3) Real time protocol (RTP).

3.1 Introduzione a RTP.

In questo capitolo vedremo il protocollo RTP, il quale costituisce un valido supporto per il trasferimento dati in tempo reale, come per esempio gli stream audio o video di una videoconferenza.

I servizi forniti da RTP sono essenzialmente i seguenti:

1) l'identificazione del payload type
2) sequence numbering
3) timestamping
4) monitoring.

Le applicazioni tipicamente pongono RTP sopra UDP. Tuttavia RTP può essere usato con altri protocolli di rete e trasporto. Il trasferimento di dati verso molteplici destinazioni avviene con una distribuzione multicast.

E' da notare che questo protocollo non prevede nessun meccanismo che assicuri una corretta trasmissione o che garantisca la qualità del servizio, in quanto questo e' compito degli strati più bassi.

Inoltre esso, non s'interessa direttamente di un errato ordine di consegna dei pacchetti. I numeri di sequenza inclusi da RTP permettono, volendo, al ricevente di ricostruire il corretto ordine.

Sebbene RTP sia stato implementato principalmente per le conferenze multicast multimediali, questa non è la sua unica applicazione. Altre possibili applicazioni sono per esempio: la memorizzazione di un flusso dati continuo, simulazioni interattive distribuite, applicazioni di misurazione e controllo.

L'Internet Engineering Task Force (IETF), ideatrice di RTP, ha suddiviso lo standard, in due parti strettamente legate (vedi fig. 3.1):

1) il real time transport protocol (**RTP**): per trasportare dati con proprietà temporali.

2) l'RTP control protocol (**RTCP**): per "monitorare" la qualità del servizio e fornire informazioni sui partecipanti di una sessione in atto.

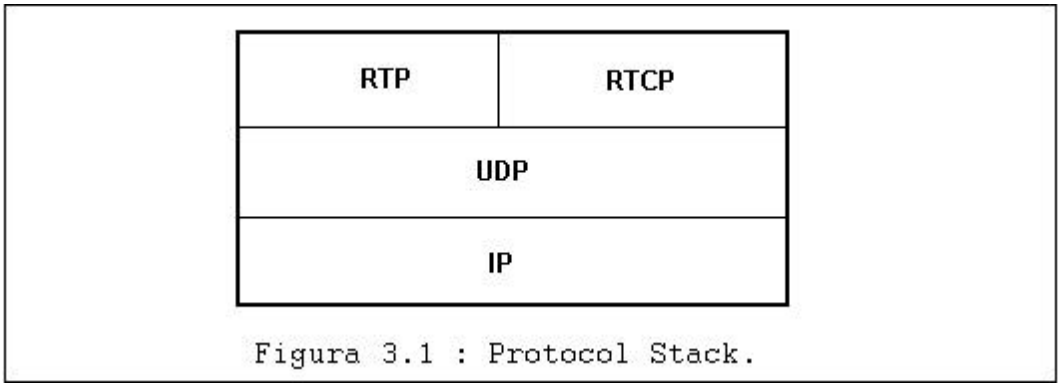

Figura 3.1 : Protocol Stack.

RTP rappresenta un nuovo stile di protocolli, che seguono i principi dell'Application Level Framing (ALF) e dell'Integrated Layer Processing (ILP) (vedi più avanti). In altre parole, esso vuole essere malleabile per fornire all'applicazione le particolari informazioni di cui essa ha bisogno. Esso sarà quindi integrato nell'applicazione, invece di essere implementato come uno strato separato. (vedi fig. 3.2)

Da notare inoltre che l'RFC1889 non specifica completamente il protocollo RTP, infatti esso è stato progettato per essere "aggiustato su misura" ogni volta che lo si implementa. Per questi motivi una completa

specifica di RTP richiede uno o più documenti d'accompagnamento (vedi fig. 3.3):

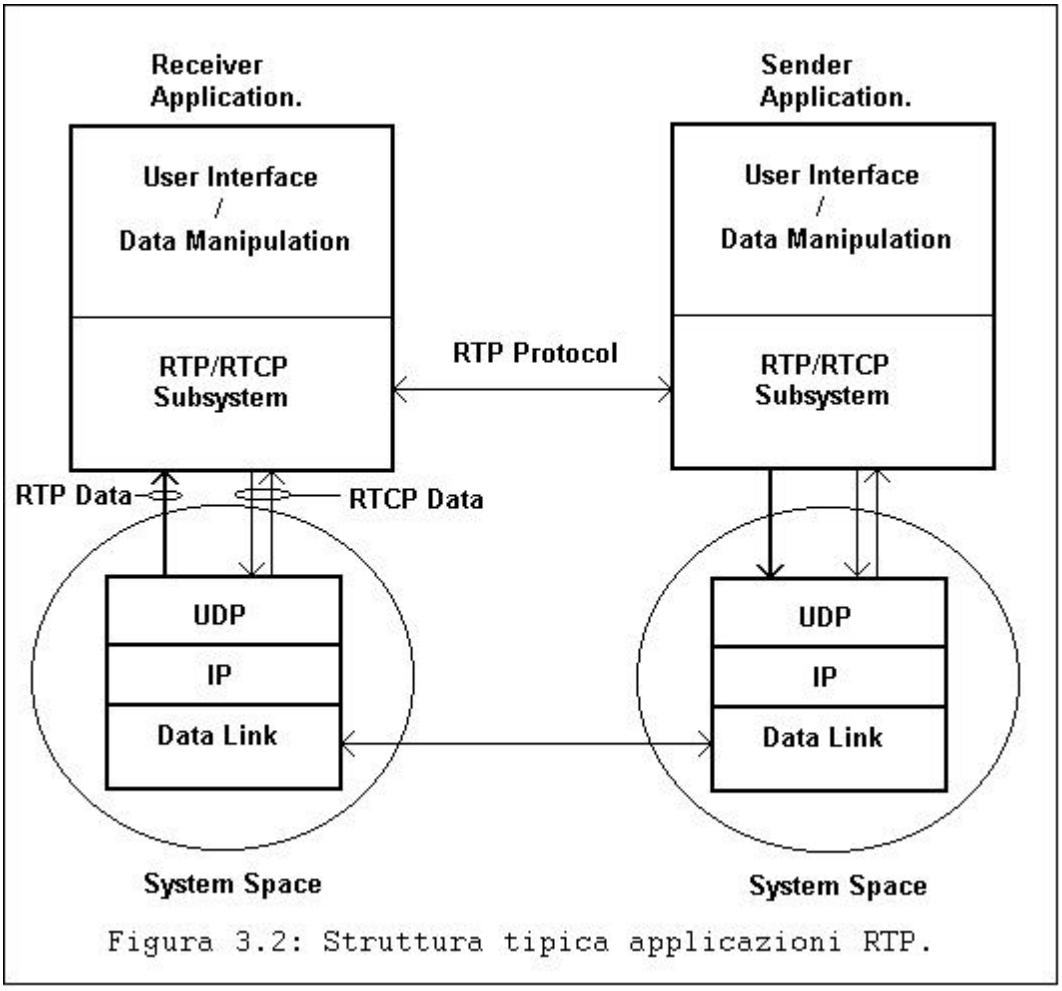

Figura 3.2: Struttura tipica applicazioni RTP.

1) documento di specifica del **profile**: definisce i payload type possibili, estensioni e campi particolari dell'header fisso. Io in particolare farò riferimento all'RFC 1890 che definisce il profile per conferenze audio/video e che è il profile usato in H.323.

2)documenti di specifica del **payload format**: specifica come un particolare payload deve essere trasportato nel pacchetto RTP (ricordo che il payload la parte di dati audio o video da inserire nei pacchetti). Da notare che questo documento non è necessario per tutti i tipi di

payload, infatti, quelli più semplici (G.711, G722, ...) sono già contemplati nel profile.

Ll'RFC 1889 propone anche alcuni algoritmi interessanti per l'implementazione di RTP.

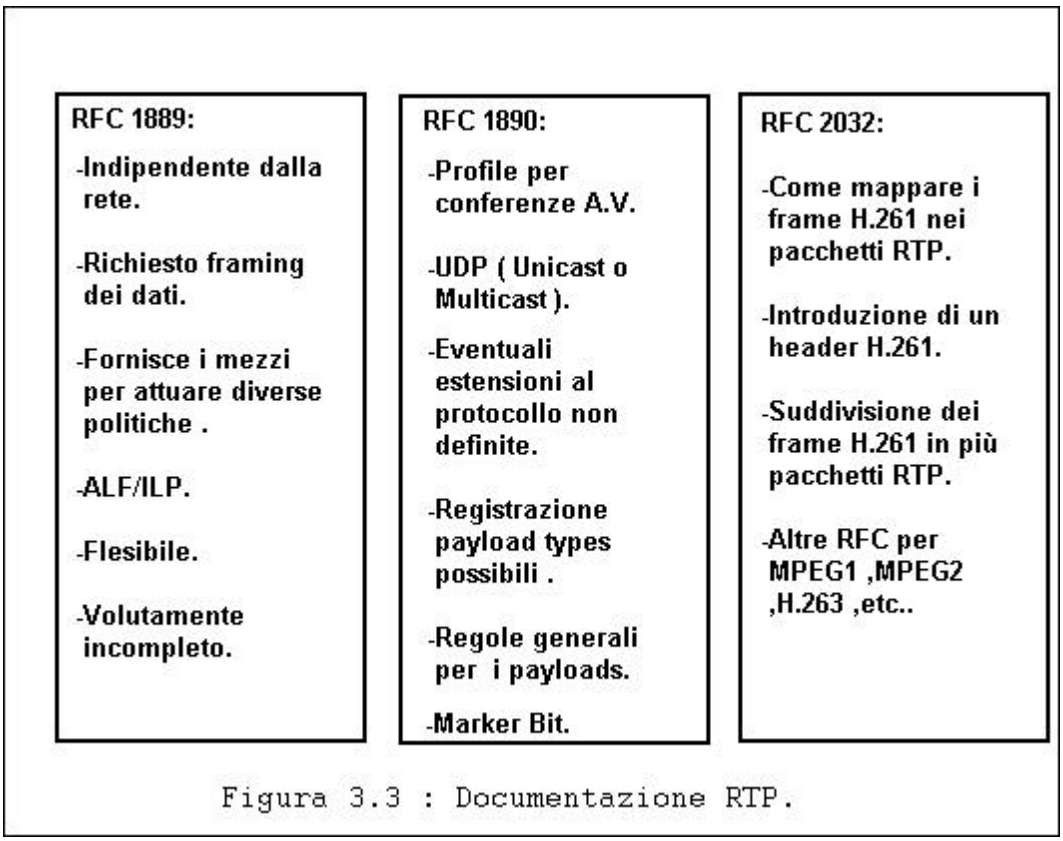

Figura 3.3 : Documentazione RTP.

