UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Facoltà di Ingegneria

Corso di Laurea Triennale in
Ingegneria Informatica e Automatica

Tesi di Laurea

Realizzazione di un VoIP gateway su piattaforma embedded Linux

---

Relatore
Prof. Daniele Frigioni

Candidato
Davide Di Gesualdo

Correlatore
Ing. Fabio Antonini

Anno Accademico
2006-2007

Alla mia famiglia,

al ricordo di nonno Antonio

RINGRAZIAMENTI

Ricordo quando, da piccolo, passavo ore e ore davanti allo schermo del televisore, a scrivere linee di codice Basic sulla tastiera del Commodore 64, senza conoscere il significato di tutte quelle istruzioni, e codici; e tutto per vedere apparire al centro della tv una pallina colorata! Ricordo anche quando papà tornò a casa con il primo x86 (era un notebook 386, schermo EGA, con 1MB di RAM e 20MB di hard disk!), e già ero affascinato da quella scatola con i bottoni, che "chissà come fa a funzionare"…

Ringrazio, innanzitutto, la mia famiglia e "la mia piccola" Isabella (che mi sopporta da più di sei anni!): senza loro non so cosa farei! La cosa che più mi dispiace è che nonno Antonio non sia qui per sostenermi, come ha sempre fatto; ma io so che da lassù lui è il più contento di tutti per me.

Un ringraziamento dovuto (ma soprattutto voluto!) va ai fantastici "ragazzi" della Kasko Networks, che mi hanno supportato (e sopportato!) nello sviluppo della tesi, in particolar modo a Fabio, il mio mentore all'interno dell'azienda, sempre alle prese con mille progetti, ma sempre disponibile per ogni spiegazione, e a Italo, il nuovo

acquisto della Kasko, che mi ha assistito nel primo periodo di tirocinio; grazie anche al neopapà "teatino" Antonello, a Carlo e a Pietro.

Ovviamente ringrazio il Prof. Frigioni, relatore della tesi, disponibile, cordiale e simpatico come pochi altri lo sono.

Ringrazio i miei amici, quelli che conosco da una vita, e di cui mi potrò sempre fidare: Daniele (meglio conosciuto come "miocuggino!"), Dario, Fabio e Domenico; e quelli che conosco da qualche anno, ma con cui ho condiviso tante di quelle esperienze da poterli quasi considerare fratelli. Mi riferisco ai "ragazzi di CasaPianola" Alessio e Andrea, e Matteo (ola amigo!), il mio barista di fiducia!

Infine, ringrazio Massimo Cretarola, l'"Ing. per eccellenza", che ne sa sempre una più del diavolo! (ma come fai!?)

Un grazie di cuore va a tutte le persone che hanno sempre creduto in me, nelle mie potenzialità, e che mi sono state vicine anche nei momenti peggiori.

INDICE

RINGRAZIAMENTI **III**

INDICE **V**

PREMESSA **X**

1. INTRODUZIONE **1**

1.1 I principali vantaggi del Voice over IP *4*

1.2 Organizzazione della Tesi *7*

2. VOIP E QUALITÀ DEL SERVIZIO **9**

2.1 VoIP *10*

2.1.1 Ritardo/Latenza 10

2.1.1.1 Il ritardo di propagazione 11

2.1.1.2 Il ritardo di gestione 11

2.1.1.3 Il ritardo di accodamento 12

2.1.2 Tremolio o Jitter 13

2.1.3 La modulazione PCM 14

2.1.4 Eco 15

2.1.5 Compressione della voce 17

2.1.5.1 Gli standard di codifica della voce 19

2.1.5.2 La valutazione soggettiva MOS 20

2.1.6 Perdita di pacchetti 22

2.1.7 Rilevamento dell'attività vocale 23

2.1.8 Conversione da digitale ad analogico 24

2.1.9 Codifica Tandem 25

2.1.10 Confronto fra chiamate: PSTN e IP 28

2.2 Qualità del Servizio *32*
2.2.1 Limiti legati all'ampiezza di banda 34
2.2.2 Il protocollo cRTP 36
2.2.3 Accodamento 39
2.2.3.1 Accodamento WFQ 40
2.2.3.2 Accodamento personalizzato 42
2.2.3.3 Accodamento a priorità 43
2.2.3.4 Accodamento CB-WFQ 44
2.2.3.5 Accodamento a bassa latenza (coda a priorità con CB-WFQ) 45
2.2.4 Classificazione dei pacchetti 46
2.2.4.1 Precedenza IP 47
2.2.4.2 Politiche di instradamento 49
2.2.4.3 RSVP 51
2.2.4.4 IP RTP Reserve 53
2.2.4.5 IP RTP Priority 53
2.2.5 Politiche di traffico 54
2.2.6 Frammentazione 55
2.2.6.1 MCML PPP 57
2.2.6.2 FRF.12 57
2.2.6.3 IP MTU e MTU 58
2.2.7 Trasporto ad alta velocità 58
2.2.7.1 POS 59
2.2.7.2 IP e ATM 60
2.2.8 Accodamento ad alta velocità 61
2.2.8.1 WRED 62

3. RTP: REAL-TIME TRANSFER PROTOCOL **63**

3.1 Cenni sul protocollo UDP *64*
3.2 Caratteristiche generali dell'RTP *66*
3.3 Il funzionamento del miscelatore *67*
3.4 L'intestazione dei pacchetti RTP *68*

3.5 Il protocollo RTCP *72*
3.6 Il formato dei pacchetti RTCP *73*
3.6.1 I messaggi RR 75
3.6.2 I messaggi SR 78
3.6.3 I messaggi SDES 79
3.6.4 I messaggi APP e BYE 80
3.7 Il protocollo RTP in un applicativo VoIP *81*

4. SISTEMI EMBEDDED LINUX **84**

4.1 Classificazione dei sistemi embedded Linux *86*
4.1.1 Grandezza 86
4.1.2 Vincoli temporali 87
4.1.3 Utilizzo della rete 87
4.1.4 Interazione con l'utente 88
4.2 Metodologie di progettazione e di implementazione *88*
4.2.1 Creare un sistema Linux target 88
4.2.2 Scegliere gli strumenti di sviluppo 89
4.2.3 Sviluppare per sistemi embedded 89
4.2.4 Configurare la rete 90
4.3 Setup e configurazione del target *90*
4.3.1 Hardware supportato: processore 90
4.3.2 Hardware supportato: bus e interfacce 96
4.3.3 Hardware supportato: I/O 98
4.3.4 Hardware supportato: storage 102
4.3.5 Hardware supportato: networking 103
4.3.6 Setup del kernel 105
4.3.7 Contenuto del Filesystem radice 107
4.3.8 Scelta del bootloader 109
4.4 Strumenti di sviluppo *114*
4.4.1 Workspace 114
4.4.2 Toolchain di sviluppo cross-piattaforma GNU 116

4.4.3 Librerie C alternative 117
4.4.4 Altri linguaggi di programmazione 119
4.4.5 Ambienti di sviluppo integrati 119
4.4.6 Emulatori di terminale 120
4.5 Configurazione della rete *121*

5. PIATTAFORMA DI SVILUPPO **125**

5.1 Piattaforma hardware *126*
5.1.1 Intel© IXP425 Network Processor 128
5.1.2 Flash/Boot ROM 128
5.1.3 Porta WAN ADSL 129
5.1.4 Interfaccia PCI 129
5.1.5 Interfacce Ethernet 130
5.1.6 Porte voce 130
5.1.7 Porte USB 131
5.1.8 Porta seriale 131
5.2 Piattaforma software *131*
5.2.1 Intel© IXP400 Software v1.4 131
5.2.2 Intel© IXP400 DSP Software v2.6.2 136
5.2.2.1 Architettura dell'Intel© IXP400 DSP Software v2.6.2 138

6. TOOLCHAIN **147**

6.1 Wireshark *148*
6.2 MontaVista Linux 3.1 *150*
6.3 Source Navigator *151*
6.4 NoMachine NX *153*
6.5 Minicom 2.0 *154*

7. INTEGRAZIONE KRTPD – IXP400 DSP SOFTWARE **156**

7.1 Premessa *156*

7.2 Panoramica sul "krtpd" *158*

7.3 Porting del krtpd verso il kernel 2.4 *159*

7.4 Preparazione di kernel e RAM disk image *160*

7.4.1 Cross-compilazione del kernel 160

7.4.2 Generazione del RAM disk image 162

7.5 Inizializzazione e configurazione IXDPG425 *163*

7.6 Adattamento krtpd *167*

7.6.1 Strutture dati e funzioni di supporto 168

7.6.2 Trasmissione 170

7.6.3 Ricezione 173

7.6.4 Ulteriori modifiche 173

7.7 Test d'utilizzo *174*

7.8 Conclusioni e possibili sviluppi futuri *181*

GLOSSARIO **182**

BIBLIOGRAFIA **203**

PREMESSA

Gli ultimi decenni hanno vissuto cambiamenti radicali nei modi e nei mezzi di comunicazione. È Internet, la rete delle reti, l'ambito in cui si assiste maggiormente ad una diversificazione delle richieste di nuovi servizi da parte degli utenti, in particolar modo della telefonia su IP. La grande attenzione attirata dallo sviluppo della tecnologia VoIP (Voice over IP) ha dimostrato come anche il mondo del business sia decisamente in sintonia con tutto ciò che permette la riduzione dei costi e l'aumento della qualità del servizio. La comunicazione su IP avviene tramite degli applicativi, detti VoIP applications, i quali possono utilizzare protocolli di sessione/trasmissione proprietari (come, ad esempio, il sempre più noto Skype), oppure protocolli OpenSource. Questi applicativi, nella maggior parte dei casi, lavorano in user space, il che può compromettere la trasmissione dei dati. Poiché da sempre tutti i sistemi operativi danno maggiore spazio di esecuzione ai processi in kernel space, si deduce che gli applicativi VoIP attuali hanno dei limiti nelle prestazioni.

La tesi si propone di adattare il driver “krtpd”, precedentemente sviluppato da Italo Armenti nella tesi dal titolo “Sviluppo di uno stack RTP in Kernel Space per applicazioni VoIP time critical in ambiente Linux”, al residential gateway platform della Intel IXDPG425.

Per conseguire gli obiettivi preposti è stato necessario modificare il comportamento del krtpd, nonché di parte del modulo governante il DSP (Digital Signal Processor) dell'IXDPG425. In particolare, il thread di trasmissione è stato rimpiazzato da una funzione, esportata nel kernel (quindi resa disponibile al DSP), che accettasse pacchetti dal DSP, ricostruisse pacchetti RTP e li inviasse in rete. Il thread di ricezione è stato modificato in modo da far ricostruire un pacchetto comprensibile dal DSP (a partire dal pacchetto RTP in ingresso); è stata poi introdotta la chiamata ad una funzione del modulo DSP, in grado di processare e riprodurre i dati audio contenuti nel pacchetto ricevuto. D'altra parte, nel modulo del DSP è stata necessaria la modifica dell'inizializzazione di alcune strutture dati (proprie del modulo), e delle modalità d'accesso alla rete. Lo scopo ultimo del lavoro è quello di ottenere comunicazioni VoIP i cui parametri prestazionali siano compatibili, o almeno confrontabili, con quelli conseguiti dalla sopraccitata tesi. In particolar modo ci si aspettano valori di jitter medio compresi tra 0.01 e 0.1 ms (il jitter è la differenza tra l'istante in cui arriva un pacchetto e l'istante in cui il pacchetto stesso è atteso).

I risultati ottenuti dai test del driver hanno dimostrato pieno allineamento con i valori attesi, nonostante le ridotte prestazioni di un sistema embedded quale l'IXDPG425: il jitter medio rilevato si è attestato circa a 0.09 ms.

Capitolo 1

INTRODUZIONE

Le tradizionali reti voce, a commutazione di circuito, PSTN (Public Switched Telephone Network) stanno ormai lasciando il posto alla rete dati, in cui viene utilizzato il VoIP come tecnologia per la comunicazione.

I limiti presentati dalla rete PSTN sono molteplici:

- I dati hanno superato la voce come traffico principale in molte reti realizzate per la voce.

  La trasmissione dati attualmente opera su reti realizzate per trasportare in modo efficiente la voce. Ma i dati hanno caratteristiche differenti, ad esempio un uso variabile dell'ampiezza di banda e la necessità di impiegare un'ampiezza di banda elevata.

  Ben presto le reti per la voce opereranno su reti realizzate secondo un approccio basato su dati. Il traffico verrà differenziato sulla base delle applicazioni e non dei circuiti

fisici. Per creare le reti ad alta velocità necessarie per trasportare tutti questi dati aggiuntivi verranno usate nuove tecnologie (Fast Ethernet, Gigabit, Ethernet e reti ottiche).

- Le reti PSTN non sono in grado di creare e fornire le funzionalità con la dovuta rapidità.

  Dato l'aumento della competizione dovuto a una sostanziale apertura del mercato delle telecomunicazioni, i LEC (Local Exchange Carrier) stanno cercando nuovi metodi per mantenere la loro attuale clientela. Il metodo principale per conservare i propri clienti consiste nell'offrire sempre più servizi e applicazioni.

  Le reti PSTN si basano su un'infrastruttura in cui solo il produttore del dispositivo è in grado di sviluppare le applicazioni per tale dispositivo. Questo significa che vi è un solo sviluppatore di soluzioni per tutte le esigenze. E' molto difficile che una sola società possa rispondere a tutte le esigenze dei propri clienti. Un'infrastruttura più aperta in cui i produttori possono fornire applicazioni, consente di sviluppare soluzioni e applicazioni più creative. Inoltre è impossibile con l'attuale architettura consentire a più produttori di scrivere nuove applicazioni per reti PSTN. Si immagini come sarebbe il mondo oggi se i produttori come Microsoft impedissero agli altri produttori di scrivere applicazioni per il proprio software.

- Le informazioni Dati/Voce/Video non possono convergere sulle attuali reti PSTN.

Utilizzando unicamente la linea analogica disponibile in molte abitazioni, è impossibile avere un accesso ai dati (a Internet), un accesso telefonico e un accesso video con un modem da 56kbit/s. Per consentire questa convergenza è necessario un accesso a banda larga e ad alta velocità come ad esempio una linea digitale (DSL – Digital Subscriber Line) via cavo o wireless. Dopo aver risolto gli ultimi problemi di ampiezza di banda, si può finalmente raggiungere una convergenza per l'utenza domestica. Nelle parti più interne delle reti PSTN questa convergenza è già iniziata.

- L'architettura realizzata per la voce non è sufficientemente flessibile per la trasmissione dei dati.

  Poiché i canali Bearer (canali B a circuiti T), il controllo di chiamata (Q.931) e la logica (le applicazioni) sono strettamente collegati ad un'unica piattaforma chiusa, è impossibile apportare quelle piccole modifiche che migliorerebbero la qualità audio.

Inoltre, le chiamate a commutazione di circuito, per definizione, richiedono l'esistenza di un circuito dedicato a 64 kbit/s fra i due telefoni. Indipendentemente dal fatto che il chiamante o il chiamato stiano parlando, questa connessione a 64 kbit/s non può essere utilizzata da altri. Questo significa che la compagnia telefonica non può utilizzare questa ampiezza di banda per altri scopi e deve fatturarla a chi consuma queste risorse.

Al contrario, le reti per dati offrono la possibilità di utilizzare l'ampiezza di banda solo nel momento in cui è richiesta. Questa differenza che può anche sembrare trascurabile, è uno dei principali vantaggi delle reti per voce basate su pacchetti.

1.1 I principali vantaggi del Voice over IP

Uno dei principali vantaggi del fatto di combinare le reti vocali e le reti dati è il risparmio economico. Se si considera un puro confronto tariffario in termini di costo al minuto, i risparmi ottenuti passando al sistema Voice Over IP potrebbero essere insufficienti per giustificare la spesa per l'installazione del servizio .

Il risparmio può variare a seconda dell'area geografica. In molti paesi, ad eccezione del nord America, un confronto diretto fra Voice Over IP e la tradizionale rete PSTN (considerando che in alcuni paesi il costo di una chiamata locale può essere di circa un dollaro al minuto) giustifica largamente l'investimento.

In nord America molte grandi aziende pagano anche meno di 3 centesimi al minuto per chiamate a lunga distanza entro gli Stati Uniti. Per queste aziende è difficile che la creazione di una nuova infrastruttura si ripaghi in tempi rapidi, a meno che non intervengano altri fattori diversi dal puro confronto tariffario.

Ad esempio, per le aziende, l'unificazione fra le reti voce e le reti dati può significare l'acquisto di un minor numero di linee PSTN. Inoltre un'infrastruttura IP richiede poche aggiunte, spostamenti e modifiche

rispetto ad una rete dati e vocale tradizionale. Questo avviene perché, con un'unica infrastruttura, si possono usare funzionalità quali DHCP (Dynamic Host Configuration Protocol) che consente ad un dispositivo (che può essere un PC o un apparecchio telefonico IP) di ricevere dinamicamente un indirizzo IP (ovvero l'indirizzo IP non deve essere configurato in modo statico nel dispositivo). Dunque, ad esempio, se si ha un apparecchio telefonico IP configurato con DHCP, si può spostare il telefono nel punto desiderato e mantenere comunque lo stesso numero. E' come spostare un computer portatile da un ufficio ad un altro, connettendosi sempre allo stesso server di rete.

Molte grosse aziende hanno calcolato costi di svariate centinaia di dollari solo per spostare un telefono (il costo può essere dovuto al prezzo della manodopera più il costo di configurazione del centralino). Tali costi sono assenti in un'infrastruttura IP poiché si deve semplicemente impostare il profilo dell'apparecchio telefonico IP, che successivamente potrà essere posizionato in qualsiasi punto della rete.

Un ulteriore vantaggio di Voice Over IP è la possibilità di avere un unico dipartimento dei servizi informatici che supporta sia la rete voce che la rete dati (in quanto le due reti si riducono a un'unica entità), anche se questo potrebbe inizialmente provocare tensioni fra le due infrastrutture.

Un vantaggio del Voice Over IP, spesso sottovalutato dalle aziende e dai fornitori di servizi, è il fatto che non è più necessario utilizzare vari strumenti, dato che l'infrastruttura è comune. Fra questi strumenti vi sono le porte fisiche per servizi come i messaggi vocali. In una rete

locale a commutazione di circuito, il servizio di messaggistica vocale viene venduto sulla base del numero di caselle e del numero di porte fisiche necessarie per supportare un determinato numero di utenti simultanei. Con Voice Over IP le porte fisiche a commutazione di circuito non sono più necessarie. Il server dei messaggi vocali deve avere semplicemente una connessione IP (che può essere Ethernet, ATM - Asynchronous Transfer Mode e così via).

Inoltre Voice Over IP consente di porre i sistemi di gestione dei messaggi vocali su piattaforme standard (ad esempio PC o macchine Unix). Quando una determinata funzionalità viene inserita in una piattaforma standard, si può prevedere una notevole riduzione nei prezzi. Ad esempio, i fornitori di sistemi di messaggi vocali richiedono da 0.5 $ a 1.5 $ per MB di spazio su disco fisso poiché utilizzano meccanismi proprietari di formattazione e memorizzazione delle chiamate vocali. Ma il prezzo medio dei dischi fissi, presso qualsiasi rivenditore di PC, è dell'ordine di 3-4 centesimi di $ per 1 MB .

