolo di esempio la descrizione XML della EIT.La figura 4.4 mostra la struttura di riferimento della EIT secondo quanto indicato dall'ETSI EN 300 468, mentre in figura 4.5 si riporta il relativo file EIT.xml.

Syntax	Number of bits	Identifier
event_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
service_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
segment_last_section_number	8	uimsbf
last_table_id	8	uimsbf
for(i=0;i<N;i++){		
event_id	16	uimsbf
start_time	40	bslbf
duration	24	uimsbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Event loop

figura4.4 (EIT)

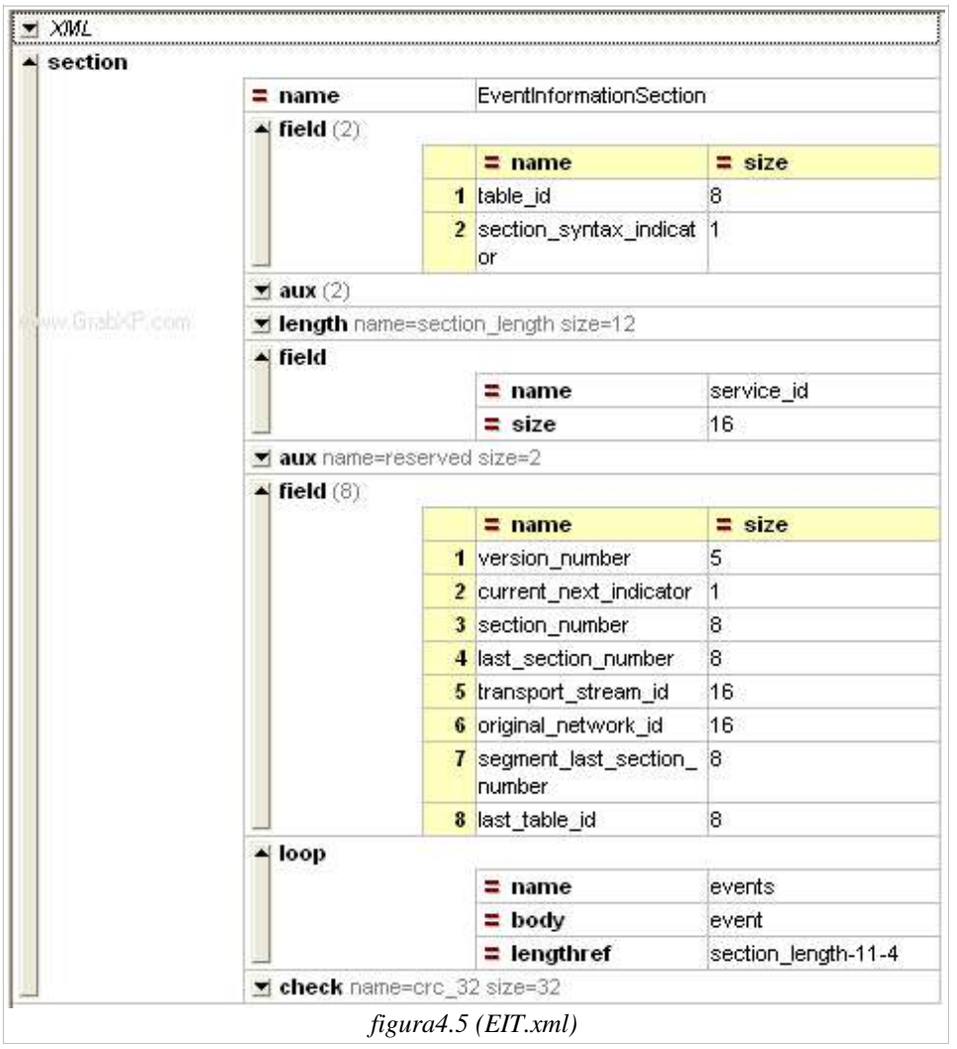

figura4.5 (EIT.xml)

La figura 4.5 non è altro che una rappresentazione un pò più intuitiva del file xml fornita dal già citato programma XMLSpy.

Ogni riquadro si riferisce ad uno specifico tag, il cui nome è indicato nella parte alta a sinistra.La gerarchia dei tag è facilmente comprensibile dall'inglobamento dei riquadri gerarchicamente subordinati entro quelli di grado superiore.

Nell'esempio si nota che il tag di root *section,* ingloba i tag *field*, *aux*, *length*, *loop* e *check.*

Per la descrizione dei vari campi della tabella EIT, come delle altre, non si è proceduto alla creazione di uno specifico tag da associare ad ognuno di essi, bensì si sono aggiunti gli attributi *name* e *size*.

Si vede infatti che tutti i campi ad eccezione dei reserved, crc check e quelli indicanti lunghezze, sono contrassegnati dal medeimo tag *field.*La distinzione è

poi fatta dal valore assegnato all'attributo name e size.Il campo *table_id, si vede ad* esempio assegnato tag *field, name= table_id e size= 8 (*quest'ultimo attributo specifica la lunghezza in bit).Il tag *length* viene usato per campi che forniscono indicazioni sulla lunghezza di una determinata entità, *aux* viene associato ai *reserved* e check ai campi di controllo del CRC.Per concludere la carrelata dei tag utilizzati per la descrizione della tabella EIT, si osserva l'inserimento del tag *loop*.Come si intuisce già dal nome, questo marcatore ingloba tutte le informazioni contenute nel loop, che nel caso specifico sono una serie di dati relativi ai vari eventi con annesso un loop più interno di descrittori.
La struttura dell'informazione all'interno dei file XML viene definita da un XML schema.Riprendendo in considerazione ad esempio il tag loop, in figura 4.6 è riportata la relativa definizione contenuta nello schema.

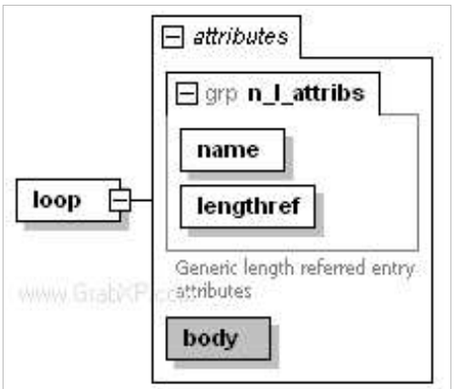

figura4.6 (XML Schema:loop)

Si nota che ogni loop prevede la presenza di due attributi che ne definiscono il nome *(name)* e la lunghezza *(lengthref)*, tutti i rimanenti contenuti informativi sono poi inglobati nell'entità contrassegnata dal tag *body*.Ovviamente nello schema si dà una descrizione anche di questo marcatore (figura 4.7).

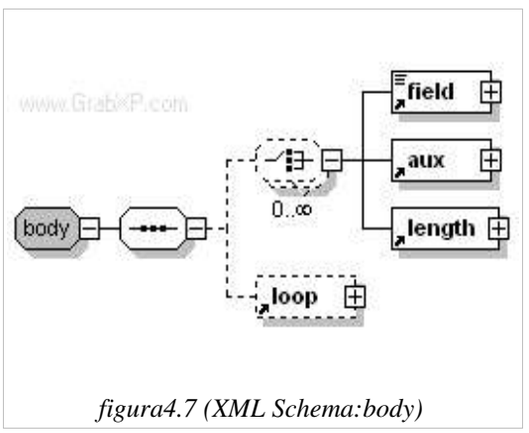

figura4.7 (XML Schema:body)

E' interessante sottolineare la presenza all'interno del *body* di un'ulteriore loop.E' infatti molto comune avere a che fare con piu loop concatenati.
Un esempio è dato propio dalla EIT: il primo loop è relativo alla lista degli eventi mentre il secondo è quello che viene associato ad ogni evento, contenente una serie di descrittori che ne vanno a completare la descrizione.
Dalla lettura del file EIT.xml si osserva che il contenuto del primo loop viene inglobato nel relativo body associato all'entità *event*.Il template xml di event è riportato in figura 4.8.

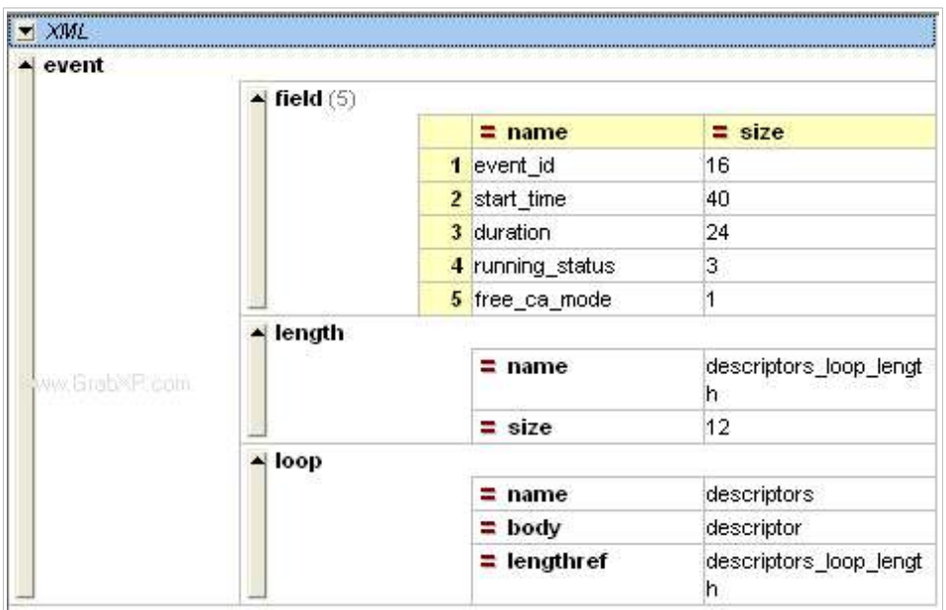

figura4.8 (EVENT.xml)

Da un confronto con la tabella di figura 4.4 si osserva la corrispondenza con i campi associati ai vari eventi (*event_id, start_time, duration, running_status, free_ca_mode)* e la presenza del secondo loop di descrittori.
Il body di quest'ultimo è associato invece all'entità *descriptor*, la quale rappresenta in formato xml i contenuti informativi dei vari descrittori.La figura 4.9 riporta come esempio il file *short_event_descriptor.*xml, mentre la figura 4.10 lo schema generico associato ad ogni descrittore.

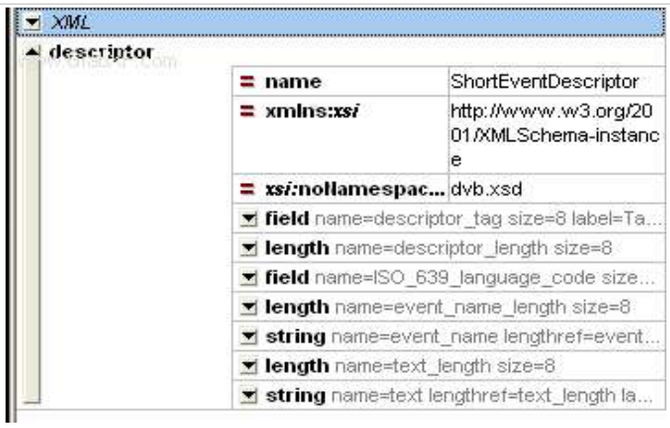

figura4.9 (short_event_descriptor.xml)

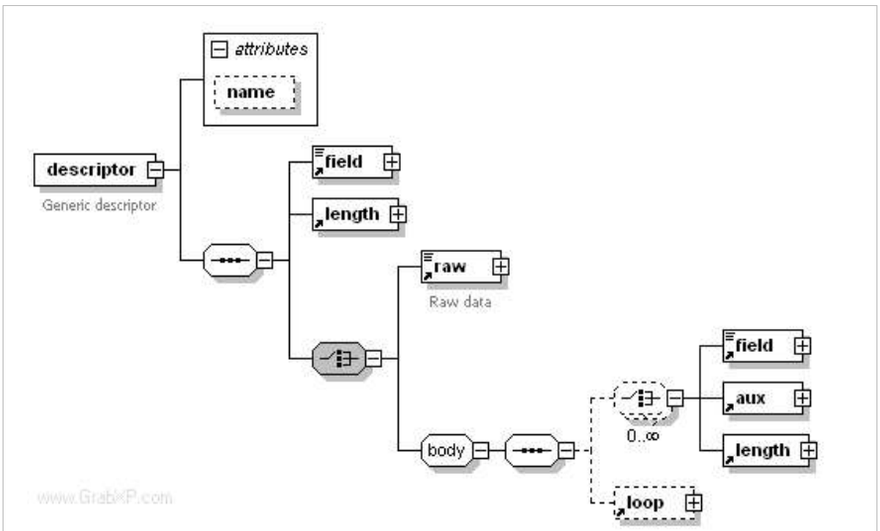

figura4.10 (XML Scchema:descriptor)

Ovviamente anche ad EVENT.xml è stato assegnato il relativo schema (vedi figura 4.11).

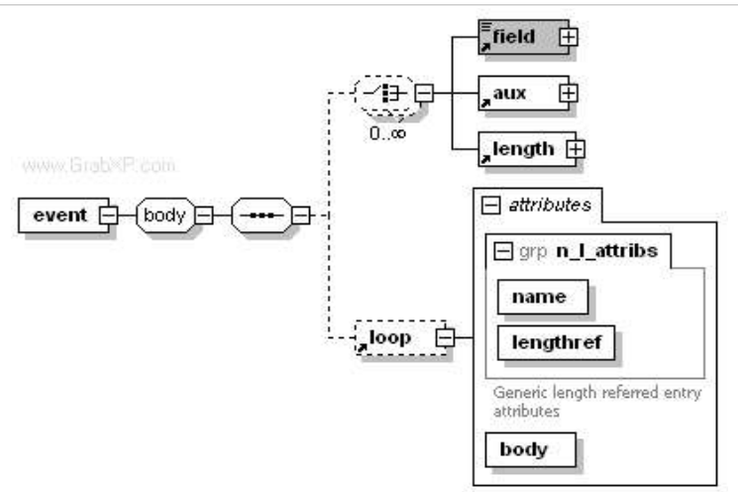

figura4.11 (XML Schema:event)

4.3 Generazione di classi DVB in C++

Lo schema sopra descritto è stato pensato sia per veicolare i dati che per descrivere la struttura.Utilizzando gli strumenti elencati nel precedente paragrafo ed in particolare un parser SAX, si è fatto in modo che a partire dai documenti XML delle PSI/SI si generassero le relative classi C++.

Il vantaggio è che una volta compilato il parser la generazione di tutte le classi avviene in maniera automatica, senza dover procedere alla loro creazione una per una.

Mediante le classi C++ si hanno a disposizione degli utili metodi di accesso ai vari campi delle tabelle.Nel paragrafo successivo si illustrerà un esempio di parsing di una sezione EIT, ove, usando i sopracitati metodi verranno estratte e visualizzate tutte le informazioni da essa contenute.

A titolo di esempio si illustra il meccanismo seguito per la generazione della classe C++ EIT.

Come già spiegato in precedenza il SAX rientra nella categoria dei parser “event-based”, i quali prevedono la generazione di un evento ogni qual volta incontrano, durante la lettura del documento xml, un tag, un attributo o del testo.

Di seguito si riporta il documento xml della tabella EIT.

```
<?xml version="1.0" ?>
<section name="EventInformationSection">
   <field name="table_id" size="8"/>
   <field name="section_syntax_indicator" size="1"/>
   <aux name="reserved_future_use" size="1"/>
   <aux name="reserved" size="2"/>
   <length name="section_length" size="12"/>
   <field name="service_id" size="16"/>
   <aux name="reserved" size="2"/>
   <field name="version_number" size="5"/>
   <field name="current_next_indicator" size="1"/>
   <field name="section_number" size="8"/>
   <field name="last_section_number" size="8"/>
   <field name="transport_stream_id" size="16"/>
   <field name="original_network_id" size="16"/>
   <field name="segment_last_section_number" size="8"/>
   <field name="last_table_id" size="8"/>
   <loop name="events" body="event" lengthref="section_length-11-4"/>
```

```
   <check name="crc_32" size="32"/>
</section>
```

Secondo quanto detto sino ad ora, è prevedibile pensare che il parser sax contenga al suo interno una serie di condizioni, finalizzate al controllo del tipo di tag, attributo o testo incontrato.Ad ognuna di esse seguirà poi un'operazione che provvede alla creazione del particolare metodo da associarvi.
Ad esempio il tag “field” viene trattato dal parser nel seguente modo:

```
//  <field name="table_id"  size="8"/>
   else if (aLocalName == "field") {
      //---- .h ---------------------------------
//       *theOut2_h << "    uint32_t               " << myAtts["name"] << ";" << std::endl;
      //---- .cpp -------------------------------
      *theOut2_cpp << "    theInts[\"" << myAtts["name"] << "\"] = aReader->getBits(" << myAtts
["size"] << ");" << std::endl;

      *theOut3_cpp << "    aWriter->setBits((theInts.find(\"" << myAtts["name"] << "\"))->second, "
<< myAtts["size"] << ");" << std::endl;
      //---------------------------------------
   }
      //---- .h ---------------------------------
      *theOut2_h << "    uint32_t               " << myName << ";" << std::endl;
      //---- .cpp -------------------------------
      *theOut2_cpp << "    " << myName << " = aReader->getBits(" << mySize << ");" << std::endl;
     //---------------------------------------
```

Senza entrare nei dettagli si osserva comunque che è effettivamente presente il controllo sul nome del tag e che, una volta verificato essere “field” , si procede alla lettura dei valori assegnati ai due attibuti “name” e “size”.
Fatto ciò viene effettuato l'inserimento del metodo relativo nel file .h e .cpp della classe EIT che si sta generando.
Il risultato finale, comprensivo dei controlli su tutti gli altri tag è il seguente:

EventinformationSection.h

```
//***************************************************************************
 * HEADER FILE GENERATED AUTOMATICALLY BY XML2DVB_LIBGEN, DO NOT EDIT!     *
 *                                                                         *
 * Copyright (C) by MBI, Sep  6 2006                                       *
 * multimedia@mbigroup.it                                                  *
 *                                                                         *
 * WARNING! All changes made in this file will be lost!                    *
 ***************************************************************************/
#ifndef EIT_H
#define EIT_H
#include <map>
#include <list>
#include "bit_reader.h"
#include "bit_writer.h"
#include "dvb.h"
#include "creator.h"
#include "event.h"
class EventInformationSection : public Dvb {
   static Creator<EventInformationSection> theCreator;
public:
   EventInformationSection() { /* EMPTY */ }
   virtual EventInformationSection & EventInformationSection::operator << ( BitReader *
aReader );
   virtual EventInformationSection & EventInformationSection::operator >> ( BitWriter * aWriter );

   uint32_t            section_length;
   uint32_t            crc_32;
};
#endif
```

EventinformationSection.cpp

```
/***************************************************************************
 * CPP FILE GENERATED AUTOMATICALLY BY XML2DVB_LIBGEN, DO NOT EDIT!        *
 *                                                                         *
 * Copyright (C) by MBI, Sep  6 2006                                       *
 * multimedia@mbigroup.it                                                  *
 *                                                                         *
 * WARNING! All changes made in this file will be lost!                    *
 ***************************************************************************/
#include "eit.h"
Creator<EventInformationSection> EventInformationSection::theCreator
```

```
("EventInformationSection");
EventInformationSection & EventInformationSection::operator << ( BitReader * aReader ) {
  uint32_t myEndLoopByteCount;
   start = aReader->getByteCount();
   aReader->resetCrc();

   theInts["table_id"] = aReader->getBits(8);
   theInts["section_syntax_indicator"] = aReader->getBits(1);
   aReader->getBits(1);
   aReader->getBits(2);
   section_length = aReader->getBits(12);
   .
   .
   .
   theInts["last_table_id"] = aReader->getBits(8);

   myEndLoopByteCount = aReader->getByteCount() + (section_length-11-4);
   while (aReader->getByteCount() < myEndLoopByteCount) {
      Event * myItem = new Event();
      (*myItem) << aReader;
      add("events", myItem);
   }
   crc_32 = aReader->getBits(32);
   if (aReader->getCrc() == 0) {
      isValid = true;
   }
   return (*this);
}
EventInformationSection & EventInformationSection::operator >> ( BitWriter * aWriter ) {
   std::map<std::string, std::list<Dvb *> >::iterator myListI;

   start = aWriter->getByteCount();
   aWriter->resetCrc();

   aWriter->setBits((theInts.find("table_id"))->second, 8);
   aWriter->setBits((theInts.find("section_syntax_indicator"))->second, 1);
   aWriter->setBits(0, 1);
   aWriter->setBits(0, 2);
   aWriter->setBits(section_length, 12);
   section_length = aWriter->getByteCount();
   .
   .
```

```
    .
    aWriter->setBits((theInts.find("last_table_id"))->second, 8);

    myListI = theLoops.find("events");
    if (myListI!=theLoops.end()) {
        for (std::list<Dvb *>::iterator myIter=myListI->second.begin();
            myIter!=myListI->second.end(); myIter++) {
            *(*myIter) >> aWriter;
        }
    }
    crc_32 = aWriter->getCrc();
    aWriter->setBits(crc_32, 32);

    section_length = aWriter->getByteCount() - section_length;
    return (*this);
}
```

In breve si fa notare che questa classe come le altre, fa un uso congiunto delle liste e delle mappe, cioè di due delle strutture dati di base definite nella standard library C++.Le liste appartengono alla categoria dei contenitori sequenziali, ove l'ordine degli elementi nel contenitore è quello in cui essi sono stati inseriti.Le mappe invece rientrano nella categoria dei contenitori associativi, i quali a differenza di quelli sequenziali, utilizzano il valore degli oggetti per creare un ordinamento e per permetterne quindi un più rapido ritrovamento.Ogni oggetto è associato a un valore detto chiave.Data una chiave è possibile trovare l'oggetto associato in maniera molto efficiente.La chiave è di un tipo scelto a piacere, ad esempio mediante una stringa si può associare ad ogni oggetto del contenitore un nome, utilizzabile per ritrovarlo ed eventualmente modificarlo.
Nel paragrafo successivo viene riportato un esempio di parsing di una sezione EIT, grazie al quale risulterà più chiara la struttura generale delle classi e dei metodi ad esse associate.