L'attuale Internet non può ancora sostenere la completa richiesta di tutti i servizi tempo reale che si vorrebbe. Sarebbe, infatti, necessaria una banda più larga, al fine di non rallentare troppo gli altri servizi.

to importante per le applicazioni che utilizzano RTP prevedere appropriate precauzioni per **limitare l'uso accidentale di una banda troppo ampia**.

3.2 Esempi d’impiego di RTP.

In una conferenza audio/video entrambi i media sono trasmessi come sessioni RTP separate e i pacchetti RTCP relativi usano due differenti coppie di porte UDP. Le sessioni RTP audio e video, non sono, perciò, accoppiate direttamente tra loro.

Il motivo di questa separazione sta nel voler permettere ai partecipanti della conferenza di ricevere soltanto uno dei due media a loro scelta. Nonostante questo è però possibile sincronizzare il playback della sorgente audio/video, usando informazioni di temporizzazione, trasportate nei pacchetti RTCP, per entrambe le sessioni.

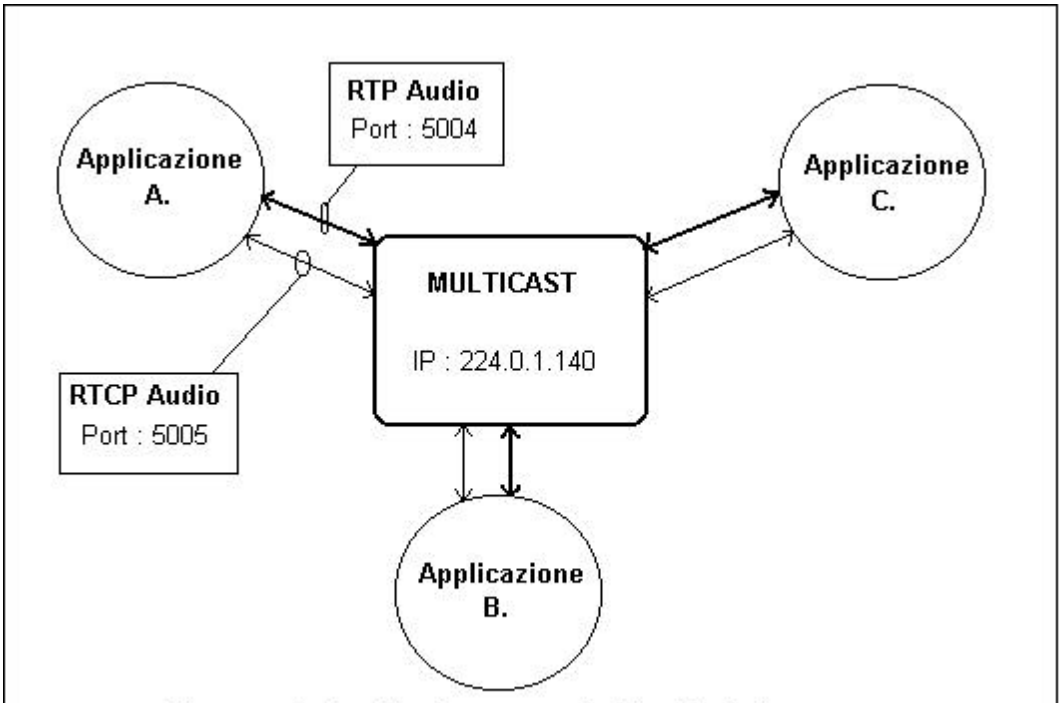

Figura 3.4: Conferenza Audio Multicast.

In figura 3.4 è mostrata una semplice **conferenza audio multicast**.

Attraverso qualche meccanismo i partecipanti sono a conoscenza dell’indirizzo IP multicast, e delle due porte che saranno usate, una per i dati audio (porta pari), una per i dati di controllo RTCP (successiva porta dispari). Tale meccanismo di accordo può essere per esempio fornito da H.323. I valori in figura sono puramente d’esempio.

Ogni partecipante invia i dati audio in pacchetti da 20mS di durata. Ognuno di tali pacchetti ha un header RTP in cui è indicata la codifica usata per l'audio (per esempio: PCM, ADPCM, …). Da notare che il payload type può cambiare durante la conferenza per reagire ad esempio ad un calo di banda. Anche qui H.323 può svolgere la funzione di stabilire quale sarà il nuovo payload type.

Nell'header sono indicati, fra l'altro, il numero di sequenza del pacchetto, e l'istante di campionamento del primo campione contenuto nel payload.

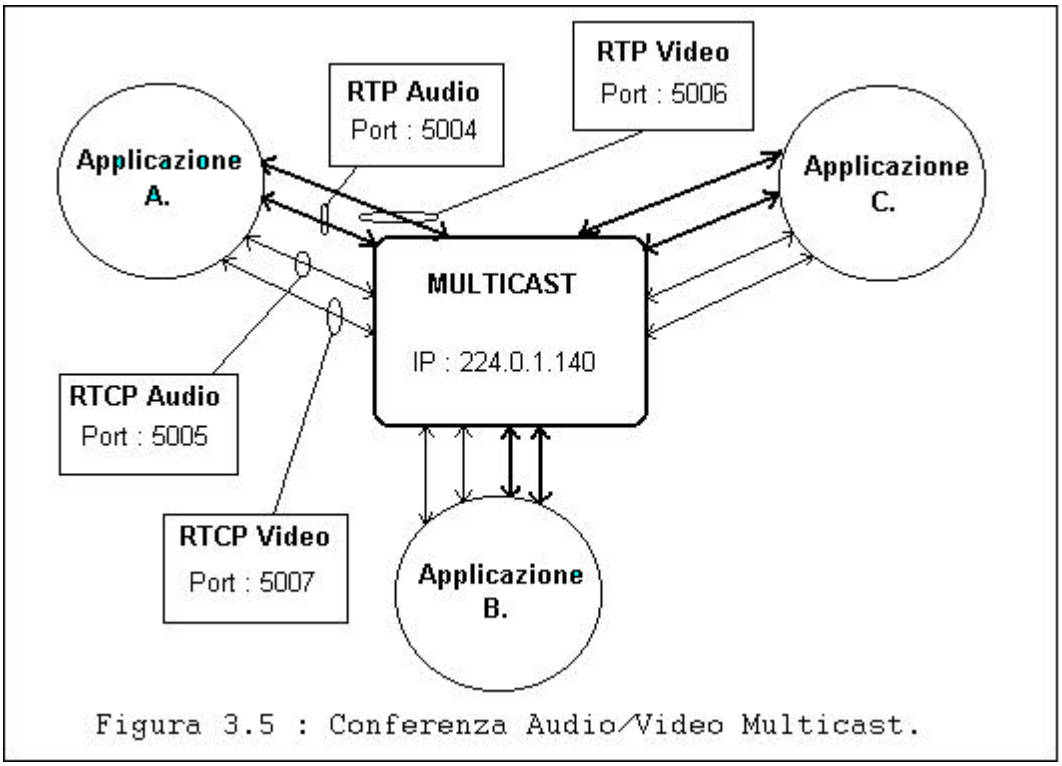

Figura 3.5 : Conferenza Audio/Video Multicast.

Periodicamente inoltre sono trasmessi dei pacchetti RTCP, per informare gli altri partecipanti sulla qualità della trasmissione, e su altre informazioni di controllo come lo user name del partecipante.

Un altro esempio può essere una **conferenza audio/video** (fig. 3.5). La prima cosa da dire è che per audio e video sono usate due separate sessioni RTP, in altre parole si usano due porte RTP separate per i dati audio/video e due porte RTCP distinte. L'unica cosa che consente l'accoppiamento fra i due

flussi dati, è il CNAME (Canonical NAME) contenuto nei pacchetti RTCP che identifica ogni partecipante.

La sincronizzazione del movimento delle labbra con la voce è consentita sempre dai pacchetti RTCP che contengono le informazioni per il corretto timing.

In questi esempi non sono state prese in considerazione altre due entità RTP, mixer e translator, che hanno lo scopo di rendere più flessibile tale standard. Le esamineremo più avanti.

3.3 RTP: transport protocol.

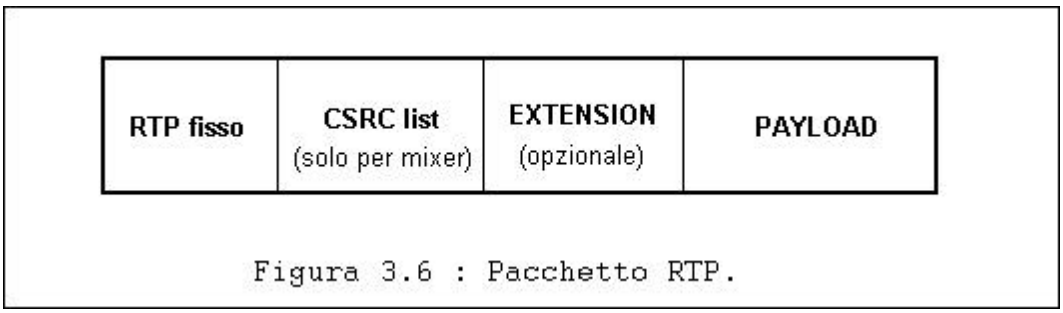

Figura 3.6 : Pacchetto RTP.

Per ogni partecipante di una conferenza, la **sessione RTP** è definita da una particolare coppia d'indirizzi di destinazione:

1) un indirizzo di rete.
2) una coppia di porte.(una per l'RTP e una per l'RTCP)

In una conferenza multimediale, ogni media ha una sua separata sessione con i propri pacchetti RTCP. Le sessioni RTP multiple sono distinte da coppie di numeri di porta differenti.

Il tipico pacchetto dati; consiste in un header fisso RTP, una eventuale lista chiamata contributing sources (CSRC), e il payload data (la sorgente dati, vale a dire i campioni audio o i dati video compressi) (fig. 3.6).

I primi dodici bytes (RTP fisso) sono presenti in ogni pacchetto RTP, mentre la lista d'identificatori di contributing sources (CSRC) è presente soltanto se inserita da un mixer.