Cosa dire poi del fatto che il server dei messaggi vocali potrebbe essere lo stesso dei messaggi di posta elettronica e si potrebbe decidere di prelevare la posta vocale via telefono o tramite un client di posta elettronica. Il personale, che si trova frequentemente a lavorare all'esterno dell'azienda, apprezzerà moltissimo la possibilità di prelevare messaggi vocali e rispondere in modo elettronico, o di inoltrare i messaggi vocali a un gruppo. Questa tecnologia esiste già e sarà ben presto ampiamente utilizzata nelle aziende e nelle reti dei fornitori di servizi.

1.2 Organizzazione della tesi

La tesi è strutturata in 6 capitoli:

- Nel Capitolo 1, c'è la parte introduttiva che illustra l'importanza ed i vantaggi dell'avvento del VoIP.
- Nel Capitolo 2 si scende nel dettaglio per quanto riguarda il VoIP. Se ne descrivono i problemi ancora aperti ed alcune soluzioni. Inoltre si introduce il concetto di qualità del servizio (QoS): gli strumenti a disposizione per la sua attuazione, le funzionalità marginali, le politiche di traffico, cosa avviene nelle parte backbone della rete e le regole empiriche da rispettare.
- Nel Capitolo 3 si descrive il protocollo RTP, le sue caratteristiche, la sua importanza nelle applicazioni multimediali in generale ed il suo utilizzo specifico nel VoIP.
- Nel Capitolo 4 si descrivono, in linea generale, i sistemi Linux embedded: cosa sono, la loro importanza, le caratteristiche e la loro configurazione; questo capitolo è necessario, in quanto l'obiettivo della tesi è proprio quello di calare un modulo RTP in un sistema embedded Linux.
- Nel Capitolo 5 viene continuato il discorso sui sistemi embedded; si parlerà, in particolare, delle piattaforme hardware e software utilizzate durante lo sviluppo della tesi: se ne descriveranno la struttura, le linee generali di funzionamento e le particolarità.

- Nel Capitolo 6 viene presentato l'insieme delle applicazioni utilizzate durante lo sviluppo della tesi.
- Nel Capitolo 7 viene, infine, illustrato il lavoro di sviluppo e messa a punto del modulo RTP in kernel space, pronto per essere utilizzato nel sistema embedded di cui al Capitolo 5; nella parte finale del capitolo, inoltre, vengono descritti ed illustrati i test effettuati sul sistema, nonché i risultati ottenuti e le conclusioni sui possibili sviluppi futuri.
- Sono inoltre presenti come appendice, il glossario dei termini e delle sigle utilizzate e la bibliografia dei testi da cui si è preso spunto per la redazione della tesi.

Capitolo 2

VOIP E QUALITÀ DEL SERVIZIO

Questo capitolo descrive nel dettaglio i problemi che presenta la tecnologia Voice Over IP e spiega come possono influenzare le reti a pacchetti. Tali problemi riguardano:

- Ritardo/latenza
- Tremolio
- La modulazione PCM
- Eco
- Compressione della voce
- Perdita di pacchetti
- Rilevamento dell'attività vocale
- Conversione da digitale ad analogico
- Codifica tandem
- Un confronto fra chiamate telefoniche a centralino IP

Per quanto riguarda la Qualità del Servizio (Qos – Quality of Service), si fa riferimento a CoS (Class of Service) ed a ToS (Type of Service).

L'obiettivo principale della classe di servizio e del tipo di servizio è quello di ottenere le caratteristiche di ampiezza di banda e di latenza richieste da una determinata applicazione.

Una classe di servizio permette di raggruppare i vari flussi di pacchetti, ognuno dei quali presenta requisiti di latenza e ampiezza di banda differenti. Il tipo di servizio è un campo dell'intestazione IP (Internet Protocol) che offre la possibilità di raggiungere una determinata classe di servizio.

Sono disponibili vari strumenti che consentono di ottenere la qualità del servizio necessaria per un utente o un'applicazione; in questo capitolo si spiega quando usarli e gli eventuali problemi che possano presentarsi nel loro utilizzo.

2.1 VoIP

2.1.1 Ritardo/latenza

In Voice Over IP per *ritardo* o *latenza* si intende il tempo necessario perché la voce, uscita dalla bocca del parlante, raggiunga l'orecchio del destinatario. In un router, la latency è il tempo che intercorre tra la ricezione e la rispedizione di un pacchetto dati.

Nelle attuali reti telefoniche esistono tre tipi di ritardi: il *ritardo di propagazione*, il *ritardo di serializzazione* ed il *ritardo di gestione*. Il ritardo di propagazione è provocato dalla velocità della luce nelle reti in fibra ottica o in rame. Il ritardo di gestione (o ritardo di elaborazione) è definito da varie cause (trasformazione in pacchetti, compressione e

commutazione dei pacchetti) ed è provocato dai dispositivi che inoltrano la trama lungo la rete. Il ritardo di serializzazione è il tempo necessario per inserire i dati nell'interfaccia. Il ritardo di serializzazione non viene descritto in dettaglio, in quanto la sua influenza sul ritardo globale è relativamente trascurabile.

2.1.1.1 Il ritardo di propagazione

La luce viaggia nel vuoto ad una velocità di 300.000 km/s e gli elettroni viaggiano in un cavo di rame o di fibra ottica a circa 200.000 km/s. Pertanto, una fibra ottica che collega due punti distanti 20.000 km, introduce un ritardo monodirezionale di circa 70ms. Anche se questo ritardo è quasi impercettibile per l'orecchio umano, i ritardi di propagazione, uniti ai ritardi di gestione, possono provocare un degrado della voce maggiormente percepibile.

2.1.1.2 Il ritardo di gestione

Come si è detto in precedenza, i dispositivi che inoltrano la trama nella rete, introducono un ritardo di gestione. I ritardi di gestione possono avere un impatto anche sulle reti telefoniche tradizionali ma rappresentano un problema di maggiore entità negli ambienti a pacchetti. Si tratta dei ritardi dovuti al campionamento, alla codifica e decodifica del parlato, alla creazione, compressione e commutazione dei pacchetti, ecc..

2.1.1.3 Il ritardo di accodamento

Una rete a pacchetti sperimenta altri tipi di ritardi. Due di questi sono il tempo necessario per trasferire un pacchetto alla coda di output (ritardo di commutazione del pacchetto) ed il ritardo di accodamento.

Quando i pacchetti vengono memorizzati in una coda a causa di una congestione dell'interfaccia di uscita, il risultato è un ritardo di accodamento. Il ritardo di accodamento si verifica quando vengono inviati più pacchetti di quanti l'interfaccia sia in grado di gestire in un determinato intervallo. Se possibile si deve mantenere questo fattore minore di 10ms, utilizzando il metodo di accodamento ideale della propria rete.

La raccomandazione G.114 ITU-T specifica che per ottenere una buona qualità della voce si dovrebbe ottenere un ritardo monodirezionale punto-a-punto non superiore a 150ms, come illustrato in *Figura 2.1*.

FIGURA 2.1: Un ritardo punto a punto

Come si può vedere nella Figura 2.01, alcune forme di ritardo sono più ingenti, anche se sono accettabili, in quanto non esistono alternative. Ad esempio, in una trasmissione via satellite sono necessari circa 250ms per raggiungere il satellite ed altri 250ms per attendere il

segnale in arrivo dal satellite. Questo introduce un ritardo totale di 500ms. Anche se la raccomandazione ITU-T indica che questo ritardo fuoriesce dall'intervallo accettabile per la qualità vocale, al giorno d'oggi molte conversazioni si svolgono su collegamenti via satellite. Per questo la qualità della voce spesso viene definita da quanto gli utenti possono accettare ed utilizzare.

In una rete non gestita e congestionata, il ritardo di accodamento può raggiungere i 2 secondi (o in alternativa, può portare alla perdita di pacchetti). Questo ritardo è inaccettabile per qualsiasi rete vocale. Il ritardo di accodamento è solo uno dei componenti del ritardo punto-a-punto.

2.1.2 Tremolio o jitter

In breve, il *tremolio* (jitter) è prodotto dalla variabilità nel tempo di arrivo dei pacchetti. Il tremolio si verifica solo nelle reti a pacchetti. In un ambiente vocale a pacchetti si attende di trasmettere in modo affidabile i pacchetti di voce ad intervalli regolari (ad esempio una trama ogni 20ms). Questi pacchetti vocali possono essere ritardati dalla rete e quindi possono non arrivare alla stazione ricevente ad intervalli regolari (ad esempio potrebbero non essere ricevuti ogni 20ms). La differenza, fra il momento in cui un pacchetto è atteso ed il momento in cui viene ricevuto, si chiama tremolio (jitter). Alcuni pacchetti potrebbero essere ritardati dalla rete e potrebbero non arrivare alla stazione ricevente con la stessa spaziatura temporale.

Diviene allora necessario immagazzinare i pacchetti in un buffer, in attesa che arrivino in tempo per poter essere riprodotti nella corretta sequenza temporale. Il buffer riduce la variabilità del ritardo d'arrivo dei pacchetti. Si noti che il tremolio ed il ritardo totale non sono la stessa cosa, anche se, il fatto di avere elevati livelli di tremolio in una rete a pacchetti, può aumentare il ritardo totale della rete. Questo è dovuto al fatto che, maggiore è il livello di tremolio, maggiori saranno le dimensioni del buffer per compensare la natura imprevedibile della rete a pacchetti.

Se la rete dati viene ben realizzata e si prendono le opportune precauzioni, il tremolio non rappresenta un grande problema ed il buffer non contribuisce in modo significativo al ritardo totale.

Buffer dinamici, che presentano una coda dinamica che si espande o riduce a seconda del tempo di arrivo dei pacchetti RTP, sono il meccanismo migliore per le reti locali a pacchetti. I buffer statici sono necessariamente troppo estesi o troppo ridotti con un degrado nella qualità audio a causa dei pacchetti persi o dell'eccessivo ritardo introdotto.

2.1.3 La modulazione PCM

Mentre la forma analogica è ideale per le comunicazioni interpersonali, non è sufficientemente efficiente per risolvere i problemi derivanti dal rumore presente sulla linea. Nei primi tipi di reti telefoniche, quando la trasmissione analogica doveva passare attraverso amplificatori di segnale, non veniva amplificata solo la voce ma anche il rumore

presente sulla linea. Questo rumore di linea portava spesso a connessioni inutilizzabili.

E' invece molto più facile distinguere dal rumore di linea i segnali dei campioni digitali (costituiti da sequenze di bit uguali a 1 o a 0). Pertanto quando i segnali analogici vengono rigenerati come campioni digitali, si mantiene un segnale chiaro. Quando divennero evidenti questi vantaggi della rappresentazione digitale, le reti telefoniche hanno adottato la modulazione PCM (Pulse Code Modulation).

La modulazione PCM converte i suoni analogici in una forma digitale campionando il suono analogico 8000 volte al secondo e convertendo ogni campione in un codice numerico. Il teorema di Nyquist stabilisce che se si esegue un campionamento di un segnale analogico a una frequenza doppia rispetto alla più elevata frequenza cui si è interessati è possibile ricostruire accuratamente la forma analogica del segnale. Poiché la maggior parte dei contenuti di una conversazione telefonica si trova sotto 4000 Hertz (4 kHz), è necessaria una frequenza di campionamento di 8000 campioni al secondo (con un intervallo di 125μs fra i campioni).

2.1.4 Eco

L'eco è un curioso fenomeno che tutti possono sperimentare con piacere in montagna ma, in una conversazione telefonica, può essere fastidioso o addirittura insopportabile, fino a rendere non intelligibile una conversazione. Il fatto di udire la propria voce mentre si parla è normale e rassicurante. Il fatto invece di ascoltare la propria voce con

un ritardo superiore a 25ms è fastidioso e può arrecare disturbo nella cadenza della conversazione.

In una rete tradizionale, l'eco è normalmente provocato da un errore di impedenza, dovuto alla conversione che avviene nella centrale di rete da quattro fili alla linea locale a due fili. L'eco della rete telefonica PSTN (Public Switched Telephone Network) viene regolato da appositi cancellatori di eco e da un attento controllo degli errori di impedenza nei punti di riflessione.

L'eco ha principalmente due caratteristiche: può essere elevato e può essere lungo; naturalmente, più è presente e più è lungo, più diventa fastidioso.

Le reti telefoniche, nelle aree del mondo in cui viene principalmente usata la voce analogica, impiegano soppressori di eco in grado di rimuovere l'eco, agendo sull'impedenza del circuito. Questo non è il meccanismo migliore per rimuovere l'eco e, infatti, provoca altri problemi. Ad esempio, in una linea dotata di un soppressore di eco, non è possibile utilizzare il sistema ISDN (Integrated Services Digital Network), poiché il soppressore di eco taglia l'intervallo di frequenze impiegato dalla trasmissione ISDN.

Nelle attuali reti a pacchetti è possibile costruire cancellatori di eco nei codecs a basso bit-rate ed inserirli in ogni linea DSP. Nelle implementazioni di alcuni produttori, la cancellazione dell'eco viene eseguita via software. Questo, però, riduce drasticamente i vantaggi della cancellazione dell'eco.

Per comprendere come funzionano i cancellatori di eco è meglio parlare, innanzitutto, delle origini dell'eco.

In questo esempio, si supponga che l'utente A stia parlando con l'utente B. Il parlato dell'utente A si chiamerà G. Quando G trova un errore di impedenza o un'altra situazione che provoca un eco, ritorna all'utente A. L'utente A può udire l'eco con un ritardo di vari millisecondi.

Per rimuovere l'eco dalla linea, il dispositivo utilizzato dall'utente A (il router A) conserva, per un determinato periodo di tempo, un'immagine inversa delle parole dell'utente A (-G). Questo cancellatore di eco ascolta il suono in arrivo dall'utente B e gli sottrae -G per rimuovere ogni eco.

I cancellatori di eco hanno un campo d'azione limitato al tempo di attesa del segnale riflesso, un fenomeno noto con il nome di "coda d'eco".

E' importante configurare il livello appropriato di cancellazione di eco, nel momento in cui si installa il dispositivo Voice Over IP. Se non si configura un livello sufficiente di cancellazione dell'eco, il chiamante potrà udire l'eco durante le conversazioni. Se si configura un eccesso di cancellazione dell'eco, il cancellatore di eco avrà bisogno di più tempo per convergere ed eliminare l'eco.

2.1.5 Compressione della voce

Vengono normalmente utilizzate due varianti di base della modulazione PCM a 64 kbit/s: μ-law e a-law. I metodi impiegati sono simili in

quanto entrambi usano una compressione logaritmica per inserire 12 o 13 bit di qualità PCM lineare in 8 bit mentre le differenze sono costituite da dettagli di compressione relativamente secondari (il metodo μ-law ha un lieve vantaggio prestazionale nel rapporto segnale - rumore a basso livello). Storicamente, in nord America si usa il metodo μ-law mentre in Europa si usa la modulazione a-law. E' importante notare che, quando si esegue una chiamata a lunga distanza, ogni eventuale conversione da μ-law ad a-law è a carico del paese μ-law.

Un altro metodo di compressione molto utilizzato è ADPCM (Adaptive Differentiale Pulse Code Modulation). Un'istanza molto utilizzata di ADPCM è ITU-T G.726 che esegue la codifica utilizzando campioni a 4 bit, con una velocità di trasmissione di 32 kbit/s. A differenza di PCM, i 4 bit non codificano direttamente l'ampiezza della voce ma le differenze di ampiezza ed il tasso di cambiamento di tale ampiezza, impiegando una semplice previsione lineare.

PCM ed ADPCM sono esempi di metodi di codifica/decodifica (codec) a forma d'onda, tecniche di compressione che impiegano le caratteristiche ridondanti della forma d'onda. Negli ultimi 10 - 15 anni sono state sviluppate nuove tecniche di compressione che impiegano nuove conoscenze nel campo della generazione della voce. Queste tecniche sfruttano procedure di elaborazione del segnale che consentono di comprimere la voce inviando solo informazioni parametriche semplificate sulla voce, richiedendo così un'ampiezza di banda minore per trasmettere tali informazioni.

Queste tecniche sono considerate codec di origine ed includono varianti quali LPC (Linear Predictive Code), CELP (Code Excited Linear Prediction) e MP - MLQ (MultiPulse, Multilevel Quantization).

2.1.5.1 Gli standard di codifica della voce

ITU-T standardizza gli schemi di codifica CELP, MP - MLQ PCM ed ADPCM nelle sue raccomandazioni della serie G. I più diffusi standard di codifica della voce per la telefonia e la trasmissione della voce a pacchetti sono i seguenti:

- G.711: descrive la tecnica di codifica della voce PCM a 64 kbit/s descritta in precedenza; la voce codificata G.711 è già nel formato corretto per l'invio digitale alla rete telefonica pubblica o per il passaggio tramite un centralino PBX.
- G.726: descrive la codifica ADPCM a 40, 32, 24 e 16 kbit/s; è anche possibile interscambiare la voce ADPCM fra, da un lato, le reti locali a pacchetti e, dall'altro, la rete telefonica pubblica o il centralino PBX, sempre che il secondo elemento sia dotato di funzionalità ADPCM.
- G.728: descrive una variante a 16 kbit/s a basso ritardo della compressione della voce CELP.
- G.729: descrive una compressione CELP che consente di codificare la voce in flussi da 8 kbit/s; esistono due varianti di questo standard (G.729 e G.729 Annex A) che differiscono enormemente per complessità computazionale; entrambe

forniscono in generale una qualità della voce pari a ADPCM a 32 kbit/s.

- G.731.1: descrive una tecnica di compressione che può essere utilizzata per comprimere la voce o altri componenti audio del servizio multimediale ad un basso bit-rate nell'ambito della famiglia H.324. A questo codec sono associati due bit-rate: 5.3 e 6.3 kbit/s. Il bit-rate più elevato si basa sulla tecnologia MP - MLQ e fornisce una qualità superiore. Il bit-rate più basso si basa sulla tecnologia CELP, fornisce una buona qualità ed offre ai progettisti di sistemi una maggiore flessibilità.

2.1.5.2 La valutazione soggettiva MOS

La qualità della voce può essere verificata in due modi: in modo soggettivo oppure oggettivo. Gli esseri umani possono eseguire un collaudo soggettivo mentre i computer, che a differenza dell'orecchio umano difficilmente vengono ingannati dagli schemi di compressione, svolgono collaudi oggettivi.

I sistemi codec vengono sviluppati e regolati sulla base di misure soggettive della qualità della voce. Le misure oggettive standard della qualità, come ad esempio la distorsione armonica totale ed i rapporti segnale-rumore, non sono in relazione con la percezione umana della qualità della voce, che resta comunque l'obiettivo finale della maggior parte delle tecniche di compressione della voce.

Una valutazione soggettiva molto comune, per quantificare le prestazioni dei sistemi codec relativi alla voce, si chiama MOS (Mean

Opinion Score). MOS esegue un test su un determinato gruppo di ascoltatori. Poiché la qualità della voce e del suono sono elementi soggettivi degli ascoltatori, per la conduzione di un buon test MOS è importante avere un'ampia gamma di ascoltatori e di materiale campionato. Gli ascoltatori possono valutare la qualità del materiale con un valore compreso tra 1 (cattivo) e 5 (eccellente). Viene poi calcolata la media dei punteggi per ottenere il valore medio.