4.4 Parsing C++ di una sezione EIT

In questo paragrafo viene presentato il metodo utilizzato per estrarre i contenuti di una sezione EIT da un dump di un Transport Stream DVB.

Come spiegato nel secondo capitolo i pacchetti TS costituenti il Transport Stream sono associati a dei sottoflussi (Elementary Stream) identificati dal campo PID dell'header.La sezione EIT, secondo specifiche, sarà da ricercarsi nell'elementary stream con PID 0x12.Al primo modulo del programma, denominato *Section_Choice,* è affidato il compito di costruire la sezione EIT e di salvarla in un file di uscita che sarà poi analizzato dal modulo *Section_Parser* per l'estrazione vera e propria dei vari campi EIT.

Nel file analizzato, il payolad di ogni pacchetto TS contiene esclusivamente una sezione e nel caso questa sia di dimensioni non multiple di 184 i rimanenti byte del payload sono riempiti dai byte di stuffing 0xFF.Questa premessa facilita il compito del programa visto che nel caso contrario si dovrebbe far riferimento al campo *pointer_field.*Infatti, se non si volessero utilizzare i byte di stuffing, finita una sezione si inizierebbe a mappare il contenuto della successiva già nei rimanenti byte del pacchetto TS.Settato il campo *payload_unit_start_indicator* ad uno, il primo byte del payload (*pointer_field)* conterrà le informazioni per capire dove finiscono i byte della sezione precedente e quindi dove iniziano quelli della successiva.Di seguito si riporta il codice Section_Choice:

SECTION_CHOICE

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif
#include <vector>
#include <iostream>
#include <cstdlib>
#include <stdio.h>
#include <stdlib.h>
#include <ts_packet.h>
#include <fstream>
#define TS_PACKET_LEN   188
#define MAX_EIT_SECT_LEN 4096
using std::streampos;
using namespace std;
```

```
int main(int argc, char **argv) {
   std::ofstream myNewFile ("section.dat", ios_base::binary);
   ifstream file;
   unsigned char mySectionBuffer[MAX_EIT_SECT_LEN];
   unsigned char myTsBuffer[TS_PACKET_LEN];
   unsigned char * myCursorTsbuffer;
   unsigned char * myCursorSectionBuffer;
   unsigned char * myPayload;
   char * myTsdumpFileName;
   int myTableid;
   int myLength;
   long int myByteReadIndex=0;
   if (argc < 3) {
      cout << "needed 2 arguments" << endl;
      exit(1);
   }
   myTsdumpFileName = argv[1];
   myTableid = strtol(argv[2],NULL,0);
   file.open(myTsdumpFileName,ios::binary);
   while(file.get()!=0x47); //synchronization phase
   file.unget();
   bool isTableIdFound = false;
   bool mySectionStarted;
   bool mySectionCompleted;
   TsPacket * myPacket;
   while (isTableIdFound == false){ //condition to check if the requested section is found
      mySectionStarted=false;
      mySectionCompleted=false;
      myCursorSectionBuffer = mySectionBuffer;
      uint16_t mySectionLength=0;
      while (mySectionCompleted==false) {
         myCursorTsbuffer = myTsBuffer;
         streampos myStreamPosition;
         myStreamPosition = file.tellg();
         file.read((char *)myCursorTsbuffer, TS_PACKET_LEN);
         if (file.eof()){
            cout << "END OF FILE REACHED, TABLE_ID DIDN'T FOUND!!!" << endl;
            exit(0);
         }
         cout << ".";
         myPacket=new TsPacket(myCursorTsbuffer); //costructor of the TsPacket type object
         bool mySectionCheck=false;
```

```
        int myPid;
        myPid=myPacket->getPid();
        if (myPacket->getPusi()==1 && myPid==0x12 && mySectionStarted==false){ //checking
        for a EIT  section beginning in a TSPacket with PID=0x12
            mySectionStarted = true;
            mySectionCheck=true; // used to skip the third if
            fprintf(stdout, "reached pusi=1, beginning of new section");
            myPacket->getPayload(&myPayload,&myLength);
            memcpy(myCursorSectionBuffer,myPayload,myLength);//copying  payload into
            mySectionBuffer
            myCursorSectionBuffer+=myLength;
            mySectionLength+=myLength;
        }
        if (myPacket->getPusi()==0 && myPid==0x12 && mySectionStarted==true){//checking
         presence of further payloads to copy
            cout << "pusi 0, writing *" << endl;
            myPacket->getPayload(&myPayload,&myLength);
            memcpy(myCursorSectionBuffer,myPayload,myLength);
            myCursorSectionBuffer += myLength;
            mySectionLength+=myLength;
        }
        if (myPacket->getPusi()==1 &&myPid==0x12 && mySectionCheck==false){//checking end
         of section
            mySectionCompleted = true;
            fprintf(stdout, "reached second pusi=1, section ended --->");
            file.seekg(myStreamPosition);
        }
      delete myPacket;
      }
      if ((mySectionBuffer[0]==myTableid) && mySectionCompleted==true) {//checking table_id
         fprintf(stdout, "tableID number 0x%x found and copied in section.dat file!\n",
mySectionBuffer[0]);
         isTableIdFound = true;
         int myIndex = 0;
         while (myIndex <= mySectionLength) && mySectionBuffer[myIndex]!=0xff{// copying the
          right section into the section.dat file till the first stuffing byte (oxff)
                  myNewFile << mySectionBuffer[myIndex++];
         }
      }
      else{
        fprintf(stdout, "wrong table_id, new section capture!\n");
      }
```

```
    }
    myNewFile.close();
}
```

Il programma prevede l'inserimento da parte dell'utente di due parametri: il nome del file da analizzare e il table_id della sezione EIT da catturare.
Per prima cosa il codice provvede all'apertura del file contenente il dump DVB e alla ricerca del *Sync_Byte* per sincronizzare lo stream sul primo byte del pacchetto TS.
Fatto ciò viene costruito l'oggetto *ts_packet* passando al relativo costruttore il puntatore ai blocchi di 188 byte letti sequenzialmente dal file.La classe *ts_packet* fornisce dei metodi per accedere ai contenuti dei vari campi dell'header (ad esempio *get_pusi()* e *get_pid()*) e per l'estrazione del payload *(get_payload())*.
Il programma procede ora al controllo, pacchetto per pacchetto, dei campi PID e Payload_Unit_Start_Indicator (PUSI).Le condizioni da verificare sono che il pacchetto in questione appartenga all'elementary stream con PID 0x12 e che il relativo PUSI sia settato ad uno.Questa condizione assicura che il payload sia associabile all'inizio della sezione.In tal caso viene estratto mediante il metodo getpayload() ed inserito nel buffer *SectionBuffer.*Letti altri 188 byte, viene creato un nuovo oggetto *ts_packet*.I casi ora possono essere due:

- La sezione appena iniziata ha una lunghezza minore o uguale a 183byte e quindi il nuovo pacchetto letto sarà da associare alla sezione successiva.Ciò comporta la presenza del PUSI settato ad uno.Questa condizione si differenzia dalla prima per il fatto che ora la sezione è già iniziata (infatti il booleano *mySectionStarted* è già stato settato a true nel loop precedente).
- La sezione ha una lunghezza maggior di 183 byte e quindi il payload del nuovo pacchetto letto appartiene ancora a lei.In tal caso il PUSI è settato a 0 e si continuerà a copiare sequenzialmente il payload in *SectionBuffer.*

Alla fine *SectionBuffer* contiene l'intera sezione.Controllato il *table_id,* se coincide con quello richiesto dall'utente, tutto il contenuto del buffer, sino al primo stuffing byte escluso (0xff), viene copiato nel file section.dat, viceversa si procede all'acquisizione di una nuova sezione.La schermata riportata in figura

4.12 è relativa all'esecuzione del programma per la ricerca della sezione EIT con *table_id* = 0x51 (TS attuale, event schedule information) nel file *eit_dump.ts.*

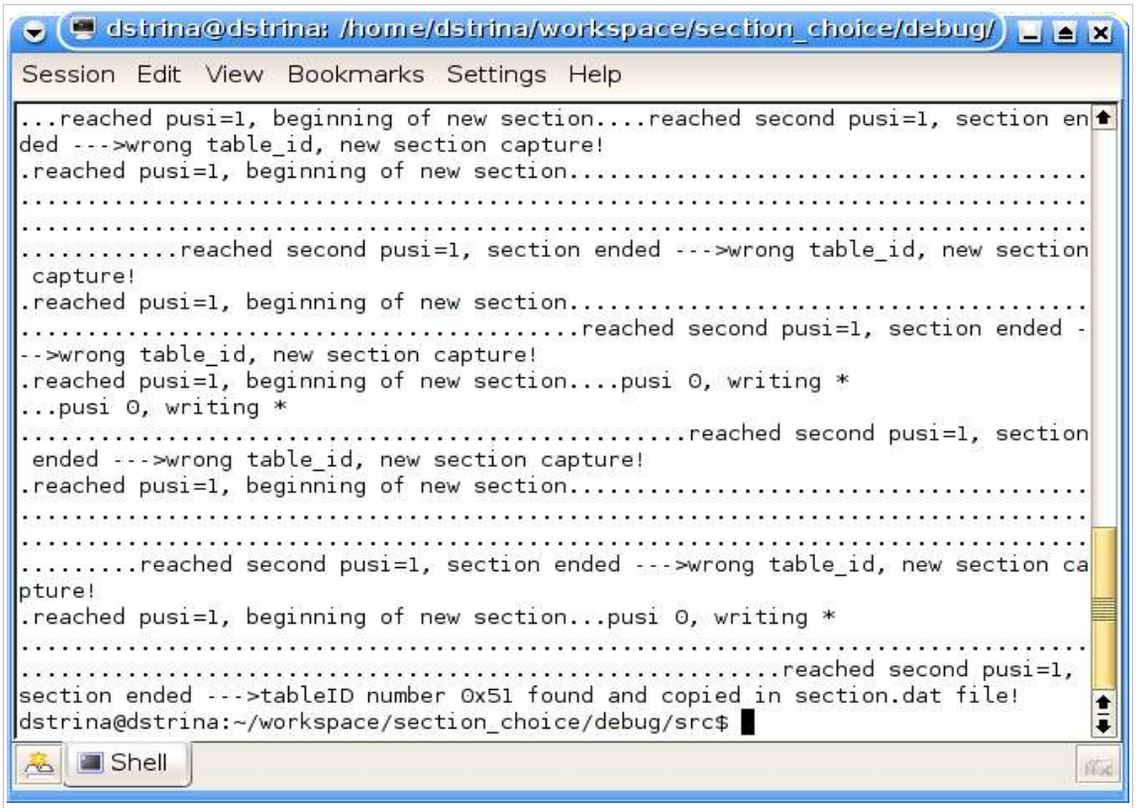

figura4.12 (esecuzione del programma section_choice)

Una volta che la sezione richiesta è stata catturata e copiata nel file *section.dat*, si passa al secondo modulo del programma, *Section_Parser*.

SECTION_PARSER

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif
#include <vector>
#include <iostream>
#include <cstdlib>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <event.h>
#include <bit_reader.h>
#include <byte_stream.h>
```

```
#include <eit.h>
#include <mjd.h>
#include <bcd_time_conv.h>
#include <map>
#include <list>
#define Sect_Packet_Len   184
#define MAX_EIT_SECT_LEN 4096
using namespace std;
int main() {
   ifstream file;
   unsigned char mySectionBuffer[MAX_EIT_SECT_LEN];
   file.open("section.dat",ios::binary);
   unsigned char * myCursorSectionBuffer;
   myCursorSectionBuffer = mySectionBuffer;
   while(!file.eof()){  //copying section into mySectionBuffer
      file.read((char*)myCursorSectionBuffer,Sect_Packet_Len);
      myCursorSectionBuffer+=Sect_Packet_Len;
   }
   uint8_t * myEitpoint;
   int alengh;
   EventInformationSection myEit;
   alengh=sizeof(mySectionBuffer);
   myEitpoint=mySectionBuffer;
   ByteBuffer     myByteStream(myEitpoint, alengh);
   BitReader      myBitReader(&myByteStream);
   myEit << &myBitReader; //Associating the EventInformationSection object to the bitreader
         method
   //using the MAP associated to the EventInformationSection to print some fields
   fprintf(stdout, "table_id = 0x%x\n", myEit.theInts["table_id"]);
   fprintf(stdout, "section_length = %i\n", myEit.section_length);
   fprintf(stdout, "service_id = 0x%x\n", myEit.theInts["service_id"]);
   fprintf(stdout, "version_number = %i\n", myEit.theInts["version_number"]);
   fprintf(stdout, "section_number = %i\n", myEit.theInts["section_number"]);
   fprintf(stdout, "transport_stream_id = 0x%x\n", myEit.theInts["trasnsport_stream_id"]);
   fprintf(stdout, "original_network_id = 0x%x\n", myEit.theInts["original_network_id"]);
   //entering into the EIT event loop by the map "theLoops"
   for (std::map<std::string, std::list<Dvb *> >::const_iterator myloop=myEit.theLoops.begin();
      myloop!=myEit.theLoops.end(); myloop++) {
      for (std::list<Dvb *>::const_iterator myList=myloop->second.begin(); myList!=myloop->
         second.end(); myList++) {
         //Associating to dvb type the Event object and accessing to its field by the relative map
         Event * myEvent;
```

```
myEvent = (Event *)*myList;
mjd myStartdate(myEvent->theInts["start_date"]);
cout << "MJD Start Date = " << myEvent->theInts["start_date"] << endl;
cout << "dd/mm/yy Start date >>" << myStartdate.getday() << "/" <<
 myStartdate.getmonth() << "/"    << myStartdate.getyear() << " Giorno della settimana:"
 << myStartdate.getweek_day() << "                Settimana Numero:" <<
 myStartdate.getweek_number() << endl;
bcd_time_conv myDuration(myEvent->theInts["duration"]);
cout << "DURATION: " << myDuration.getHourDec() << ":" << myDuration.getMinDec() <<
 ":" <<  myDuration.getSecDec() << endl;
fprintf(stdout, "Event_id = 0x%x\n", myEvent->theInts["event_id"]);
//entering into the EIT event descriptor loop by the map "theLoops"
for (std::map<std::string, std::list<Dvb *> >::const_iterator myD=(*myEvent).
 theLoops.begin(); myD!=(*myEvent).theLoops.end(); myD++) {
  for (std::list<Dvb *>::const_iterator myDesc=myD->second.begin(); myDesc!=myD-
   >second.end    (); myDesc++) {
      //Associating to the dvb type the Descriptor object and accessing to its field by the
      relative map
      Descriptor * myDescriptor;
      myDescriptor = (Descriptor *)*myDesc;
      fprintf(stdout, "Descriptor_tag = ox%x\n", myDescriptor->theInts["descriptor_tag"]);
      fprintf(stdout, "Descriptor_length = %i\n", myDescriptor->descriptor_length);
      vector<uint8_t> myData;
      myData=myDescriptor->data; //Descriptor data following the desc_length field!
      int cont=0; //language code field analysis
      char ch;
      cout << "LANGUAGE CODE>>";
      while(cont!=3){
         ch=myData[cont];
         cout << ch;
         cont++;
      }
      cout << endl;
      int myStringLength=myData[cont]; //event name analysis
      cout << "EVENT NAME>>";
      while(cont<=myStringLength+3){
         ch=myData[cont];
         cout << ch;
         cont++;
      }
      cout << endl;
      myStringLength=myData[cont]; //text analysis
```

```
                int myEndText = cont + myStringLength;
                cout << "TEXT>>";
                while(cont!=myEndText){
                   ch=myData[cont];
                   cout << ch;
                   cont++;
                }
                cout << endl;
             }
          }
       }
    }
}
```

In questo codice compaiono i riferimenti alle classi DVB introdotte nel precedente paragrafo, oltre ad altre create ad hoc per convertire la data dal formato MJD a quello Gregoriano (*mjd.h*) e il campo durata dal formato bcd a ore/minuti/secondi (*bcd_time_conv.h*).

Una volta aperto il file *section.dat,* se ne copia l'intero contenuto nel buffer *mySectionBuffer.*

Intervengono adesso le classi *ByteBuffer* e *BitReader*.La prima è una classe creata per gestire le funzionalità di base di uno stream, tra le quali quella del calcolo del crc a 32bit.La seconda permette di applicare operazioni di bitwise sui bit letti da un bytestream.Possono essere letti dati di una lunghezza arbitraria sino a 32bits.

Il codice associa l'oggetto EventInformationSection alla classe bitreader, dopodichè si procede alla stampa di alcuni suoi campi richiamando le coppie chiave valore della relativa mappa.Si riporta di seguito una parte di questa mappa:

```
.
.
.
theInts["table_id"] = aReader->getBits(8);
theInts["section_syntax_indicator"] = aReader->getBits(1);
aReader->getBits(1);
aReader->getBits(2);
section_length = aReader->getBits(12);
```

```
theInts["service_id"] = aReader->getBits(16);
aReader->getBits(2);
theInts["version_number"] = aReader->getBits(5);
theInts["current_next_indicator"] = aReader->getBits(1);
theInts["section_number"] = aReader->getBits(8);
theInts["last_section_number"] = aReader->getBits(8);
theInts["transport_stream_id"] = aReader->getBits(16);
theInts["original_network_id"] = aReader->getBits(16);
theInts["segment_last_section_number"] = aReader->getBits(8);
theInts["last_table_id"] = aReader->getBits(8);
.
.
.
```

E' interessante notare che tutte le sezioni e descrittori sono associabili alla generica classe *dvb.h*.E' qui che sono definiti i tre differenti tipi di mappe utilizzate;

```
std::map<std::string, uint32_t>        theInts;
std::map<std::string, std::string>     theStrings;
std::map<std::string, std::list<Dvb *> >  theLoops;
```

Tutte prevedono che la chiave sia di tipo stringa, mentre ciò che cambia è il valore ad essa associato.In *theInts* è un *uint32_t* e in *theString* una stringa.
TheLoops, come si intuisce dal nome, gestisce i loop, nei quali i valori mappati non sono altro che una lista di puntatori ad oggetti DVB, i quali a loro volta possono contenere altri sotto loop.
Nel caso specifico della EIT, si incontra un primo loop contenente oggetti *event.*
Dichiarato l'iteratore *myLoops* del tipo *theLoops*, cioè *map<std::string, std::list<Dvb *>,* per accedere poi agli oggetti *event* si dichiara un'altro iteratore *myList del tipo list<Dvb *>,* specificando che il particolare tipo DVB in questione è l'*event,* la cui mappa è quella sottostante.

```
.
.
.
theInts["event_id"] = aReader->getBits(16);
theInts["start_date"] = aReader->getBits(16);
```

```
theInts["start_time"] = aReader->getBits(24);
theInts["duration"] = aReader->getBits(24);
theInts["running_status"] = aReader->getBits(3);
theInts["free_ca_mode"] = aReader->getBits(1);
descriptors_loop_length = aReader->getBits(12);

myEndLoopByteCount = aReader->getByteCount() +    (descriptors_loop_length);
while (aReader->getByteCount() < myEndLoopByteCount) {
   Descriptor * myItem = new Descriptor();
   (*myItem) << aReader;
   add("descriptors", myItem);
}
.
.
.
```

Come noto l'oggetto *event* prevede la presenza di un'ulteriore loop di oggetti *descriptor*, ai quali si accede nella stessa maniera.