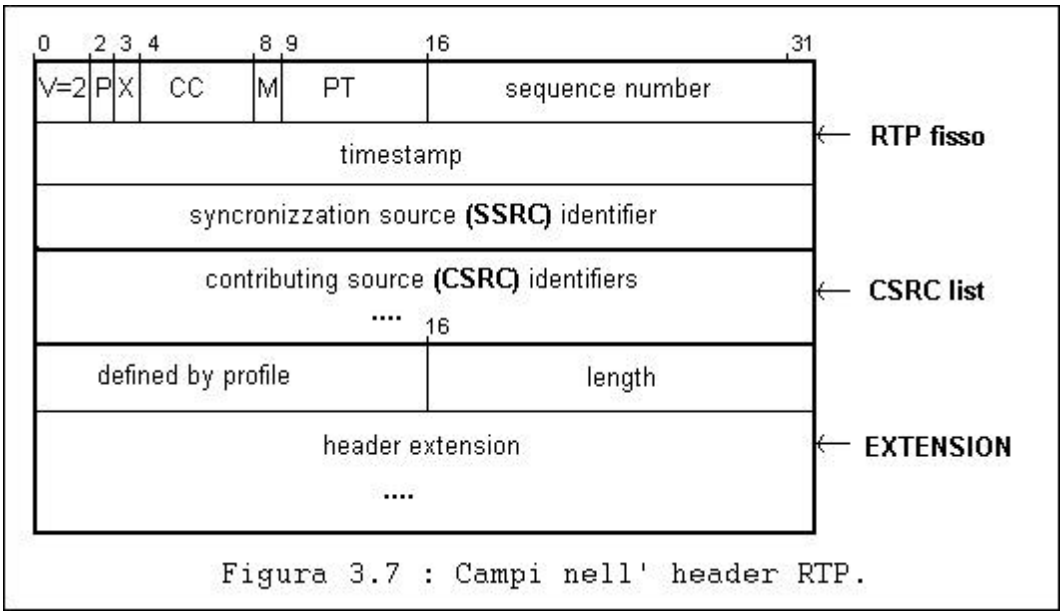

Figura 3.7 : Campi nell' header RTP.

I campi hanno i seguenti significati (vedi Fig. 3.7):

version (V): 2 bits, questo campo identifica la versione di RTP. La versione definita da questa specifica è 2. (Il valore 1 è usato nella prima versione RTP e il valore 0 è usato dal protocollo inizialmente implementato nel VAT audio tool).

padding (P): 1 bit, se il bit padding è attivo, il pacchetto contiene alla fine uno o più padding bytes .(byte riempitivi addizionali, non facenti parte dei dati). L'ultimo byte padding contiene un valore di quanti bytes padding dovrebbero essere ignorati. Questi particolari bytes possono essere necessari per un'eventuale criptaggio con blocchi di misura fissa, o per trasportare pacchetti RTP in un'unità dati per protocolli di strati inferiori.

extension (X): 1 bit, se il bit d'estensione è settato, la parte fissa della testata è seguita esattamente da un'estensione header, con un formato da definire.

CSRC count (CC): 4 bits, il CSRC count contiene il numero di CSRC che seguono l'header fisso

marker (M): 1 bit , l'interpretazione del marker è definita da un documento (profile)

payload type (PT): 7 bits , questo campo identifica il formato del payload RTP, e determina la sua interpretazione.

sequence number : 16 bits , il sequence number s'incrementa di un'unità ogni pacchetto RTP spedito, così il ricevitore può risalire ad un'eventuale perdita di pacchetti e ristabilirne la corretta sequenza. Il valore iniziale del sequence number è casuale per rendere difficili eventuali attacchi sul criptaggio.

timestamp: 32 bits, il timestamp riflette l'istante di campionamento del primo byte del pacchetto RTP. L'istante di campionamento deve essere derivato da un clock che s'incrementa monotonamente e linearmente nel tempo. Questo permette i controlli di sincronizzazione e le misure temporali sul campionamento dei pacchetti. Nel caso i pacchetti RTP siano generati periodicamente; bisognerà considerare l'istante di campionamento nominale, determinato dal clock di campionamento e non come lettura del clock di sistema. Il valore iniziale del timestamp è casuale, come per il sequence number. La frequenza di tale clock dipende dal tipo di codifica usata dal payload.

Molti pacchetti RTP consecutivi possono avere gli stessi timestamp se essi sono (logicamente) generati nello stesso istante. Pacchetti consecutivi possono contenere timestamp non monotoni se il dato non è trasmesso nell'ordine di campionamento (dipende dai tipi di compressione usati).

synchronization source (SSRC): 32 bits, la sorgente di uno stream di pacchetti RTP, è identificata da un identificatore SSRC, trasportato nell'header RTP in modo da non dipendere dall'indirizzo di rete.

Tutti i pacchetti con la stessa synchronization source devono avere lo stesso clock e lo stesso generatore di sequence number, in modo che il

ricevente raggruppi correttamente i pacchetti per il playback. I synchronization source possono essere riferiti a sorgenti di segnali come un microfono, una videocamera, o un mixer RTP (...il mixer ha un suo SSRC il translator invece non è considerato una sorgente).

L'SSRC è scelto in maniera random per essere probabilisticamente unico per una sessione RTP.

Se un partecipante genera stream multipli in una sessione RTP, per esempio videocamera e microfono, ogni sorgente deve essere identificata con un SSRC differente. Sebbene la probabilità che sorgenti multiple siano associate allo stesso SSRC è bassa, tutte le implementazioni RTP devono essere preparate ad evitare e risolvere le collisioni.

contributing source (CSRC): fino a 15 campi, 32 bits. Sono la lista degli SSRC che hanno contribuito a formare uno stream prodotto da un mixer. Questa lista è chiamata CSRC list. Di solito la funzione del mixer è proprio quella di condensare più flussi RTP, in un unico flusso. Il numero di CSRC è dato dal CC field. Se ci sono più di 15 contributing sources, soltanto 15 possono essere identificati.

length: lunghezza in parole di 32-bit dell'estensione; per fare il parsing correttamente.

header extension: l'estensione dell'header è prevista solo per un uso limitato come ad esempio la sperimentazione. L'importante di fatto è saperla evitare per fare il parsing dei pacchetti correttamente. Tutte le informazioni su un'eventuale estensione sono a carico di un profile.

3.4 RTCP: control protocol.

L'RTP control protocol è basato sulla trasmissione periodica dei pacchetti di controllo a tutti i partecipanti ad una sessione. Il protocollo sottostante deve provvedere alla separazione dei pacchetti di dati e di controllo, usando porte con numeri diversi.

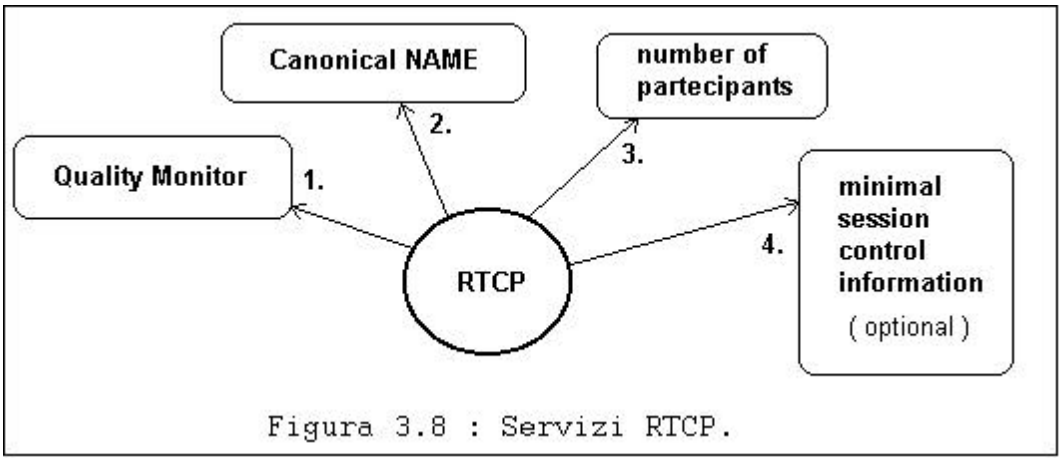

Figura 3.8 : Servizi RTCP.

L'RTCP esegue quattro funzioni:

1)La funzione principale è provvedere al **controllo della qualità** della trasmissione dati. Questa funzione è permessa dai reports RTCP. E' anche possibile, per una entità non direttamente coinvolta nella sessione, ricevere informazioni di feedback e di agire quindi da “monitor" di rete diagnosticando eventuali problemi di congestione.

2)RTCP trasporta un identifier di sorgente RTP chiamato canonical name **(CNAME)**. Poiché l'SSRC identifier può cambiare in caso di problemi di conflitto o di restart di programmi, i riceventi richiedono il CNAME per ricondursi al partecipante. Essi richiedono il CNAME anche per riorganizzare sessioni multiple trasmesse da uno stesso mittente (per es.: video e audio).

3)Le prime due funzioni richiedono che tutti i partecipanti spediscano pacchetti RTCP. Per questo la frequenza alla quale vengono spediti i pacchetti, deve essere controllata in funzione del numero di partecipanti. Spedire pacchetti RTCP "tutti a tutti", permette di risalire al **numero di partecipanti**, quindi di permettere il calcolo di tale frequenza (vedi RTCP Transmission Interval).

4)Una quarta funzione opzionale regola un controllo minimo sulle sessioni in atto, dove i partecipanti entrano ed escono senza parametri di negoziazione. Ciò serve a realizzare una **conferenza con libero accesso**.

Le funzioni 1-3 sono obbligatorie, mentre la 4 può non essere implementata.

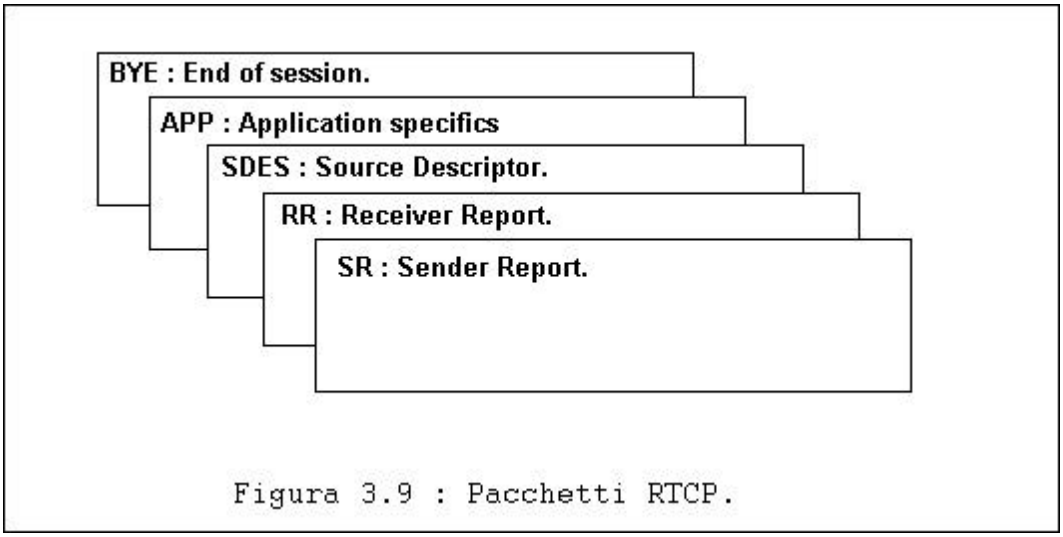

Figura 3.9 : Pacchetti RTCP.