La verifica MOS viene utilizzata anche per confrontare la bontà di un codec in determinate circostanze con livelli di rumore di sottofondo differenti, codifiche e decodifiche multiple e così via. I dati raccolti possono essere utilizzati per eseguire un confronto fra codec.

Anche se la valutazione MOS è un metodo soggettivo per determinare la qualità della voce, non è l'unico metodo impiegabile. ITU-T ha ulteriormente raffinato la raccomandazione P.861 che descrive i modi in cui si può determinare oggettivamente la qualità della voce utilizzando il metodo di misura PSQM (Percentual Speech Quality Measurement).

PSQM presenta molti difetti quando viene utilizzato nel campo della voce. Un difetto è il fatto che ciò che viene udito dalla "macchina" o da PSQM non è ciò che viene percepito dall'orecchio umano. In altri termini, si può ingannare l'orecchio umano e percepire una voce di buona qualità mentre in un computer ciò non è possibile.

Inoltre PSQM è stato sviluppato in modo da individuare i difetti provocati dalla compressione e decompressione e non dalla perdita di pacchetti o dal tremolio.

2.1.6 Perdita di pacchetti

Nelle reti a commutazione di pacchetto la perdita di un pacchetto è un problema comune: i protocolli UDP/IP non possono garantire la consegna dei pacchetti, né che questa avvenga nel giusto ordine, dato che alcuni pacchetti possono venire abbandonati durante periodi di congestione e carichi eccessivi.

Data la natura Real Time del traffico vocale, la ritrasmissione non è una scelta accettabile; per compensare questa perdita si usano approcci del tipo interpolazione del parlato (ripetendo l'ultimo pacchetto ricevuto) e trasmissione di informazioni ridondanti.

Poiché la perdita di pacchetti riguarda solo 20ms di parlato, l'ascoltatore medio non noterà alcuna differenza nella qualità della voce.

Quando si ha la perdita di un pacchetto, la stazione ricevente attende un determinato periodo di tempo (sulla base delle dimensioni del buffer di tremolio) e poi intraprende una strategia di occultamento dell'errore.

La strategia dell'implementazione G.729 di Voice Over iP replica l'ultimo pacchetto ricevuto, in modo che l'ascoltatore non percepisca alcuna pausa di silenzio. Poiché un pacchetto riguarda una perdita di soli 20ms, è probabile che l'ascoltatore non noti alcuna differenza. Si può ottenere questa strategia di occultamento dell'errore solo quando la perdita riguarda un solo pacchetto. Se si perdono più pacchetti consecutivi, la strategia di occultamento viene eseguita una sola volta.

Grazie alla strategia di occultamento dell'errore di G.729, si può dire che il metodo di compressione G.729 tollera una perdita di pacchetti media del 5% (packet loss rate) nell'ambito di un'intera chiamata. Una perdita di pacchetti superiore al 10% è da considerarsi intollerabile a meno che lo schema di codifica non sia particolarmente robusto.

2.1.7 Rilevamento dell'attività vocale

Nelle normali conversazioni, uno degli interlocutori parla e l'altro resta in ascolto. Le attuali reti allocano sempre un canale bidirezionale a 64 kbit/s indipendentemente dal fatto che parli uno solo dei due interlocutori. Questo significa che in una normale conversazione si perde almeno il 50% dell'ampiezza totale di banda. Il livello effettivo dell'ampiezza di banda sprecata può essere molto più elevato se si prende un campione statistico delle interruzioni e delle pause contenute in una normale conversazione interpersonale.

Utilizzando Voice Over IP è possibile impiegare questa ampiezza di banda sprecata per altri scopi, utilizzando un sistema di rilevamento dell'attività vocale. Il rilevamento dell'attività vocale controlla l'ampiezza del parlato in decibel e decide quando si può evitare di trasmettere la voce in una trama.

Tipicamente, quando il rilevamento dell'attività vocale percepisce una caduta dell'ampiezza della voce, attende un periodo prefissato e poi smette di inserire le trame vocali nei pacchetti. Questo periodo di tempo fissato è chiamato persistenza e generalmente ha una durata di 200ms.

In ogni tecnologia occorre scendere a compromessi. Il rilevamento dell'attività vocale ha alcuni problemi intrinseci a determinare l'inizio e la fine del parlato ed a distinguere il parlato dal rumore di sottofondo. Questo significa che se ci si trova in un locale rumoroso, il rilevamento dell'attività vocale non sarà in grado di distinguere fra la voce ed il rumore di sottofondo. Questo problema è anche noto come soglia segnale-rumore. In questi casi il sistema di rilevamento dell'attività vocale si disattiva all'inizio della chiamata.

Un altro problema intrinseco del rilevamento dell'attività vocale consiste nel rilevare il momento in cui inizia il parlato. Tipicamente l'inizio di una sequenza viene eliminato. Normalmente la persona che ascolta il parlato non nota questa eliminazione.

2.1.8 Conversione digitale/analogico

Anche le conversioni da analogico a digitale o da digitale ad analogico presentano dei problemi nelle reti odierne. Ogni volta che si verifica una conversione da digitale ad analogico e viceversa, il parlato o la forma d'onda diventano meno "realistici". Anche se le attuali reti sono in grado di gestire almeno sette conversioni da digitale ad analogico prima che venga influenzata la qualità della voce, il parlato compresso è meno resistente a queste conversioni.

E' importante notare che la conversione da digitale ad analogico deve essere gestita assolutamente in un ambiente di parlato compresso. Quando si usa G.729, due sole conversioni da digitale ad analogico provocano una drastica riduzione della valutazione MOS. L'unico modo

per gestire la conversione da digitale ad analogico è quello di progettare ambienti Voice Over IP con il minor numero possibile di conversioni.

Anche se le conversioni da digitale ad analogico riguardano tutte le reti vocali, le reti Voice Over IP con codec PCM (G.711) sopportano meglio i problemi provocati dalle conversioni digitale - analogico rispetto alle attuali reti telefoniche.

2.1.9 Codifica tandem

Tutte le reti a commutazione di circuito odierne funzionano sulla premessa di eseguire commutazioni delle chiamate a livello di collegamento dei dati. I commutatori di circuito (le centrali telefoniche) sono organizzati secondo un modello gerarchico in cui le centrali in posizione più elevata nella gerarchia sono chiamate centrali tandem.

Le centrali tandem non hanno un collegamento diretto con le linee locali ma operano come centrali telefoniche di livello superiore. Nel modello gerarchico possono esistere vari livelli di centrali telefoniche tandem. Questo offre a tutti la possibilità di avere un apparecchio telefonico senza per questo dover avere una connessione diretta con ogni altro apparecchio telefonico.

Volendo fare un esempio, prendiamo una chiamata che deve essere trasportata attraverso tre centrali telefoniche. Una chiamata vocale che passa attraverso due centrali terminali e una centrale tandem non sperimenta alcun degrado nella qualità della voce poiché queste centrali usano canali a 64 kbit/s.

Se le centrali terminali comprimono la voce e la centrale tandem deve decomprimere e ricomprimere la voce, si può percepire un drastico degrado della qualità della voce. Anche se la decompressione e ricompressione della voce non sono più molto comuni, occorre considerare questo fattore durante la progettazione delle reti a pacchetti.

Il degrado della voce si verifica quando per ogni chiamata telefonica viene eseguito più di un ciclo di compressione / decompressione. La *Figura 2.2* presenta un esempio di questa situazione.

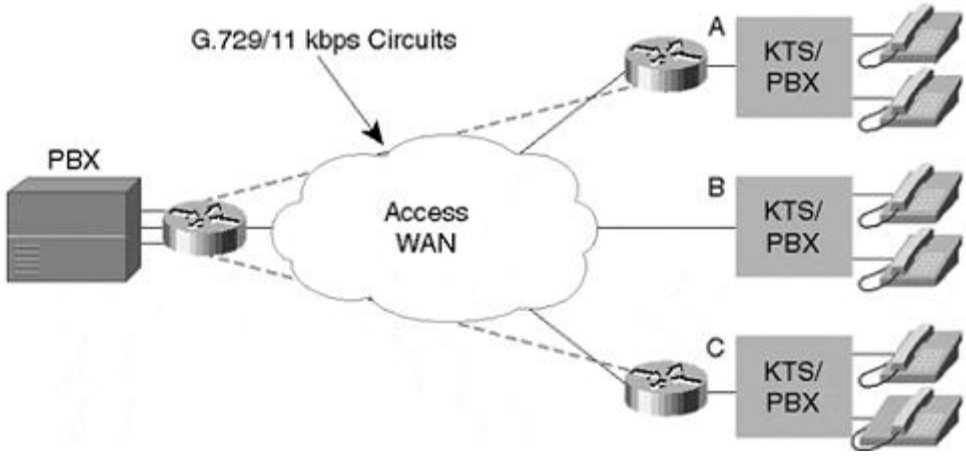


FIGURA 2.2: Codifica tandem in Voice Over IP

La *Figura 2.2* rappresenta tre router Voice Over IP che connettono tramite linee private il centralino PBX della sede centrale e tre centralini PBX di altrettanti sedi remote. La rete è progettata per conservare tutte le informazioni del piano di numerazione nel centralino PBX. Questa è una situazione molto comune nelle aziende

che preferiscono mantenere centralizzata l'amministrazione del piano di numerazione.

Un problema della codifica tandem in Voice Over IP è che se un utente telefonico della sede B vuole chiamare un utente che si trova nella sede C, occorre utilizzare due porte Voice Over IP della sede centrale A. Inoltre occorre eseguire due cicli di compressione e decompressione con un degrado nella qualità della voce.

I vari sistemi codec reagiscono differentemente alla codifica tandem. G.729 può sopportare due cicli di compressione / decompressione mentre G.723.1 è più sensibile alla ripetizione dei cicli di compressione.

Ad esempio si supponga che un utente della sede remota B voglia chiamare un utente della sede remota C. La chiamata passa attraverso il centralino PBX B, viene compressa e trasformata in pacchetti nel router Voice Over IP B e viene inviato al router Voice Over IP A della sede centrale che decomprime la chiamata e la invia al centralino PBX A. Il centralino PBX A commuta nuovamente la chiamata al proprio router che comprime e trasforma in pacchetti la chiamata e la invia alla sede remota C dove viene decompressa e inviata al centralino PBX C. Questo processo è chiamato compressione tandem e deve essere evitato in tutte le reti in cui esiste compressione.

Si può rivedere lo stesso esempio dei tre centralini PBX connessi tramite tre router Voice Over IP; configurando in modo differente questi router, si semplifica il flusso delle chiamate evitando la codifica tandem (vedere la *Figura 2.3*)

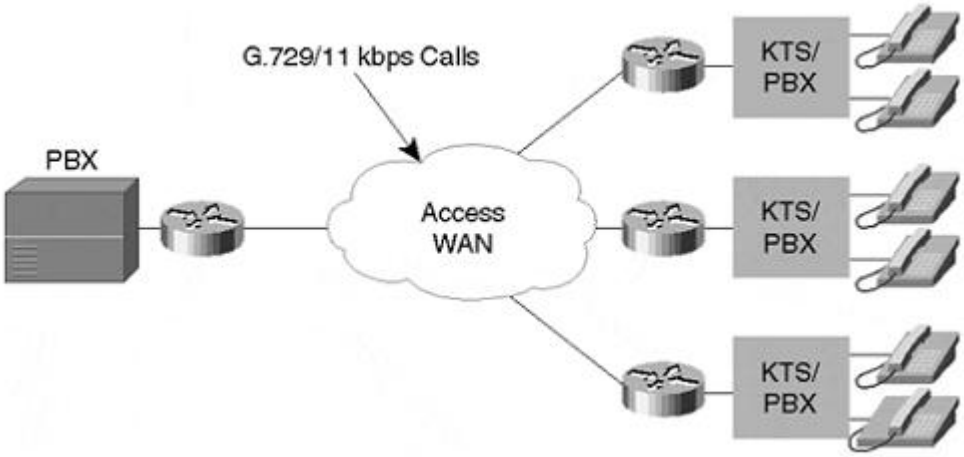


FIGURA 2.3: Voice Over IP senza codifica tandem

Nella *Figura 2.3* si può vedere uno dei punti di forza del protocollo IP: non è necessario affittare una linea privata dalla società telefonica per eseguire una chiamata fra due centralini PBX. Se esiste una rete dati che connette le sedi, Voice Over IP può impiegare tale rete.

Il piano di numerazione viene trasferito dal centralino PBX della sede centrale ai router Voice Over IP. Questo consente a ogni dispositivo Voice Over IP di prendere le decisioni di indirizzamento della chiamata eliminando la necessità di impiegare linee private affittate. Il vantaggio principale di questa soluzione è la rimozione dei cicli di compressione e decompressione.

2.1.10 Confronto fra chiamate: PSTN e IP

Per semplificare il flusso di una chiamata attraverso una centrale telefonica e IP, questa parte del capitolo descrive i modi in cui si può

chiamare il proprio vicino utilizzando da un lato la rete PSTN e dall'altro Internet. Anche se attualmente Internet non rappresenta il luogo migliore per garantire una buona qualità della voce, è adatta per lo scopo prefissato ovvero quello di mostrare come si svolge il flusso di chiamata.

Si comincia con la descrizione del flusso di chiamata su rete PSTN e si può confrontare questo con il flusso di chiamata telefonico IP, notando le analogie nell'instaurazione della chiamata.

Supponiamo che Bob chiami la sua vicina Alice. Entrambi sono connessi alla stessa centrale telefonica terminale e pertanto non è necessario accedere alla rete SS7. Si svolgono le seguenti operazioni.

1. Bob solleva il telefono.
2. La centrale telefonica terminale invia a Bob il tono di linea.
3. Bob compone il numero telefonico di Alice.
4. La centrale telefonica terminale raccoglie ed analizza il numero, per determinare la destinazione della chiamata. La centrale terminale sa che la chiamata giunge dall'abitazione di Bob poiché riceve una chiamata attraverso la sua porta.
5. La centrale analizza il numero (digitato da Bob) per determinare se si tratta di un numero locale che può servire direttamente (se Alice non è servita dalla stessa centrale telefonica terminale, questa centrale ricerca nelle proprie tabelle di instradamento il modo in cui connettere questa chiamata).
6. La centrale determina la linea dell'abbonato Alice.

7. La centrale terminale invia un segnale sulla linea di Alice per far squillare il suo telefono.
8. Viene inviato un segnale anche a Bob che sente che il telefono di Alice è libero e sta squillando (lo squillo del telefono di Alice ed il tono di squillo che Bob può udire nel telefono non sono necessariamente sincronizzati).
9. Alice solleva il telefono.
10. La centrale telefonica terminale connette il percorso vocale fra Bob e Alice. Si tratta di un circuito a 64 kbit/s full-duplex DS-0 (Digital Service, Level 0) ed instaura la comunicazione vocale.

Per la chiamata via Internet sia Bob che Alice devono essere connessi ad Internet o a qualche altra rete. Supponendo che questa rete esista o che entrambi abbiano un collegamento ad Internet, vengono svolte le seguenti operazioni.

1. Alice lancia la propria applicazione di telefonia via Internet compatibile H.323.
2. Bob ha già lanciato la propria applicazione di telefonia via Internet.
3. Alice sa che il "nome" di Bob in Internet (o la sua voce sul server DNS) è *bob.vicino.com*, lo inserisce in un'apposita sezione della propria applicazione di telefonia via Internet e preme INVIO.
4. L'applicazione di telefonia via Internet converte il nome *bob.vicino.com* in un nome di host DNS e accede al server DNS

configurato in modo statico nella macchina di Alice per risolvere il nome e ottenere il relativo indirizzo IP.

5. La macchina DNS invia l'indirizzo IP di Bob.
6. L'applicazione di telefonia via Internet di Alice prende l'indirizzo IP di Bob e gli invia un messaggio H.225.
7. Il messaggio H.225 chiede al PC di Bob di iniziare a squillare.
8. Bob fa clic sul pulsante di accettazione della chiamata che dice alla sua applicazione di telefonia via Internet di inviare un messaggio di connessione H.225.
9. L'applicazione di telefonia via Internet di Alice inizia la negoziazione H.245 con il PC di Bob.
10. La negoziazione H.245 termina e i canali logici vengono aperti. Ora Alice e Bob possono comunicare tramite la rete a pacchetti.

Se tutto questo è complesso, ciò è dovuto al fatto che il Voice over IP è complesso. L'esempio non mostra tutti i passi e omette vari dettagli che un provider deve considerare per creare una rete Voice over IP. Ma nonostante questa complessità Voice over IP presenta numerosi vantaggi. Grazie all'universalità del protocollo IP, quando una chiamata è trasformata in pacchetti può essere destinata al proprio vicino di casa o a un parente in Norvegia.

2.2 Qualità del servizio

In una rete ben ingegnerizzata occorre fare attenzione a separare le funzioni che si svolgono ai margini della rete dalle funzioni che si svolgono nel nucleo centrale o backbone della rete. Separando le funzionalità marginali da quelle backbone è possibile raggiungere la migliore qualità di servizio possibile.

In alcune situazioni si potrebbe anche non dover impiegare alcuno strumento di qualità di servizio e ottenere comunque la qualità di servizio adatta per le applicazioni. Ma in generale ogni rete presenta dei problemi specifici che possono essere risolti utilizzando uno o più strumenti di qualità di servizio.

In questa seconda parte di capitolo si descrive i seguenti strumenti relativi alle funzionalità marginali di una rete.

- Ampiezza di banda aggiuntiva.
- Il protocollo cRTP (Compressed Real Time Protocol).
- L'accodamento:

    Weighted Fair Queuing (WFQ).

    Accodamento personalizzato (CQ – Custom Queuing).

    Class-Based Weighted Fair Queuing (CB-WFQ).

    Accodamento a priorità – CB-WFQ.
- La classificazione dei pacchetti:

    Precedenza IP.

    Politiche di instradamento.

    RSVP (Resource Reservation Protocol).

    IP RTP Riserve.

IP RTP Priorità.

- Definizione e politica dei flussi della rete:

  GTS (Generic Traffic Shaping).

  FRTS (Frame Relay Traffic Shaping).

  CAR (Committed Access Rate).

- Frammentazione:

  MCML PPP (Multi-Class Multilink Point-to-Point Protocol).

  FRF.12 (Frame Relay Forum 12).

  MTU.

  IP MTU (IP Maximum Transmission Unit).

Inoltre questa seconda parte di capitolo descrive i seguenti elementi riguardanti le funzionalità backbone di una rete.

- Trasporto ad alta velocità:

  Packet over SONET (POS).

  IP + ATM (Asynchronous Transfer Mode).

- Accodamento ad alta velocità:

  WRED (Weighted Random Early Drop/Detect).

Voice Over IP presenta vari tipi di problemi. La qualità di servizio può aiutare a risolvere alcuni di questi problemi e in particolare la perdita di pacchetti, il tremolio e il ritardo di gestione.

Alcuni dei problemi che la qualità di servizio non può risolvere sono il ritardo di propagazione (al momento attuale non esiste alcuna tecnica in grado di superare il limite imposto dalla velocità della luce), il ritardo dei codec, il ritardo di campionamento e il ritardo di digitalizzazione.