La sezione EIT analizzata contiene solo lo ShortEventDescriptor.Si fa notare che nel codice si è utilizzata la generica classe *descriptor,* valida per qualsiasi descrittore.Infatti come si vede dal relativo codice riportato di seguito, sono previsti i due campi, *descriptor_ tag descriptor_length,* comuni a tutti i descrittori e un vettore *data* contenente la rimanente parte.

```
.
.
.
theInts["descriptor_tag"] = aReader->getBits(8);
descriptor_length = aReader->getBits(8);

myEndLoopByteCount = aReader->getByteCount() + (descriptor_length);
while (aReader->getByteCount() < myEndLoopByteCount) {
   data.push_back((uint8_t)aReader->getBits(8));
}
.
.
.
```

Sarebbe stato possibile utilizzare la classe *ShortEventDescriptor*, che mette a disposizione metodi più specifici.Ad esempio quello relativo all'estrazione del

campo *ISO_639_language_code.*

theInts["ISO_639_language_code"] = aReader->getBits(24);

La schermata di figura 4.13 riporta l'analisi effettuata da *SectionParser* della sezione catturata precedentemente da *Section_Choice.*

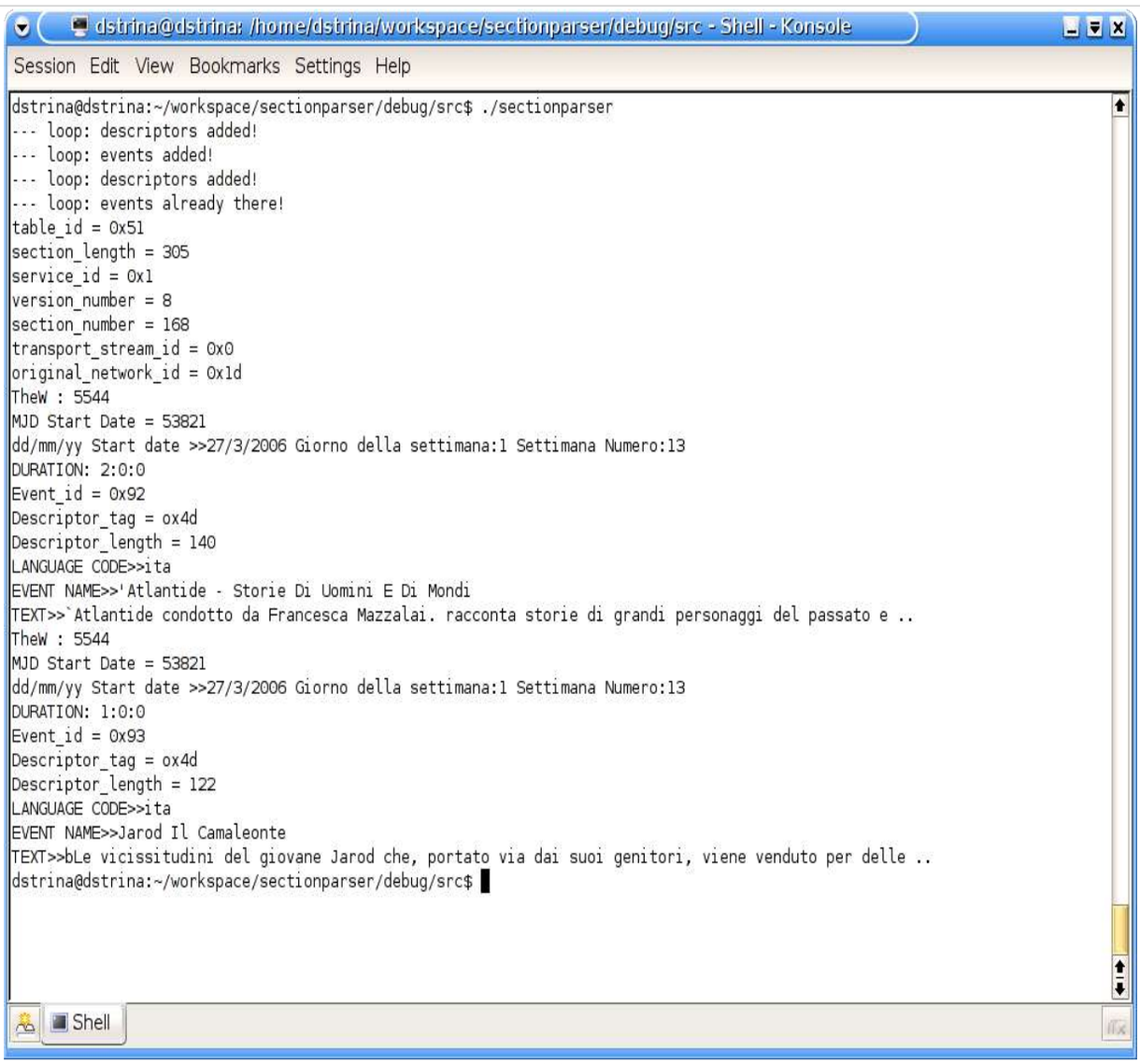

figura4.13 (esecuzione del programma SectionParser)

4.5 Generazione di flussi DVB, MBI Eracle.

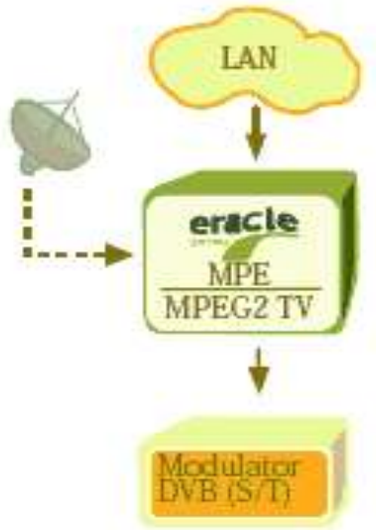

figura4.11 (Sistema Eracle)

Eracle è un sistema gateway IP-DVB che permette la trasmissione di flussi dati e flussi audio/video MPEG2 su un unico transport stream di uscita.E' possibile associare ad un PID il traffico sulla base di diversi parametri: indirizzo IP, protocollo di livello superiore (TCP/UDP,ICMP,IGMP), porta sorgente o di destinazione, etc.L'allocazione dinamica della banda implementata da Eracle si basa sull'algoritmo token bucket.

Il sistema permette di multiplexare flussi audio e video MPEG2 con il traffico dati in tempo reale, minimizzando i costi della banda satellitare utilizzata.Inoltre supporta flussi MPEG2-TS over IP in conformità alla rfc 2250, o soluzioni proprietarie quali quelle di MINERVA Networks o MBI Niagara Streaming Server.

I pacchetti TS, prelevati dal flusso d' ingresso, vengono trasmessi sullo stream TS di uscita mediante porta DVB/ASI.

Il processo di incapsulamento di pacchetti dati IP in DVB viene implementato mediante il protocollo MPE.

E' configurabile tramite interfaccia WEB via HTTP o HTTPS.

Prevede la presenza di un'apposita sezione con informazioni di debug utili per risolvere problemi legati ad errate configurazioni.Consente di monitorare in tempo reale costantemente l'occupazione della banda sul canale di uscita da parte dei PID configurati.Le prestazioni del gateway possono essere controllate tramite protocollo SNMPottenendo grafici in continuo aggiornamento.

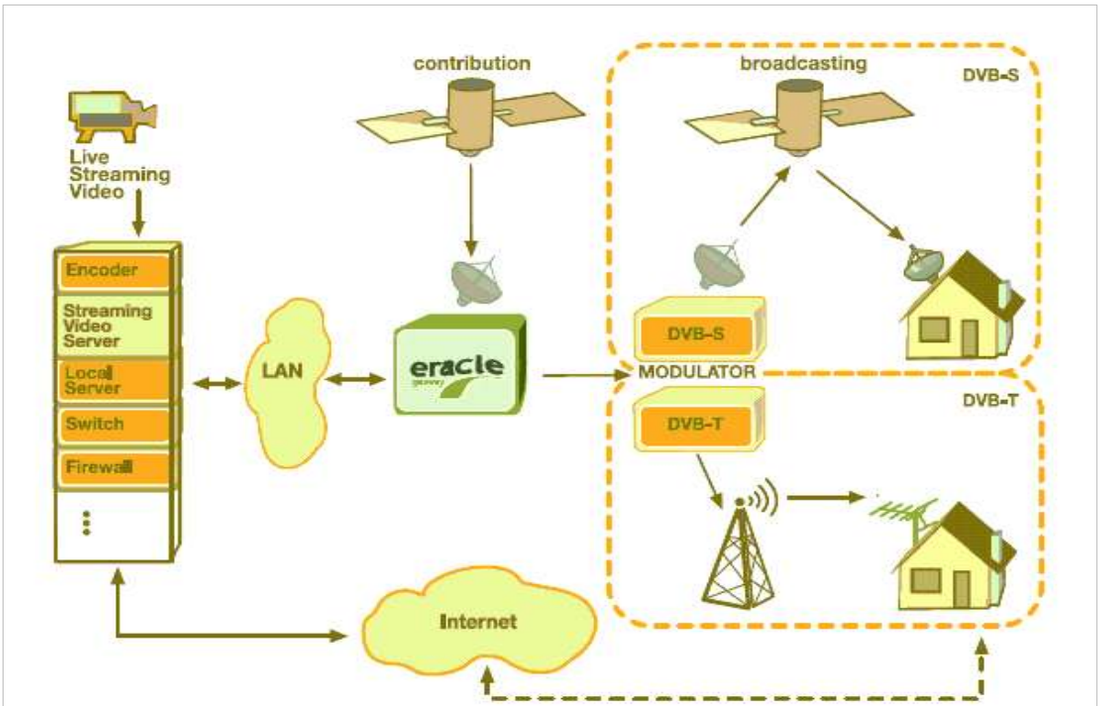

figura4.12 (Eracle:scenario di funzionamento)

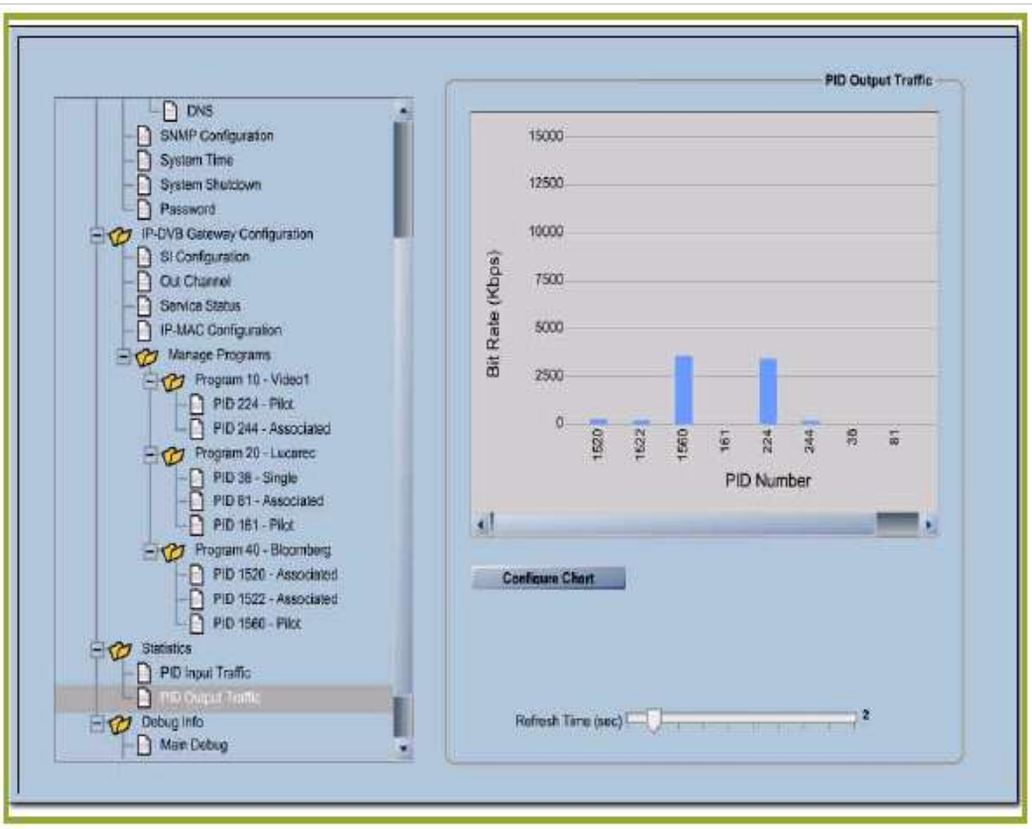

figura4.13 (Eracle:interfaccia)

Eracle può essere facilmente integrato in ogni tipo di soluzione complessa, disponendo di una interfaccia di configurazione basata su protocollo XML-RPC. Di seguito si riporta un riassunto delle caratteristiche e specifiche tecniche dell'apparato:

Caratteristiche

- compatibile con la maggior parte dei modulatori DVB-S e DVB-T
- da MPEG-2 su Ip a MPEG2 DVB/ASI
- da Ip data a MPE encapsulation (ETSI-EN 301-192)
- multiplexing di pacchetti MPEG2 DVB/ASI e DVB/MPE
- tabelle SI/PSI supportate: PAT, PMT, NIT, SDT (conforme a EN 300-468)
- supporto per Ip unicast e multicast (RFC 1112) banda per PID gestita secondo l'algoritmo token bucket
- 8192 PID supportati di cui 250 contemporaneamente
- server basato su OS Linux, montato su rack
- prestazioni controllabili tramite protocollo SNMP
- interfaccia web di configurazione

Applicazioni

- broadcast di canali MPEG2 TV audio/video
- servizi IP su banda larga: trasferimento di file, Web surfing, download di software, e-mail, e-learning, trasmissioni in multicast

Specifiche tecniche

Sistema di controllo
- interfaccia Web di amministrazione
- protocollo SNMP
- interfaccia di controllo XML-RPC

Protocolli network
- UDP, TCP
- supporto multicast (RFC 1112)
- MPEG over IP

IP Encapsulation
- Data Streaming
- DVB MPE

Filtraggio IP
- IP adress
- protocolli supportati (TCP/UDP, ICMP, IGMP)
- TCP/UDP source/destination port

Input
- fino a 4 10/100 porta Fast Ethernet o fino a 2 Gigabit Ethernet (opzionale)
- 300 diversi input IP simultanei
- DVB-MPEG2 TS su cards DVB-S o DVB-T

Output
- interfaccia ASI di output fino a 160 Mbit/s

Riferimenti bibliografici:

- *W3C Xml tutorial (http://www.w3schools.com/xml)*
- *Xml tutorial:Learn About Key E-commerce Trends and Technologies at Your Own Pace.* (*www.gxs.com/pdfs/Tutorial/Tutor_XML_GXS.pdf*)
- *Gateway MBI Eracle data sheet (http://www.mbigroup.it/en/products/eracle/eracle_en.pdf)*

5. Sistema MBI EPG

5.1 Architettura del sistema

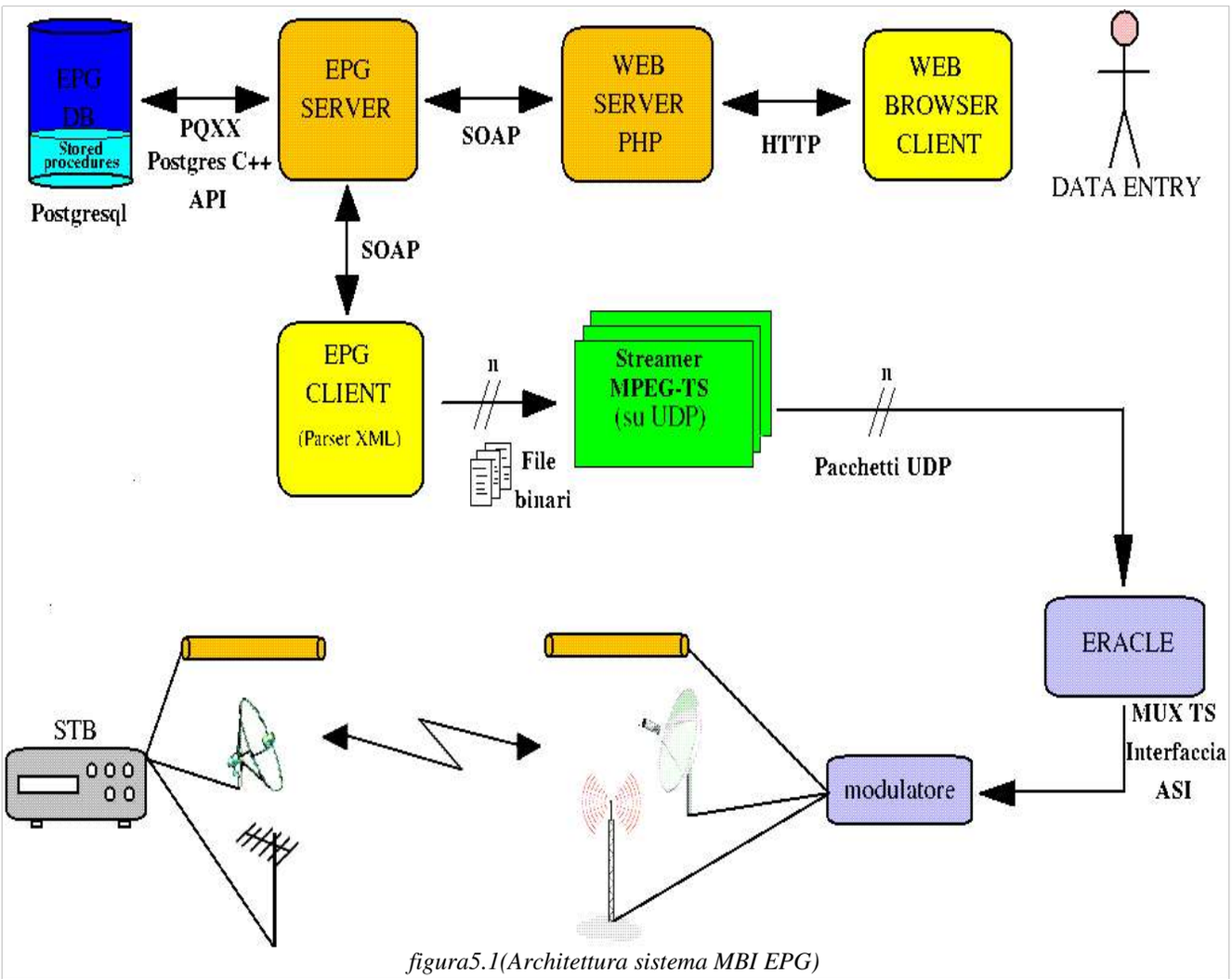

figura5.1(Architettura sistema MBI EPG)

In figura 5.1 è mostrata l'architettura end to end del sistema MBI EPG.Nel presente paragrafo se ne fornisce una prima presentazione riservandosi di approfondire l'analisi dei principali blocchi nei successivi.

Il database utilizzato nel progetto è Postgresql, l'organizzazione interna dei dati verrà spiegata in seguito, per il momento si sottolinea solo che le sezioni PSI/SI, tutti i relativi descrittori e lookup table sono state memorizzate in formato XML.Dal punto di vista del database (DB) ciò vuol dire avere semplicemente a che fare con dati di tipo testo, il contenuto XML è riconosciuto come tale solo quando viene processato mediante XSLT dal web server.Il DB è fornito anche di

una tabella contenente le informazioni di login dei vari utenti autorizzati all'uso del sistema.
Il web server PHP mette a disposizione le pagine HTML utilizzate per creare l'interfaccia web dell'EPG client.Attraverso di essa l'utente del sistema MBI EPG può inserire ed editare i contenuti delle sezioni PSI/SI.
Un ruolo molto importante è ricoperto dal server EPG il quale fornisce dei metodi che permettono di interrogare e gestire il DB remoto.Come indicato nello schema, la comunicazione dei client con il server EPG avviene tramite remote procedure call SOAP, mentre quella con il DB Postgresql mediante l'apposita libreria C++ libpqxx.
L'EPG client (Parser XML) ha il compito di prelevare i documenti XML dal DB via EPG server e in seguito ad un loro parsing di fornire in uscita i relativi file binari, i quali vengono pacchettizzati dallo Streamer MPEG-TS UDP.Il sistema Eracle effettua la multiplazione degli stream in ingresso per poi inviare il segnale in uscita all'apparato di trasmissione.

5.2 Database EPG

Come gia accennato nel precedente paragrafo, il sistema MBI EPG fa uso del database Postgresql versione 7.4.7.Tale DB appartiene alla categoria dei Relational DataBase Management System (RDBMS).Il modello relazionale è un modello logico di rappresentazione dei dati che si basa sul concetto di relazione e di tabella.La gestione avviene mediante il linguaggio di alto livello SQL (Structured Query Language).Entrando nello specifico del progetto MBI EPG si descrive ora come sono organizzati i dati all'interno del DB.

• Tabella *t_user:*

La prima tabella chiamata in causa dall'accesso al sistema di Data Entry (procedura di login) è quella denominata *t_user*.Qui sono memorizzati tutti i dati (tra i quali username e password, quest'ultima criptata) degli utenti registrati e quindi autorizzati all'utilizzo del sistema (vedi figura 5.2).Fra i campi contenuti in ogni riga è stato inserito anche il booleano *is_administrator.*Infatti, la gestione degli utenti non è demandata esclusivamente, come di solito succede, al gestore della base di dati, ma è stato prevista la creazione di un amministratore al quale è concesso accedere alla tabella utenti sia in lettura che in scrittura.

oid	user_id serial	username text	password text	firstname text	surname text	company text	email text	phone text	fax text	notes text	is_administrator bool
16443	18	nbonelli	secret	Nicola	Bonelli	MBI	nbonel	050	050	test	FALSE
16443	21	rsciascia	c36407b8c	Roberto	Sciascia	MBI	rsciasc	050	050	test	FALSE
16443	24	dstrina	10c2a6f3d	Diego	Strina	MBI	dstrina	050	050	Tesista	FALSE
16443	25	admin	f6fdffe48c9	"	"	"	"	"	"	"	TRUE
16443	27	avaccaro	733521ff0a	Attilio	Vaccaro	MBI	avacca	050	050	test	FALSE
16443	28	dsparano	b32d579af	Daniele	Sparano	MBI	dspara	050	050	test	FALSE

figura5.2 (Tabella t_user)

```
CREATE TABLE t_user
(
  user_id serial NOT NULL,
  username text NOT NULL,
  "password" text NOT NULL,
  firstname text,
  surname text,
  company text,
  email text,
  phone text,
  fax text,
  notes text,
  is_administrator bool,
  CONSTRAINT pk_t_user PRIMARY KEY (user_id),
  CONSTRAINT uq_t_user_username UNIQUE (username)#
```

Dall'analisi del sovrastante codice SQL usato per la creazione della tabella *t_user,* si nota che ai campi *username* e *user_id* sono stati assegnati rispettivamente i vincoli *(constraint) unique* e *primary key.*Il primo indica che il dato contenuto nella rispettiva colonna *(username)* deve essere unico rispetto a tutti quelli della altre righe, mentre il secondo oltre a garantirne l'unicità assicura che non sia nullo.

La *primary key,* date le sue caratteristiche, può quindi essere usata come unico identificatore per le righe della tabella.