RTCP si basa sulla trasmissione di cinque **tipi di pacchetto** (Fig. 3.9):

-**SR (sender report)**: per portare statistiche di ricezione e di trasmissione effettuate dai partecipanti trasmettono dati RTP.

-**RR (receiver report)**: per le statistiche di ricezione di un partecipante che riceve solo dati RTP.

-**SDES (source descriptor)**: per gli elementi di descrizione , incluso CNAME .

-**BYE** : indica la fine di una partecipazione.

-**APP** : pacchetto per funzioni specifiche di una applicazione.

Ogni pacchetto è costituito da una prima parte fissa, e da una seconda parte variabile. Più pacchetti RTCP possono essere inviati in uno stesso pacchetto UDP . Ogni pacchetto deve essere però processato dalla applicazione in modo indipendente l'uno dall'altro.

I seguenti **vincoli** sono però imposti :

- I pacchetti SR e RR devono essere spediti tutte le volte che i vincoli di banda lo consentono , cioè devono sempre esserci in un pacchetto composto.

- I nuovi ricevitori devono ottenere il CNAME delle sorgenti il più presto possibile per associare i diversi stream RTP.

- Il numero di pacchetti in un pacchetto composto, inizialmente deve essere limitato per aumentare la probabilità di un successo nella validazione dei pacchetti stessi.

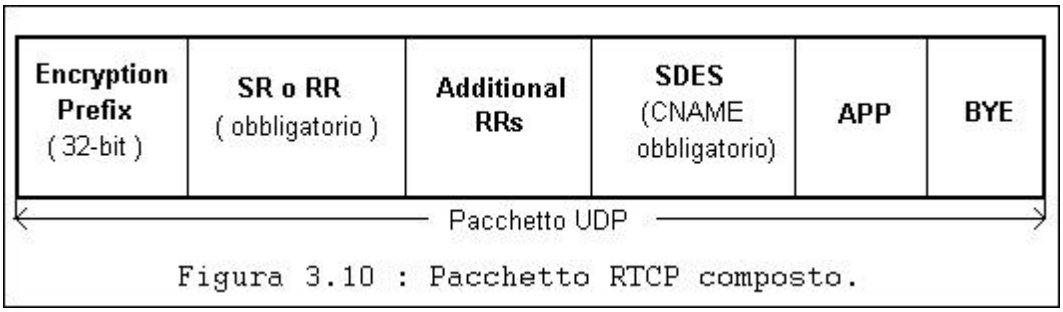

Figura 3.10 : Pacchetto RTCP composto.

Il pacchetto composto, di almeno due pacchetti RTCP, avrà il seguente **formato** (Fig. 3.10):

- **prefisso di criptazione**: solo se il pacchetto è criptato esso è preceduto da una quantità random di 32-bit ricalcolata per ogni pacchetto (vedi RFC1321 e RFC2437 per chiarimenti sulla criptazione).

-**SR o RR**: il primo pacchetto deve essere uno fra questi due per facilitare la validazione.

-**RRs addizionali**: pacchetti che vanno aggiunti, se il numero di sorgenti di cui si sono fatte le statistiche supera 31.

-**SDES**: un pacchetto SDES contenente CNAME deve essere presente in ogni pacchetto composto. Altri SDES opzionali possono essere inclusi.

-**BYE o APP:** in particolare il pacchetto BYE deve essere l'ultimo.

3.5 La banda impiegata.

RTP è un protocollo progettato per sessioni che vanno da pochi a migliaia di partecipanti, quindi non solo per servizi di videoconferenza ma anche per servizi di video on demand, o trasmissioni “TV” su Internet. Non stupisce quindi l’attenzione che viene data al controllo della banda impiegata.

Prima di tutto va notato che la banda impiegata dalle trasmissioni audio sono auto limitate perché in generale parla un solo interlocutore alla volta. Per quanto riguarda il traffico video, esso cresce con il numero dei partecipanti, ma si suppone che in una conferenza il numero dei partecipanti sia limitato mentre in una trasmissione “TV”, la sorgente video è una sola. Si conclude quindi che anche il traffico video è auto limitato.

Ciò non può dirsi vero per il traffico dei pacchetti RTCP. Si pensi ad esempio proprio ad una trasmissione “TV”, in cui ogni ricevente manda il suo pacchetto RTCP. L’intervallo di trasmissione fra un pacchetto RTCP e l’altro va quindi regolato in base al numero di partecipanti.

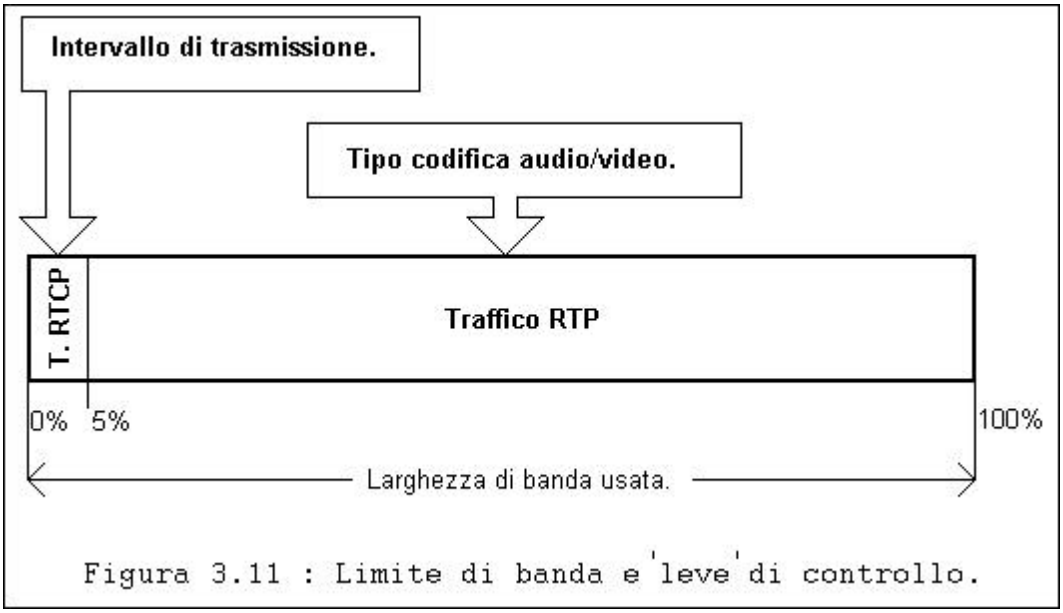

Figura 3.11 : Limite di banda e 'leve' di controllo.

Per ogni sessione si suppone quindi che il traffico sia sottoposto ad un **limite di banda,** (fig. 3.11) da dividere fra i partecipanti. Tale limite di banda può essere scelto secondo qualche criterio (ad esempio il costo della banda, o una percentuale della banda disponibile). Fatto sta che l’applicazione può

obbligare l’impiego di banda sotto un certo limite. Tale controllo si esercita usando tipi di codifica più o meno compressi.

La banda è calcolata con gli header introdotti da UDP (8 byte) e da IP (20 byte). L’header del link-level non è incluso in quanto, nel suo viaggio, il pacchetto può essere incapsulato in diversi link-level headers (Abbiamo a che fare con una internet!).

Il traffico di controllo deve essere limitato ad una piccola frazione della banda impiegata. Viene suggerito di usare il 5% del limite di banda previsto per la sessione (la frazione può in ogni caso essere specificata nel profile). Ciò viene fatto calcolando opportunamente l’intervallo di trasmissione fra un pacchetto RTCP e l’altro; per questo calcolo occorre tenere conto del numero di partecipanti. Nell’rfc1889 viene suggerito un algoritmo per calcolare tale intervallo e come tenere conto dei partecipanti alla sessione.

Nei pacchetti RTCP oltre ai report e al CNAME possono essere contenute informazioni addizionali, viene quindi suggerito che tali informazioni addizionali non superino il 20% del traffico di controllo per far sì che il CNAME sia spedito con sufficiente frequenza.

3.6 I pacchetti RTCP in dettaglio.

Incominciamo con il pacchetto SR (sender report). Tutti i pacchetti RTCP iniziano con una parte comune a tutti per facilitare il parsing (header). In particolare essi sono divisi in diverse sezioni.

Esaminiamo i campi della sezione **header**:

version (V): 2 bits. Come pacchetti RTP.

padding (P): 1 bits. Come pacchetti RTP.

reception report count (RC): 5 bits. Numero di blocchi di report.

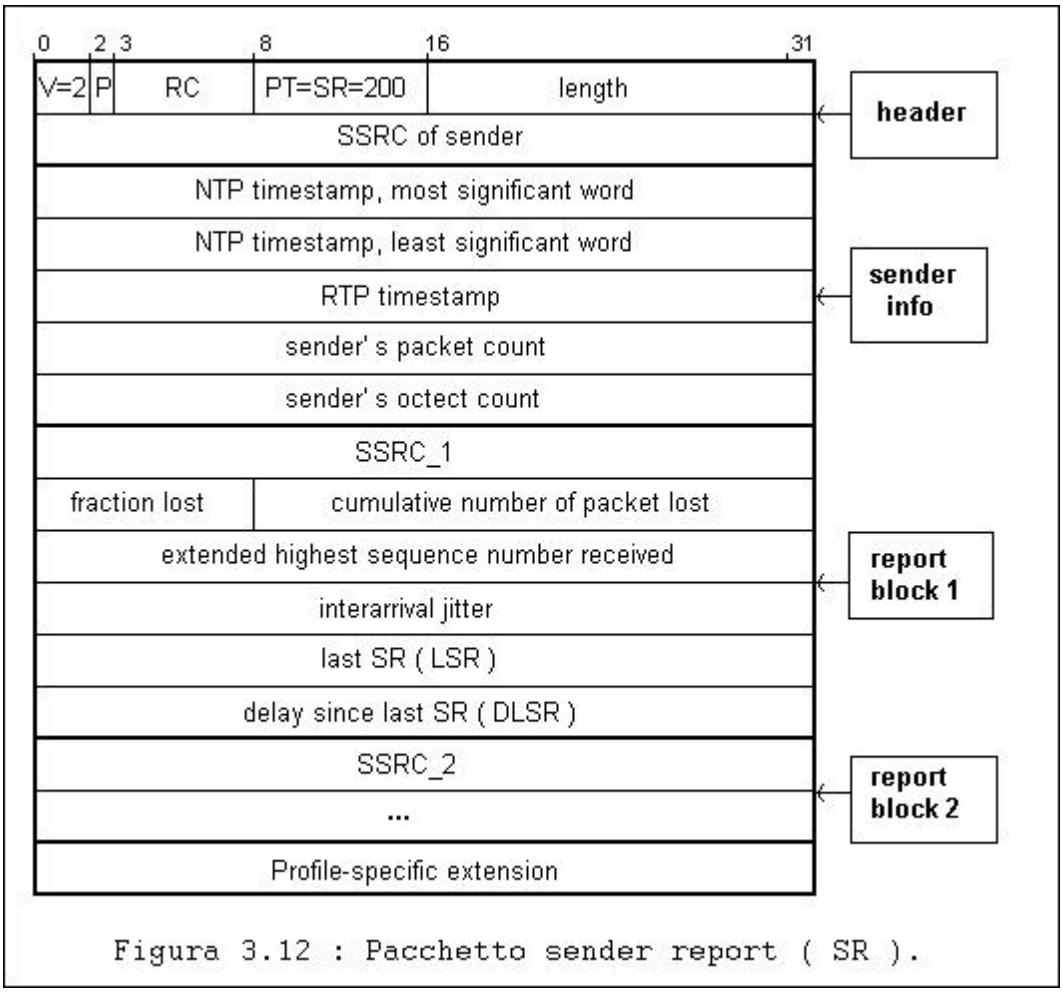

Figura 3.12 : Pacchetto sender report (SR).