Una chiamata telefonica Voice Over IP può essere equivalente a ogni altro grosso investimento. Pertanto è importante sapere quali parti del budget non possono essere cambiate e quali si ha la possibilità di controllare.

2.2.1 Limiti legati all'ampiezza di banda

Il primo grande problema quando si tratta di prestazioni Voice Over IP è rappresentato dai vincoli legati all'ampiezza di banda. A seconda del codec utilizzato e del numero di campioni vocali per pacchetto, l'ampiezza di banda per chiamata può aumentare enormemente. Per una descrizione delle dimensioni dei pacchetti e dell'ampiezza di banda consumata si consulti la tabella seguente:

CODEC	**AMPIEZZA DI BANDA**	**AMPIEZZA DI BANDA CON CRTP (INTESTAZIONE 2 BYTE)**	**LATENZA DI CAMPIO-NAMENTO**
G.729 w/ un campione da 10ms per trama	40 kbit/s	9,6 kbit/s	15ms
G.729 w/ quattro campioni da 10ms per trama	16 kbit/s	8,4 kbit/s	45ms

G.729 w/ due campione da 10ms per trama	24 kbit/s	11,2 kbit/s	25ms
G.711 w/ un campione da 10ms per trama	112 kbit/s	81,6 kbit/s	10ms
G.711 w/ due campione da 10ms per trama	96 kbit/s	80,8 kbit/s	20ms

Dopo aver osservato questa tabella ci si potrebbe chiedere perché si consumano 24 kbit/s di ampiezza di banda quando si usano codec da 8 kbit/s. Questo è dovuto al protocollo IP. G.729 con due campioni da 10ms contiene 20 byte per trama e questo produce un totale di 8 kbit/s. Le intestazioni IP, RTP e UDP del pacchetto aggiungono 40 byte a ogni trama. Pertanto in questo caso il protocollo IP aggiunge un'intestazione di dimensioni pari al doppio del carico utile.

Utilizzando G.729 con due campioni da 10ms come esempio, senza la compressione dell'intestazione RTP, vengono consumati 24 kbit/s per ogni direzione della chiamata. Anche se questo può essere un valore sopportabile per circuiti T1 (1.544 Mbit/s), E1 (2.048 Mbit/s) o circuiti ancora più veloci, si tratta di una grossa quantità (il 42%) per un circuito da 56 kbit/s.

Inoltre si deve tenere in considerazione che l'ampiezza di banda indicata nella tabella precedente non include intestazioni del livello 2

(PPP, Frame Relay e così via); include solo le intestazioni introdotte dal livello 3 (il livello di rete) e dai livelli sovrastanti. Pertanto la stessa chiamata G.729 può consumare un'ampiezza di banda differente a seconda del tipo di livello di collegamento utilizzato (Ethernet, Frame Relay, PPP e così via).

2.2.2 Il protocollo cRTP

Per ridurre la percentuale di ampiezza di banda consumata per una chiamata vocale G.729, si può usare il protocollo cRTP che consente di comprimere l'intestazione IP/RTP/UDP in 2 o 4 byte (vedere la *Figura 2.4*).

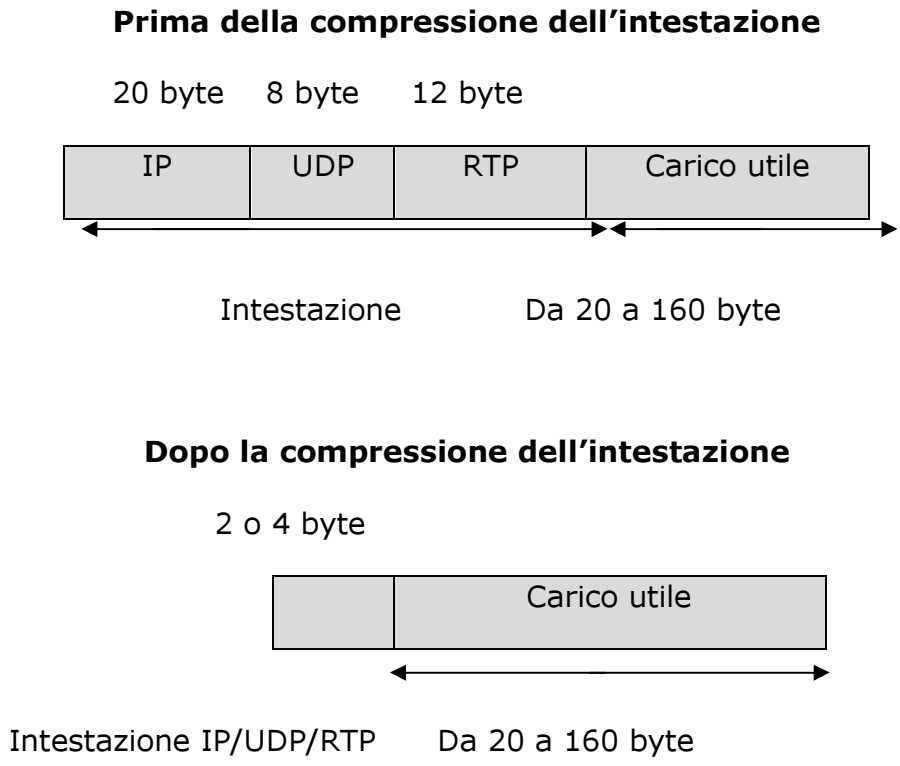


FIGURA 2.4: Compressione dell'intestazione cRTP

Con cRTP, la quantità di traffico per chiamata Voice Over IP, si riduce da 24 kbit/s a 11.2 kbit/s. Questo rappresenta un grande miglioramento per i collegamenti a bassa ampiezza di banda. Ad esempio, un collegamento a 56 kbit/s sarà in grado di trasportare 4 chiamate Voice Over IP G.729 da 11.2 kbit/s ciascuna. Senza cRTP, sarebbe possibile trasportare solo due chiamate Voice Over IP G.729.

Per evitare l'inutile consumo di ampiezza di banda, il protocollo cRTP viene applicato collegamento per collegamento. Questo schema di compressione riduce l'intestazione IP/UDP/RTP a 2 byte (se non viene usato il checksum UDP) o in 4 byte (se viene usato il checksum UDP).

cRTP usa una tecnica simile alla compressione dell'intestazione TCP. Nella compressione dell'intestazione TCP, il primo dimezzamento della velocità di trasferimento dei dati è dovuto al fatto che, la metà dei byte delle intestazioni IP e TCP, rimane costante per tutta la durata della connessione.

Il guadagno più ingente deriva, però, dal fatto che le differenze da pacchetto a pacchetto seguono spesso un andamento costante, anche quando per ogni pacchetto cambia il valore dei campi dell'intestazione. Pertanto, l'algoritmo può semplicemente aggiungere una unità ad ogni valore ricevuto. Poiché il compressore ed il decompressore mantengono condivise sia l'intestazione non compressa che le differenze di primo ordine nello stato della sessione, il protocollo cRTP deve solo comunicare il fatto che la differenza di secondo ordine è

zero. In questo caso il decompressore può ricostruire l'intestazione originale senza alcuna perdita di informazioni, semplicemente aggiungendo le differenze di primo ordine all'intestazione non compressa che è stata memorizzata.

Così come la compressione dell'intestazione TCP/IP gestisce uno stato condiviso fra più connessioni TCP simultanee, questa compressione IP/UDP/RTP deve conservare lo stato per più contesti di sessione. Un contesto di sessione è definito dalla combinazione degli indirizzi IP di origine e di destinazione, delle porte UDP di origine e di destinazione e del campo SSRC (Synchronization Source RTP). Per l'implementazione del compressore si può usare una funzione hash che impiega i campi per indicizzare una tabella dei contesti di sessione memorizzati.

Il pacchetto compresso trasporta un piccolo valore intero chiamato identificatore del contesto di sessione (CID) che indica il contesto di sessione all'interno del quale deve essere interpretato il pacchetto. Il decompressore potrà usare il codice CID per indicizzare la tabella dei contesti di sessione memorizzati.

Il protocollo cRTP può comprimere i 40 byte dell'intestazione per produrre 2 o 4 byte. Pertanto, nel 98% dei casi, verrà inviato un pacchetto compresso. Tuttavia, periodicamente, occorre inviare un'intestazione non compressa, per verificare che entrambi i lati della connessione usino lo stesso stato. Talvolta si verificano dei cambiamenti nei campi normalmente costanti. In questi casi l'intestazione IP/UDP/RTP non può essere compressa e deve essere inviata per esteso.

Si dovrebbe usare il protocollo cRTP in ogni interfaccia WAN dove l'ampiezza di banda rappresenta un problema ed esiste un ingente traffico RTP. Non si deve usare cRTP su interfacce ad alta velocità, in quanto gli svantaggi superano i vantaggi.

Come per ogni compressione, la CPU risulta maggiormente impegnata per comprimere il pacchetto. Questo aumenta il tempo di utilizzo della CPU da parte del router. Pertanto, occorre bilanciare i vantaggi (i minori requisiti in termini di ampiezza di banda) con gli svantaggi (un maggior utilizzo della CPU). Un router con livelli di utilizzo della CPU elevati può sperimentare problemi durante l'esecuzione di altre operazioni. Pertanto, si consiglia come regola empirica di mantenere l'utilizzo della CPU inferiore al 60% - 70%, in modo da garantire che la rete funzioni senza problemi.

2.2.3 Accodamento

Quello dell'accodamento è un concetto piuttosto semplice. Si può paragonare un sistema di accodamento ad un sistema di autostrade. Si supponga di trovarsi in autostrada e di guidare ad una velocità adeguata. Quando ci si avvicina ad una casello occorre rallentare, fermarsi e pagare il pedaggio. Durante il tempo necessario per pagare, si forma una coda di lettura, creando una congestione.

Anche nelle code si applica il concetto FIFO (First-In First-Out): il primo ad entrare in una coda è anche il primo ad uscirne. L'accodamento FIFO è stato il primo sistema di accodamento utilizzato

nei router e può essere utile ancora oggi in determinate topologie di rete.

Le reti odierne, che presentano un'ampia variabilità di applicazioni, protocolli ed utenti, richiedono un modo di classificazione del traffico. Tornando all'esempio del casello, è necessario prevedere una corsia speciale per l'accodamento delle vetture. In molti casi esiste anche una corsia rapida per il pagamento elettronico.

Ci sono vari strumenti di accodamento che consentono all'amministratore di rete di specificare quale tipo di traffico ha caratteristiche di importanza tali da ignorare l'accodamento. La tecnica di accodamento più diffusa si chiama WFQ.

2.2.3.1 Accodamento WFQ

L'accodamento FIFO inserisce tutti i pacchetti ricevuti in una coda e li trasmette nel momento in cui risulta disponibile l'ampiezza di banda necessaria. Al contrario, l'algoritmo WFQ (Weighted Fair Queuing) usa più code per distinguere i flussi, assegnando un'ampiezza di banda uguale per ciascun flusso. Questo evita che un'applicazione, ad esempio FTP (File Transfer Protocol), consumi tutta l'ampiezza di banda disponibile.

L'algoritmo WFQ garantisce che le code non restino mai a corto di ampiezza di banda e che il traffico mantenga un servizio prevedibile. I flussi di dati di basso volume ricevono un servizio preferenziale, trasmettendo l'intero carico utile in modo sincronizzato. I flussi di

traffico di alto volume condividono la capacità rimanente, ottenendo un'ampiezza di banda uguale o proporzionale.

L'algoritmo WFQ è simile al multiplex a divisione di tempo TDM in quanto suddivide in modo uguale l'ampiezza di banda disponibile fra i vari flussi, in modo che nessuna applicazione risulti mai bloccata. WFQ è, però, superiore a TDM poiché, quando un flusso risulta assente, WFQ può regolare dinamicamente l'uso dell'ampiezza di banda disponibile per i flussi rimanenti.

L'algoritmo WFQ identifica dinamicamente i flussi di dati sulla base di vari fattori. Viene stabilita una scala di priorità fra questi flussi di dati sulla base dell'ampiezza di banda che essi consumano. Questo algoritmo consente di eseguire una buona condivisione dell'ampiezza di banda senza l'impiego di liste di accesso o di altre operazioni amministrative che richiedono comunque del tempo. WFQ determina le caratteristiche di un flusso utilizzando l'indirizzo di origine e di destinazione, il tipo di protocollo, il numero di socket o di porta ed i valori di qualità di servizio e di tipo di servizio.

L'algoritmo WFQ fa in modo che le applicazioni a bassa ampiezza di banda, che rappresentano la maggior parte del traffico, abbiano tutta l'ampiezza di banda che richiedono, lasciando a disposizione lo spazio rimanente per il traffico con ampiezza di banda superiore. Questo tipo di accodamento riduce l'effetto di tremolio ed offre una condivisione efficiente dell'ampiezza di banda disponibile.

La valutazione eseguita da WFQ si basa fondamentalmente su sei meccanismi: IP Precedence, FECN (Frame Relay Forward Explicit

Congestion Notification), BECN (Frame Relay Backward Explicit Congestion Notification), RSVP, IP RTP Priority e IP RTP Reserve.

Il campo IP Priority può contenere un valore compreso fra 0 (l'impostazione predefinita) e 7. A mano a mano che aumenta il valore di precedenza, l'algoritmo alloca per tale conversazione o flusso un'ampiezza di banda maggiore. Questo consente al flusso di trasmettere con maggiore frequenza.

Nelle reti Frame Relay, i bit FECN e BECN segnalano normalmente la presenza di una congestione del traffico. Quando viene individuata una congestione, i pesi utilizzati dall'algoritmo cambiano in modo che la conversazione che sperimenta la congestione trasmetta con minore frequenza.

L'amministratore di rete deve preoccuparsi di garantire che i pesi usati nell'algoritmo WFQ siano regolati correttamente. Questo evita che un'applicazione richieda, o usi, una priorità più elevata del necessario.

L'algoritmo WFQ non è adatto per essere utilizzato su interfacce che operano a velocità superiori a 2.048 Mbit/s.

2.2.3.2 Accodamento personalizzato

L'accodamento personalizzato consente agli utenti di specificare una percentuale dell'ampiezza di banda disponibile per un determinato protocollo. E' possibile definire fino a 16 code di output più una coda aggiuntiva per i messaggi di sistema. Ogni coda viene servita in modo sequenziale ed a turni, trasmettendo una percentuale del traffico per ogni coda prima di procedere con la coda successiva.

Il router determina il numero di byte che devono essere trasmessi per ogni coda sulla base della velocità dell'interfaccia e della percentuale di traffico configurato. In altre parole, l'ampiezza di banda inutilizzata potrà essere utilizzata da un altro tipo di traffico, finché la coda non tornerà a richiedere la percentuale che le è stata destinata.

L'accodamento personalizzato richiede una conoscenza dei tipi di porta e dei tipi di traffico. Questo significa che produce una grande quantità di carico per l'amministratore. Ma, al termine di queste valutazioni, l'accodamento personalizzato offre un approccio a granularità più elevata che, in alcuni casi, è preferibile rispetto all'accodamento standard.

2.2.3.3 Accodamento a priorità

L'accodamento a priorità consente all'amministratore di rete di configurare quattro priorità di traffico: high, normal, medium e low. Il traffico in arrivo viene assegnato ad una delle quattro code di output. Il traffico che viene inserito nella coda ad alta priorità viene servito finché la coda non è vuota e poi vengono trasmessi i pacchetti contenuti nella successiva coda di priorità.

Questo tipo di accodamento garantisce che il traffico più importante riceva sempre l'ampiezza di banda richiesta ma a scapito delle altre applicazioni.

Pertanto, è importante conoscere il flusso del traffico, in modo da non bloccare le applicazioni per assenza di ampiezza di banda. L'accodamento a priorità è particolarmente adatto a situazioni in cui il

traffico a priorità più elevata consuma una quantità ridotta dell'ampiezza di banda disponibile sulla linea.

L'accodamento a priorità consente all'amministratore di rete di rallentare determinate applicazioni. Una coda a priorità mal configurata potrebbe trovarsi a servire una sola coda ignorando completamente tutte le altre. Questo provoca il blocco di alcune applicazioni. Se un amministratore di sistema di rete tiene in considerazione questo problema, l'accodamento a priorità può rappresentare l'alternativa più adatta per determinati clienti.

2.2.3.4 Accodamento CB-WFQ

L'accodamento CB-WFQ presenta tutti i vantaggi dell'accodamento WFQ con la funzionalità aggiuntiva di un supporto granulare per determinate classi di traffico definite dall'amministratore di rete. CB-WFQ può funzionare anche su interfacce ad alta velocità e consente di definire cosa costituisce una classe, sulla base di criteri che non rientrano nei confini del flusso. Utilizzando CB-WFQ è possibile creare una classe specifica per il traffico vocale. L'amministratore di rete definisce queste classi di traffico tramite liste di accesso. Queste classi di traffico determinano il modo in cui i pacchetti vengono raggruppati in code.

La funzionalità più interessante di CB-WFQ è il fatto che consente all'amministratore di rete di specificare esattamente l'ampiezza di banda che deve essere allocata per classe o per traffico. CB-WFQ può

gestire fino a 64 classi e controllare i requisiti di ampiezza di banda di ciascuna classe.

Con l'accodamento WFQ standard, vi sono dei pesi che determinano la quantità di ampiezza di banda allocata per conversazione. Pertanto, tutto dipende dal numero di flussi e dal traffico presente in un determinato momento.

Con CB-WFQ, ad ogni classe viene associata una coda distinta. E' possibile allocare una determinata quantità minima di ampiezza di banda garantita per la classe, misurata in percentuale o in kbit/s. Le altre classi possono condividere l'ampiezza di banda inutilizzata in proporzione ai pesi loro assegnati. Quando si configura CB-WFQ, occorre considerare che l'allocazione dell'ampiezza di banda non significa necessariamente che il traffico appartenente ad una classe sperimenta bassi ritardi; tuttavia è possibile regolare i pesi in modo da emulare una coda a priorità.

2.2.3.5 Accodamento a bassa latenza (coda a priorità con CB-WFQ)

L'accodamento a bassa latenza è stato sviluppato per fornire una priorità assoluta al traffico vocale rispetto a tutto l'altro traffico presente in un'interfaccia.

La caratteristica della bassa latenza introduce in CB-WFQ la funzionalità di accodamento rigidamente a priorità di IP RTP Priority, necessaria per il traffico in tempo reale, sensibile ai ritardi, qual è, ad

esempio, la voce. L'accodamento a bassa latenza consente di utilizzare un rigido accodamento a priorità.

Anche se è possibile accodare vari tipi di traffico con un rigido accodamento a priorità, si consiglia di dirigere verso questa coda solo il traffico vocale. Questo consiglio si basa sul fatto che il traffico vocale presenta un buon comportamento ed invia i pacchetti ad intervalli regolari. Le altre applicazioni, che trasmettono ad intervalli irregolari, se configurate in modo errato, possono provocare il collasso dell'intera rete.

Con l'accodamento a bassa latenza è possibile specificare il traffico in moltissimi modi, per garantire un rigido invio a priorità. Per indicare che il flusso vocale deve essere accodato rigidamente a priorità, si può usare una lista di accesso. Questa situazione è differente rispetto alla priorità IP RTP Priority che consente solo di specificare un intervallo di porte UDP.