- **Tabella *t_entity_templates*:**

Con l'analisi di questa tabella si comincia ad entrare più nel dettaglio dell'organizzazione dei dati del sistema MBI EPG.Il nome *t_entity_templates* da già indicazioni sulle finalità d'uso di questi dati.Template è un termine inglese che può essere tradotto in modello.La tabella infatti contiene i modelli di tutte le entità utilizzate per la gestione delle sezioni PSI/SI.Nel caso specifico si definisce modello il documento XML che ne descrive esclusivamente la grammatica e che quindi non è ancora stato popolato.Come sarà mostrato in seguito infatti, in risposta alla selezione da parte dell'utente, di una tabella da editare, si richiede al DB l'invio del relativo modello XML, il quale verrà processato dal parser XSLT per presentarne i relativi campi in maniera intuitiva via interfaccia web.

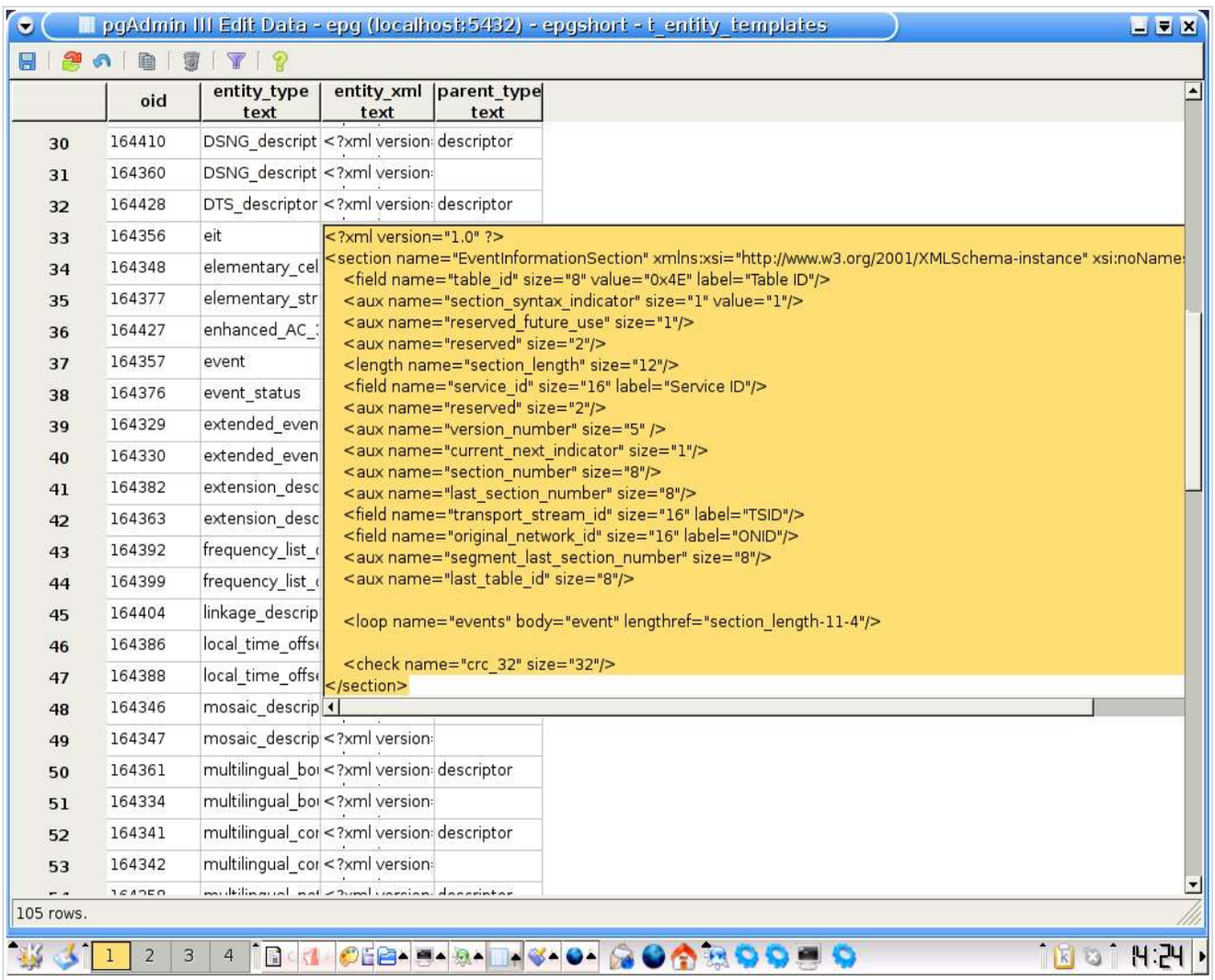

figura5.3 (Tabella t_entity_templates)

Dalla figura 5.3 si nota la presenza delle tre colonne di tipo testo, *entity_type, entity_xml e parent_type.Entity_type* identifica univocamente ogni entità attraverso il relativo nome.Nel codice SQL utilizzato per la creazione della tabella è infatti presente la seguente linea:

CONSTRAINT pk_t_entity_templates PRIMARY KEY (entity_type),

Come già visto per la *t_user*, anche la *t_entity_templates* presenta una colonna alla quale è associata la *primary key*.E' questa una regola generale da seguire per tutte le tabelle contenute nel DB e ciò è previsto proprio dalla teoria dei relational database.
Nella colonna *entity_xml* sono inseriti i già citati modelli XML.In figura 5.3, a titolo di esempio, è stato evidenziato il contenuto relativo all'entità *eit* e si trova conferma del fatto che esso rappresenti solo la descrizione xml non popolata, della sezione EIT.
L'ultima colonna, la *parent_type,* è stata inserita per distinguere i descrittori dalle altre entità.Si vedrà infatti che dall'interfaccia web, una volta entrati ad esempio nella pagina da cui è possibile editare la EIT, mediante selezione del pulsante *"add descriptor"* sarà possibile associare ad un evento uno o più descrittori.La scelta averrà mediante consultazione di una drop down list, la quale elenca tutti i descrittori messi a disposizione dal sistema.La creazione di tale lista passa appunto da una query rivolta al DB che seleziona tutte le entità in *entity_templates* il cui campo *parent_type contiene* il testo *descriptor.*

- **Tabella *t_entities:***

Non appena l'utente decide di aggiungere ad esempio una NIT, si chiama in causa la tabella *t_*entities.La prima operazione effettuata dal sistema è sempre quella di recuperare il relativo modello da *t_enity_templates* per presentarne i campi da editare via interfaccia web, ma parallelamente viene allocata la stessa entità in *t_entities.*
In figura 5.4 si riporta la struttura di questa tabella.

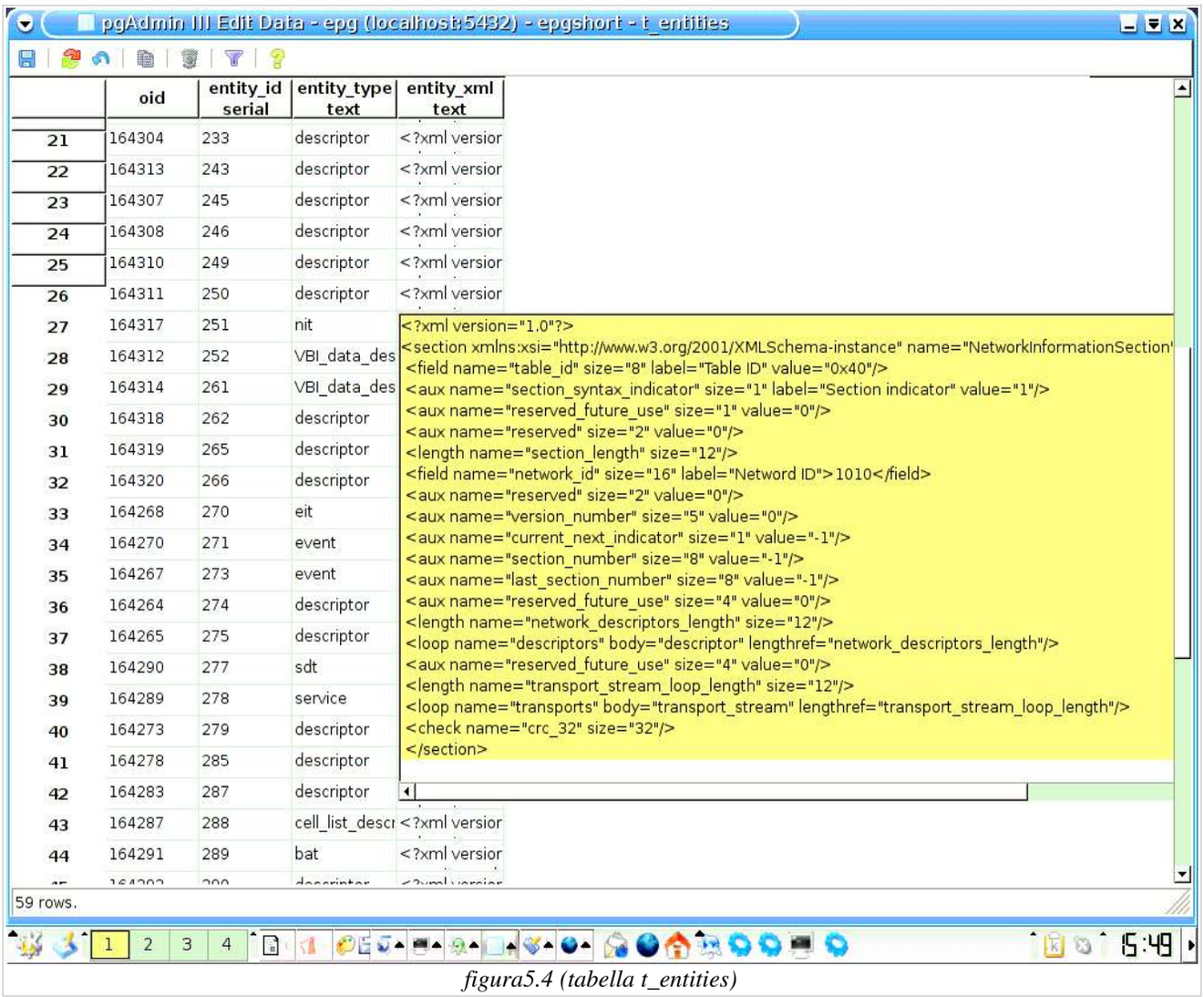

figura5.4 (tabella t_entities)

Oltre *alle* già viste colonne *entity_type* e *entity_xml,* questa tabella prevede la presenza della colonna di tipo serial *entity_id.*Il tipo *serial* non è un vero e proprio tipo, ma solo una notazione convenzionale per associare un identificatore unico agli elementi della colonna.Infatti ad ogni nuovo istanziamento il sistema ne aumenta serialmente il valore e ciò assicura che l'*entity_id* sia associabile ad una unica entità.

La differenza sostanziale della colonna *entity_xml* dall'omonima della tabella *t_entity_templates* è intuibile già da una rapida analisi del contenuto evidenziato in figura.Si ha ancora a che fare con un documento XML ma ora non è più solo un modello, bensì comincia ad essere popolato.Al momento della cattura dell'immagine era appena stato inserito il valore 1010 nel campo *network_id.*

- **Tabella *t_ent_ent:***

L'uso di questa tabella è strategico per la gestione dei loop contenuti nelle sezioni.Per illustrarne il principio di funzionamento si ritorna all'esempio della NIT, della quale sono stati riportati in figura 5.5 la tabella definita dallo standard DVB e a seguire la relativa descrizione XML.

Syntax	Number of bits	Identifier
network_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
network_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
network_descriptors_length	12	uimsbf
for(i=0;i<N;i++){ Network descriptor loop		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0;i<N;i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0;j<N;j++){ Transport stream loop		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

figura5.5 (semantica della tabella NIT)

```
<?xml version="1.0"?>
 <section  name="NetworkInformationSection"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
 instance" xsi:noNamespaceSchemaLocation="dvb.xsd">
<field name="table_id" size="8" label="Table ID" value="0x40"/>
<aux name="section_syntax_indicator" size="1" label="Section indicator" value="1"/>
<aux name="reserved_future_use" size="1" value="0"/>
<aux name="reserved" size="2" value="0"/>
```

```
<length name="section_length" size="12"/>
<field name="network_id" size="16" label="Network ID"/>
<aux name="reserved" size="2" value="0"/>
<aux name="version_number" size="5" value="0"/>
<aux name="current_next_indicator" size="1" value="-1"/>
<aux name="section_number" size="8" value="-1"/>
<aux name="last_section_number" size="8" value="-1"/>
<aux name="reserved_future_use" size="4" value="0"/>
<length name="network_descriptors_length" size="12"/>
<loop name="descriptors" body="descriptor" lengthref="network_descriptors_length"/>
<aux name="reserved_future_use" size="4" value="0"/>
<length name="transport_stream_loop_length" size="12"/>
<loop name="transports" body="transport_stream" lengthref="transport_stream_loop_length"/>
<check name="crc_32" size="32"/>
</section>
```

Questa sezione contiene al suo interno due loop separati, identificati nel relativo documento XML dai tag *loop*.La descrizione della semantica di ognuno di essi è affidata ad altri due documenti XML (riportati di seguito), anch'essi contenuti nella tabella *t_enity_templates*.

```
<?xml version="1.0"?>
<transport_stream  name="Transport"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dvb.xsd">
   <field name="transport_stream_id" size="16" label="tsid"/>
   <field name="original_network_id" size="16" label="onid"/>
   <aux name="reserved" value="15" size="4"/>
   <length name="transport_descriptors_length" size="12"/>
   <loop name="descriptors" body="descriptor" lengthref="transport_descriptors_length"/>
</transport_stream>

<?xml version="1.0" ?>
<descriptor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dvb.xsd">
   <field name="descriptor_tag" size="8" label="Tag" lookup="t_descriptors"/>
   <length name="descriptor_length" size="8"/>
   <raw name="data" lengthref="descriptor_length"/>
</descriptor>
```

La loro associazione alla sezione di appartenenza è resa possibile dall'attributo *body* del tag *loop,* infatti il nome ad esso assegnato è l'*entity_type* del modello contenuto in *t_entity_templates.*

Quando il parser XSLT, nell'analisi del documento incontra *loop*, crea sull'interfaccia web un campo ove listarne il contenuto.Ovviamente se l'XML analizzato è un modello, tale campo sarà vuoto, per aggiungere elementi basterà allora selezionare l'apposito pulsante.Se ad esempio se ne volesse inserire uno nel loop tranports, il sistema effettua una query ove richiede il modello identificato dal campo *entity_type=transport_stream.*Ciò permette all'XSLT di fare il parsing dell'XML ad esso associato e di mostrare via web i relativi campi editabili.

E' a questo punto che entra in gioco la tabella *t_ent_ent,* la cui struttura è mostrata in figura 5.6.

	oid	entity_id int4	loop_id int4	target_id int4
1	164244	250	1	252
2	164245	243	1	261
3	164246	251	1	262
4	164247	251	1	265
5	164248	251	1	266
6	164249	270	1	271
7	164250	270	1	273
8	164251	251	1	274
9	164252	150	1	275
10	164253	277	1	278
11	164254	287	1	288
12	164255	278	1	290
13	164256	300	1	307
14	164257	300	1	308
15	164258	300	1	310

figura5.6 (tabella t_ent_ent)

La creazione dell'entità *tranport_stream* segue esattamente lo stesso processo di istanziamento descritto in precedenza, e cioè l'inserimento nella tabella *t_entities* della sua riga identificata univocamente dal campo *entity_id.*Oltre a questa riga però, ne viene inserita una seconda in *t_ent_ent.*
Come mostrato in figura questa tabella contiene tre colonne: *entity_id*, *loop_id*, *target_id.*
La riga in questione quindi avrà nella prima colonna lo stesso *entity_id* istanziato in *t_*entities.Visto poi che l'elemento *transport_stream* appartiene al secondo loop la colonna *loop_id* ha valore 2.Per finire *target_id* indica l'*entity_id* dell'entità padre a cui associare *transport_stream.*
Il procedimento è ricorsivo in caso siano presenti loop di loop.

- **Tabella *t_lookup:***

Durante l'inserimento dei dati nelle sezioni PSI/SI, accade spesso di dover editare campi i cui valori non possono essere liberamente scelti dall'utente, ma che appartengono ad un set di valori prefissato dallo standard.
Per poter gestire questa casistica si è aggiunto ai tag field dell'XML l'attributo *lookup* e si è listato l'elenco dei valori da cui sciegliere in formato XML.
In figura 5.7 è riportata la tabella *t_lookup* con evidenziato il contenuto della colonna *xml* di *t_content*.
Tale XML lista i valori a cui fanno riferimento i campi *content_nibble_level_1/2* del *content_descriptor*.
Se durante una sessione di data entry l'utente decide ad esempio di aggiungere tale descrittore, l'XSLT effettua il parsing del relativo modello XML e scorrendo i vari tag giunge a quello relativo ai *content_nibble_level_1/2.*

```
<field name="content_nibble" size="8" label="Content Nibble" lookup="t_content"/>
```

Fra i suoi attributi si trova anche quello di *lookup*.Il nome ad esso associato viene utilizzato per fare una query al DB ove si richiede l*'xml* dell'entità *t_type=t_content* contenuta nella tabella *t_lookup*.A questo punto l'XSLT fa il parsing di tale XML per creare sull'interfaccia web una drop down list da cui selzionare il valore appropriato.

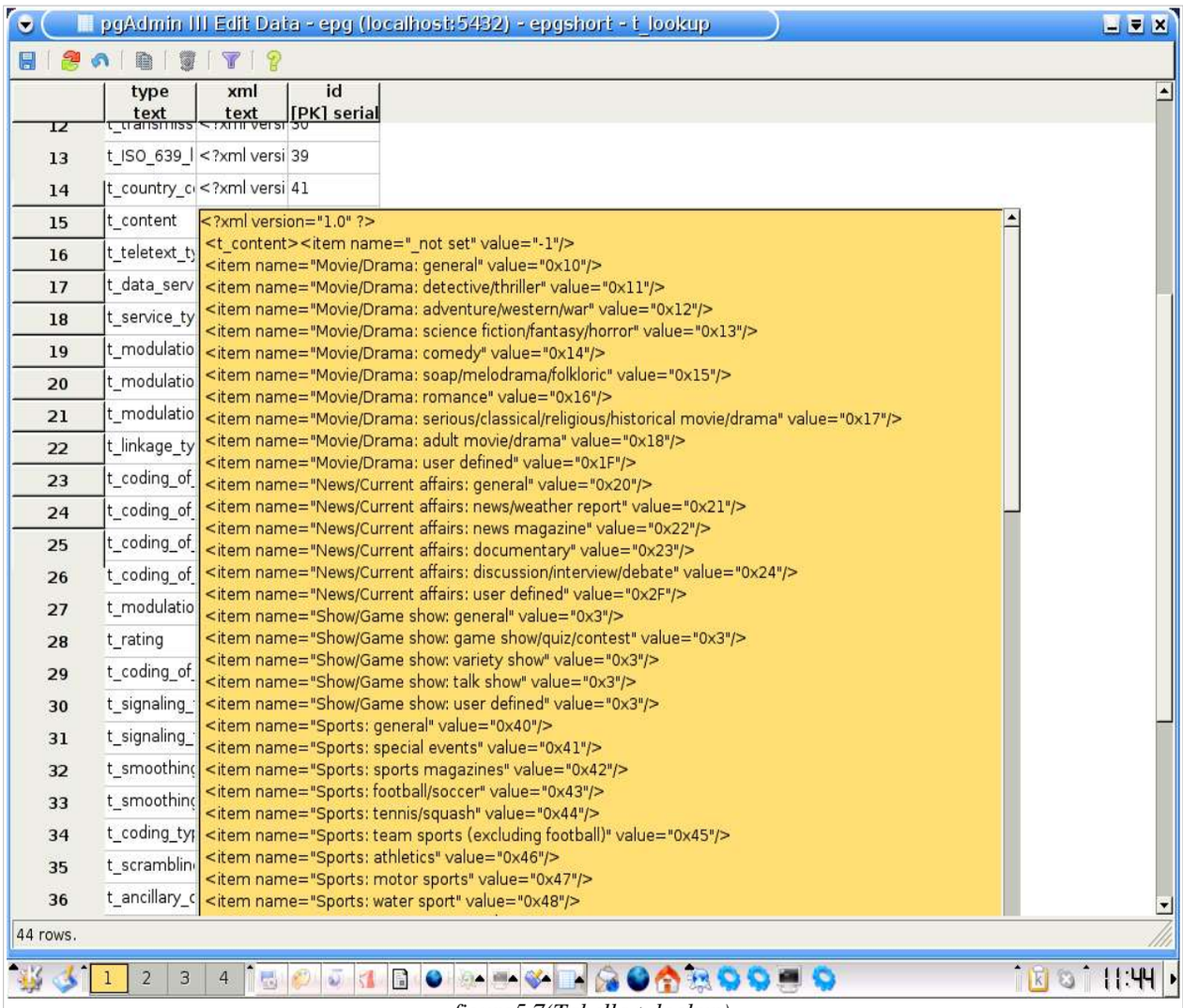

figura5.7(Tabella t_lookup)

5.3 Modulo EPG Server

Come già accennato ad inizio capitolo il modulo EPG Server fornisce dei metodi che permettono di interrogare e gestire il DB remoto.

L'esecuzione di ognuno di essi è legata alla specifica azione intrapresa dall'utente via interfaccia web.Se ad esempio si decide di visualizzare il contenuto di una determinata entità, verrà richiamato il metodo *getEntity,* il quale non fa altro che eseguire una query in linguaggio SQL al DB.

Prima di presentare più nel dettaglio un elenco di queste funzioni, si analizza il metodo di interfacciamento utilizzato per consentire la comunicazione del server con gli altri moduli client del sistema.

5.3.1 SOAP

Una delle prerogative del progetto MBI EPG è stata quella di creare un sistema distribuito e cioè un sitema i cui moduli potessero essere implementati su più macchine remote.Nella simulazione che sarà presentata in seguito si vedrà infatti che database, EPG Server e Web Server sono collocati su tre distinti computer.Il protocollo scelto per permettere la comunicazione fra EPG Server e relativi Client è il Simple Object Access Protocol (SOAP).