packet type (PT): 8 bits. Costante che identifica il pacchetto SR.

length: 16 bits. Lunghezza di questo pacchetto RTCP espressa in parole da 32-bit. Questo campo viene usato per separare i pacchetti in un pacchetto composto.

SSRC: 32 bits. Come pacchetti RTP.

Esaminiamo i campi della sezione **sender info**:

NTP timestamp: 64 bits. Indica l'istante in cui è stato mandato questo pacchetto. Il formato NTP (wallclock time) è specificato in [*].Può essere usato per calcolare il tempo di trasmissione del pacchetto. Vale 0 se l'applicazione non ha nozione del wallclock time.

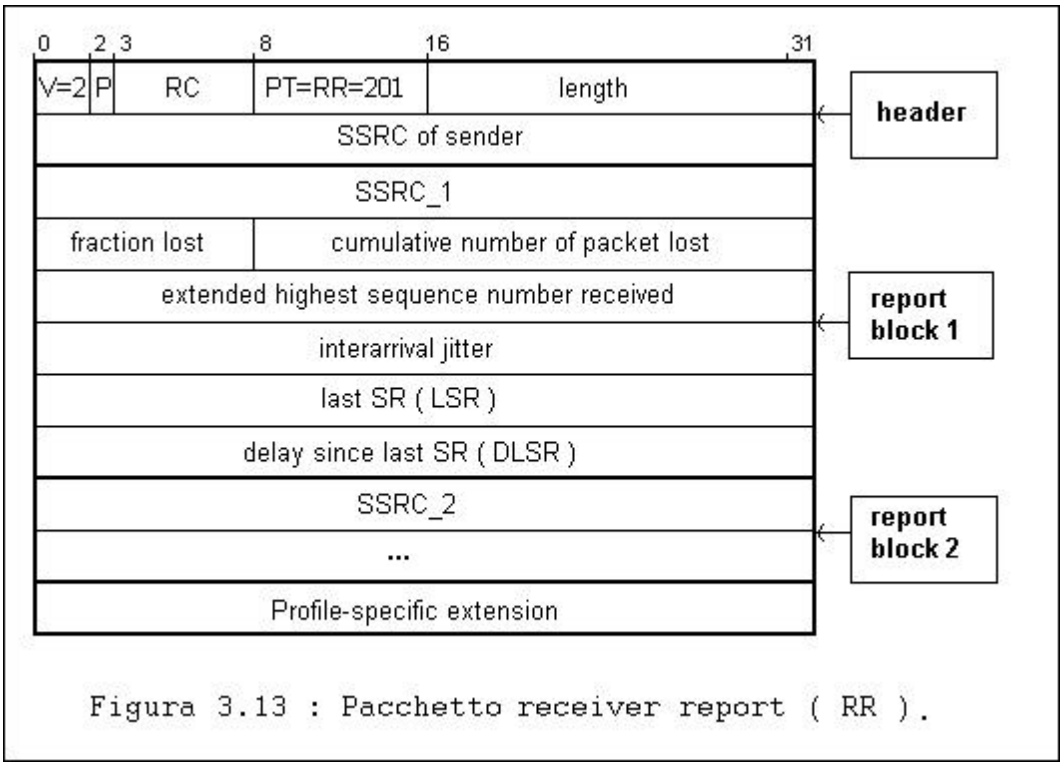

Figura 3.13 : Pacchetto receiver report (RR).

RTP timestamp: 32 bits. Corrisponde allo stesso istante del precedente campo ma specificato secondo lo stesso formato e offset usato nei pacchetti RTP. Questo campo è utile per la sincronizzazione fra sorgenti diverse.

sender's packet count: 32 bits. Numero di pacchetti RTP spediti dall'inizio della sessione.

sender' s octect count: 32 bits. Numero di byte spediti come payload nei pacchetti RTP, dall'inizio della sessione.

Esaminiamo la sezione **report block**:

SSRC_n: 32 bits. SSRC del sender cui si riferisce il report block.

fraction lost: 8 bits. Numero dei pacchetti RTP persi dal precedente r eport spedito (per il formato di questo campo vedi rfc1889/A.3).

cumulative number of packet lost: 24 bits. Numero di pacchetti RTP persi, spediti da SSRC_n. Esso viene calcolato dal numero di sequenza dei pacchetti RTP.

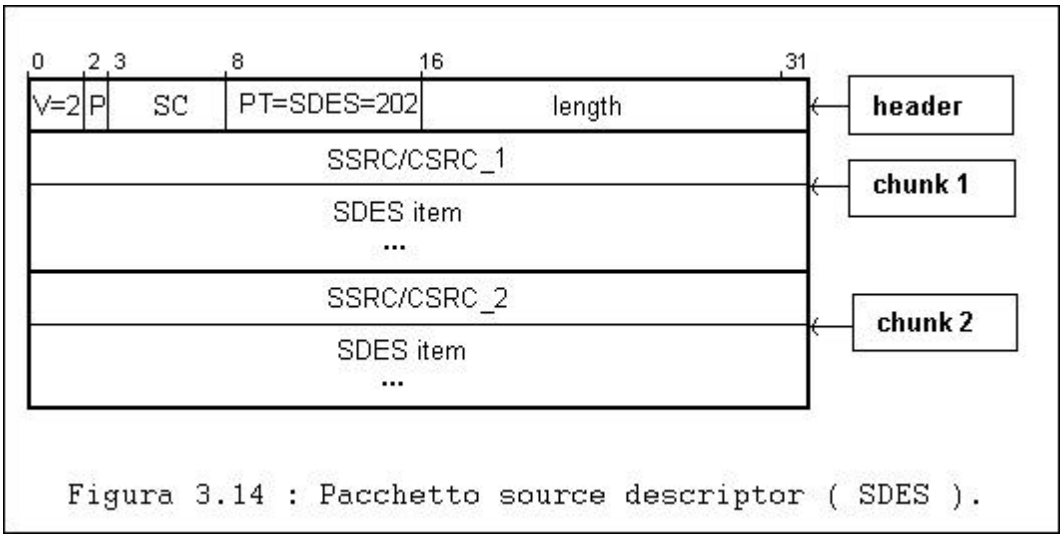

Figura 3.14 : Pacchetto source descriptor (SDES).

extended highest sequence number received: 32 bits. Numero di sequenza dell'ultimo pacchetto RTP ricevuto. In realtà il numero viene esteso da 16 a 32 bits per evitare ripetizioni cicliche.

Interarrival jitter: 32 bits. Stima della varianza statistica dell'intervallo di tempo d'arrivo dei pacchetti RTP. RTP propone anche l'algoritmo per calcolare il jitter.

last SR timestamp (LSR): 32 bits. Parte centrale del timestamp NTP del più recente pacchetto RTCP ricevuto da SSRC_n.

delay since last SR (DLSR): 32 bits. Il ritardo espresso in 1/65536 di secondo fra l'ultimo pacchetto RTCP ricevuto da SSRC_n e la spedizione di questo report.

Passiamo quindi al receiver report (RR) (fig. 3.13). Esso è in sostanza uguale al SR, solo che manca la sezione sender info e il campo PT deve essere settato al valore 201.

Sottolineo infine che questi pacchetti possono essere estesi, secondo un profile. Le estensioni, vanno fatte, solo se ritenute strettamente necessarie. Esse devono essere costituite da pochi byte, e tali da rendere semplice il parsing.

I pacchetti RTCP di report sono quindi molto importanti per determinare la qualità della trasmissione, ovvero il numero di pacchetti persi ,

il delay di trasmissione , il jitter. Consentono anche di compiere la sincronizzazione di diversi stream RTP.

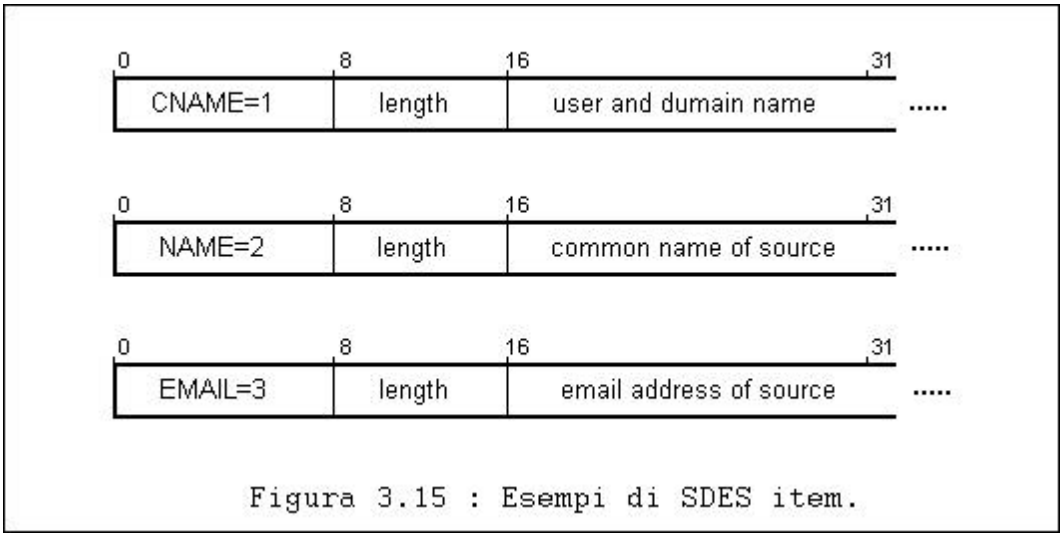

Figura 3.15 : Esempi di SDES item.