Questo meccanismo si è dimostrato potente e fornisce ai pacchetti vocali le caratteristiche di priorità, latenza e tremolio necessarie per ottenere una buona qualità della voce.

2.2.4 Classificazione dei pacchetti

Per ottenere il tipo di consegna dei pacchetti desiderato, occorre sapere come distribuire adeguatamente i pesi WFQ. Questa parte del capitolo descrive varie tecniche di valutazione dei "pesi" ed il loro uso nelle reti per ottenere il livello di qualità di servizio desiderato.

2.2.4.1 Precedenza IP

Con precedenza IP si fa riferimento ai tre bit del campo ToS di un'intestazione IP come illustrato nella *Figura 2.5*. Questi tre bit consentono di definire otto diverse classi di servizio, elencate nella *Figura 2.6*. I livelli di precedenza 6 e 7 sono riservati per le informazioni di gestione della rete (aggiornamenti di instradamento, pacchetti "welcome" e così via). Questo lascia a disposizione i sei livelli rimanenti per il normale traffico IP.

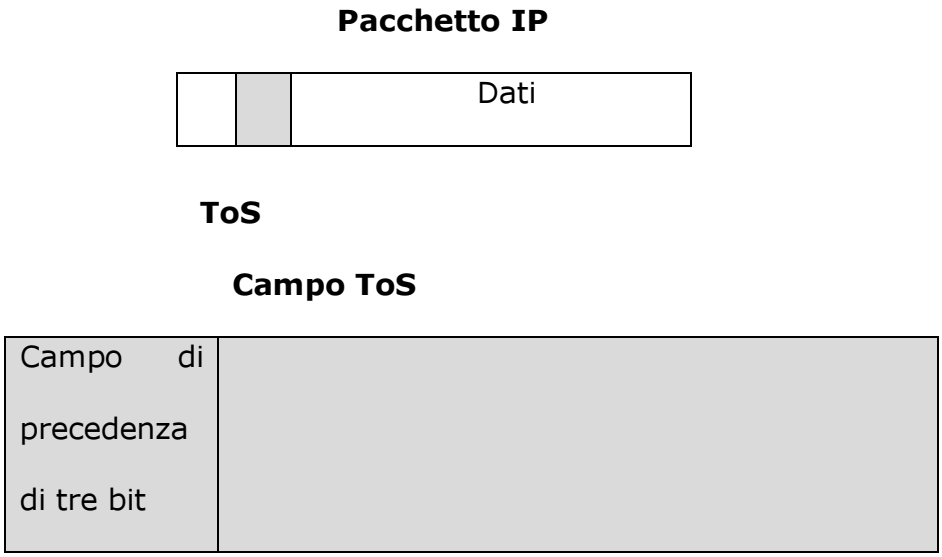


FIGURA 2.5: Il Campo ToS nell'intestazione IP

TIPO DI SERVIZIO	SCOPO
Routine	Set routine precedence (0)
Priority	Set priority precedence (1)

Immediate	Set immediate precedence (2)
Flash	Set Flash precedence(3)
Flash-override	Set Flash override precedence(4)
Critical	Set critical precedence (5)
Internet	Set internetwork control precedence(6)
Network	Set network control precedence(7)

FIGURA 2.6: I valori del campo ToS (Precedenza IP)

La precedenza IP consente ad un router di raggruppare i flussi del traffico sulla base degli otto livelli di precedenza e di accodare il traffico sulla base di queste informazioni e dell'indirizzo di origine, dell'indirizzo di destinazione e del numero di porta.

Si può considerare la precedenza IP come un meccanismo di qualità di servizio in banda. Infatti non sono previsti segnali aggiuntivi, né esiste un incremento nel traffico dei pacchetti. Per questi vantaggi, quello della precedenza IP è il meccanismo di qualità di servizio più impiegato nelle reti a larga scala.

Un modo per impostare i bit di precedenza IP è quello di considerare il numero di telefono di destinazione (il numero chiamato); impostare la precedenza in questo modo è facile e consente di creare vari tipi di classi di servizio a seconda del tipo di destinazione.

La precedenza IP non ha alcun meccanismo interno per rifiutare le impostazioni errate. Dunque, l'amministratore della rete deve prendere delle precauzioni, per assicurarsi che le impostazioni di precedenza IP della rete rimangano fedeli al piano originario.

2.2.4.2 Politiche di instradamento

E' possibile configurare una determinata politica per i flussi di traffico, evitando, quindi, di fare affidamento completamente sui protocolli di instradamento, per determinare l'inoltro e l'instradamento del traffico. La politica di instradamento consente anche di impostare il campo della precedenza IP, in modo che la rete possa utilizzare varie classi di servizio.

Le politiche di instradamento possono basarsi sugli indirizzi IP, sui numeri di porta, sui protocolli o sulle dimensioni dei pacchetti. Si può usare uno di questi descrittori e creare una politica di instradamento molto semplice, oppure si possono usare tutti i descrittori e creare una politica molto più complessa.

Tutti i pacchetti ricevuti in un'interfaccia, avendo attivato una politica di instradamento, passano attraverso appositi filtri di pacchetti chiamati *mappe di instradamento*. Le mappe di instradamento indicano la direzione che deve essere presa dai pacchetti.

E' anche possibile segnalare le mappe di instradamento con "permit" o "deny". Quando un'istruzione è contrassegnata con "deny", i pacchetti che rispondono a determinati criteri vengono rinviati lungo i normali canali di inoltro (in altre parole, viene eseguito un instradamento

basato sulla destinazione). Solo se l'istruzione è contrassegnata con "permit" ed i pacchetti rispondono ai criteri indicati, vengono applicate tutte le clausole specificate.

Se l'istruzione è contrassegnata con "permit" ed i pacchetti non rispondono ai criteri, verranno inoltrati lungo i normali canali di instradamento.

Da notare che la politica di instradamento è specificata nell'interfaccia che riceve i pacchetti e non nell'interfaccia che invia i pacchetti.

Per stabilire i criteri di corrispondenza si possono usare delle liste di controllo degli accessi standard o estese; le liste degli accessi IP standard specificano i criteri per l'indirizzo di origine mentre le liste degli accessi estese specificano criteri basati sull'applicazione, sul tipo di protocollo, sul tipo di servizio e sulla precedenza.

La clausola di corrispondenza è stata estesa per includere la lunghezza del pacchetto che deve essere compresa fra i due valori specificati. In questo modo, l'amministratore della rete potrà utilizzare anche la lunghezza dei pacchetti come criterio per distinguere fra traffico interattivo e standard (in genere quest'ultimo usa pacchetti di maggiori dimensioni).

Il processo della politica di instradamento segue la mappa di instradamento fino a trovare una corrispondenza. Se nella mappa di instradamento non si trova alcuna corrispondenza o se la voce della mappa di instradamento è contrassegnata da "deny" invece che da "permit", viene utilizzato un normale instradamento basato sulla destinazione.

2.2.4.3 RSVP

RSVP consente ai punti terminali della rete di segnalare il tipo di qualità di servizio richiesto da una determinata applicazione. Si tratta di un enorme progresso rispetto alla necessità, da parte della rete, di stabilire, in modo pressoché cieco, la qualità di servizio richiesta dalle applicazioni.

Gli amministratori di rete possono usare RSVP come lista dinamica degli accessi. Questo significa che gli amministratori di rete non devono preoccuparsi dei numeri di porta o dei flussi di pacchetti, poiché RSVP segnala tali informazioni nella sua richiesta originaria. RSVP è un protocollo di segnalazione punto-a-punto fuori banda, che richiede una certa ampiezza di banda ed una latenza per ogni tratta della rete che supporta RSVP. Se un nodo della rete (un router) non supporta RSVP, RSVP si porterà sul nodo successivo. Un nodo della rete ha la possibilità di approvare o di rifiutare la richiesta basandosi sul carico dell'interfaccia per la quale è stato richiesto il servizio.

RSVP è come un'ambulanza che "apre" il traffico. Basta seguire da vicino l'ambulanza. Chi si trova alla guida dell'ambulanza, potrà dire, ad ogni incrocio, che la vettura che segue l'ambulanza deve godere degli stessi privilegi dell'ambulanza. Ad ogni incrocio, viene assegnato il diritto di decidere se la vettura che segue l'ambulanza ha un livello di importanza sufficiente per avere questi particolari privilegi (nel caso di IP se ha il diritto di ricevere l'ampiezza di banda e la latenza richieste).

Le applicazioni ricevono una risposta che dice se la loro richiesta di qualità di servizio è stata approvata o rifiutata. Alcune applicazioni

trasmettono i propri dati a tutti, senza considerare la qualità di servizio; altre applicazioni, più intelligenti, scelgono invece di non trasmettere o scelgono un altro percorso. Nel caso di Voice Over IP tale percorso può anche essere la rete commutata PSTN. E' interessante notare che chi richiede i livelli di servizio in RSVP è la stazione ricevente e non la trasmittente. Questo consente a RSVP di adattarsi alla tecnologia IP multicast in cui esiste un trasmettitore e più ricevitori.

RSVP non è un protocollo di instradamento e non modifica la tabella di instradamento IP sulla base dei flussi di traffico o dei livelli di congestione. Semplicemente, RSVP attraversa la rete IP e consente ai protocolli di instradamento IP di scegliere il percorso ottimale. Questo percorso ottimale potrebbe anche non essere il percorso ideale come qualità del servizio. RSVP non può regolare i router in modo da modificare questo comportamento.

Anche se RSVP è uno strumento importante per la qualità di servizio, questo protocollo non risolve tutti i problemi. RSVP ha principalmente tre difetti: la scalabilità, il controllo delle ammissioni ed il tempo necessario per instaurare una prenotazione RSVP.

RSVP deve ancora essere impiegato in ambienti a larga scala. Nella situazione peggiore per RSVP, un router backbone deve gestire varie migliaia di prenotazioni RSVP ed accodare ciascun flusso sulla base di queste prenotazioni.

I problemi di scalabilità di RSVP lo relegano ai margini della rete e spingono ad usare altri strumenti di qualità di servizio per la parte

backbone della rete. A lungo termine, la IETF (Internet Engineering Task Force) elaborerà dei metodi per utilizzare meglio RSVP e per aumentare la sua scalabilità.

2.2.4.4 IP RTP reserve

Si può paragonare IP RTP Reserve ad un sistema di prenotazione RSVP "statico". Quando si usa IP RTP Reserve, non è necessario usare la precedenza IP o RSVP.

Anche se IP RTP Reserve classifica i pacchetti sulla base di un intervallo di porte UDP, consente anche di specificare l'ampiezza di banda, che può essere gestita a priorità in tale intervallo di porte.

La prenotazione "statica" di IP RTP Reserve consente di classificare con un peso più elevato il traffico "riservato". Quando questo traffico è assente, ogni altro flusso di traffico può usare l'ampiezza di banda aggiuntiva non impiegata.

IP RTP Reserve non rappresenta la soluzione più scalabile fra quelle disponibili. Consente di impiegare un intervallo di solo 100 porte UDP e non consente di controllare le ammissioni. Poiché manca questo controllo viene pesato ogni flusso di pacchetti che rientra nell'intervallo IP RTP Reserve specificato.

2.2.4.5 IP RTP priority

Quando è attivata la valutazione dei "pesi" WFQ ed è configurata IP RTP Priority, si crea una rigida coda a priorità. Si può usare la

funzionalità IP RTP Priority per attivare un rigido schema di accodamento a priorità per dati sensibili ai ritardi.

Si può identificare il traffico vocale sulla base del numero di porta UDP e classificarlo in una coda a priorità. In questo modo la voce avrà a disposizione un servizio preferenziale rispetto all'altro traffico. Questo è lo schema di classificazione consigliato per le reti Voice Over IP dotate di collegamenti a bassa ampiezza di banda (fino a 768 kbit/s).

2.2.5 Politiche di traffico

Le sezioni precedenti hanno descritto i modi in cui si possono accodare i vari flussi di traffico e definire delle priorità fra questi flussi. Questo è un elemento importante della qualità di servizio ma talvolta è necessario regolare con precisione o limitare la quantità di traffico che un'applicazione può inviare attraverso le interfacce o le reti. Per questo si utilizzano strumenti di limitazione della velocità come il CAR e strumenti di controllo come GTS e FRTS.

La differenza principale fra questi due strumenti di regolazione del traffico sta nel fatto che i primi riducono la velocità del traffico sulla base della politica impostata mentre i secondi, in genere, creano un buffer del traffico in eccesso ed attendono il successivo intervallo libero per trasmettere i dati.

Gli strumenti di limitazione della velocità e di controllo del traffico operano in modo analogo, nel senso che entrambi identificano le situazioni in cui il traffico supera le soglie impostate dall'amministratore di rete.

Spesso questi due strumenti vengono utilizzati insieme. Il controllo del traffico viene utilizzato ai margini della rete (ovvero nella parte della rete che raggiunge il cliente), per assicurare che il cliente utilizzi l'ampiezza di banda in base alle proprie esigenze.

CAR viene frequentemente utilizzato nelle reti dei provider per garantire che un abbonato non superi l'ampiezza di banda definita dal contratto.

Il CAR (Committed Access Rate) permette agli amministratori di rete di impostare azioni di superamento (per esempio scartare pacchetti o contrassegnarli con bassi valori di precedenza IP) o conformità (per trasmettere il pacchetto) del traffico, limitando il traffico in base alla precedenza, all'indirizzo MAC, all'indirizzo IP o ad altri parametri.

Per il controllo del traffico sia GTS (Generic Traffic Shaping) che FRTS (Frame Relay Traffic Shaping) vengono utilizzati per controllare l'uso dell'ampiezza di banda disponibile, definire le politiche di gestione del traffico, regolare il flusso per evitare congestioni. La principale differenza è che FRTS utilizza un sistema di accodamento personalizzato, o a priorità, mentre GTS utilizza il WFQ.

2.2.6 Frammentazione

Il motivo che richiede la frammentazione dei pacchetti è semplice. I pacchetti di maggiori dimensioni (1500 byte di MTU) richiedono più tempo per il trasferimento lungo collegamenti a bassa ampiezza di banda (fino a 768 kbit/s). La frammentazione suddivide i pacchetti di

maggiori dimensioni in pacchetti più piccoli. Si può eseguire l'operazione ai livelli 2 o 3 del modello di riferimento OSI.

In molte applicazioni per dati, la latenza provocata dai collegamenti a bassa ampiezza di banda non ha una grande importanza per l'utente finale. Al contrario, nelle applicazioni in tempo reale, questo può provocare molti problemi (voce a scatti, perdita di trame, caduta delle chiamate).

Un pacchetto di 1500 byte in movimento lungo un circuito da 56 kbit/s richiede 214ms per attraversare il circuito. La raccomandazione ITU-T prevede una latenza massima della voce in senso unidirezionale di 150ms. Pertanto un circuito a 56 kbit/s con un pacchetto da 1500 byte consuma l'intero ritardo accumulabile per la trasmissione in Voice Over IP.

Da sola la frammentazione non è sufficiente per rimuovere il problema della latenza su circuiti a bassa ampiezza di banda. Il router deve essere in grado di eseguire l'accodamento di frammenti o di pacchetti più piccoli invece che dei pacchetti originali non frammentati.

La frammentazione aiuta anche ad eliminare i problemi di bloccaggio, cioè il periodo di tempo per il quale un pacchetto può consumare l'ampiezza di banda disponibile nella rete costringendo così ad accodare gli altri pacchetti di applicazioni operanti in tempo reale. E' opportuno mantenere il ritardo di bloccaggio a valori non superiori all'80% delle dimensioni totali del pacchetto vocale.

Per ottenere il valore esatto si utilizzano alcuni algoritmi.

2.2.6.1 MCML PPP

L'estensione MCML per PPP è stata realizzata per fornire un meccanismo che consentisse di svolgere la frammentazione al livello OSI senza coinvolgere i livelli superiori. Il protocollo Multilink PPP ha già un meccanismo di frammentazione che consente di inviare i pacchetti attraverso due circuiti fisici differenti e di riassemblarli all'altro capo della comunicazione. Era però necessario un metodo per utilizzare il protocollo Multilink PPP su un unico mezzo fisico applicando più code e che consentisse l'estrazione dalla coda dei singoli frammenti e non dell'intero pacchetto.

MCML PPP richiede ancora che i frammenti vengano classificati in base alla precedenza IP e a RSVP e che siano accodati con una valutazione WFQ.

2.2.6.2 FRF.12

FRF.12 consente alle reti Frame Realy di operare in modo simile a MCML PPP. Poiché questa frammentazione si verifica anche al livello di collegamento, i protocolli di livello superiore non sono interessati dalla frammentazione.

Come nel caso di MCML PPP i pacchetti vengono frammentati all'ingresso della rete geografica e riassemblati quando vengono ricevuti dal router.

FRF.12 specifica anche l'interconnessione fra reti Frame Relay e ATM. Questo consente ai pacchetti frammentati da FRF.12 sul lato Frame Relay di essere riassemblati sul lato ATM.

2.2.6.3 IP MTU e MTU

Su Interfacce di rete geografiche che non supportano MCML PPP o FRF.12 è possibile impostare il valore MTU dell'interfaccia o del protocollo a un valore inferiore obbligandolo così alla frammentazione.

Normalmente il valore MTU su un'interfaccia è di 1500 byte. Con FRF.12 o MCML PPP è possibile cambiare le dimensioni del pacchetto inviato all'interfaccia senza disturbare il flusso dei pacchetti. Quando si riducono le dimensioni del valore MTU e IP MTU, la variazione dei pacchetti vale solo per la durata della trasmissione di quegli stessi pacchetti.

D'altro canto, cambiare le dimensioni dei pacchetti IP su base generale può provocare vari problemi. Ad esempio si può provocare un degrado globale delle prestazioni della stazione ricevente che si trova a gestire una grande quantità di piccoli pacchetti. Inoltre l'intestazione del pacchetto deve essere replicata per ogni frammento del pacchetto.

Un altro grave problema di IP MTU e MTU è il fatto che se in un pacchetto è impostato il bit DNF (Do Not Fragment), tale pacchetto verrà scartato. E' questo il caso di aggiornamenti di instradamento, gli aggiornamenti Frame Relay LMI (Local Management Interface) e per altri protocolli che non supportano la frammentazione.

2.2.7 Trasporto ad alta velocità

Con questo paragrafo si inizia a parlare delle parte backbone della rete. Poiché questa è completamente differente dalla parte che si trova ai margini, i meccanismi per la qualità di servizio sono diversi.

Si può definire trasporto ad alta velocità qualsiasi interfaccia di velocità superiore a T1. In genere alle interfacce ad alta velocità non possono essere applicate le stesse regole e politiche impiegate per le interfacce a bassa velocità. Ciò è dovuto principalmente al fatto che più politiche e meccanismi di qualità si applicano e maggiore è il tempo impiegato da un router per inoltrare un pacchetto. Mentre questa situazione è accettabile per le interfacce a bassa velocità, le interfacce ad alta velocità hanno a disposizione un tempo inferiore per identificare e accodare i pacchetti.

Dunque, a mano a mano che i clienti impiegano interfacce a velocità sempre più elevate è importante fornire loro delle opzioni che li mettano in grado di determinare il modo in cui attivare correttamente la qualità di servizio sulle proprio reti.