Nel contesto del progetto si è utilizzato SOAP come protocollo per poter effettuare Remote Procedure Call (RPC) su HTTP.Questa è però solo una delle sue possibili implementazioni.Infatti SOAP è da intendersi più genericamente come un leggero protocollo di comunicazione che può essere usato per mandare messaggi, codificati in XML, fra più nodi distribuiti su una rete.

Il protocollo di trasporto non è predefinito, si può infatti usare un qualunque protocollo in grado di trasportare informazioni testuali (infatti XML non è altro che testo).In realtà si tende a privilegiare HTTP per la sua ampia distribuzione e perchè attraverso di lui è possibile utilizzare facilmente i meccanismi di richiesta/risposta tipici di SOAP.

Il messaggio SOAP (vedi figura 5.8) è composto da:

- un elemento radice obbligatorio denominato *envelope, usato* per esprimere la struttura completa del messaggio SOAP, al suo interno trovano spazio tutti gli altri elementi sintattici del messaggio.Tutto quello che è racchiuso nell'Envelope fa parte del messaggio SOAP, ne è escluso tutto ciò che è al di fuori.

- un elemento header opzionale, progettato per facilitare ed individuare le elaborazioni richieste ai processori SOAP intermedi incontrati nel cammino del messaggio dal mittente al destinatario finale.
- Un elemento body obbligatorio, indirizzato al destinatario, che contiene le informazioni vere e proprie del messaggio.

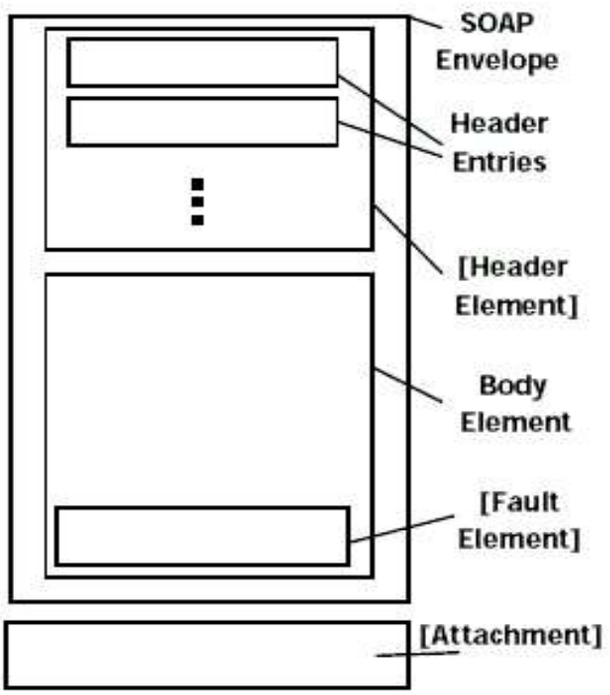

figura5.8 (messaggio SOAP)

Come detto in precedenza il sistema MBI EPG fa uso di SOAP in un contesto di Remote Procedure Call.In generale per invocare una RPC sono necessarie le seguenti informazioni:

- l'URI del nodo SOAP obbiettivo;
- il nome del metodo o della procedura;
- una firma opzionale del metodo o della procedura;
- i relativi parametri;
- uno o più header opzionali per informazioni supplementari;

La gestione di RPC attraverso messaggi SOAP prevede i seguenti componenti

nella sezione Body:

- **Call**: la chiamata al metodo remoto ed i suoi parametri.
- **Response**: il risultato dell'elaborazione del metodo remoto (con eventuali parametri di ritorno).

L'invocazione RPC *call* è modellata come una struttura nel body del SOAP, contenente un elemento figlio per ogni parametro di in/out.La struttura ha lo stesso nome e lo stesso tipo del metodo invocato.
Ogni parametro di in/out è visto come un elemento figlio della struttura, dove il nome rappresenta il nome del parametro e il tipo rappresenta il tipo di parametro.Questi devono apparire nello stesso ordine in cui sono definiti nel metodo.
A titolo di esempio si riporta il messaggio SOAP relativo alla chiamata del già citato metodo *getEntity.*

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ns="urn:epgserver">
 <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ns:getEntity>
   <anEntityId></anEntityId>
  </ns:getEntity>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

L'elemento *getEntity* indica il metodo da invocare, dove l'elemento figlio *anEntityId* è il relativo parametro di ingresso.
La risposta RPC è modellata nel Body SOAP allo stesso modo della chiamata, cioè come una struttura contenente un elemento per ogni parametro di in/out.L'ordine di ritorno deve rispettare quello specificato dal metodo, preceduto dall'eventuale valore di ritorno.La struttura di risposta ha lo stesso nome del metodo invocato nella *request* con l'aggiunta del suffisso *Response*.L'eventuale

valore di ritorno del metodo, in genere, ha nome “result”.Non si hanno parametri di ritorno se il metodo invocato è void.Di seguito il messaggio SOAP di ritorno alla chiamata *getEntity.*

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:ns="urn:epgserver">
 <SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <ns:getEntityResponse>
   <aResult></aResult>
  </ns:getEntityResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

L'elemento *getEntityResponse* contiene la risposta del metodo invocato in precedenza e l'elemento figlio *aResult* il relativo parametro di uscita.

5.3.2 Metodi implementati

I metodi implementati dall'EPG Server, come già accennato ad inizio paragrafo, sono finalizzati all'esecuzione di query verso il DB Postgresql.Riprendendo in esame il metodo *getEntity*, si riporta parte del codice C++ ad esso associato.

```
StrTable EpgServer::getEntity(int anEntityId) {
   pqxx::result myResult;
   std::cout << "called getEntity( " << anEntityId << " )"<< std::endl;
   std::string myQuery = std::string("select entity_id,entity_xml from t_entities where entity_id =")
                  + intToString(anEntityId);
   myResult = doQuery(myQuery);
   return result2StrTable(myResult);
```

Pqxx è l'API C++ ufficiale di Postgresql.Questa libreria viene usata, ed è questo il caso, quando si scrive software C++ che ha necessità di accedere a tale DB.Senza entrare troppo nel dettaglio del codice, risulta subito evidente la riga (evidenziata in rosso), contenente la query in questione:

select entity_id,entity_xml from t_entities where entity_id = anEntityId;

GetEntity richiede come parametro di ingresso l'*entityId* (identificato nel codice dall'intero *anEntityId)* dell'entità che si vuole visualizzare.Ricordando quanto illustrato nel paragrafo relativo all'organizzazione del DB, risulta chiaro che la tabella da interpellare è proprio la *t_entities,* in cui la Primary Key è data dalla colonna *entity_id.*La query richiede quindi l'*entity_xml* (cioè il documento XML che descrive l'oggetto selezionato), dell'entità contenuta nella tabella *t_entities* e identificata univacamente dall'*entity_id* di valore *anEntityId.*
Mettendo ora insieme tutto ciò con quanto detto su SOAP, dovrebbe risultare più chiaro il processo di scambio dati fra client e server.La sottostante figura 5.9 ne riassume i passi principali.

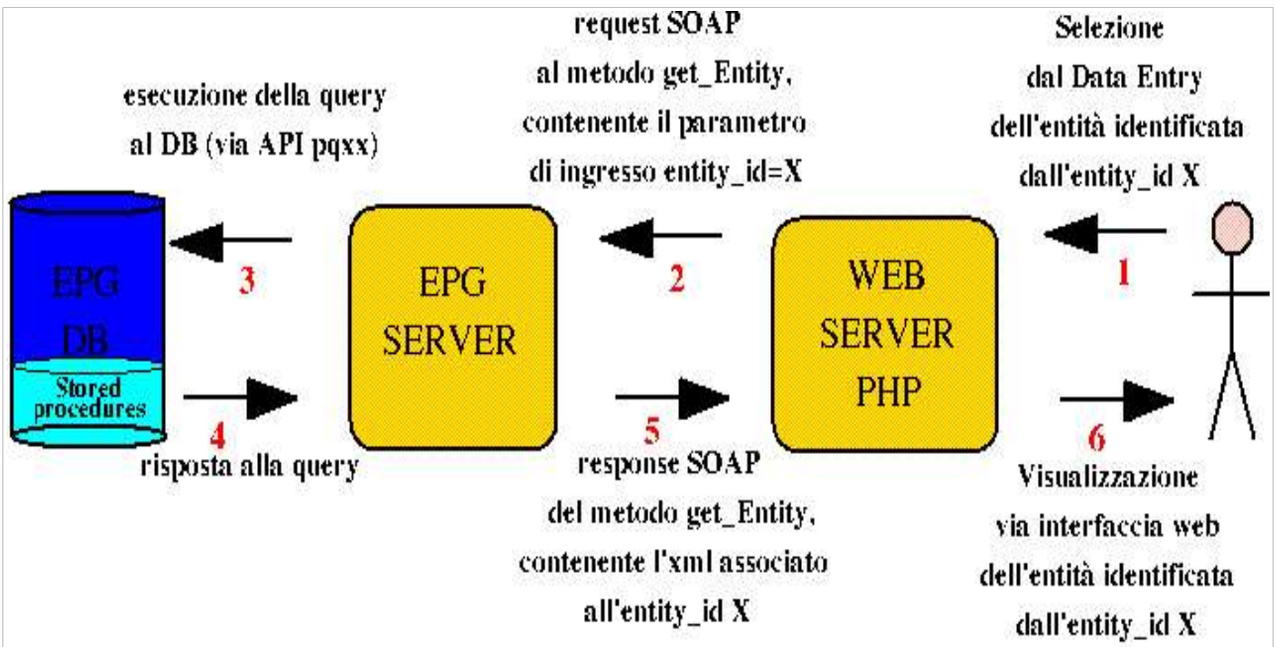

figura5.9 (metodo get_Entity:RPC SOAP)

Gli altri metodi più significativi, utilizzati nel sistema MBI EPG sono i seguenti:

- ***login***:

  il metodo *login* riceve in ingresso i due paramteri necessari all'autenticazione dell'utente, cioè le stringhe contenenti UserName e Password.La procedura prevede l'esecuzione di una query alla tabella *t_user,* ove si richiede la selezione della riga identificata dai sopracitati parametri di ingresso.Se la query va a buon fine, cioè se viene trovato fra la lista degli utenti registrati, quello avente username e password inseriti, l'autenticazione ha esito positivo e la sessione può iniziare.

- ***getEntityTye***:

  la query associata a g*etEntityType* richiede al DB l'*entity_id* e l'*entity_xml* di tutte gli oggetti di un determinato tipo (indicato dalla stringa associata a *EntityType)* contenuti nella tabella *t_entities*.Questa funzione viene chiamata in causa quando l'utente seleziona un determinato tipo di entità da editare o creare, ad esempio la NIT.La prima cosa fatta dal sistema è infatti quella di mostrargli tutte le NIT che già esistono nel DB, sull'interfaccia sarà poi visualizzato anche il pulsante da selezionare per crearne di nuove.

- ***setEntity***:

  setEntity è il metodo chiamato dal sistema nel momento in cui si seleziona il sopracitato pulsante per creare una nuova entità.E' interessante notare che la query in questione è stata direttamente memorizzata come stored procedure sul DBMS.Quest'ultima non è altro che una funzione scritta in linguaggio SQL.Alla fine il risultato della sua esecuzione, sempre nel caso della creazione di una NIT, è l'istanziamento di una riga nella tabella *t_entities* con *entity_type=nit* e *entity_xml* il modello xml della NIT.

- ***relateEntities***:

  relateEntities è l'importante metodo che permette di registrare le relazioni che intercorrono fra i vari loop e le entità padri.Quando si decide di editare il loop di una sezione, viene richiamato dal DB il modello XML del body associato a tale loop e poi lo si istanzia come una nuova entità nella *t_entities*.A questo punto il sistema provvede a richiamare automaticamente la *relatedEntities*, la quale riceve in ingresso l'*entity_id* dell'entità padre, l'*entity_id* dell'entità figlia (il body del loop appena inserito in *t_entities*) e il *loop_id.*Quest'ultimo parametro indica, nel caso l'entità padre abbia più loop, a quale di questi si faccia riferimento.Il risultato è l'inserimento nella tabella *t_ent_ent* della riga contenente le informazioni base per poter ricollegare l'entità figlia a quella padre.

- ***getLookupTable***:

  getLookupTable è il metodo chiamato dal sistema nel caso in cui l'utente stia procedendo alla creazione di una nuova entità o all'editazione di una già esistente e fra i relativi campi da riempire ne risultino alcuni i cui valori fanno riferimento ad un set prefissato dallo standard.La scelta viene fatta mediante consultazione di una drop down list, la formazione della quale passa attraverso una funzione che ha il compito di trasferirgli i valori da listare dal DB.Tale funzione è proprio la *getLookupTable.*
  La query ad essa associata è molto semplice, si seleziona cioè dalla tabella *t_lookup* il campo *xml* dell'oggetto identificato dal *type=LookupTableType* (dove *LookupTableType* è la stringa passata come parametro di ingresso).

5.3.3 Shell di debugging

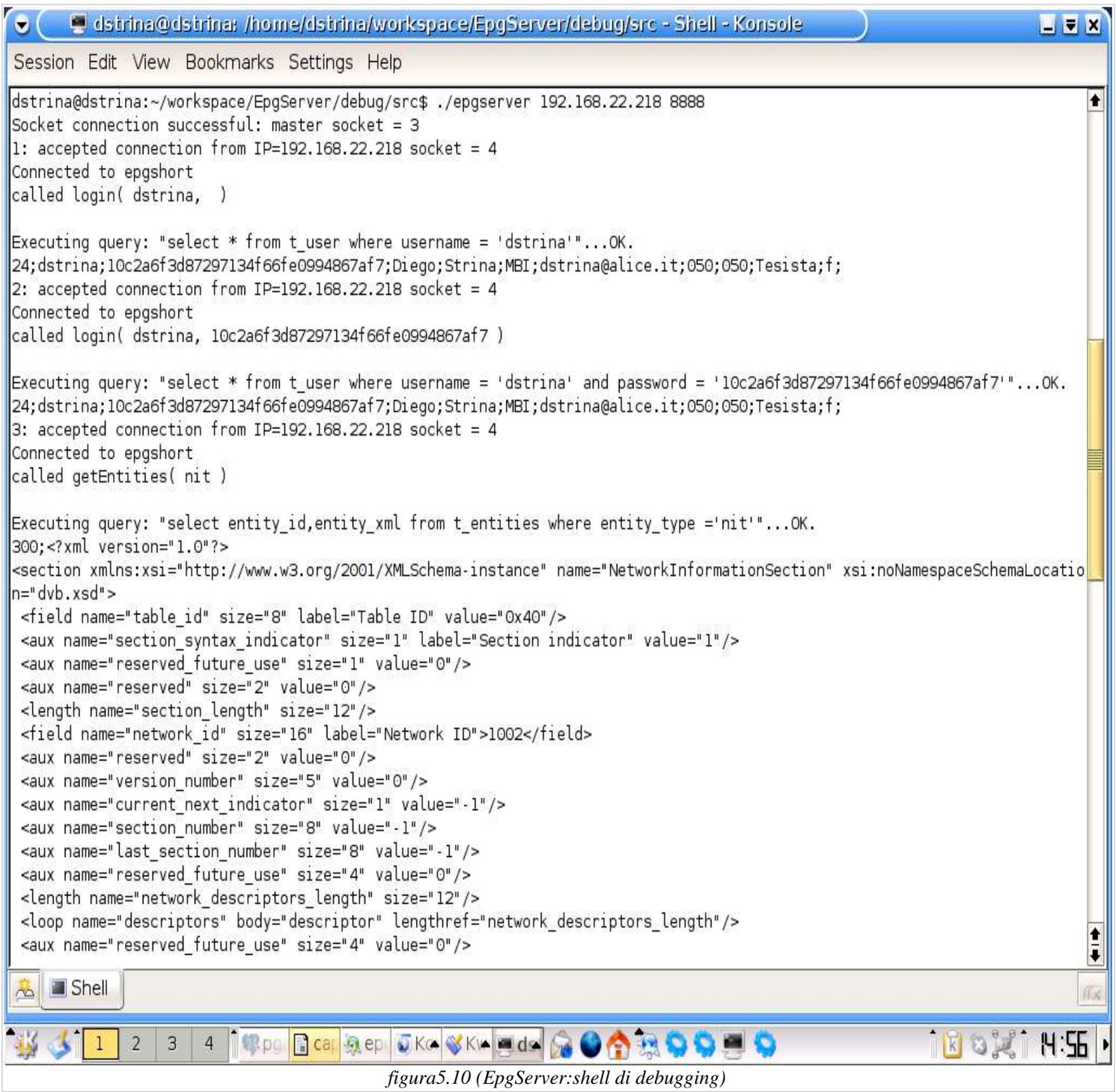

figura5.10 (EpgServer:shell di debugging)

Si osserva che i codici delle funzioni presentate in questo paragrafo (ma ciò vale anche per tutte le altre), prevedono la stampa sullo stream di uscita standard cout (vedi righe evidenziate in verde) di informazioni di debugging, utilizzate per monitorare l'esecuzione delle query, eventuali situazioni di errore e l'effettivo passaggio dei dati fra client e server.La figura 5.10 mostra la shell dell'epgserver.Leggendo quanto su di essa stampato si possono ripercorrere le operazioni effettuate.

- Per attivare la connessione con il server sono stati inseriti i seguenti parametri d'ingresso:

  -Indirizzo IP del computer sul quale risiede il server (nell'esempio in figura l'indirizzo è il 192.168.22.218)
  -Porta da aprire per la comunicazione con il server (in questo caso la 8888)

  Una volta passati i corretti parametri viene visualizzato il messaggio indicante l'avvenuta connessione (cioè l'apertura del socket sul quale il server viene messo in ascolto):

  Socket connection successful: master socket = 3

- Successivamente si è proceduto alla login dell'utente con UserName dstrina.

  1: accepted connection from IP=192.168.22.218 socket = 4
  Connected to epgshort
  called login(dstrina,)
  *Executing query: "select * from t_user where username = 'dstrina'"...OK.*
  24;dstrina;10c2a6f3d87297134f66fe0994867af7;Diego;Strina;MBI;dstrina@alice.it;050;050;Te sista;f;
  2: accepted connection from IP=192.168.22.218 socket = 4
  Connected to epgshort
  called login(dstrina, 10c2a6f3d87297134f66fe0994867af7)
  *Executing query: "select * from t_user where username = 'dstrina' and password = '10c2a6f3d87297134f66fe0994867af7'"...OK.*
  24;dstrina;10c2a6f3d87297134f66fe0994867af7;Diego;Strina;MBI;dstrina@alice.it;050;050;Te sista;f;

L'utente in questione sta eseguendo la login dallo stesso computer ove risiede il server, infatti l'indirizzo IP è lo stesso (192.168.22.218).Viene allora creato un secondo socket utilizzato per lo scambio dati con il server e instaurata la connessione con il DB postgresql (il DB di sistema è stato chiamato epgshort). Inseriti i dati di autenticazione via interfaccia web, si chiama il metodo login che, come già visto in questo paragrafo, esegue la query alla tabella *t_user*.

- L'autenticazione ha avuto esito positivo e l'utente può iniziare la sessione di data entry.La prima operazione effettuata è stata quella di listare tutte le NIT presenti sul DB.Dalle informazioni di debugging effettivamente si vede la chiamata al metodo *getEntities*, al quale è stato passato come parametro di ingresso la stringa *"nit".*I valori di ritorno di tale metodo sono l'*entity_Id* e *l'entity_xml* di tutti gli oggetti identificati da *entity_type=nit.*

called getEntities(nit)
Executing query: "select entity_id,entity_xml from t_entities where entity_type ='nit'"...OK.
300;<?xml version="1.0"?>
<section xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
name="NetworkInformationSection" xsi:noNamespaceSchemaLocation="dvb.xsd">
<field name="table_id" size="8" label="Table ID" value="0x40"/>
<aux name="section_syntax_indicator" size="1" label="Section indicator" value="1"/>
<aux name="reserved_future_use" size="1" value="0"/>
<aux name="reserved" size="2" value="0"/>
<length name="section_length" size="12"/>
<field name="network_id" size="16" label="Network ID">1002</field>
<aux name="reserved" size="2" value="0"/>
<aux name="version_number" size="5" value="0"/>
<aux name="current_next_indicator" size="1" value="-1"/>
<aux name="section_number" size="8" value="-1"/>
<aux name="last_section_number" size="8" value="-1"/>
<aux name="reserved_future_use" size="4" value="0"/>
...
...
...

5.4 Client di inserimento dati (GUI)

In questo paragrafo si fornisce una descrizione della GUI (Graphic User Interface) utilizzata per implementare il client di inserimento dati.Vengono presentate le varie schermate, a partire da quella di login, che l'utente incontra durante una tipica sessione di Data Entry.In figura 5.11 si riporta la pagina di login attraverso la quale avviene l'autenticazione al sistema MBI EPG.