Il pacchetto **SDES** (Fig. 3.14) è costituito da un header e da zero o più chunk (porzioni), ognuno dei quali contiene una descrizione della sorgente.

Vediamo i campi non già visti:

source count (SC) : 5 bits. Indica il numero di chunk che seguono.

SSRC/CSRC: 32 bits. Indica la sorgente cui si riferisce il chunk.

SDES item: questo è un elemento costituito da tre campi (Fig. 3.15):

- un identificatore del tipo di item
- un campo che indica la lunghezza dell'item in parole da 32 bits.
- il campo che porta l'informazione sulla sorgente.

Rimangono infine i pacchetti BYE e APP . Per questi basta fare riferimento alla figura 3.16, che in pratica si commenta da sola.

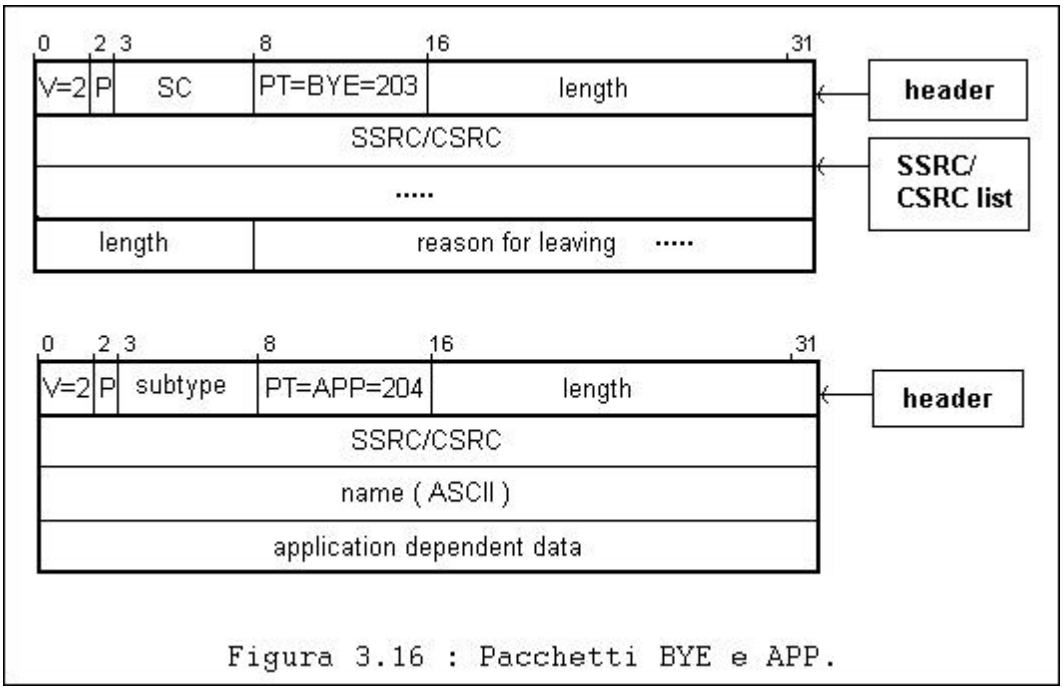

Figura 3.16 : Pacchetti BYE e APP.

3.7 Le entità: mixer e translator.

Il protocollo RTP prevede anche due entità opzionali (mixer e translator) che lo rendono più flessibile e adattabile ai casi reali. Prenderò in esame queste due entità senza approfondire troppo in quanto esse non sono prese in considerazione da H.323. L'importante è solo essere in grado di fare correttamente il parsing di pacchetti che provengono da un mixer. Bisogna anche aggiungere che le funzioni di un mixer e di un translator di fatto possono essere svolte da un MCU/H.323 quindi a mio avviso , dal punto di vista di H.323 , ciò ne sminuisce l'importanza. Inoltre per quanto riguarda i translator , vedremo che di fatto essi sono essenzialmente "trasparenti", nel senso che la loro presenza non dovrebbe modificare il comportamento dei componenti H.323.

Comunque questo è un argomento piuttosto avanzato e non mi sembra il caso di inoltrarsi troppo.

Passiamo a fare i due esempi di impiego più importanti per mixer e translator.

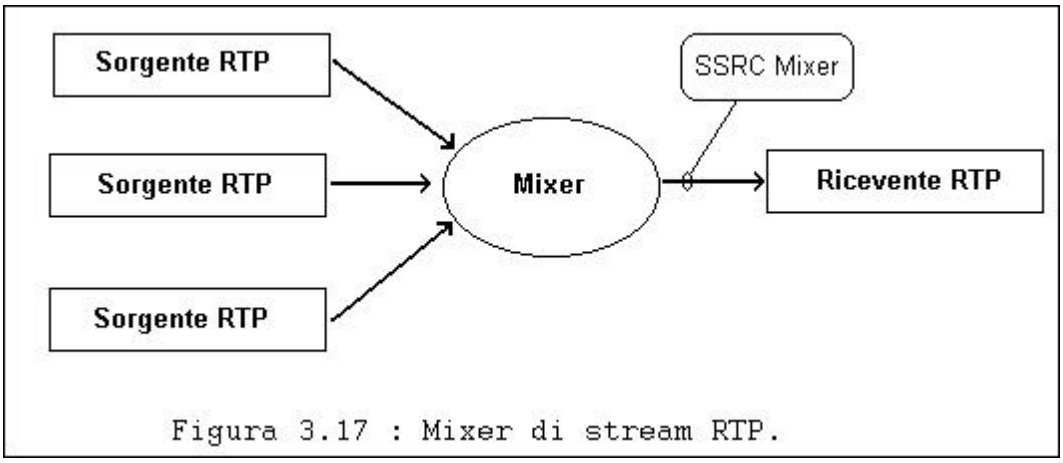

Figura 3.17 : Mixer di stream RTP.

Fino ad ora abbiamo supposto che tutti gli end-point ricevano dati nello stesso formato; in realtà ciò può essere scomodo.

Consideriamo il caso in cui i partecipanti abbiano tutti a disposizione una larga banda ,meno uno che dispone di una connessione "lenta".

Piuttosto che vincolare tutti ad una conferenza eseguita con risorse più basse, riducendo la qualità della sorgente, un'entità RTP, chiamata **mixer**, posto vicino all'area a banda stretta, può rielaborare i dati per metterli in un diverso formato (fig. 3.17). Il mixer cioè miscela gli audio/video stream ricevuti da tutti gli altri partecipanti, in un unico stream, compatibile con il collegamento a bassa velocità.

E' da notare che questo stream può essere unicast verso un processo, o multicast su molteplici processi di differenti host. L'header del pacchetto RTP prevede un campo per le informazioni di mixing . Come visto in precedenza i pacchetti RTP generati da un mixer , contengono una CSRC-list che è la lista degli SSRC (sorgenti) che hanno contribuito alla formazione di quello stream.

Vediamo ora il tipico esempio di utilizzo di un translator.

Alcuni dei partecipanti alla conferenza potrebbero essere connessi con collegamenti non direttamente raggiungibili via IP multicast. Per esempio, essi potrebbero trovarsi "dietro" un'applicazione firewall che non permette ad alcuni pacchetti IP di passare. Per questi siti, non il mixer ma un altra entità RTP, chiamata **translator**, può rivelarsi necessario (fig. 3.18). Due sono i translator installati: uno su entrambi i lati del firewall. Quello esterno, provvede a instradare su una connessione sicura i pacchetti multicast

pervenuti verso il firewall, mentre quello interno li ridistribuisce verso il gruppo multicast circoscritto alla rete interna (realizzando una conferenza ibrida).

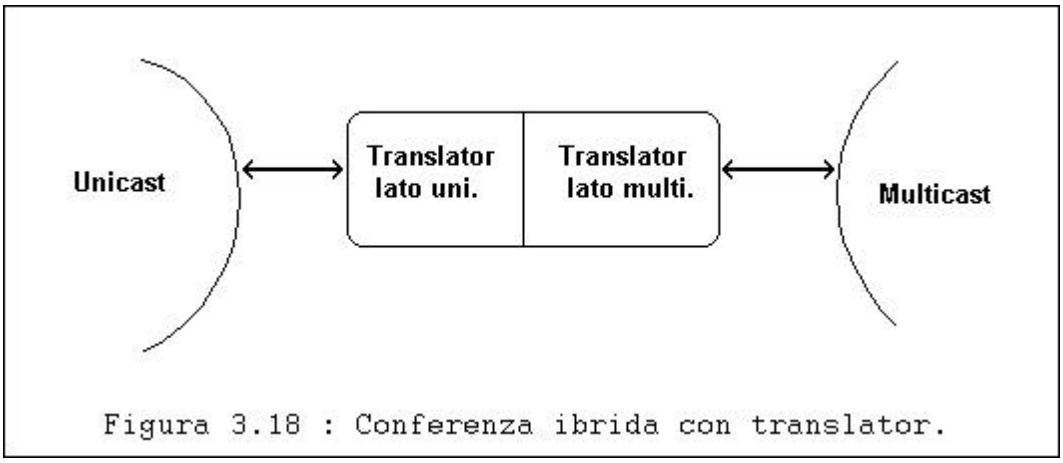

Figura 3.18 : Conferenza ibrida con translator.

Mixer e translator possono però essere progettati per gli scopi più disparati. Ad esempio un mixer può “unire “ più flussi video per creare

Per riassumere la più importante differenza fra translator e mixer è che i primi inoltrano i pacchetti RTP senza cambiare l’SSRC mentre i mixer mettono nei pacchetti un proprio SSRC e aggiungono la CSRC-list.(ovvero il terminale è consapevole di ricevere da un mixer ma non si accorge di ricevere da un translator)

3.8 Profile per conferenze audio/video.

Diamo quindi un breve sguardo al profile per conferenze audio video. In particolare questo profile è fatto per conferenze con controllo di sessione minimo. Esso non prevede quindi alcuna negoziazione di parametri e nessun controllo di accesso alla conferenza.

Questo profile è quindi stato progettato di fatto per essere usato in congiunzione con un protocollo di più alto livello.

Codifica	sample/frame	bits/sample	mS/frame	commento
1016	frame	N/A	30	
DVI4	sample	4		
G721	sample	4		
G722	sample	8		
G728	frame	N/A	2.5	
GSM	frame	N/A	20	
L8	sample	8		Linear audio data
L16	sample	16		
LPC	frame	N/A	20	Linear predictive
MPA	frame	N/A		MPEG-I , MPEG-II
PCMA	sample	8		G711/ A-law
PCMU	sample	8		G711/ u-law
VDVI	sample	var.		