2.2.7.1 POS

Con la capacità IP di viaggiare direttamente su un'infrastruttura di rete SONET (Synchronous Optical NETwork) è nata l'esigenza di determinare la priorità del traffico su un'interfaccia ad alta velocità. Prima di inserire il protocollo IP direttamente su SONET, veniva impiegato il meccanismo di trasporto ATM. Anche se è possibile fornire degli schemi di priorità utilizzando strumenti di qualità e di servizio ATM, sono stati sviluppati meccanismi di qualità di servizio specifici per IP in grado di gestire interfacce ad alta velocità.

Uno di questi è l'MDRR (Modified Deficit Round Robin) che estende il sistema DDR (Deficit Round Robin) per introdurre le priorità per il

traffico in tempo reale come Voice Over IP. In MDRR i pacchetti IP vengono inviati a code aventi classi di servizio differenti a seconda dei bit di precedenza.

Tutte le code vengono servite secondo uno schema circolare (l'algoritmo Round Robin) tranne una: una coda a priorità utilizzata per gestire il traffico vocale.

DDR fornisce un sistema di accodamento simile al WFQ ma per interfacce ad alta velocità, MDRR estende tale protocollo includendo una coda ad alta priorità che viene trattata in modo differente dalle altre code associate alle classi di servizio.

Per ogni insieme di code (caratterizzate dalle classi di servizio), MDRR include una coda a bassa latenza e alta priorità per il traffico Voice Over IP e comunque per il traffico in tempo reale.

Questo consente ai provider e a tutti coloro che hanno bisogno di impiegare un sistema di accodamento ad alta velocità di assicurare che i pacchetti o Voice Over IP abbia priorità più elevata rispetto a tutto l'altro traffico.

2.2.7.2 IP e ATM

Attualmente molte reti utilizzano come livello di trasporto ATM con IP. Per questo motivo è necessario mappare la qualità di servizio IP su una rete ATM.

Ciò è possibile in due modi. Il primo metodo non fa altro che mappare i valori di precedenza di un pacchetto IP su circuiti PVC ATM differenti. Questo consente all'amministratore di rete di avere vari circuiti PVC e

di allocare il traffico più importante su un circuito ATM VBR (Variable Bit Rate) e il traffico meno importante su un circuito ATM UBR (Unspecified Bit Rate).

E' possibile avere circuiti PVC di velocità variabile e, da un lato continuare a inviare deliberatamente i dati a un PVC congestionato e dall'altro inviare la voce e tutto il traffico in tempo reale a un PVC meno congestionato.

Il secondo metodo mappa le priorità IP su ATM utilizzando tecniche di accodamento come WFQ per definire le priorità dei vari flussi inviati ai PVC.

2.2.8 Accodamento ad alta velocità

Come si è detto in precedenza, i meccanismi di accodamento gestiscono le congestioni e determinano la priorità del traffico più importante.

I sistemi per evitare le congestioni affrontano un problema simile ma da un punto di vista completamente differente. Invece di gestire la congestione esistente, si cerca di evitare del tutto che si formi una congestione. Semplificando, si può evitare la congestione scartando i pacchetti di vari flussi per rallentare le applicazioni in modo che riducano la frequenza di invio dei pacchetti.

Questo evita un fenomeno chiamato sincronizzazione globale che si verifica quando vengono trasmessi più flussi TCP IP che poi terminano di trasmettere contemporaneamente. Questa situazione è provocata dalla mancanza di qualità di servizio nel backbone del provider.

2.2.8.1 WRED

RED (Random Early Detection) è un meccanismo che evita le congestioni che può essere molto utile nelle reti ad alta velocità. WRED si basa sul fatto che molti sistemi di trasporto dei dati sono sensibili alla perdita di pacchetti e quanto meno rallentano momentaneamente nel caso in cui si perda del traffico.

Per segnalare ad una stazione TCP di smettere di trasmettere, basta scartare un po' del traffico inviato da essa. WRED è l'implementazione di questo meccanismo che evita una situazione di sincronizzazione globale.

WRED combina le funzionalità dell'algoritmo RED e la precedenza IP. Questa combinazione offre la possibilità di gestire il traffico preferenziale per i pacchetti di priorità più elevata. Questo algortimo può scartare selettivamente il traffico a bassa priorità quando l'interfaccia inizia ad essere in stato di congestione e offre caratteristiche prestazionali differenziate a seconda delle classi di servizio richieste.

Capitolo 3

RTP: REAL-TIME TRANSFER PROTOCOL

RTP è lo standard per la trasmissione del traffico sensibile ai ritardi lungo reti a pacchetti. RTP si basa su UDP e IP e fornisce alle stazioni riceventi varie informazioni aggiuntive che non sono contenute nei flussi senza connessione UDP/IP. Due elementi importanti delle informazioni sono la sequenzialità e la cronologia. RTP usa le informazioni sulla sequenza per determinare se i pacchetti stanno arrivando secondo l'ordine corretto e le informazioni di cronologia per determinare il ritardo nell'arrivo dei pacchetti (jitter/tremolio). E' possibile usare il protocollo RTP per trasmissioni "on-demand" o per servizi interattivi quali la telefonia via Internet. Il protocollo RTP è costituito da una parte di dati e una parte di controllo; quest'ultima si chiama RTCP (Real Time Control Protocol).

La parte di dati RTP è un protocollo leggero che offre il supporto per applicazioni operanti in tempo reale, il rilevamento delle perdite e l'identificazione del contenuto.

Il protocollo RTCP fornisce il supporto per conferenze in tempo reale via Internet di gruppi di qualsiasi dimensione. Questo supporto include l'identificazione dell'origine e il supporto dei gateway come ad esempio i bridge audio e video e i traduttori da multicast a unicast. Inoltre offre un rilevamento della qualità del servizio dai ricevitori al gruppo multicast e un supporto per la sincronizzazione di vari flussi di dati.

L'utilizzo di RTP è importante per il traffico in tempo reale ma esistono alcuni problemi. Le intestazioni IP/RTP/UDP occupano rispettivamente 20, 8 e 12 byte per un totale di 40 byte, ovvero il doppio del carico utile quando si usa G.729 con due campioni vocali (20ms). Utilizzando il meccanismo cRTP (Compression header RTP, vedi paragrafo 2.2.2) si può comprimere questa intestazione in 2 o 4 byte.

3.1 Cenni sul protocollo UDP

Il protocollo UDP, diversamente dal TCP, è un protocollo senza connessione con un'intestazione più compatta che si traduce in un minor carico della rete.

L'intestazione UDP contiene solo quattro campi: la porta di origine, la porta di destinazione, la lunghezza e il checksum UDP. I primi due campi identificano i punti in cui i processi di origine e destinazione del livello superiore ricevono i servizi UDP. Il campo lunghezza indica la

lunghezza dell'intestazione e dei dati e il campo checksum offre la possibilità di verificare l'integrità dei pacchetti. Il checksum UDP è opzionale.

Il protocollo UDP viene utilizzato da Voice Over IP per trasportare il traffico vocale. Non viene utilizzato il protocollo TCP in quanto le funzioni di controllo del flusso e di ritrasmissione dei pacchetti audio non sono necessarie. Poiché per trasmettere il flusso audio viene impiegato il protocollo UDP, questo continua a trasmettere indipendentemente dal tasso di perdita dei pacchetti che può andare dal 5% fino al 50%.

Se per Voice Over IP venisse utilizzato il protocollo TCP, la latenza necessaria per i messaggi di acknowledgement e di ritrasmissione renderebbe inaccettabile la qualità della voce. Con Voice Over IP e altre applicazioni in tempo reale, è più importante controllare la latenza che garantire un trasferimento affidabile di ogni singolo pacchetto.

Il protocollo TCP viene però utilizzato per l'instaurazione della chiamata nella maggior parte dei protocolli di segnalazione Voice Over IP.

Bisogna fare un cenno anche al protocollo RUDP (Reliable User Datagram Protocol) che aumenta l'affidabilità del protocollo senza connessione UDP. Fondamentalmente RUDP invia più copie dello stesso pacchetto e consente alla stazione ricevente di eliminare i pacchetti non necessari o ridondanti. Questo meccanismo aumenta le probabilità che uno dei pacchetti riesca a giungere al ricevente.

Esistono poche implementazioni di questa tecnica, però, a causa delle considerazioni relative all'ampiezza di banda (occorre utilizzare un'ampiezza di banda doppia o tripla). Tuttavia i clienti che hanno a disposizione un'ampiezza di banda quasi illimitata possono considerare la possibilità di impiegare tale meccanismo per migliorare l'affidabilità e la qualità della voce.

3.2 Caratteristiche generali dell'RTP

Il protocollo RTP si colloca al livello trasporto della scala OSI, e considera di avere a disposizione le funzioni offerte da UDP: quindi una connessione orientata ai datagrammi, completa di un controllo sul contenuto dei pacchetti, ottenuto attraverso una checksum, e di funzionalità di multiplexing alla sorgente ed alla destinazione, ottenute attraverso le porte.

Poiché il protocollo deve essere in grado di integrarsi in protocolli specifici, è necessario che la descrizione del contenuto informativo del pacchetto (il payload) possa essere definito dinamicamente, e comunque rientrare nell'intestazione del pacchetto. Questa descrizione potrà non essere standard, ed essere specifica per l'applicazione in cui RTP viene integrato.

Le informazioni trasferite devono essere collocate rispetto a due distinte specifiche: la sequenza di spedizione, che deve essere in grado di riprodurre caratteristiche simili a quelle di un circuito virtuale in cui l'ordine di spedizione dei dati viene ricostruito all'arrivo, e la sequenza

temporale, potenzialmente indipendente ed incoerente rispetto alla prima, e che tiene conto solo dell'istante in cui l'informazione è stata prodotta dalle sorgenti.

Infine, seppure il protocollo non fornisca garanzie circa la temporizzazione delle comunicazioni, nondimeno deve fornire il modo per poterle controllare. Come già detto quindi, al protocollo RTP è affiancato un protocollo di controllo, Real Time Control Protocol (RTCP) per controllare, attraverso messaggi di eco e simili, che le prestazioni della rete siano quelle attese.

3.3 Il funzionamento del miscelatore

Il funzionamento di un host che operi da miscelatore illustra bene le potenzialità del protocollo RTP.

I flussi di informazioni multimediali possono generalmente essere miscelati: non solo su domini diversi (ad esempio audio e video) ma anche sullo stesso dominio. Ad esempio più immagini possono essere presentate simultaneamente sullo schermo, o più suoni possono essere riprodotti simultaneamente attraverso un altoparlante.

Per ottenere questo è necessario che i diversi flussi siano sincronizzati, e che l'effetto della miscelazione sia di presentare insieme eventi effettivamente simultanei. Altrimenti il significato della comunicazione può essere assolutamente falsato.

Utilizzando RTP è possibile miscelare i flussi di informazione proveniente da fonti diverse: inoltre questa funzionalità può essere

associata ad uno specifico host, che riceve i flussi provenienti dai diversi partecipanti ad una videoconferenza (ad esempio), e fornisce a ciascuno una riproduzione "miscelata" di ciò che avviene nei diversi siti coinvolti.

Questa soluzione consente anche di ridurre notevolmente il carico di rete, ciò che è sempre opportuno.

Un'altra funzione simile alla miscelazione è la traduzione. L'organizzazione della rete, statica o dinamica che sia, può rendere un protocollo di codifica dell'informazione più opportuno di altri su determinate connessioni.

Un host traduttore può trasformare il flusso informativo ricodificandolo nella forma più adatta. In questo caso, oltre alla necessaria informazione di sincronizzazione, sarà anche necessaria una descrizione del genere di codifica utilizzata per rappresentare il flusso informativo. Questa descrizione potrà essere specifica per l'applicazione, ma dovrà essere presente nella parte descrittiva di ogni singolo pacchetto ricevuto.

Questa funzione è associata, in RTP, ad un campo specifico nell'intestazione del pacchetto, che descrive la codifica del payload, l'informazione trasferita nel pacchetto.

3.4 L'intestazione dei pacchetti RTP

Si può vedere l'intestazione di un pacchetto RTP in figura:

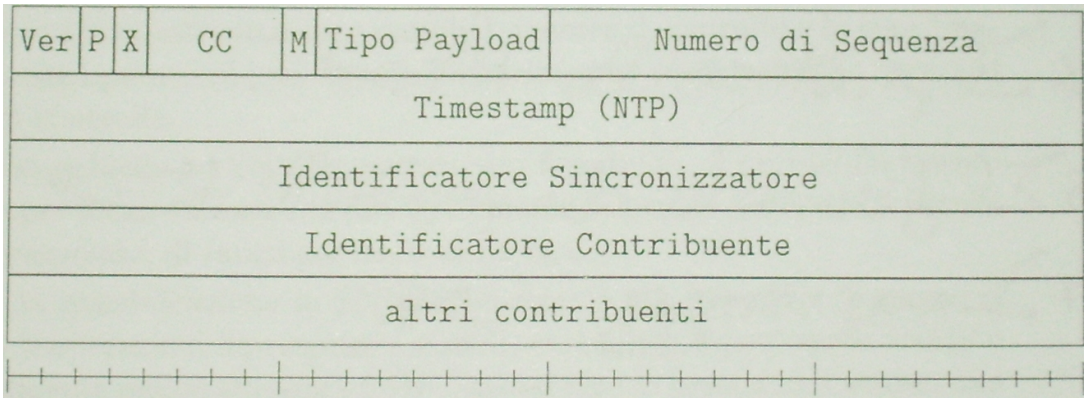


FIGURA 3.1: Intestazione di un pacchetto RTP

Ver: il campo indica il numero di versione, attualmente il 2.

P: il campo segnala la presenza di ottetti di padding, necessari a riallineare i dati al termine del pacchetto. Si rende necessario per alcuni algoritmi di crittazione.

X: il campo segnala che lo header RTP è esteso con informazioni di intestazione specifiche per l'applicazione in cui RTP è stato integrato. Come abbiamo visto RTP non è concepito come un protocollo a sé stante, ma come modulo che può essere integrato in una applicazione più complessa. Questo bit segnala che, oltre lo header RTP, è presente anche uno header per l'applicazione.

CC: il valore del campo corrisponde al numero di host che contribuiscono al flusso informativo trasportato dal pacchetto, fino ad un massimo di 16.

M: è un marker generico, al quale l'applicazione specifica può attribuire un significato determinato (ad esempio può indicare il primo pacchetto spedito). Anche questo dettaglio è in vista della integrazione del protocollo RTP nella applicazione.

Tipo Payload (Payload Type): il campo specifica il tipo di payload. Alcuni codici sono descritti da uno standard, ma il significato di questo campo viene ancora lasciato all'applicazione, e quindi definito solo dopo l'integrazione di RTP nella applicazione (per sapere i diversi tipi di payload in dotazione si veda il paragrafo 2.1.5.1).

Numero di Sequenza: il campo indica il Numero d'ordine riferito al flusso di dati, e viene incrementato di una unità ad ogni pacchetto spedito.

Timestamp (NTP): nel campo è indicato un valore numerico che corrisponde ad una misura dell'istante in cui il primo ottetto nel campo dati viene prodotto. La misurazione del tempo deve avere caratteristiche di crescita lineare nel tempo, ma non deve effettivamente aderire ad uno standard di rappresentazione del tempo: anche questo può dipendere dal tipo di carico utile. La granularità di rappresentazione del tempo deve essere tale da consentire la corretta sincronizzazione della comunicazione.

Identificatore Sincronizzatore (Synchronization Source - SSRC): il campo indica l'Identificatore Sincronizzatore, cioè l'identificatore della sorgente che ha fornito la sincronizzazione per il flusso di informazione: nel caso l'informazione sia frutto di un'operazione di miscelazione, qui si tratterà del miscelatore. Il numero è scelto a caso, ma le probabilità di collisione sono molto basse: quindi non coincide con un indirizzo Internet, e lo stesso host può essere sorgente di flussi diversi. Ad esempio, se lo stesso host genera audio e video, per ciascuno di questi sarà scelto un SSRC differente, ed i due flussi distinti. Questa soluzione è preferibile a miscelare audio e video nello stesso flusso.

Identificatore Contribuente (Contributing Source - CSRC): il campo contiene l'Identificatore Contribuente, cioè l'identificatore di una delle sorgenti che contribuiscono al flusso: infatti questo può essere frutto di una miscelazione di più flussi. La lista, la cui lunghezza è indicata nel campo CC, non può contenere più di 16 identificatori.

Altri contribuenti: il campo indica l'eventuale presenza nell'intestazione di informazioni extra. L'aggiunta di esse è a discrezione dell'applicazione.

3.5 Il protocollo RTCP

Il protocollo RTCP serve a scambiare informazioni riguardanti il funzionamento RTP. Il funzionamento di questo protocollo avviene su una porta distinta da quella utilizzata da RTP. Quindi il funzionamento di RTP comporta l'uso di due porte: una per RTP e l'altra per RTCP.

I messaggi di RTCP vengono sempre diffusi a tutti i partecipanti alla sessione: quindi diventa particolarmente interessante un supporto che consenta il multicast.

Lo scopo primario RTCP è quello di consentire al mittente di conoscere la reazione del destinatario, in modo che il primo abbia informazioni sulla qualità della comunicazione. Infatti una funzione essenziale di un protocollo a livello trasporto è proprio il controllo del flusso, che si ottiene tramite questo genere di meccanismo, detto feedback.

Tra l'altro, la diffusione di questi messaggi rende possibile anche il controllo di determinati aspetti della comunicazione (ad esempio codifica o routing) da parte di entità terze (third party), non direttamente interessate nelle informazioni scambiate, ma responsabili della comunicazione.

Un'altra funzione del protocollo RTCP consiste nell'associare agli identificatori delle sorgenti di sincronismo uno specifico indirizzo al livello trasporto (in Internet una coppia costituita da un indirizzo Internet e una porta). L'informazione può essere necessaria per collocare la sorgente di un flusso di informazioni nell'Internet, invece che nello spazio anonimo dei generatori di sincronismo.

Inoltre, il multicast dei messaggi RTCP consente ad ogni partecipante di conoscere il numero di partecipanti alla sessione: questo può essere indispensabile, ad esempio, per moderare il numero di messaggi di controllo, che rischierebbero altrimenti di saturare la rete.

Una ulteriore funzionalità associata a RTCP consiste nello scambiare informazioni identificative dei partecipanti, collocandoli nel mondo fisico: nome, indirizzo di posta elettronica, collocazione geografica ecc.

Poiché i pacchetti RTCP non vengono spediti in risposta ad eventi specifici, ma periodicamente, è possibile economizzare rispetto al carico imposto alla rete racchiudendo, quando possibile, messaggi provenienti da diversi partecipanti.

3.6 Il formato dei pacchetti RTCP

I pacchetti RTCP sono di 5 tipi diversi:

SR sta per *sender report* e contiene il rapporto di ricezione e trasmissione da parte di partecipanti che trasmettono.

RR sta per *receiver report* e contiene il rapporto di ricezione da parte di partecipanti che non trasmettono.

SDES sta per *source description* e contiene la descrizione (compreso l'indirizzo al livello trasporto) di un partecipante.

BYE Indica l'abbandono della sessione.

APP Indica che il tipo è specifico per una certa applicazione.