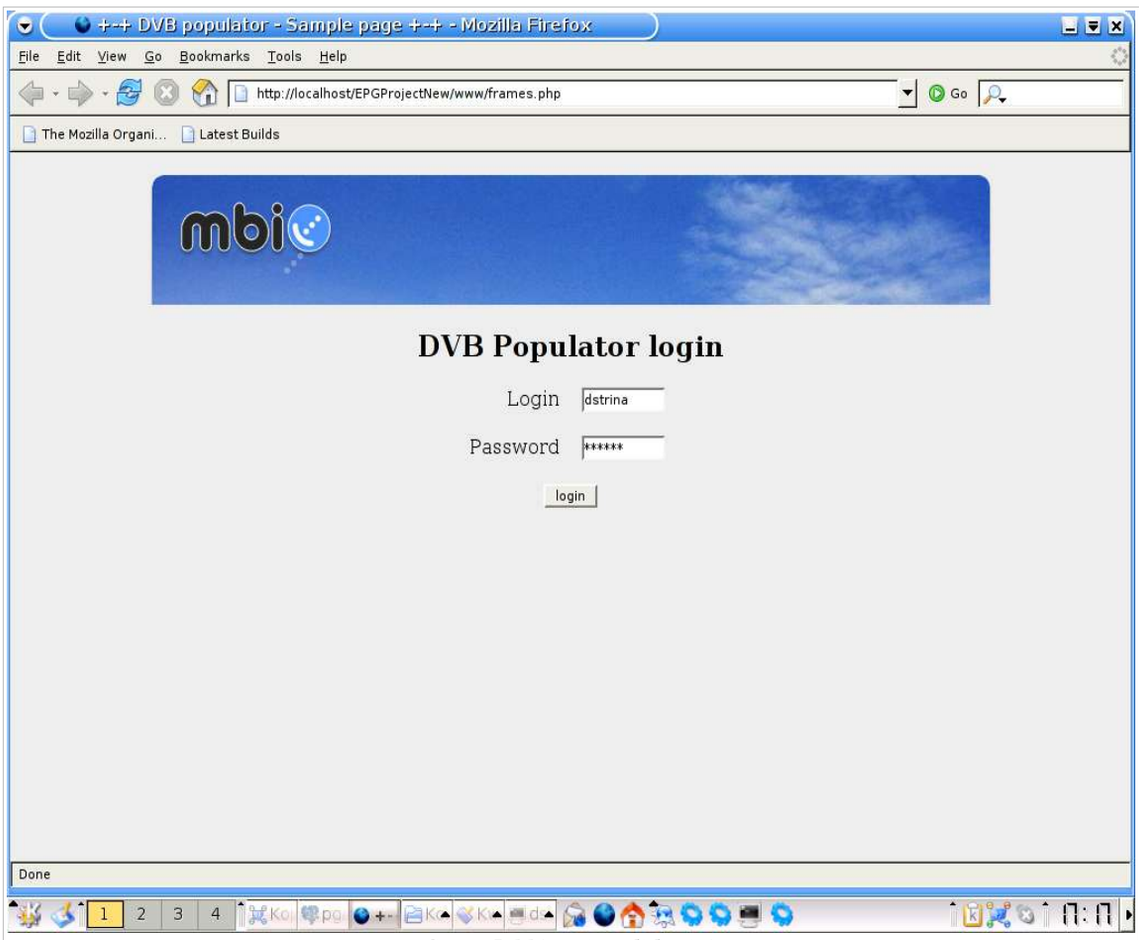

figura5.11(pagina di login)

Ad autenticazione avvenuta si accede alla vera e propria pagina di Data Entry (figura 5.12).Qui si trova l'elenco, diviso in due parti, delle varie entità trattate dal sistema.La prima parte comprende le sezioni vere e proprie, mentre la seconda, posizionata più in basso, le sotto entità collegate alle prime.Tali entità non sono altro che gli oggetti che vanno a comporre i vari loop delle sezioni.Ad esempio tutti gli oggetti evento aggiunti nel loop delle sezioni EIT, sono visualizzabili tramite selezione del pulsante *Events,* da *Descriptor* si accede invece ai

descrittori componenti i loop di descrittori etc.

figura5.12(pagina principale di Data Entry)

Selezionando ad esempio la NIT (vedi figura 5.13), sul lato destro dello schermo vengono presentate tutte le entità contenute nella tabella *t_entities* identificate dall'*entity_type nit.*E' prevista la presenza di più pulsanti che permettono di visualizzarne i contenuti, di editarli, di farne l'update e di cancellare l'intera sezione.

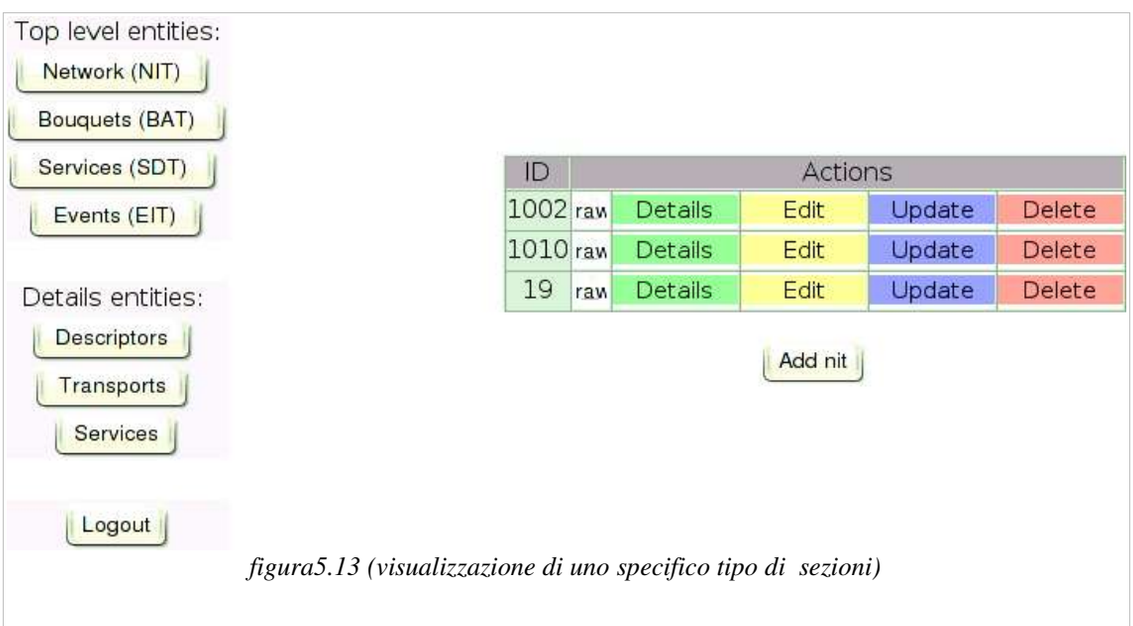

figura5.13 (visualizzazione di uno specifico tipo di sezioni)

Mediante il pulsante *Add nit* si può creare una nuova sezione (NIT).

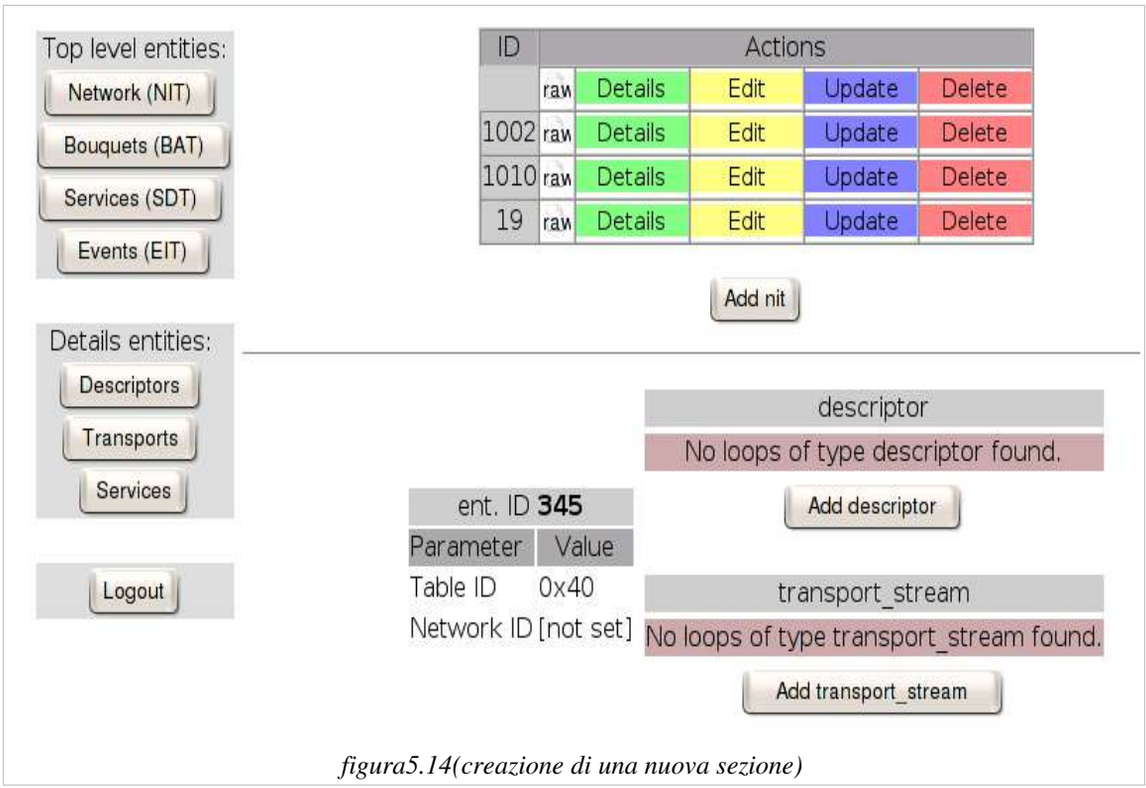

figura5.14(creazione di una nuova sezione)

Non tutti i dati sono direttamente editabili dall'utente, si vede infatti che nella NIT, escludendo i due loop, è possibile il riempimento solo del campo *Network ID*.Gli altri campi verranno inseriti automaticamente da un secondo programma.Nel caso ad esempio del dato *section_length*, tale programma procederà al calcolo della lunghezza (in byte) dell'intera sezione (partendo dai dati immediatamente seguenti *section_length*) e inserirà il risultato ottenuto in questo campo.
Sulla destra viene indicato il contenuto dei loop *descriptor* e *transport*.Visto che la sezione è stata appena creata sono ovviamente vuoti.Per aggiungere un elemento basta selezionare l'apposito pulsante.In figura 5.15 è riportata la schermata relativa al caso in cui si sia scelto di aggiungere un oggetto transport nel secondo loop della NIT.Ora i campi editabili sono solo il *transport_stream_id (tsid) e* l'*original_network_id (onid),* infatti il *transport_descriptor_length* viene riempito in un secondo momento dal sopracitato programma.Secondo specifiche il loop transport contiene un sotto loop di descrittori e infatti l'interfaccia ne riporta i dettagli sul lato destro dello schermo.Per aggiungere un nuovo descrittore basta selezionare il pulsante *Add descriptor.*

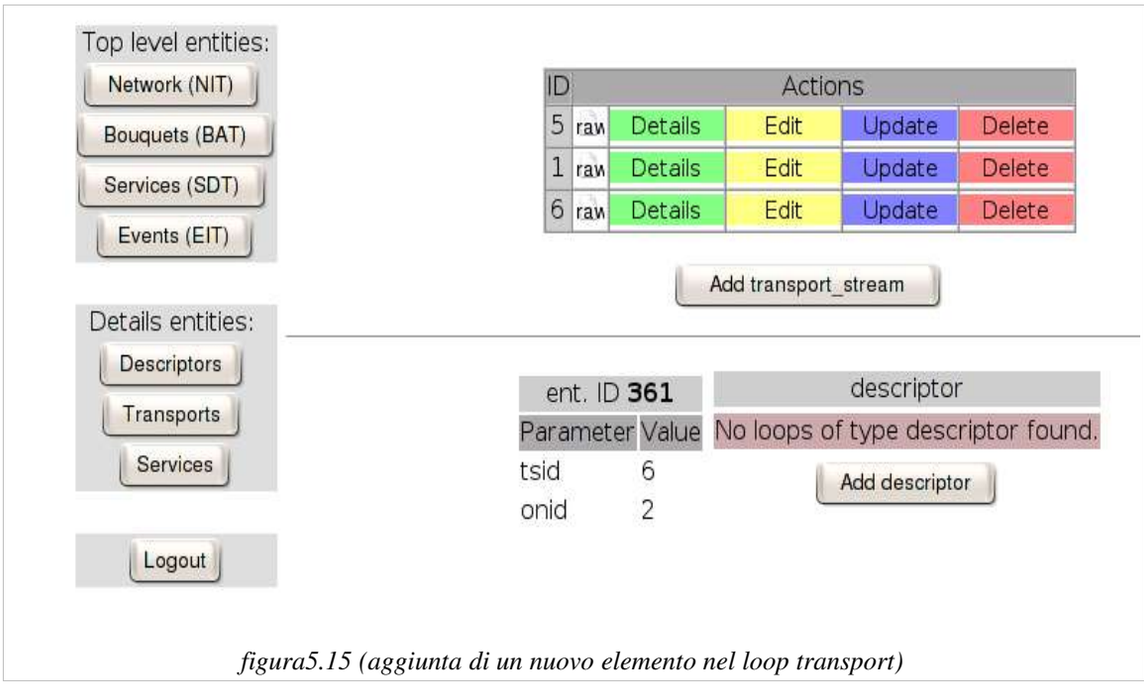

figura5.15 (aggiunta di un nuovo elemento nel loop transport)

A questo punto viene presentata la pagina contenente la drop down list di tutti i descrittori trattati dal sistema (figura 5.16).

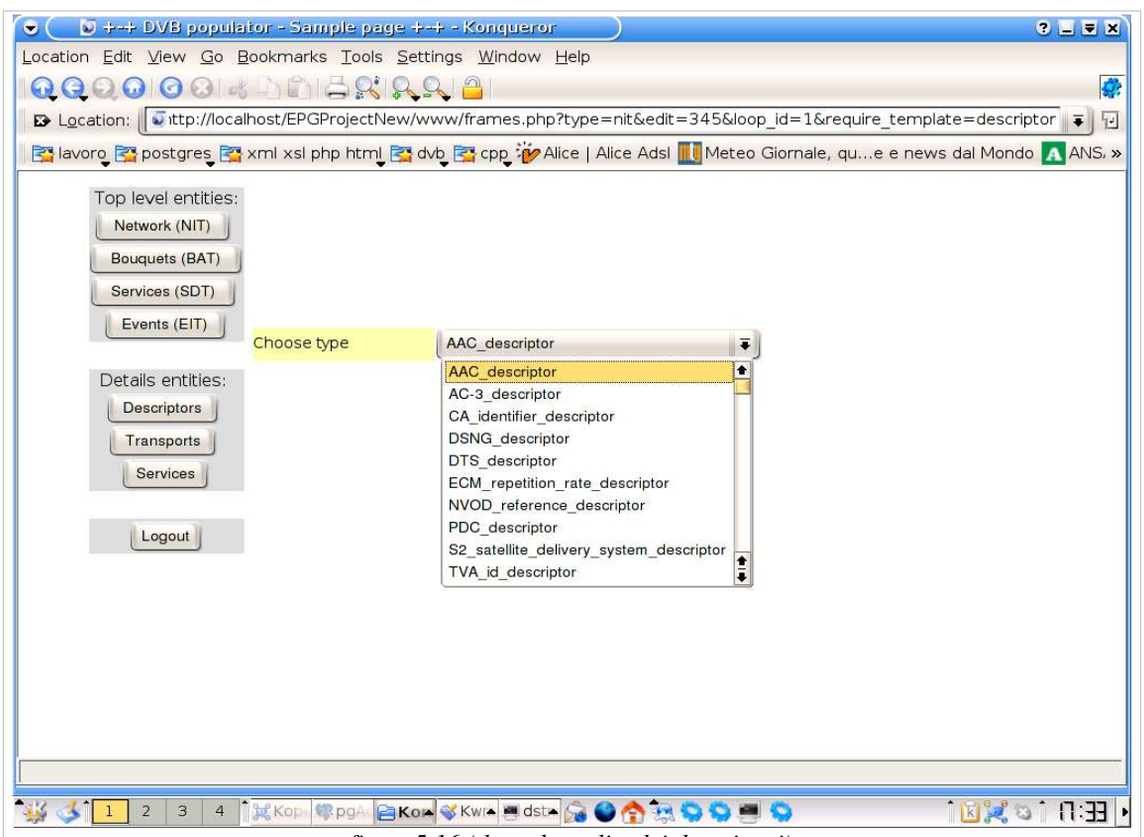

figura5.16 (drop down list dei descrittori)

Sceltone uno viene caricata la relativa pagina di editazione.La figura 5.17 riporta quella dello *short_event_descriptor*.

figura5.17 (pagina di editazione dei descrittori)

Per concludere si mostra la pagina della gui (figura 5.18) riservata agli utenti aventi diritti di amministratore.Le informazioni listate sono quelle relative alla tabella *t_user* del DB.L'amministratore può intervenire su questa tabella, aggiungendo o cancellando gli utenti che hanno accesso al sistema MBI EPG.

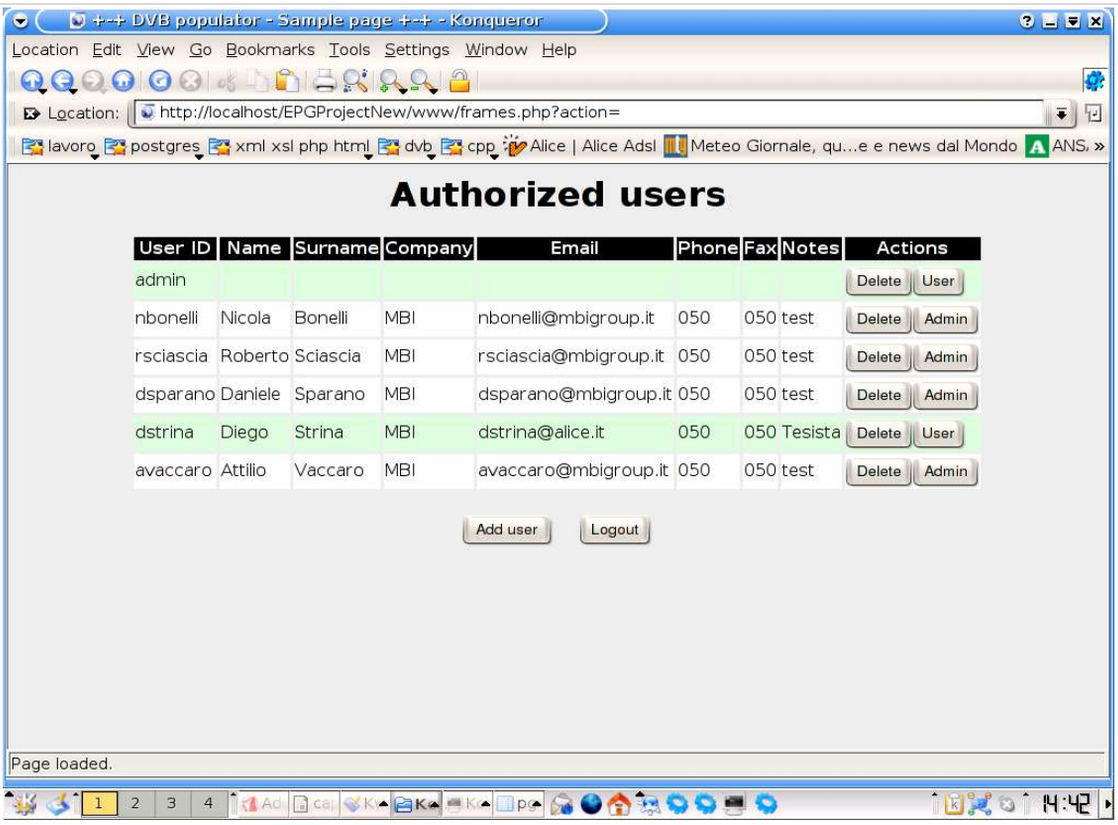

figura5.18 (pagina admin))

5.5 Simulazione del sistema MBI EPG su local-loop DVB-T

Come verifica finale del progetto MBI EPG si è effettuata una simulazione su un local loop DVB-T appositamente installato in azienda.La figura 5.19 mostra lo schema a blocchi dell'intero sistema.