Tabella 3.1 : Proprietà di alcune codifiche audio.

Passiamo quindi ad analizzare le scelte fatte:

-**RTP data header**: l'header RTP è utilizzato senza estensione o altre modifiche di alcun tipo.

-**RTCP packets**: i pacchetti RTCP sono quelli definiti dallo standard, senza estensioni.

- **intervallo RTCP**: l'intervallo di trasmissione RTCP è quello suggerito nella rfc1889 (vedi gli algoritmi proposti) che propone per il traffico di controllo il 5% della banda.

- **pacchetto SDES**: il pacchetto SDES, contenente l'item CNAME, deve essere sempre spedito nei pacchetti composti. Altri tipi di item possono essere spediti solo ogni 5 intervalli di trasmissione RTCP.

- **protocolli sottostanti**: per il trasporto dei pacchetti RTP si può usare solo unicast o multicast UDP.

Questo profile definisce inoltre un insieme di codifiche standard che possono essere usate come payload RTP. Tali codifiche sono registrate presso

l’Internet Assigned Numbers Autority (IANA).

Ad ogni codifica viene associato quindi un numero che la identifica univocamente.

Vediamo ora le regole generali da utilizzare per le **codifiche audio** (tab. 3.1).

Le applicazioni che non inviano pacchetti, durante il silenzio devono porre il marker bit a 1 nel primo pacchetto dopo un periodo di silenzio (talkspurt). Le applicazioni senza soppressione del silenzio useranno questo bit settato a zero.

Il clock usato per generare i timestamp RTP è uguale a quello dato dalla frequenza di campionamento, in modo indipendente dal numero di canali utilizzato (stereo, quadrifonia …) (in ogni caso non mi addentro sulla trasmissione di più canali audio).L’intervallo di default utilizzato per impacchettare l‘audio è di 20mS. Tale intervallo più è lungo, meno header-overhead introduce nei pacchetti e più aumenta il delay di trasmissione. Perciò tale intervallo può essere variato secondo la particolare applicazione. Comunque il buffer di ricezione deve essere di dimensioni tali da poter ricevere fino a 200 mS di audio.

Le codifiche audio si dividono in due tipi, quelle **sample-based** (ovvero l’audio è rappresentato da uno stream di campioni, in cui ogni campione è rappresentato da un numero fisso di bits) e quelle **frame-based** (ovvero l’audio è rappresentato da una sequenza di blocchi di dimensioni fisse). Con le codifiche sample-based ogni pacchetto RTP contiene semplicemente una sequenza di campioni, e la durata di un pacchetto è determinata dal numero di campioni presenti nel pacchetto. Per le codifiche frame-based si possono mettere più frame nello stesso pacchetto RTP. Avendo i frame di lunghezza fissa, è facile capire quanti ce ne sono in un pacchetto. Se una codifica audio non si inquadra nella precedente descrizione generale, è necessario definire un payload format che descrive come va messa la codifica sul pacchetto.

Per quanto riguarda le codifiche video, non sono poste regole generali e bisogna fare di volta in volta riferimento a un payload format (ne vedremo uno di esempio successivamente).

Nella tabella 3.2 sono riportati i clock e l'identificatore di alcune codifiche.

Per concludere le porte UDP da utilizzare devono avere un valore sopra 5000 in quanto nel sistema UNIX le porte sotto 5000 sono assegnate dal sistema operativo. L'altro vincolo è che bisogna utilizzare due porte consecutive, in cui quella pari (la più bassa) va per il traffico RTP. Per questo profile sono state registrate le porte 5004 e 5005 ma non è obbligatorio usare queste.

PT	Codifica	audio/video	clock (Hz)	canale audio
0	PCMU	A	8000	1
1	1016	A	8000	1
2	G721	A	8000	1
3	GSM	A	8000	1
5	DVI4	A	8000	1
6	DVI4	A	16000	1
7	LPC	A	8000	1
8	PCMA	A	8000	1
9	G722	A	8000	1
10	L16	A	44100	2
11	L16	A	44100	1
14	MPA	A	90000	
15	G728	A	8000	1
25	CelB	V	90000	
26	JPEG	V	90000	
28	nv	V	90000	
31	H.261	V	90000	
32	MPV	V	90000	
33	MP2T	AV	90000	

Tabella 3.2 : Payload Type (PT), per alcune codifiche.

3.9 Payload format per H.263.

Un payload format, come già detto, specifica come una certa codifica deve essere incapsulata in un pacchetto RTP. In questo caso voglio analizzare

brevemente il payload format per una sequenza di bit H.263, se non altro a titolo d'esempio.

H.263 è una codifica definita da una raccomandazione ITU-T, per codifiche video a velocità dati molto bassa.

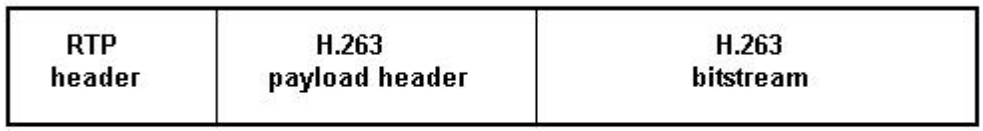

Figura 3.19 : Struttura del pacchetto video H.263.

Siccome è desiderabile che ogni pacchetto possa essere processato indipendentemente dagli altri, alcune informazioni vengono introdotte in ogni pacchetto. Viene perciò introdotto un payload header H.263, oltre alla sequenza di bits H.263. In realtà questo header può essere di tre tipi possibili per adattarsi di volta in volta alle esigenze dell'applicazione.

Il payload format specifica anche il significato del marker bit, che viene settato a 1 quando il pacchetto corrente trasporta la parte finale di un frame video. Il clock per la generazione dei timestamp è posto a 90000Hz.

Aggiungo infine che nel payload header H.263 vengono inserite informazioni anche per il ricovero da perdita di pacchetti.

3.10 Scalabilità di sessioni RTP.

Ricordo la proprietà di RTP di poter effettuare sessioni con migliaia di partecipanti. Per raggiungere quest'obiettivo non ho detto che in realtà l'intervallo di trasmissione dei pacchetti RTCP non è completamente deterministico, ma in realtà può variare casualmente entro un certo intervallo, questo per evitare che tutti trasmettano i pacchetti nello stesso istante.

Queste non sono proprietà fondamentali per la videoconferenza (almeno per la videoconferenza classica cui partecipano al più una manciata di persone), quindi ho preferito non approfondire, anche se tale problema è molto importante e delicato.

3.11 ALF/ILP: una nuova generazione di protocolli.

Ora voglio introdurre una breve panoramica sui due principi di progetto di un protocollo su cui si basa RTP.

Tali principi sono stati proposti da D.Clark e L.Tennenhouse in "Architectural considerations for a new generation of protocol".

Essi hanno l'obiettivo di fare i conti con reti a sempre più elevata velocità, e con applicazioni che devono fornire una larga gamma di servizi come la comunicazione audio, video, o il real time rendering. Ciò implica che le applicazioni devono fare attenzione a parametri come delay o jitter che non erano contemplati dalle precedenti architetture. Tutto ciò deve essere realizzato con un occhio di riguardo all'efficienza dell'implementazione, in quanto le moli di dati da trattare diventano rilevanti.

Si vuole innanzi tutto che un protocollo eviti di porre inutili vincoli al progettista, ritardando al massimo le decisioni ingegneristiche.

Le funzioni di un protocollo si possono dividere in due raggruppamenti:

- **Manipolazione dati**: criptazione, spostare i dati da/allo spazio d'indirizzamento dell'applicazione, formattazione di presentazione (codifiche e decodifiche che tengono conto di diversi possibili formati).

- **Controllo del trasferimento**: controllo del flusso dati onde evitare congestioni, rilevamento problemi di trasmissioni in rete, multiplexing

dei flussi dati, timestamping per i protocolli real-time, framing e acknowledgment. Alcuni di tali controlli sono integrati alle funzioni di manipolazione (controlli in-band).

Da un'analisi fatta da D.Clark e L.Tennenhouse emerge che il codice per implementare le funzioni di controllo è tutto sommato snello, mentre i passi di elaborazione dei dati possono essere assai costosi. In particolare, si punta il dito sulle conversioni di presentazione (si pensi ad esempio, alla decodifica da eseguire su un flusso di comunicazione video H.261).

Sottolineo quindi l'importanza delle conversioni di presentazione che permettono di aumentare molto la generalità di un'applicazione. Abbiamo, infatti, appena visto nel precedente capitolo che vogliamo poter trasmettere con diversi tipi di codifiche a seconda delle capacità dei coinvolti nella conferenza, e della larghezza di banda di trasmissione a disposizione.

Bisogna quindi far sì che sia possibile fare le conversioni di presentazione nell'applicazione stessa in modo tale da mantenere i dati sempre nello stesso spazio d'indirizzamento, evitando copie inutili.

Altro punto da non trascurare è il riordino e la perdita dei dati. Infatti, può essere difficile nelle applicazioni real time eseguire la conversione di presentazione, in quanto essa non può avere luogo immediato. Questo perché prima, bisogna porre rimedio ad un ordine sbagliato dei dati o ad una loro perdita.

Per permettere un'elaborazione in presenza di dati fuori ordine, il tipico approccio è di introdurre dei punti di sincronizzazione nel flusso (da essi si può ripartire nel caso di una perdita). Altro sistema è far sì che la conversione sia relativa ad un solo pacchetto. Tutto questo per permettere all'applicazione di procedere senza dover richiedere una ritrasmissione, che nel caso delle applicazioni real-time, avrebbe poco senso.

L'applicazione deve capire la conversione effettuata dalla presentazione, in quanto altrimenti essa non sarebbe in grado di mettere in relazione i dati persi dalla rete con i corrispondenti dati a livello dell'applicazione.

D.Clark e L.Tennenhouse propongono quindi che l'applicazione rompa i dati in adatti aggregati denominati Application Data Units (ADUs). Per capire veramente cosa siano tali ADU bisogna descriverne le caratteristiche.

Le ADUs prendono il posto dei pacchetti e possono essere spezzettate per la trasmissione (per poi venire ricomposte al ricevente). **Il punto fondamentale è che le ADU possono essere processate fuori ordine** (ma solo se complete); è quindi possibile determinare la posizione originale di un'ADU rispetto le altre. Le dimensioni di un'ADU non devono essere troppo grandi, per non fare alzare troppo la probabilità che ne venga perso un pezzo in trasmissione (il che determinerebbe la perdita di tutta l'ADU). Esiste anche un limite inferiore alla loro grandezza in quanto a seconda dell'applicazione, esisterà una minima quantità di dati per procedere all'elaborazione. Questo principio di progetto si chiama **Application Level Framing** (ALF).