Tutti si aprono con una identica sequenza, simile a quella RTP, e rappresentata in figura:

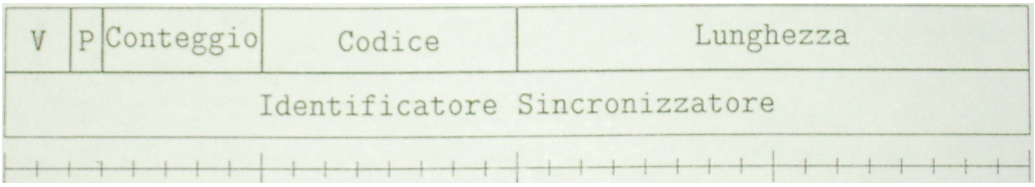


FIGURA 3.2: Intestazione di un pacchetto RTCP

V: il campo indica il numero di versione, sempre la 2.

P: segnala la presenza di padding, come per lo header RTP.

Conteggio: il campo indica il numero di elementi racchiusi nel pacchetto: infatti un pacchetto RTCP può contenere un numero variabile di elementi, tutti dello stesso tipo.

Codice: il campo codifica in quale delle categorie dette sopra ricade il pacchetto: **SR** (codice=200), **RR** (201), **SDES** (202), **BYE** (203), **APP** (204). In quest'ultimo caso il campo **conteggio** assume un significato diverso da quello detto sopra.

Lunghezza: il campo indica la lunghezza del pacchetto, corrispondente al numero di parole di quattro ottetti comprese nel pacchetto (compreso le header) decrementato di una unità.

Identificatore di Sincronizzatore: il campo indica il mittente del pacchetto.

3.6.1 I messaggi RR

Questo tipo di messaggio è composto di un numero variabile di blocchi, di lunghezza costante e corrispondente a 6 parole composte da 4 ottetti ciascuna. Ciascun blocco contiene un rapporto sui pacchetti ricevuti da un certo mittente. Il formato corrispondente è illustrato in figura:

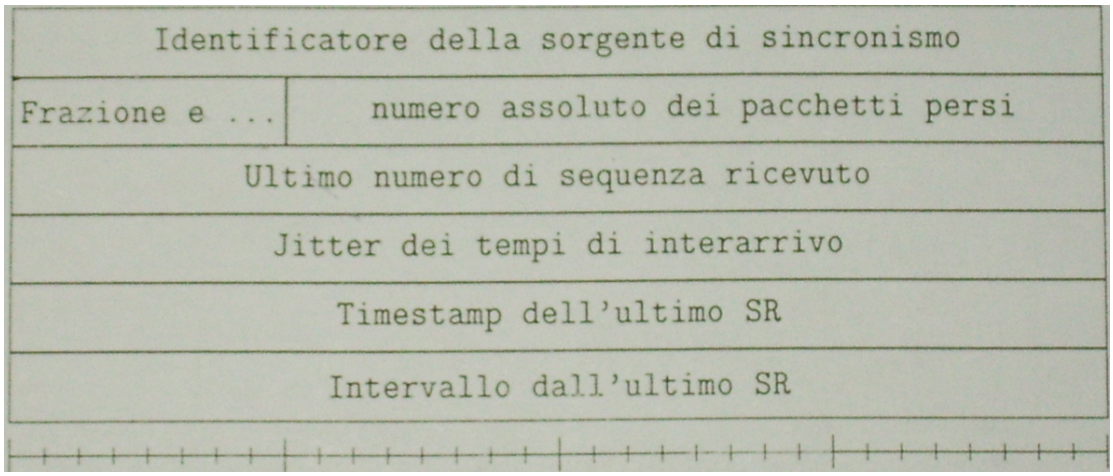


FIGURA 3.3: Intestazione di un messaggio RR

Identificatore di SSRC: l'identificatore del partecipante a cui si riferisce il rapporto.

Pacchetti persi: il campo contiene la frazione di pacchetti che il destinatario non ha ricevuto: il mittente verrà dunque a sapere in

quale misura i suoi messaggi vengono perduti senza raggiungere quello specifico destinatario. La frazione è rappresentata in virgola fissa, con la virgola collocata all'estrema sinistra. Quindi la cifra indicata corrisponde a 256-esimi, e per ottenere la frazione decimale è necessario dividere il valore del campo per 256.
Vengono conteggiati tutti i pacchetti perduti, a partire dal numero d'ordine iniziale a quello che viene riscontrato con questo pacchetto.

Ultimo numero d'ordine ricevuto: questo campo serve da riscontro. Il numero d'ordine (che è di soli 16 bit) viene esteso con un prefisso che consente sessioni prolungate senza ripetizioni di numeri d'ordine.

Jitter dei tempi di interarrivo: questo campo contiene un parametro numerico che serve a valutare la regolarità della comunicazione. Il jitter quantifica la differenza media tra i ritardi di propagazione dei pacchetti. Quindi si tratta di una valutazione che non tiene conto del ritardo di propagazione in assoluto, ma della regolarità di tale ritardo. Se S_i ed S_j sono i tempi di spedizione di due pacchetti, e R_i ed R_j sono i tempi di ricezione degli stessi pacchetti, la formula per il calcolo della deviazione è la seguente:

$$Deviazione(i, j) = \left|(R_j - S_j) - (R_i - S_i)\right|$$

Per ottenere il jitter, si accumulano i valori di deviazioni successive, dando maggior peso a quelli più recenti. La formula è la seguente:

$$Jitter(i) = Jitter(i-1) + \frac{Deviazione(i-1, i) - Jitter(i-1)}{16}$$

L'effetto è quello di ottenere una valutazione media della variabilità dei tempi di comunicazione che dia maggior peso ai dati più recenti, e meno a quelli lontani nel passato.

Timestamp dell'ultimo SR: questo campo contiene i 32 bit centrali della rappresentazione NTP del tempo di ricezione dell'ultimo pacchetto RTCP ricevuto da quella sorgente. Da notare che questa regola non è necessariamente la stessa utilizzata per assegnare i timestamp dei pacchetti RTCP dei dati. Il timestamp così prodotto ha una risoluzione di 15μs circa.

Intervallo dall'ultimo SR: in questo campo viene indicato il ritardo, espresso con la stessa notazione del campo precedente, tra la ricezione dell'ultimo pacchetto da quella sorgente, e la spedizione di questo pacchetto, misurati sul clock locale.

Utilizzando il campo precedente e questo campo, la sorgente di sincronismo che riceve questo messaggio può ottenere un campione del ritardo di comunicazione del messaggio: se il rapporto riporta un timestamp dell'ultimo SR S, ed un intervallo dall'ultimo SR *d*, la sorgente di sincronismo, ricevendo il rapporto e leggendo il clock interno *t* potrà stimare il tempo di andata e ritorno (roundtrip):

$$Roundtrip = (t - S) - d$$

3.6.2 I messaggi SR

I messaggi SR si distinguono dagli RR perché si aprono con un rapporto sull'attività del mittente del messaggio. Poi proseguono con una sequenza di rapporti di ricezione analoghi a quelli descritti per i messaggi RR.

Il blocco del messaggio SR relativo al mittente ha il formato illustrato nella figura:

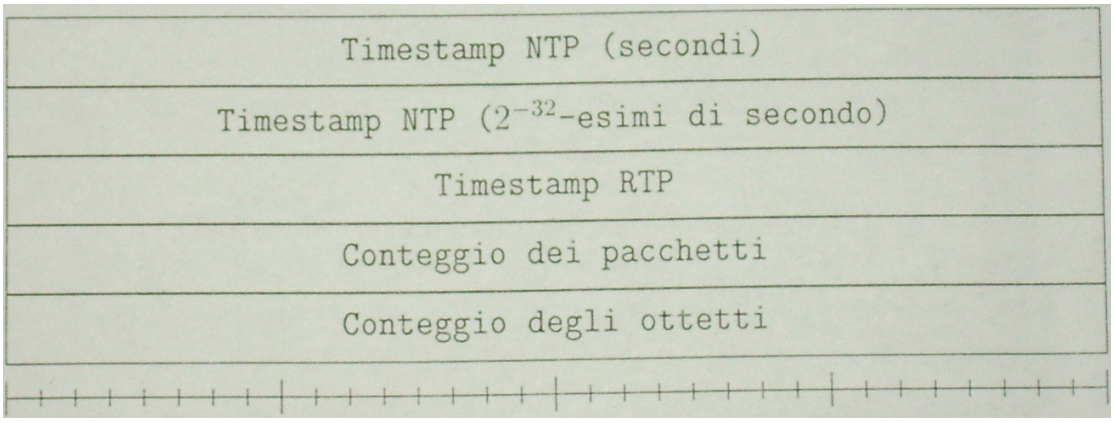


FIGURA 3.4: Intestazione di un messaggio SR

Timestamp NTP: il campo indica il tempo trascorso dal 1 Gennaio 1970. La rappresentazione del tempo secondo lo standard occupa 64 bit, corrispondenti ad 8 ottetti. I primi 4 indicano i secondi interi dall'epoca iniziale, mentre i successivi 4 indicano la frazione di secondo, ancora in virgola mobile collocata all'estrema sinistra. Quindi la risoluzione è di 2^{-4} μs circa. Da qui viene estratto il valore S utilizzato per la determinazione del roundtrip.

Timestamp RTP: il campo contiene il timestamp prodotto con la stessa regola utilizzata per produrre i timestamp nei pacchetti RTP. Questa informazione, insieme alla precedente, consente di sincronizzare tra loro i flussi provenienti da sorgenti distinte, a condizione che il loro meccanismo di generazione dei timestamp NTP sia sincronizzato.

Conteggio dei pacchetti: il campo indica il numero di pacchetti generati dalla sorgente, dall'inizio della trasmissione.

Conteggio ottetti: il campo indica il numero totale di ottetti di carico utile prodotti dalla sorgente, dall'inizio della trasmissione. Può essere utilizzato per valutare la velocità con cui la sorgente produce dati.

3.6.3 I messaggi SDES

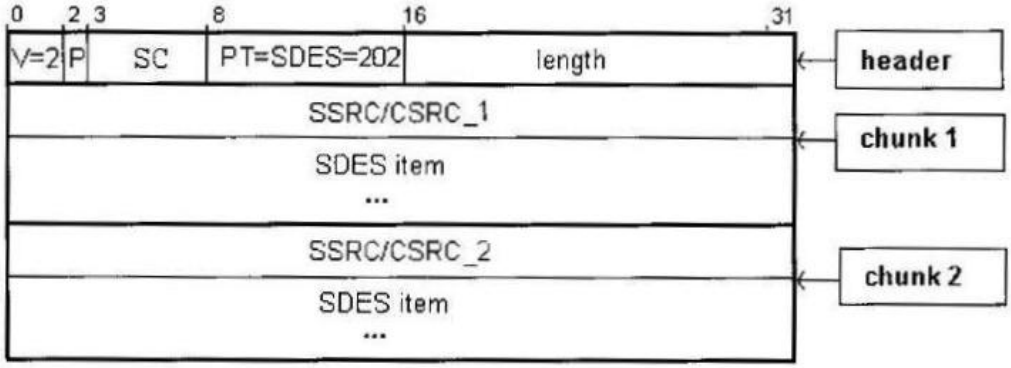


FIGURA 3.5: Formato di un blocco SDES

Il pacchetto SDES è costituito da un header e da zero o più porzioni (chunk) ognuna delle quali contiene una descrizione della

sorgente.

I campi non ancora visti negli altri pacchetti RTCP sono:

source count (SC): 5 bit, indica il numero di chunk che seguono.

SDES item è costituito da 3 campi:

- un identificatore del tipo di item (NAME,CNAME,EMAIL..)
- un campo con la lunghezza dell'item
- il campo che porta l'informazione sulla sorgente.

3.6.4 I messaggi APP e BYE

Panoramica dei pacchetti:

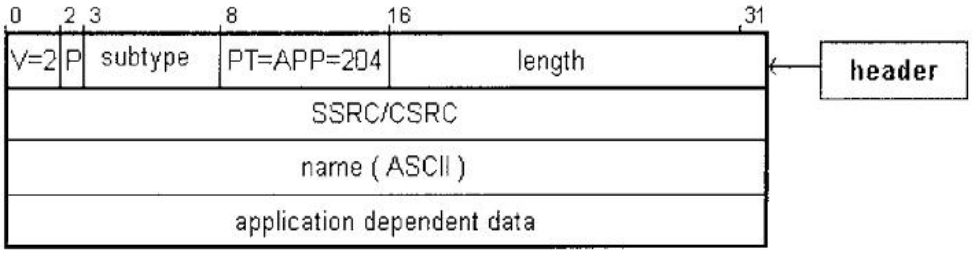


FIGURA 3.6: Formato di un blocco APP

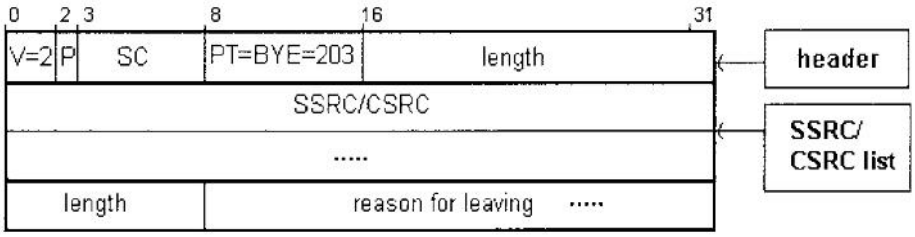


FIGURA 3.7: Formato di un blocco BYE

3.7 Il protocollo RTP in un applicativo VoIP

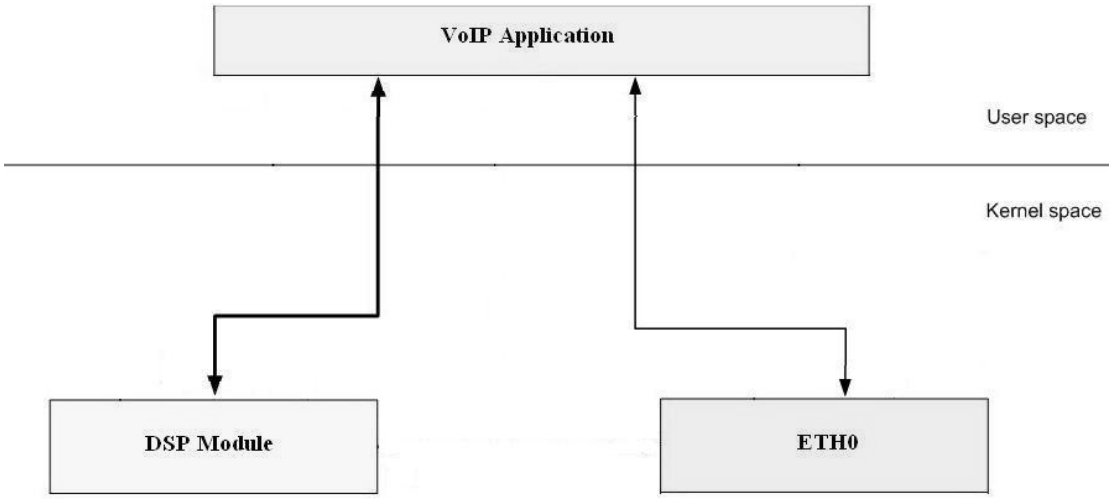


FIGURA 3.8: Grafico di funzionamento di un applicativo VoIP

Un applicativo VoIP, come già detto nel capitolo precedente, è una tecnologia che rende possibile effettuare una conversazione telefonica sfruttando una connessione Internet o un'altra rete dedicata che utilizza il protocollo IP, anziché passare attraverso la rete telefonica tradizionale (PSTN). Poiché l'RTP è un protocollo del livello applicazioni, ogni programma in user space che prevede il suo uso ha la piena facoltà di costruirne l'header e di aggiungere informazioni proprie al payload del pacchetto.

In *Figura 3.8* viene mostrato in linea di massima il funzionamento di un applicativo VoIP.

Tralasciamo la fase di chiamata ad un altro calcolatore nella rete (tramite il protocollo SIP) i cui dettagli non sono scopo di questo

paragrafo. Partiamo, invece, dal momento in cui due calcolatori hanno già instaurato una comunicazione tramite il proprio applicativo VoIP, cioè nel momento in cui i due user sono in contatto ed iniziano a parlare. E' qui che inizia l'uso del protocollo RTP.

L'applicativo VoIP tramite il driver DSP cattura un pezzetto di voce (come già detto nel capitolo precedente, di solito ogni trama di voce viene presa ogni 30 millisecondi). Dopo questa operazione, l'applicativo codifica la trama secondo un codec già deciso in precedenza (compito del protocollo SIP) e costruisce il pacchetto RTP. Viene settato l'header, alcuni campi hanno bisogno di nota:

P: se ci sono delle informazioni da allineare assume valore diverso da 0.

X: se sono presenti informazioni aggiuntive specifiche dell'applicativo assume valore diverso da 0 (la loro ubicazione nel pacchetto dovrebbe essere subito dopo l'header).

M: di solito assume valore 1 se l'header è relativo al primo pacchetto spedito dall'applicativo; valore 0 altrimenti.

PT (Payload Type): è il numero relativo al tipo di codec utilizzato per codificare/decodificare le trame.

Synchronization Source (SSRC): l'identificatore della sorgente. Questo valore varia per ogni sessione, se lo stesso calcolatore è in comunicazione con diversi altri, l'applicativo distingue le varie conversazioni tramite questo valore.

In seguito nel payload del pacchetto viene inserita la trama codificata; la spedizione di esso avviene tramite il driver di rete Eth0.

Nel momento in cui l'applicativo, invece, riceve un pacchetto dal driver di rete, questo dapprima legge l'header RTP per sapere se ci sono informazioni aggiuntive o dati da riallineare. Recupera l'eventuale header relativo all'applicativo, dopodichè decodifica la trama secondo il codec indicato e lo manda in output tramite il driver DSP.

Gli applicativi VoIP tuttora in commercio vengono eseguiti in user space. Questo vuol dire che se il calcolatore ha molti processi in corso, possono verificarsi dei rallentamenti nelle conversazioni. Tra l'altro, poiché ogni sistema operativo dà una precedenza particolare a tutti i processi a livello kernel, un applicativo VoIP in user space è già svantaggiato in partenza.

Capitolo 4

SISTEMI EMBEDDED LINUX

I sistemi embedded sono presenti ovunque nella nostra vita quotidiana. Telefoni cellulari, sistemi di navigazione aerea, lettori MP3, stampanti, router e tutti i dispositivi a microprocessore che ci circondano sono sistemi embedded.

Una delle ultime tendenze è quella di utilizzare il sistema operativo Open Source Linux nei sistemi embedded; i motivi sono molteplici: la sua flessibilità, la sua robustezza, il suo prezzo, la comunità di sviluppatori ed il numero sempre crescente di dispositivi supportati.

Il termine "Linux" è spesso utilizzato in maniera intercambiabile per riferirsi al kernel di Linux, ai sistemi Linux o ad una distribuzione Linux. In realtà, la parola Linux si riferisce al kernel mantenuto da Linus Torvalds e distribuito sotto lo stesso nome dal repository principale o dai vari mirror.