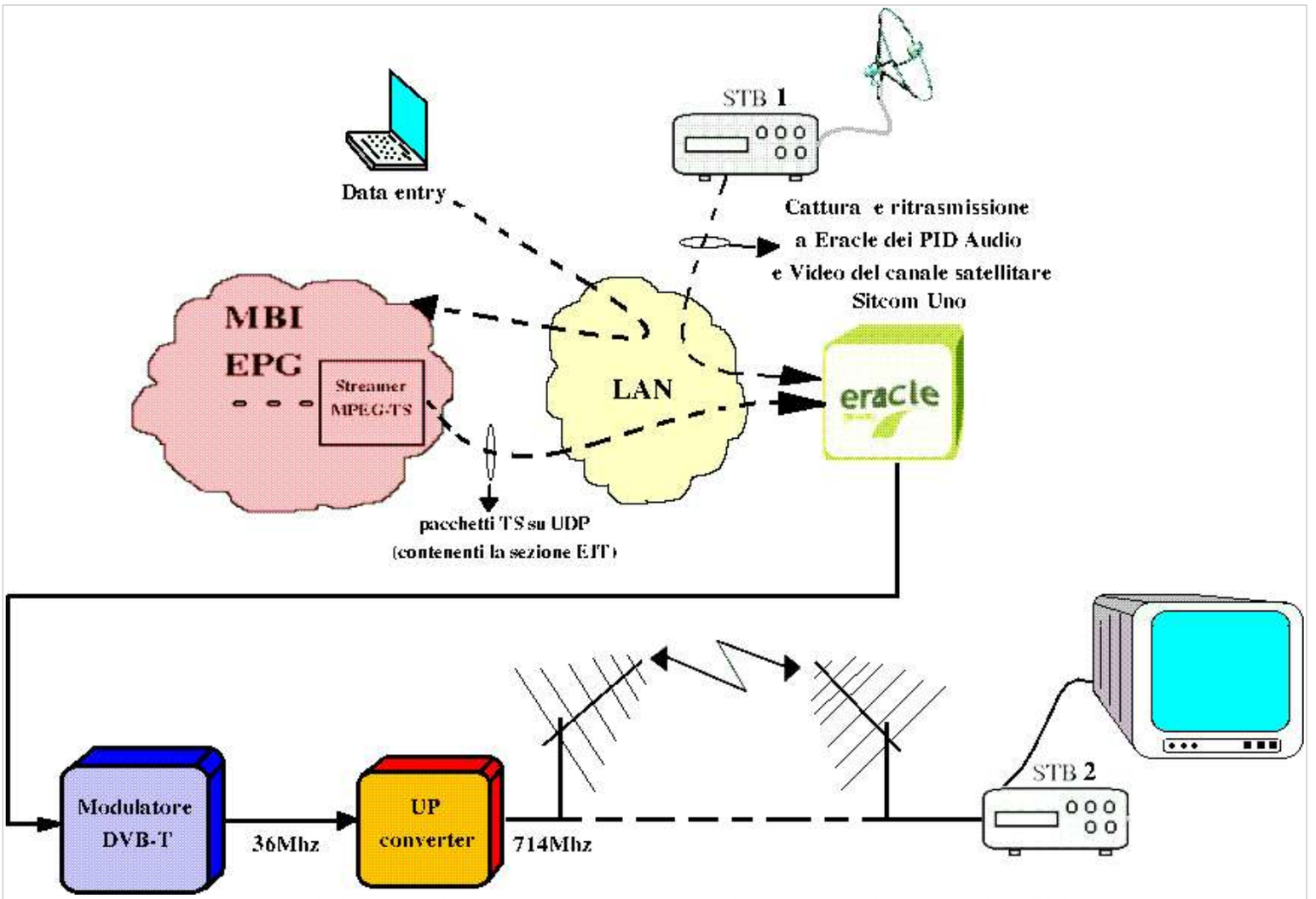

figura5.19 (Schema a blocchi del sistema utilizzato per la simulazione del progetto MBI EPG)

Nello schema, il sistema MBI EPG è stato condensato in un unico elemento.Per la sua struttura interna si faccia riferimento alla figura 5.1 di inizio capitolo.

L'unico blocco evidenziato è lo Streamer MPEG-TS, cioè quello responsabile della trasmissione ad Eracle dei file binari relativi alle tabelle PSI/SI.

Un ruolo centrale nell'intero schema è ricoperto da Eracle.Il suo compito è quello di costruire il Transport Stream di uscita, comprensivo delle principali tabelle PSI/SI (PAT, PMT, SDT...), utilizzate poi dal STB2 per poter visualizzare il

programma trasmesso e i contenuti informativi ad esso collegati.
I contenuti audio e video del canale DVB-T di test sono stati presi dal canale satellitare Sitcom Uno.A tal fine si è utilizzato il STB1 collegato ad una parabola puntata sul satellite Atlantic Bird 1 12.5°W.Su questo STB è stato installato il sistema operativo *Debian Linux*.Per lo streaming vero e prorio si è fatto uso del programma *Linux Dvbstream*.Questa applicazione è lo strumento base per interagire con la scheda di ricezione satellitare ed è in grado di inviare in LAN, tramite il protocollo rtp l'intero trasport stream di un transponder DVB o una sua parte.Nel caso specifico, come sarà mostrato in seguito, sono stati filtrati e ritrasmessi solo i PID contenenti l'Audio e il Video.
Il Transport stream costruito da Eracle viene inviato al modulatore DVB-T, dal quale esce ad una frequenza intermedia di 36Mhz.Successivamente tale segnale viene processato dall' UpConverter (il quale opera uno shift in frequenza a 714Mhz) e inviato all'antenna di trasmissione.Nelle prove effettuate in azienda si è cmq fatto a meno dell'antenna, infatti il cavo coassiale in uscita dall'UpConverter è stato direttamente collegato al STB2.
La simulazione prevede la creazione di un canale televisivo di test, denominato *DIEGO*, al quale si associerà, mediante il sistema MBI EPG una semplice sezione EIT.Si è scelto di generare una EIT *Present*, contenente le informazioni del programma attualmente trasmesso sul canale.In tale sezione si fa uso unicamente dello *Short Event Descriptor.*
Il primo passo è il settaggio su Eracle del programma di test.Attraverso la relativa interfaccia, riportata in figura 5.20, si possono definire le tabelle che il sistema riceverà dall'esterno.In questo caso l'unica tabella che viene inviata è la EIT, motivo per cui si seleziona solo lei.Ovviamente il PID ad essa destinato è quello previsto dallo standard, cioè il 18 (0x12).
Lo Streamer MPEG-TS, responsabile dell'invio della EIT, opera due pacchettizzazioni successive: la prima è quella relativa alla creazione dei pacchetti TS a partire dal formato binario dalla tabella EIT, la seconda è il loro incapsulamento in pacchetti di trasporto UDP.Si ricorda ciò perchè Eracle ha appunto bisogno di conoscere il protocollo di trasporto utilizzato dal flusso di dati esterni e il campo *Input Packet type* è destinato a contenere proprio tale specifica.Si nota infatti che è stato selezionato UDP.
In una seconda pagina, non riportata, si sono definite le regole di filtraggio IP da

applicare ai dati EIT in ingresso al sistema, in particolare si è data istruzione di accettare pacchetti provenienti da qualsiasi sorgente e destinati all'indirizzo multicast 224.0.1.3.

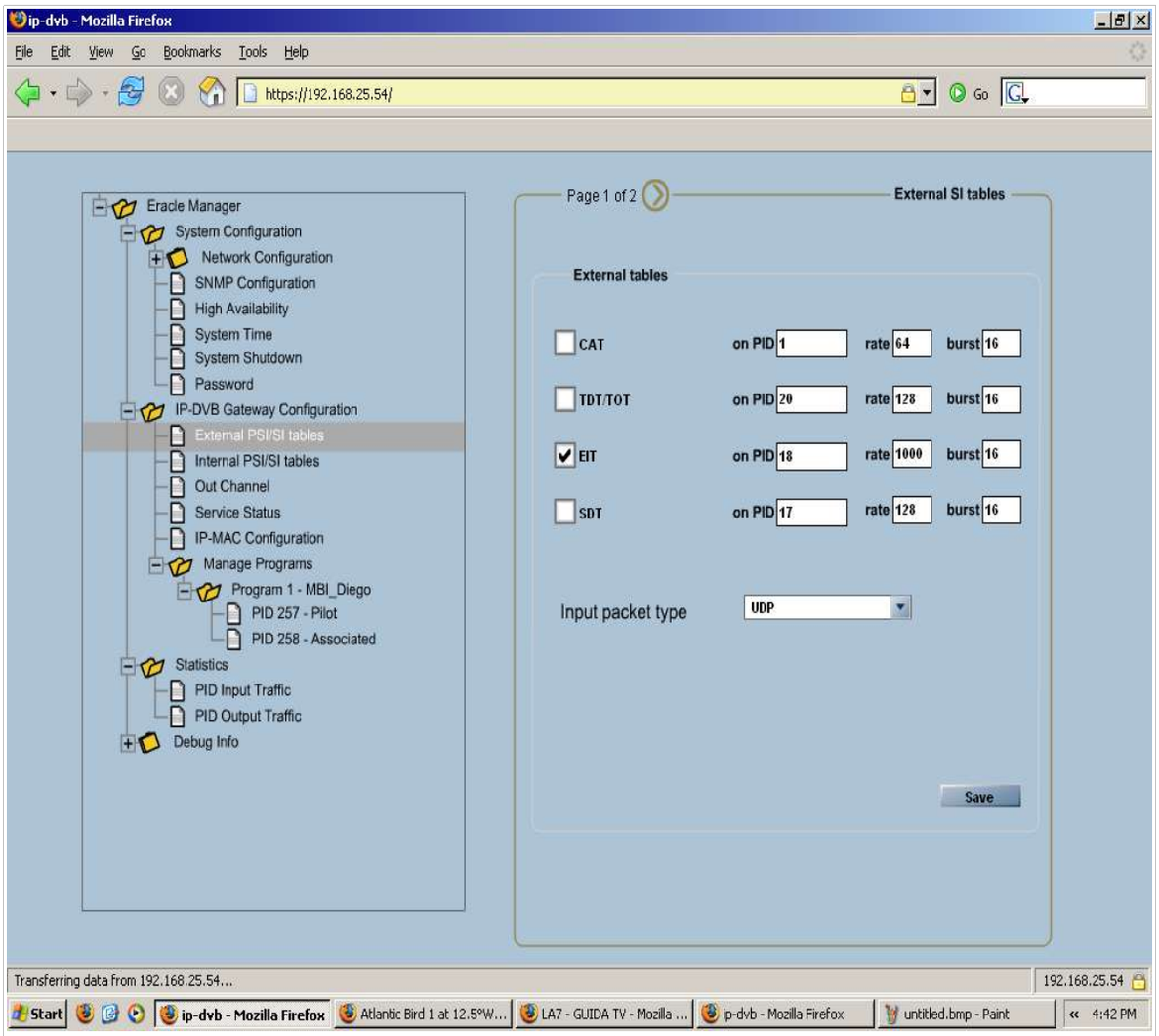

figura5.20 (Interfaccia Eracle:settaggio della EIT)

Passando alla pagina dedicata alle tabelle PSI/SI generate internamente al sistema (figura 5.22), si è proceduto alla definizione del Transport Stream ID e dell'Original Network ID.

Ovviamente questi dati potevano essere scelti a caso, ma si è voluto mantenere quelli del canale satellitare (Sitcom Uno) dal quale sono stati prelevati gli stream Audio e Video.Da una rapida consultazione del sito internet http://www.lyngsat.com (vedi figura 5.21) è stato possibile risalire ai valori originali che sono nell'ordine, 514 e 29.Nella simulazione si è comunque attivata

la trasmissione della sola PAT e SDT, escludendo la NIT.L'unico valore preso in considerazione dal sistema è quindi il Transport stream ID.

Atlantic Bird 1 © Lyngemark Satellite, last updated 2006-10-11 - http://www.lyngsat.com/ab1.html

Freq. Tp		Provider Name Channel Name			Video Encryption	SR-FEC SID-VPID		NID-TID Audio	Beam	Source Updated
12515 H tp F1		***TV Italia***			DVB	17455-3/4		29-514	Europe	T Viererbe 060607
		La 7	A	T F		1	257	258 I		
		MTV Italia	A	T F		2	513	514 I		
		La 7 Sport	A	F		4	2257	2258 I		
		La 7 Cartapiú A	A		Irdeto 2	6	290	291 I		
		La 7 Cartapiú B	A		Irdeto 2	7	293	294 I		
		La 7 Cartapiú C	A		Irdeto 2	8	296	297 I		
		La 7 Cartapiú D	A		Irdeto 2	9	299	300 I		
		La 7 Cartapiú E	A		Irdeto 2	10	302	303 I		
		Telemarket	A	F		12	284	285 I		
		Rete Capri	A	F		14	270	271 I		
		La 7 Cartapiú F	A		Irdeto 2	15	317	318 I		
		Sitcom Uno	A	N F		17	280	281 I		

figura5.21 (Dettagli canale Sitcom Uno)

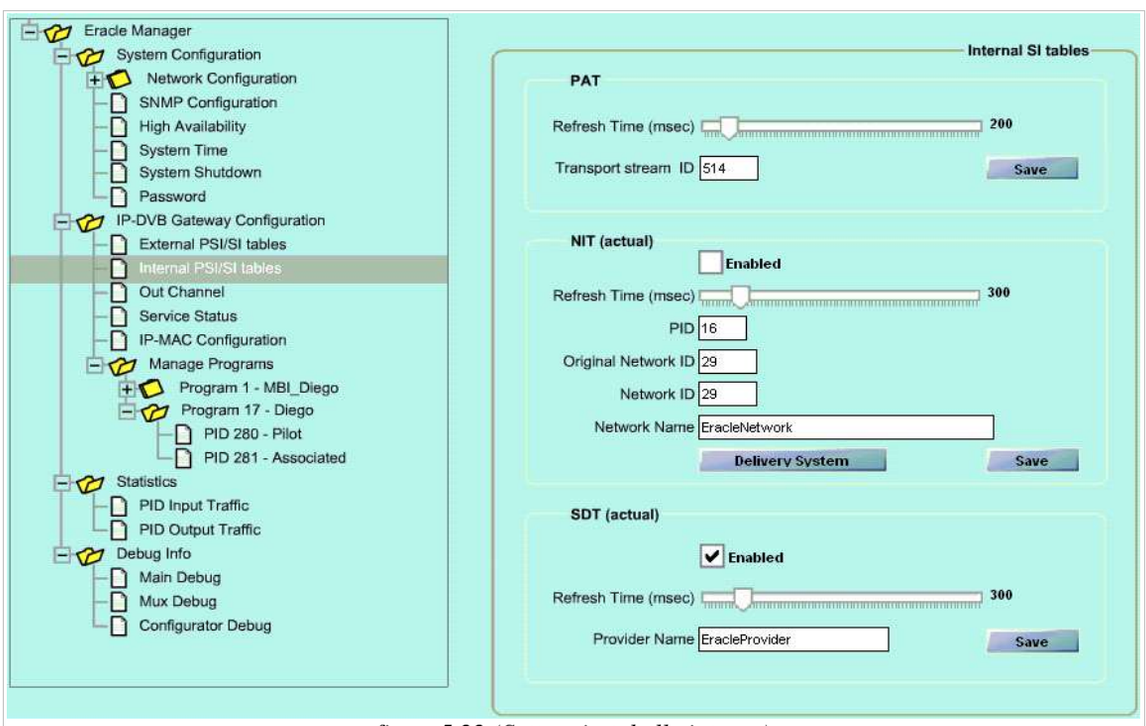

figura5.22 (Settaggio tabelle interne)

In figura 5.23 è riportata la pagina dalla quale si possono settare i principali parametri dei programmi contenuti nel TS.Il *program_name* del canale di test è

DIEGO, mentre il *program_number* 17 (anche in questo caso si è fatto riferimento ai valori originali di SitCom Uno).Come PID della PMT generata da Eracle si è scelto 400.Questo valore è quello che il sistema associa nella PAT al programma numero 17.Essendo poi previsto l'invio della sola EIT Present si seleziona il campo *EIT present following flag*.Il sistema, in risposta a tale operazione setterà ad uno il campo *EIT_present_following_flag* della SDT.

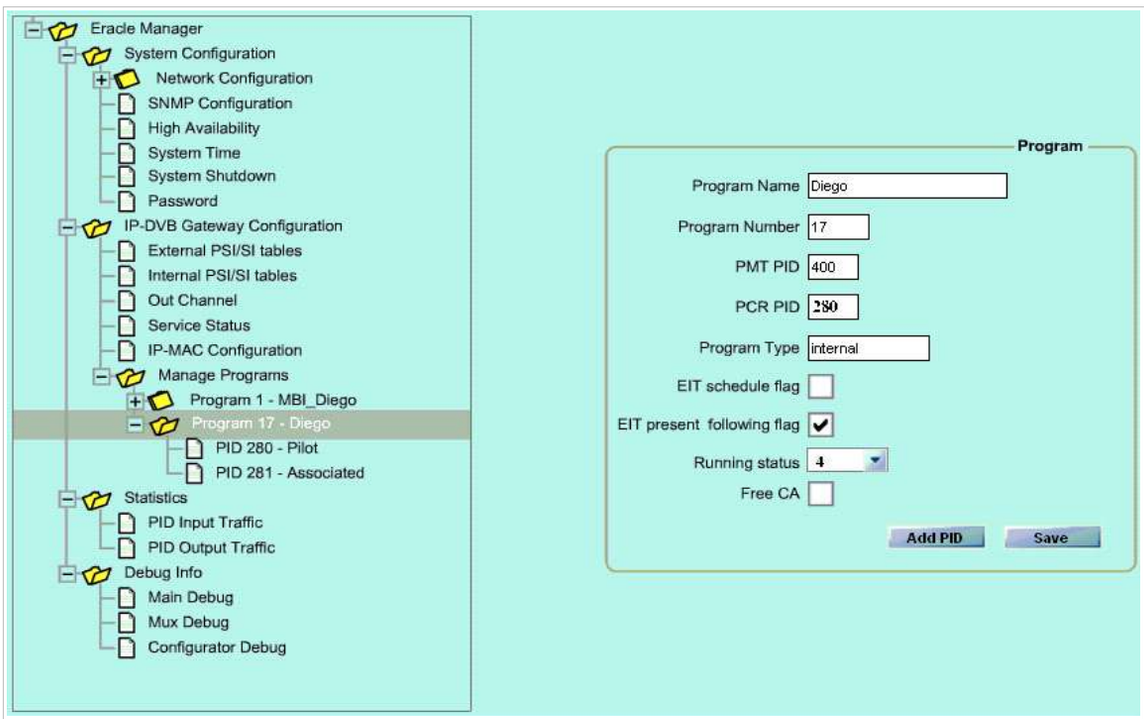

figura5.23 (Settaggio del programma)

Per completare il profilo del canale non resta che aggiungere le informazioni relative agli stream Video e Audio da associarvi (vedi figure 5.24).Facendo sempre riferimento alla figura 5.21, si nota che i valori dei PID di SitCom Uno sono 280 per il Video e 281 per l'Audio.E' importante che si inserisca l'esatto valore nel campo *Stream_type,* il quale identifica il tipo di contenuto dello stream (2 per il Video e 3 per l'Audio).Nel campo *Packet_Type* si è selezionato *RTP,* il protocollo di trasporto utilizzato dall'applicazione *Dvbstream* per trasmettere i PID 280 e 281.Come per la EIT,in una seconda pagina, non riportata, si sono definite le regole di filtraggio IP da applicare a tali pacchetti: si accettano pacchetti provenienti da qualsiasi sorgente e destinati all'indirizzo multicast 224.0.1.2.

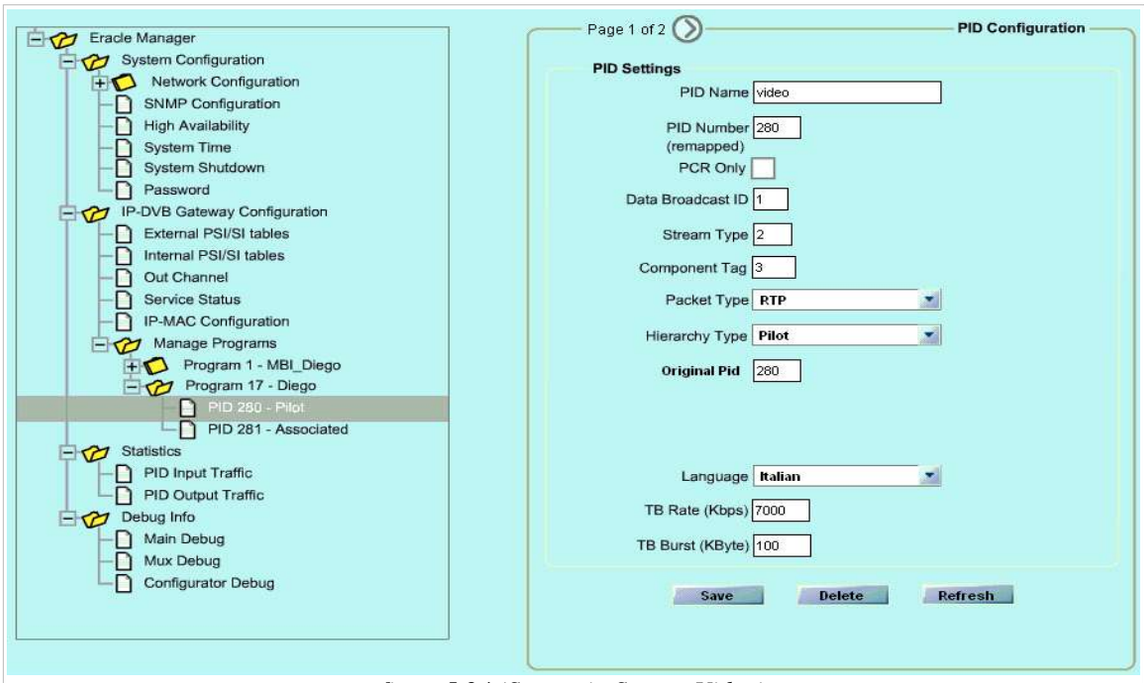

figura5.24 (Settaggio Stream Video)

Concluso il settaggio del canale di test *DIEGO,* viene avviato sul STB1 lo streaming Audio/Video verso Eracle attraverso la seguente linea di comando:

"dvbstream -f 12515 -p H -s 17455 -i 224.0.1.2 20"

I parametri passati a *Dvbstream* sono la frequenza del trasponder (12.515Ghz), la polarizzazione del segnale (orizzontale), il symbol rate (17455000 simboli al secondo) e l'indirizzo multicast (224.0.1.2).

A questo punto si passa alla generazione della sezione EIT.Superata la procedura di autenticazione si accede all'interfaccia di Data Entry del sistema MBI EPG, dalla quale, una volta entrati nella pagina relativa alle EIT, viene creata una nuova sezione.La figura 5.25 mostra in dettaglio la fase di editazione della nuova entità.Si può notare che i valori dei campi *Transport_stream_id*, *Service_id* e *Original_network_id* sono coerenti con quelli inseriti su Eracle.Inoltre avendo selezionato il pusante *Add eit Present/Following* il *Table_id* viene automaticamente messo a 0x4E.Visto poi che si è interessati alla generazione di una EIT Present si aggiunge il flag nel relativo campo.L'inserimento di questa informazione si tradurrà semplicemente nel settaggio a 0 del campo *section_number* della EIT.