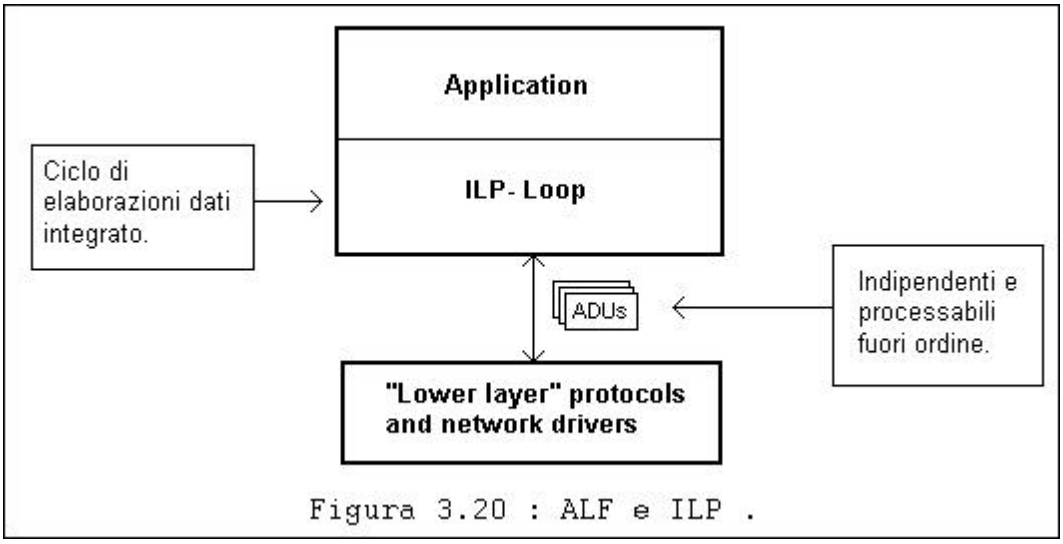

Figura 3.20 : ALF e ILP .

La stratificazione dei protocolli è molto potente in quanto consente, ad un certo livello, di disinteressarsi del sottostante, e questo facilita enormemente l'implementazione. Il problema è che ciò può entrare in conflitto con l'efficienza del sistema. Come alternativa viene quindi introdotto **Integrated Layer Processing** (ILP), che cattura l'idea che il protocollo deve essere strutturato in modo tale da permettere al progettista la possibilità di realizzare tutti i passi di manipolazione, integrati in uno o due cicli di trattamento dati (ILP loop), invece di fare tali passi serialmente come viene fatto di solito oggi. Sfortunatamente i protocolli tradizionali spesso

impongono dei vincoli di precedenza che limitano l'opportunità di una tale ottimizzazione.

Per permettere la realizzazione di ILP, l'architettura del protocollo deve essere tale da consentire che le interazioni tra i passi di elaborazione (sia di manipolazione dati sia di controllo del trasferimento) non interferiscano con la loro integrazione. La conclusione è che un tale protocollo deve minimizzare i vincoli di precedenza imposti all'implementazione. Un'architettura a vincoli ridotti dovrebbe anche facilitare implementazioni pipelined.

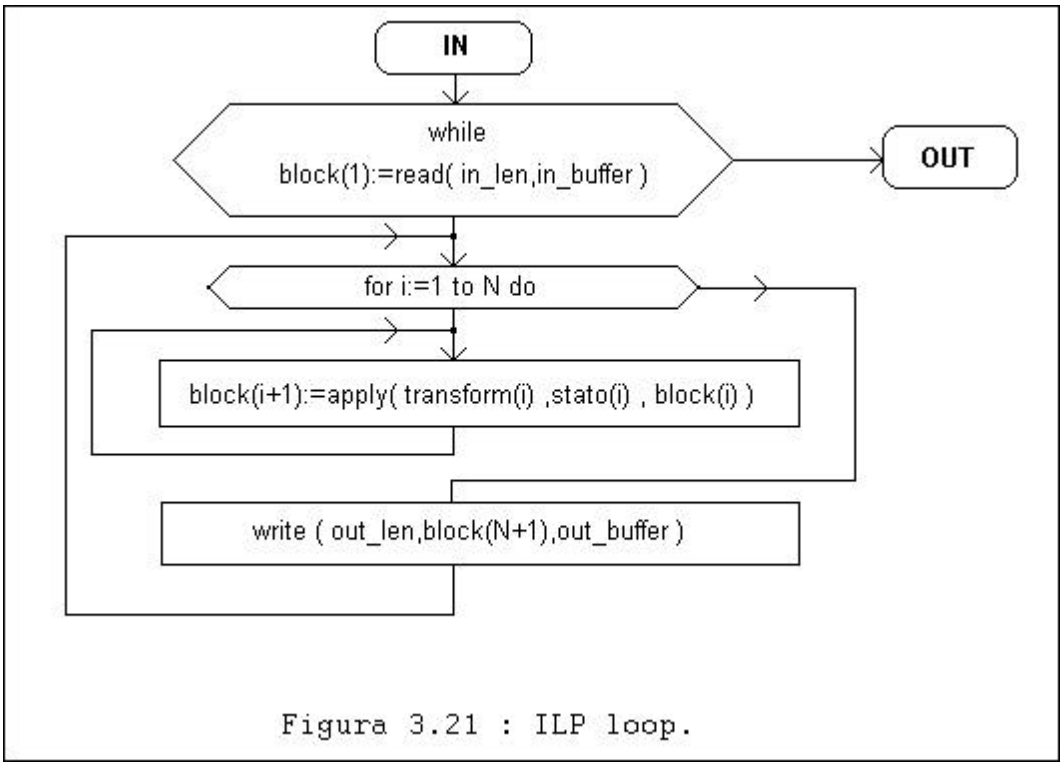

Figura 3.21 : ILP loop.

Application Level Framing alla luce di tutto questo, diventa una via per arrivare a Integrated Layer Processing, attraverso una comune struttura dati (ADU) su cui sono definite le manipolazioni.

Questa discussione non deve essere intesa nel senso che la stratificazione mal si adatta alla progettazione di un protocollo, bensì il suo ruolo principale è l'isolamento semantico di moduli funzionali.

Il livello di rete e i livelli sottostanti, possono operare senza interessarsi del significato dei simboli introdotti dai livelli superiori.

La stessa cosa però non si può dire per il livello di trasporto e per i livelli soprastanti (in particolare per le applicazioni real-time), infatti, qui, tipi di organizzazione differente possono meglio adattarsi alla decomposizione funzionale di uno specifico livello.

Il grosso svantaggio cui può portare un approccio ILP, è che può portare a progetti molto complessi in cui ogni pila di protocolli può avere la propria particolare implementazione! Tuttavia D.Clark e L.Tennenhouse credono che esistano degli approcci minimizzino gli effetti di un tale inconveniente.

Le conclusioni sono quindi che RTP ha una struttura tale da consentire di realizzare tutto questo in cui ad esempio non c'è completa indipendenza fra l'header introdotto da RTP e il payload header introdotto per la codifica H.261.

3.12 Conclusioni su RTP.

Abbiamo visto che RTP è un protocollo molto potente e versatile per la trasmissione di dati in rete con caratteristiche real-time.

Questo non vuol dire che assicura che i pacchetti siano consegnati in tempo reale, bensì trasporta dei dati con proprietà temporali.

Sebbene RTP si appoggi a un servizio di trasporto non affidabile, esso fornisce gli appigli necessari a rendere la trasmissione affidabile, inoltre tramite la misura del delay e del jitter consente di controllare il livello di congestione della rete.

Io ho esaminato RTP con l'utilizzo di UDP, ma in realtà esso può venire utilizzato con un qualsiasi servizio di trasporto, basta che quest'ultimo provveda ad effettuare il framing dai dati, cioè la scomposizione di dati in pacchetti. Notare che spetta a questo servizio fornire la lunghezza dei pacchetti.

RTP è anche, tutto sommato, leggero dal punto di vista dell'overhead introdotto sul traffico di rete. Bisogna dire che sono comunque in corso di studio tecniche per ridurlo ulteriormente. Tali tecniche non sono tuttavia

essenti da critiche anche se personalmente penso che possano trovare impiego effettivo soprattutto per sistemi wireless.

Abbiamo inoltre visto che per sincronizzare sorgenti provenienti da uno stesso terminale o da terminali diversi, bisogna affidarsi ai timestamp-NTP.

Ricordo infine che se si usa RTP in congiunzione con H.323, anche se i payload type vengono contrattati via H.245, non bisogna fare il render senza verificare il campo PT, in quanto questo può venire usato per indicare pacchetti speciali per servizi che qui non ho descritto. I pacchetti con PT diverso vanno quindi ignorati.

Altro punto cui avevo già accennato è che RTP è stato per lo più implementato entro le applicazioni. Purtroppo a causa del fatto che RTP è strettamente accoppiato all'applicazione, bisogna dire che un'eventuale libreria RTP perde di significato, e di fatto vere e proprie librerie non si trovano, al più esistono alcuni strumenti RTP che possono essere utilizzati per trarne alcuni moduli (vedremo alcuni di questi strumenti nel cap.5).

RTP ha trovato ormai largo impiego sia perché viene utilizzato non solo nella videoconferenza (H.323) ma anche da altre applicazioni come lo streaming real-time (vedi RTSP). RTP ha già dimostrato quindi di essere un protocollo potente e flessibile; ha cioè raggiunto una certa maturità.

Riferimenti:

D.D.Clark , D.L.Tennenhouse “Architectural considerations for a new generations of protocols”
Computer communications rewiew , Vol 20 SEPT. ’90.

B.Ahlgren, P.Gunningberg, K Moldeklev.“Increasing Communication Performance with a Minimal–Copy Data Path Supporting ILP and ALF”
www.sics.se.

RFC[1889] RTP: A transport protocol for real-time applications.
H.Schulzrinne, S. Casner , R. Frederick, V. Jacobson.
www.freenic.net

RFC[1890] RTP: Profile for audio and video conferences with minimal control
H.Schulzrinne, S. Casner , R. Frederick, V. Jacobson.

RFC[2032] RTP: Payload format for H.261 Video Streams
T.Turletti, C.Huitema.

RFC[2190] RTP: Payload format for H.263 Video Streams
C.Zhu.

(altri documenti d’interesse non strettamente pertinenti).

RFC[2508] Compressing IP/UDP/RTP headers for low speed serial links.
S.Casner, V.Jacobson.

RFC[2326] Real Time Streaming Protocol (RTSP).
H.Schulzrinne, A. Rao, R. Lan phier.

RFC[1321] The MD5 Message-Digest Algorithm

RFC[2437] Cryptografy Specifications Version 2.0

“Virtual Realty Transfer Protocol” (VRTP)
http://www.stl.nps.navy.mil/~brutzman/vrtp/