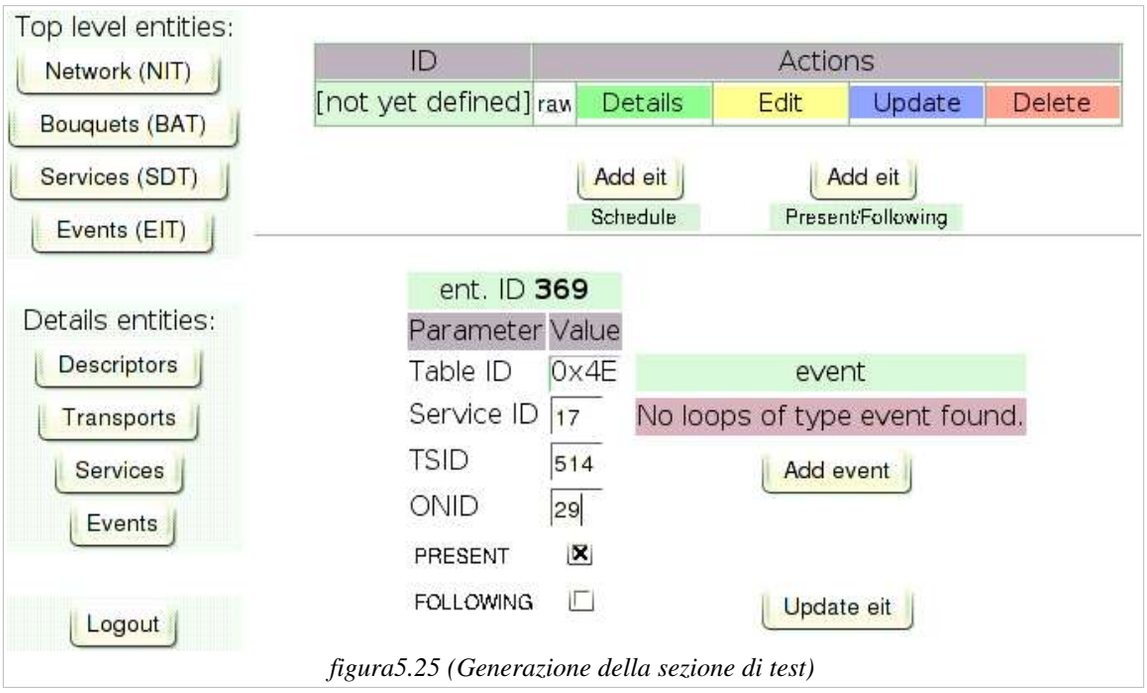

figura5.25 (Generazione della sezione di test)

In figura 5.26 si riporta la pagina in cui sono stati inseriti i dati dell'evento.Perchè questo venga visualizzato sul teleschermo come Present è necessario che l'ora di sistema, cioè quella del STB, cada nell'intervallo temporale che intercorre fra l'ora di inizio dell'evento stesso e quest'ultima più la durata dell'evento.

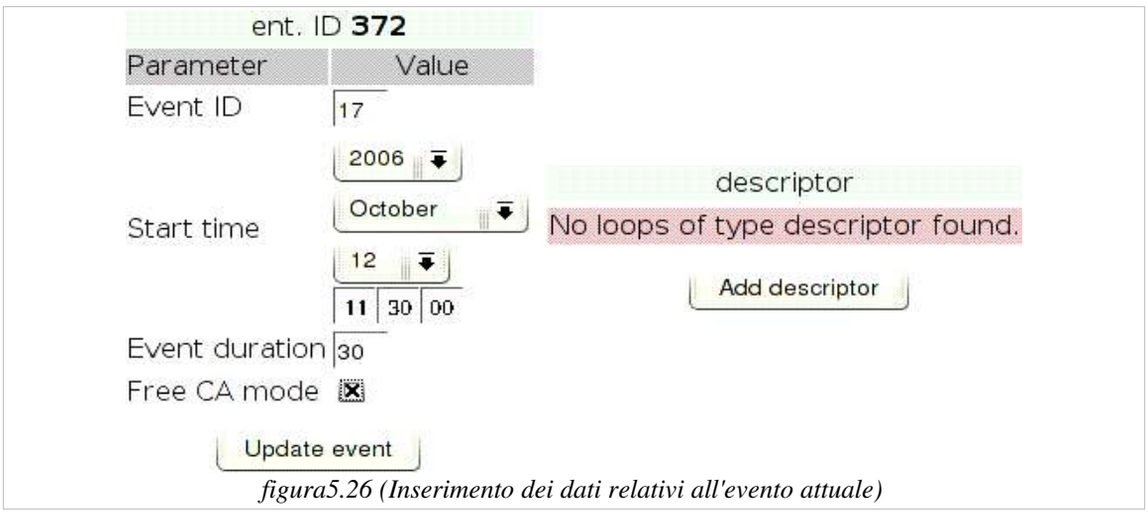

figura5.26 (Inserimento dei dati relativi all'evento attuale)

E' utile sottolineare che il STB utilizzato è provvisto appunto di un orologio e calendario interno, presi come riferimento per la visualizzazione delle informazioni di programmazione.Nel caso però che si fosse mandato oltre alla

EIT anche una TDT esterna, sarebbe stata quest'ultima a comunicare l'ora e la data esatte.Per finire non resta che aggiungere lo *Short Event* Descriptor.Qui viene definita la lingua (Italian), il nome (*Evento MBI*) e il testo assocciato all'evento *(Test end2end tesi Diego Strina).*

figura5.27 (Aggiunta del descrittore ShortEventDescriptor)

A questo punto sul DB sono presenti tutte le entità XML della EIT appena editata.Tali documenti vengono passati al Parser XML che fornisce in uscita il file binario della EIT di test, il quale viene successivamente processato dallo Streamer MPEG-TS.La seguente linea di comando è relativa a quest'ultimo passaggio:

"./dvb_streamer -f /home/dstrina/workspace/eit.dat -P 0x12 -d 224.0.1.3 -p 2000 -t 3 -i eth0 -w 100"

Vengono passati l'URL del file binario da processare, il PID da assegnare ai pacchetti TS generati a partire da tale file (visto che è uno stream EIT il valore è 0x12), l'indirizzo multicast di destinazione, la porta UDP, il valore del campo *time to live,* l'interfaccia di rete da utilizzare e l'intervallo di ripetizione in millisecondi.
La figura 5.28 riporta la pagina di Eracle da cui è possibile monitorare l'occupazione di banda dei vari stream in ingresso al sistema.Il grafico è relativo all'occupazione di banda della EIT, oscillante intorno ad un valore medio di 17 Kbit/sec.Da un rapido conto si può subito verificare l'esattezza di questa indicazione.La sezione creata ha una lunghezza di 80 byte, rientra quindi completamente in un unico pacchetto TS avente una dimensione fissa di 188 byte (ai rimanenti 104 byte del payload viene assegnato il valore 0xff).A questo pacchetto sono da aggiungere l'header UDP (8 byte) e IP (20 byte), portando

alla lunghezza totale di 216 byte.Considerato il rate di ripetizione di 100 millisecondi si ottiene una banda di:

216*8=1728 *lunghezza in bit*

1728/(100*10^{-3})=17,28 Kbit/sec *banda occupata dallo stream EIT*

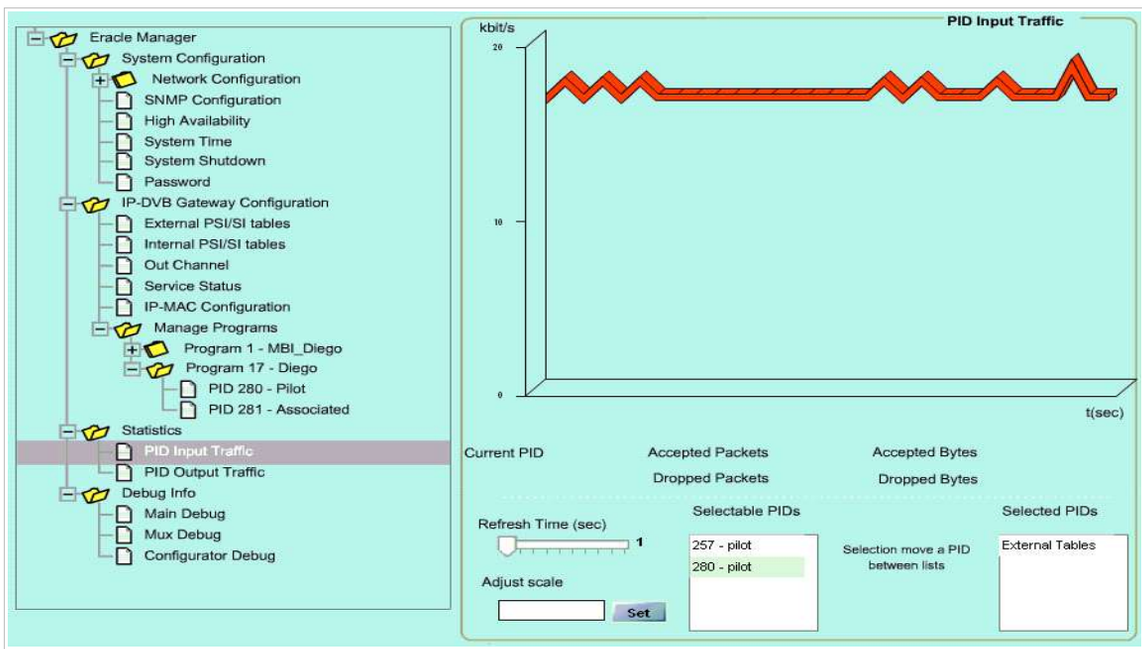

figura5.28 (Interfaccia Eracle:banda occupata dallo strem EIT)

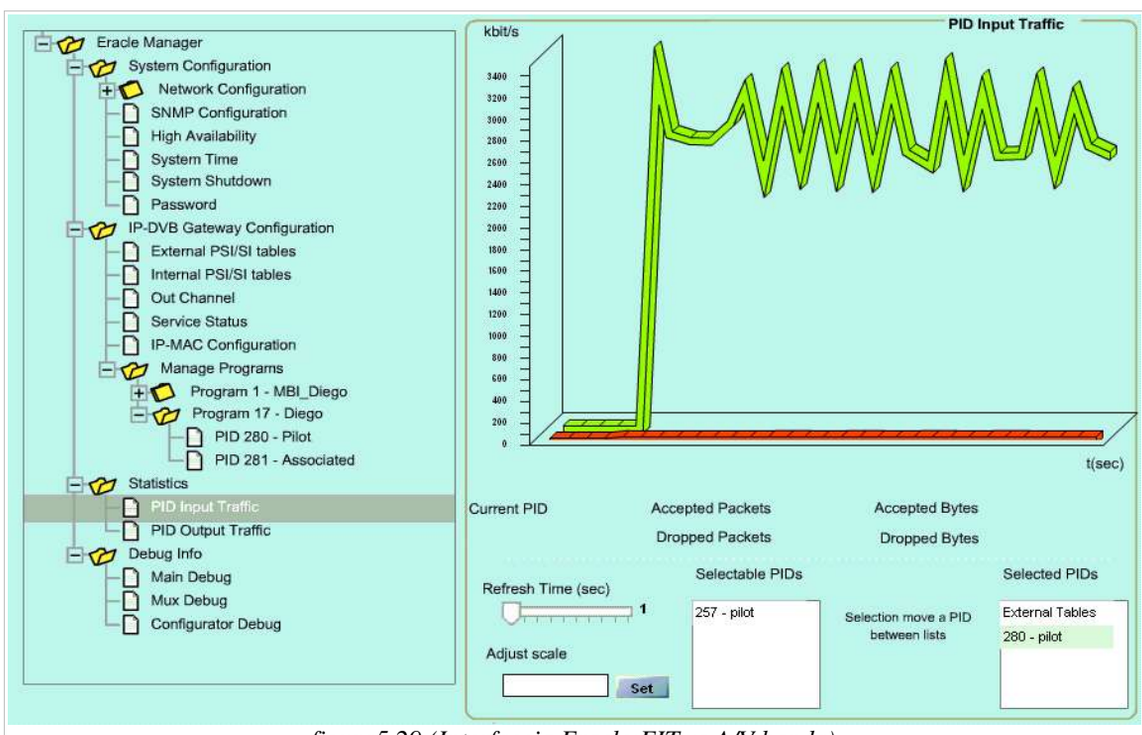

figura5.29 (Interfaccia Eracle:EIT vs A/V banda)

In figura 5.29 si può notare lo stesso grafico al quale è stato aggiunto il segnale Audio/Video.La banda ad esso associato è di circa 3Mbit/sec.In uscita Eracle trasmette il Transport Stream composto dagli elementary stream della EIT, del segnale Audio/Video e delle tabelle PAT, PMT e SDT.La verifica finale del lavoro svolto sin quì è l'effettiva visualizzazione dei dati relativi all'evento di test sul teleschermo.Per ricevere il canale di test si avvia la ricerca automatica sul STB.Il ricevitore una volta trovata la portante a 714Mhz, memorizza il canale in essa contenuto (per l'appunto il canale *DIEGO)*.La foto di figura 5.30 mostra che effettivamente una volta sintonizzatisi su di esso compare nel campo riservato all'evento attuale il nome *"Evento MBI"* il cui testo associato è proprio quello inserito nello *Short event descriptor "Test end2end tesi Diego Strina"*.Si verifica inoltre la corrispondenza della data e ora di inizio (Giovedì 12 Ottobre 11:30) e delle durata (30 minuti).La foto di figura 5.31 mostra la schermata relativa all'EPG, ove si possono vedere i dati relativi all'unico evento presente nell'elementary stream riservato alle EIT.

figura5.30 (Visualizzazione della EIT Present sul teleschermo)

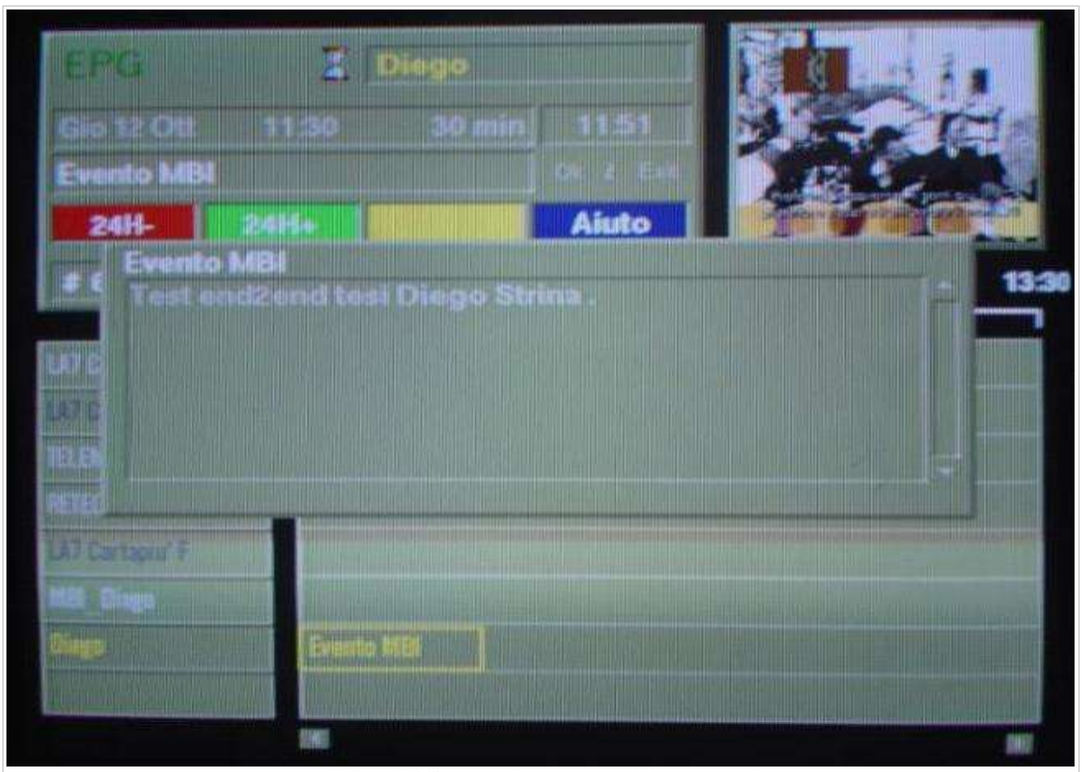

figura5.31 (EPG: visualizzazione dell'evento di test)

Per concludere si riportano in figura 5.32 e 5.33 le foto del laboratorio MBI dove è stata effettutata la simulazione e dei due apparati Eracle/modulatore DVB-T.

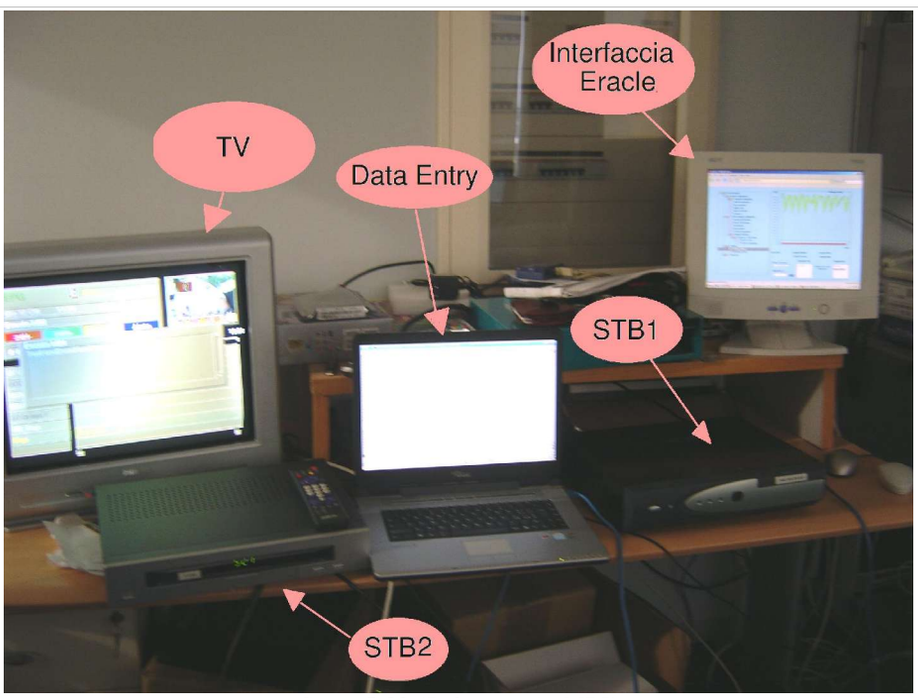

figura5.32 (Laboratorio MBI)

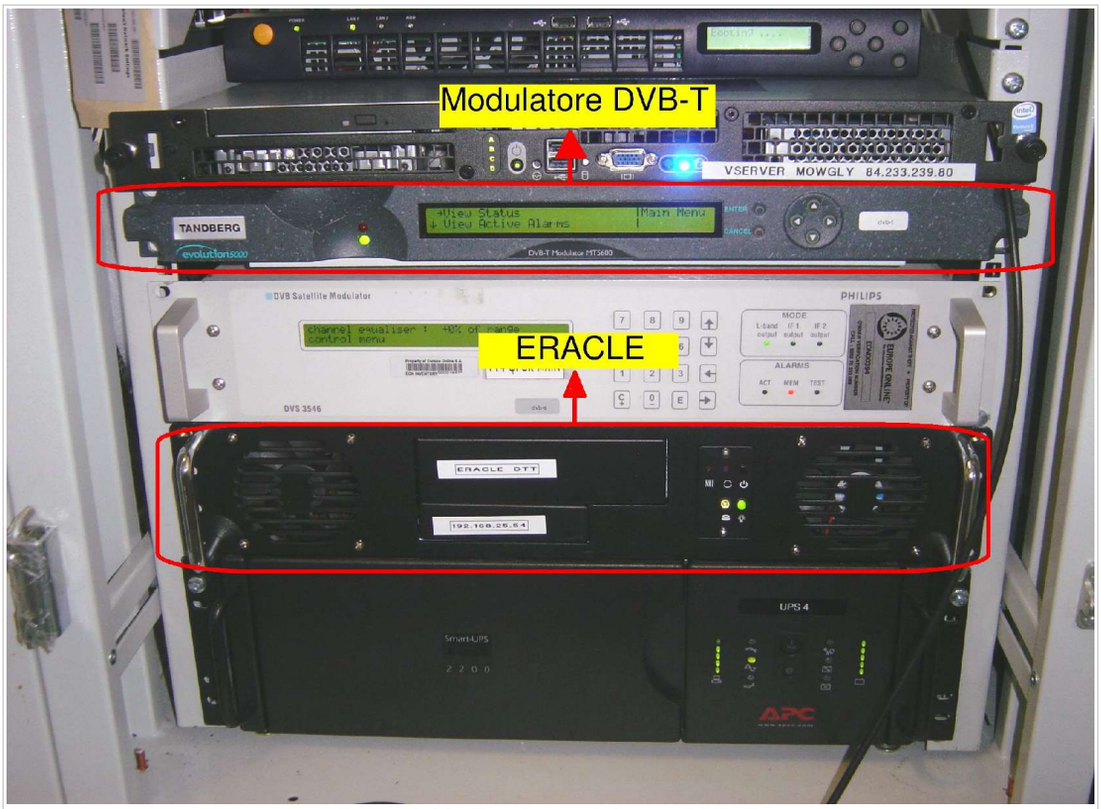

Figura5.33 (Eracle, modulatore DVB-T)

Riferimenti bibliografici:

- *Tutorial: XML messaging with SOAP (ibm.com/developerWorks)*
  www.digilife.be/quickreferences/PT/XML%20messaging%20with%20SOAP.pdf
- *Understanding SOAP, Matt Baker University of Arkansas*
  www.csce.uark.edu/~aapon/courses/gridcomputing/notes/SOAP.pdf