Nella prima accezione, Linux include solo il kernel, e nessun'altra applicazione. Non appena il codice del kernel viene eseguito, prende il

controllo dell'hardware sottostante e fornisce un livello di astrazione verso il livello applicativo.

Se consideriamo un sistema Linux, intendiamo un sistema hardware che utilizza il kernel di Linux ed altre applicazioni software (come, ad esempio, le librerie C ed altre utility GNU, oppure il sistema X Window); in questo senso, un sistema Linux può essere personalizzato a piacere dell'utente/sviluppatore.

Se parliamo di distribuzione Linux, infine, intendiamo un insieme di applicazioni e librerie software, oltre al kernel del sistema operativo, distribuiti per fornire all'utente una serie di procedure di installazione e configurazione hardware/software facilitate. Esempi di note distribuzioni sono la Debian, la Red-Hat, la Slackware, la Mandriva, la MontaVista, ecc.

Combinando i termini "sistema embedded" e "Linux", otteniamo una duplice interpretazione possibile: "sistema embedded Linux", e "distribuzione embedded Linux".

Un sistema embedded Linux si riferisce semplicemente ad un sistema embedded basato sul kernel di Linux, e non implica l'utilizzo di alcuna libreria o applicazione specifica.

Una distribuzione embedded Linux, invece, può includere un framework di sviluppo per sistemi embedded Linux (cross-compilatori, debuggers, software per la gestione dei progetti) e vari applicativi mirati all'utilizzo in sistemi embedded Linux (librerie particolari, script, file di configurazione).

4.1 Classificazione dei sistemi embedded Linux

I parametri utilizzati per classificare i sistemi embedded Linux possono essere i seguenti: grandezza, vincoli temporali, utilizzo della rete, interazione con l'utente.

4.1.1 Grandezza

La grandezza di un sistema embedded Linux è determinata da un numero di fattori differenti. Innanzitutto, la grandezza fisica: alcuni sistemi possono essere molto grandi (come sistemi di cluster), altri molto piccoli (come il Linux watch dell'IBM). Più significativa è la misura delle grandezze dei componenti elettronici del sistema, come la velocità della CPU, la misura della RAM e della memoria di massa.

I sistemi embedded Linux, quindi, possono essere piccoli, medi o grandi; quelli piccoli sono caratterizzati da processori a basse frequenze, con un minimo di 2MB di ROM e 4MB di RAM. I sistemi di grandezza media sono equipaggiati con una CPU media, con circa 32MB di ROM e 64MB di RAM; la maggior parte dei dispositivi orientati al consumo che utilizzano Linux appartengono a questa categoria. Sono infatti inclusi PDA, lettori MP3 e dispositivi di rete. I sistemi grandi sono caratterizzati da processori molto veloci (o un insieme di processori combinati) con molta memoria RAM e memorie di massa molto capaci. Di solito, questi sistemi sono utilizzati in ambienti che richiedono grande potenza di calcolo per portare a termine il proprio lavoro. Esempi di questi sistemi sono i commutatori delle compagnie telefoniche o i simulatori di volo.

C'è da notare che Linux non supporta (almeno nativamente) processori con architettura minore di 32 bit. Effettivamente, in accordo con gli standard tradizionali dei sistemi embedded, i sistemi che utilizzano Linux dovrebbero essere classificati come grandi sistemi; è pur vero che i processori sono sempre più veloci, le memorie più capaci e meno costose, cosicché i grandi sistemi potrebbero presto diventare lo standard.

4.1.2 Vincoli temporali

Ci sono due tipi di vincoli temporali per i sistemi embedded: stringent e mild.

Il vincolo "stringent" richiede che il sistema reagisca all'interno di un arco temporale predefinito. Questa categoria di sistemi si dice "real-time", e si differenzia in "hard real-time" (le operazioni da eseguire devono essere portate a termine immediatamente; il fallimento potrebbe portare a conseguenze catastrofiche) e "soft real-time" (il fallimento delle operazioni da eseguire non è così grave come per i sistemi hard rea-time).

Il vincolo "mild", invece, non impone restrizioni sui tempi di esecuzione delle operazioni.

4.1.3 Utilizzo della rete

L'utilizzo della rete (networability) definisce se un sistema può essere connesso ad una rete. Ai giorni nostri ci si aspetta che tutto sia accessibile tramite la rete. Uno dei motivi per cui gli sviluppatori sono

sempre più portati ad utilizzare Linux nei sistemi embedded è proprio la sua capacità di sfruttare la rete.

4.1.4 Interazione con l'utente

Il grado di interazione con l'utente varia enormemente da un sistema ad un altro. Alcuni sistemi, come i PDA, sono concentrati sull'interazione con l'utente, mentre altri, come sistemi di controllo di processi industriali, possono solo avere LED e bottoni. Altri sistemi possono non avere affatto interfaccia.

4.2 Metodologie di progettazione e di implementazione

La progettazione e l'implementazione di un sistema embedded Linux fanno parte di un processo che include più compiti, alcuni dei quali possono essere svolti in parallelo, altri possono essere omessi (se viene utilizzata una distribuzione).

4.2.1 Creare un sistema Linux target

Creare un sistema Linux target significa configurare e interfacciare insieme i componenti appropriati del sistema. Per fare ciò, bisogna seguire quattro passi fondamentali:

- determinare i componenti del sistema
- configurare e compilare il kernel

- creare il filesystem radice
- mettere a punto il software di boot e la configurazione

Nel prossimo paragrafo saranno descritti in dettaglio i componenti usuali dei sistemi embedded, nonché alcune considerazioni sulla configurazione del kernel, sulla creazione dell'alberatura radice e sull'installazione del bootloader.

4.2.2 Scegliere gli strumenti di sviluppo

Sviluppare per un sistema embedded è differente rispetto a sviluppare software per workstation o server. Innanzitutto, l'ambiente target è spesso diverso dalla macchina sulla quale si sviluppa. Ciò non preclude l'utilizzo della multiarchitettura di Linux, la quale permette di testare le applicazioni sulla stessa macchina su cui si sta sviluppando, apportando solo qualche (o nessuna) modifica; anche se non tutte le applicazioni possono essere provate in questo modo, questo approccio, generalmente, fa risparmiare molto tempo.

Nel paragrafo 4.4 sono presentate alcune soluzioni per quanto riguarda gli strumenti ed i linguaggi di sviluppo.

4.2.3 Sviluppare per sistemi embedded

Uno dei vantaggi nell'utilizzare Linux come sistema operativo in un sistema embedded è che il codice sviluppato per Linux dovrebbe essere eseguito in maniera identica sia sul sistema embedded che su una workstation. Nonostante ciò sia in buona parte vero, bisogna ricordare che le operazioni e le richieste dei sistemi embedded

differiscono molto da workstation e ambiente server.

4.2.4 Configurare la rete

La rete permette ad un sistema embedded di interagire con il mondo esterno. In un sistema Linux embedded bisogna scegliere l'hardware per il networking, i protocolli di rete ed i servizi da offrire (anche per la sicurezza). Un aspetto interessante nello sfruttare i servizi di rete è l'aggiornamento del sistema tramite la rete stessa, in modo da evitare interventi sul posto.

4.3 Setup e configurazione del target

4.3.1 Hardware supportato: processore

Linux supporta un gran numero di architetture, ma non tutte sono utilizzate nelle configurazioni embedded.

- **x86:** la famiglia degli x86 inizia con il 386 introdotto dalla Intel nel 1985, e include tutti i discendenti di questo processore, compresi il 486 e la famiglia dei Pentium (compatibili con i processori AMD). In ogni modo, la Intel rimane il distributore più grande di questa famiglia di processori. Nonostante il x86 sia la piattaforma più popolare e pubblicizzata per eseguire Linux, essa rappresenta solo una piccola frazione dei sistemi embedded presenti sul mercato. Nella maggior parte dei casi, i progettisti preferiscono processori ARM, MIPS e PowerPC a quelli appartenenti alla famiglia i386, per motivi di complessità e costi.

Nonostante ciò, molte applicazioni e add-on vengono sviluppati per i386 prima di essere portate sull'architettura target. Lo stesso kernel di Linux, originariamente, fu scritto per i386, prima di essere portato ad un'altra architettura.

Essendo la maggior parte dei sistemi embedded con architettura i386 molto simili (o identici) a workstation e server per quanto riguarda le funzionalità e la programmazione, il kernel non mostra differenze al variare delle CPU x86. L'architettura i386 è quella più documentata: esistono numerosi libri e documenti online in molte lingue che descrivono questa architettura. Inoltre, anche i distributori forniscono documentazione, spesso molto completa.

- **ARM:** ARM, acronimo di "Advanced RISC Machine" (macchina RISC avanzata), è una famiglia di processori mantenuta e promossa dalla ARM Holdings Ltd. Al contrario di altri produttori di chip come IBM, Motorola e Intel, ARM Holdings non produce propri processori, ma progetta i core delle CPU (basati sui core ARM) e vende i progetti ai propri clienti, i quali producono i chip. Questo approccio offre molti vantaggi alle parti coinvolte, ma, al tempo stesso, crea una certa confusione negli sviluppatori che utilizzano questa architettura per la prima volta. Tutti i processori ARM contengono lo stesso insieme di istruzioni, il che permette di produrre software totalmente compatibile per ogni variante di core. Ciò non significa che tutti i processori ARM possono essere programmate e configurate allo stesso modo,

ma solo che il linguaggio assembly ed i risultanti eseguibili sono identici per ogni CPU ARM. Ad oggi, questa architettura è prodotta da Intel, Toshiba, Samsung e molti altri. L'architettura ARM è molto popolare in molti campi, ed è utilizzata in centinaia di prodotti e servizi. Linux supporta circa 10 CPU ARM, 16 piattaforme differenti, e più di 200 schede ad esse relative.

- **IBM/Motorola PowerPC:** l'architettura PowerPC (PPC) è il risultato della collaborazione tra IBM, Motorola e Apple. Essa eredita delle idee dal lavoro svolto dalle tre compagnie, in special modo dalla "Performance Optimizarion With Enhanced RISC" (POWER) della IBM. Il PPC è maggiormente conosciuto per il suo utilizzo nei Mac della Apple, ma esistono anche workstation della IBM e di altri produttori basate su questo processore. Insieme alla i386 e ad ARM, il PPC è un'architettura consolidata all'interno del kernel di Linux; ciò si nota considerando il gran numero di macchine e sistemi con PPC che utilizzano Linux.

  Per fornire compatibilità con l'hardware, ogni tipo di architettura PPC ha le proprie funzioni di basso livello, raggruppate in files. Inoltre, numerose applicazioni che girano su i386 sono disponibili anche per PPC. Esiste già il supporto per Java, ed OpenOffice è già stato portato per PPC. La comunità Linux per PPC è attiva in molte aree di sviluppo, dalle workstation ai sistemi embedded. Il sito principale per Linux PPC è http://penguinppc.org/, mantenuto dai membri della comunità,

e non affiliato ad un particolare produttore. Il sito contiene documentazione e collegamenti, e dovrebbe essere considerato il punto di partenza per lo sviluppo di Linux per PPC. Esiste anche http://www.linuxppc.org/, affiliato alla distribuzione LinuxPPC, che era la home iniziale per il porting di Linux al PPC. Le aziende che si occupano del PPC si sono accordate su un'architettura standard per sviluppare schede basate su questo chip. Lo standard è stato fornito inizialmente tramite il "PowerPC Reference Platform" (PReP), rimpiazzato dal "Common Hardware Reference Platform" (CHRP). Altra documentazione sulle versioni a 32bit del PPC è disponibile online, nei siti della IBM e della Motorola.

- **MIPS:** il MIPS è figlio di John Hennessey, più conosciuto dagli studenti di Informatica di tutto il mondo per il suo libro sulle architetture, scritto con David Patterson, ed è il risultato del progetto "Microprocessor without Interlocked Pipeline Stages" (MIPS) di Stanford. MIPS è famoso per essere stata la base delle workstation e dei server venduti dalla SGI e delle console di gioco, come i sistemi Nintendo 64bit e Sony Playstation 1 e 2, ma si trova in molti altri sistemi embedded. Proprio come ARM, la MIPS Technologies Inc fornisce la licenza dei core a terzi, ma esistono molte implementazioni degli insiemi di istruzioni, che differiscono tra loro anche di molto. Implementazioni di core MIPS a 32bit sono prodotte da IDT, LSI, NEC, QED, SandCraft e Toshiba.

Il porting iniziale di Linux verso l'architettura MIPS fu svolto principalmente per supportare workstation basate su MIPS; più tardi il porting è stato sviluppato anche per sistemi embedded. Per far funzionare i vari MIPS ed i sistemi su di esso basati, la struttura della porzione di kernel di Linux è divisa in directory, ognuna basata sul tipo di sistema su cui eseguire il kernel stesso. Inoltre, la configurazione del kernel per un dato sistema MIPS è influenzata dalla scheda utilizzata.

In realtà, il supporto di Linux per MIPS è limitato, se comparato ad altre architetture, come la i386 e la PowerPC. In effetti, poche delle principali distribuzioni sono state effettivamente portate per MIPS. Quando sono disponibili, il supporto per MIPS dei produttori è pressoché limitato ad architetture embedded. Tra le distribuzioni più famose, ricordiamo il porting della Debian (sia big endian che little endian) e della Red Hat 7.1. Per maggiori informazioni sul porting di Linux per MIPS, c'è il sito ufficiale http://www.linux-mips.org/, che contiene una lista di sistemi supportati, documentazione, collegamenti ed altre risorse utili.

- **Hitachi SuperH:** nello sforzo per accrescere i suoi microcontrollori a 8 e 16bit della linea H8, la Hitachi ha introdotto la linea di processori a 32bit SuperH. La SuperH Inc concede in licenza i SuperH proprio come la ARM Holding Ltd fa con ARM e la MIPS Technologies Inc con MIPS. Le prime implementazioni del SuperH (SH-1, SH-2) e le loro varianti non

avevano una MMU, inclusa a partire dalla versione SH-3. I processori della serie SuperH sono utilizzati dalla Hitachi per i propri prodotti, in molti sistemi embedded come PDA e le console Sega Saturn e Dreamcast. Non possedendo una MMU, i primi processori SuperH non sono supportati da Linux; ad oggi, invece, è possibile utilizzare con Linux sia SH-3 che SH-4 (ma solo i modelli 7707, 7708, 7709, 7750, 7751, ST40). Non essendoci uno standard nell'architettura SuperH, bisogna consultare la documentazione del singolo sistema per ottenere dettagli sulla struttura e sulle funzionalità dell'hardware. Esistono, però, manuali che descrivono le operazioni e il set di istruzioni dei vari processori.

- **Motorola 68000:** la famiglia Motorola 68000 è conosciuta nel gergo di Linux come "m68k", ed è stata supportata nei suoi modelli equipaggiati con MMU per un po' di tempo, e nelle versioni senza MMU a partire dalla serie 2.5. La m68k è seconda solo alla x86 per popolarità negli anni 80. A parte l'utilizzo in molti sistemi come Atari, Apple e Amiga, e workstation della HP, Sun e Apollo, questa architettura fu spesso scelta per lo sviluppo di sistemi embedded. Recentemente, però, l'interesse si è spostato verso altre piattaforme, come ARM, MIPS e PowerPC. Linux supporta molti sistemi basati sul m68k, a partire dalle workstation già menzionate, inclusi sistemi VME e BVM. Essendo queste versioni molto differenti l'una dall'altra, l'alberatura del kernel è costruita in maniera tale da facilitare le variazioni e le

aggiunte di altri sistemi basati sul Motorola 68000. Ogni sistema ha i propri moduli per interfacciarsi con l'hardware.

Per informazioni aggiornate sul porting, il supporto hardware e le relative risorse, si può consultare il sito http://www.linux-m68k.org/. Una distribuzione che ha sviluppato molto per m68k è la Debian. Non essendoci una piattaforma standard, non esiste un unico documento sui sistemi basati su m68k, ma è possibile trovare molti libri e risorse online che trattano l'utilizzo di questo processore e la relativa programmazione.

4.3.2 Hardware supportato: bus e interfacce

Bus e interfacce sono i connettori delle CPU alle periferiche, e sono parte integrante di un sistema. Segue una carrellata sui bus e sulle interfacce utilizzabili nei sistemi embedded e supportati da Linux. In realtà, Linux supporta moltri altri bus, come SBus, NuBus, TurboChannel e MCA, ma questi non sono utilizzati in sistemi embedded.

- **ISA:** ISA sta per *Industry Standard Architecture*, ma fu progettato ed utilizzato nei core PC-AT. La sua semplicità favorì la proliferazione di molti dispositivi per PC che, a sua volta, favorì l'utilizzo del PC nelle applicazioni embedded. Si può accedere ai dispositivi ISA direttamente tramite le porte I/O comprese nel set di istruzioni del x86, quindi il kernel non ha bisogno di utilizzare dei driver per questo bus.
- **PCI:** PCI sta per *Peripheral Component Interconnection*, ed è

gestito dal *PCI Special Interest Group*; progettato per rimpiazzare l'ISA, è stato utilizzato in molte architetture differenti, compresi PPC e MIPS. A differenza dell'ISA, il bus PCI ha bisogno di supporto software per essere utilizzato, cosicché il kernel prevede il codice necessario all'inizializzazione ed alla configurazione dei dispositivi PCI.

- **PCMCIA:** PCMCIA sta per Personal Computer Memory Card International Association; esso è sia il nome di un bus, sia il nome dell'organizzazione che promuove e si occupa del relativo standard. Da quando è stato pubblicato il primo standard, che supportava solo schede a 16bit, altri standard sono stati scritti, incluse le specifiche per i *CardBus* a 32bit che per USB *CardBay*.
- **CompactPCI:** le specifiche per CompactPCI furono inizialmente scritte dalla Ziatech, e furono sviluppate dai membri della *PCI Industrial Computer Manifacturer's Group*, che supervisiona le specifiche e promuove l'uso del CompactPCI. Queste specifiche forniscono una piattaforma aperta e versatile per applicazioni con alte prestazioni. Il suo successo è fondato in gran parte sulle scelte tecniche fatte dai progettisti. Il riuso di idee e la compatibilità con lo standard ISA sono stati fondamentali per la riuscita di questo bus.
- **Porte parallele:** nonostante non sia un bus nel senso tradizionale, le porte parallele si trovano in molti computer, e possono essere utilizzate per connettere un gran numero di periferiche, compresi scanner, dischi e adattatori di rete. Il

supporto di Linux per le porte parallele è notevole.

- **SCSI:** SCSI sta per *Small Computer Systems Interface*, e fu introdotta dalla Shugart Associates. Nonostante fu introdotta principalmente come interfaccia ad alto throughput per hard disk nelle workstation e nei server, SCSI è un'interfaccia generica che può essere utilizzata per connettere diversi tipi di periferiche. Però, solo un piccolo segmento dei sistemi embedded utilizza dispositivi SCSI.
- **USB:** USB sta per *Universal Serial Bus*, ed è sviluppato e mantenuto da un gruppo di aziende che formano la *USB Implementers Forum*. Inizialmente fu sviluppato per rimpiazzare quelle interfacce più lente, come le porte parallele e seriali tradizionali. Oggi l'USB è l'interfaccia più usata per le periferiche, grazie alla facilità di utilizzo ed alle alte velocità di scambio dati. L'incremento nell'utilizzo del bus USB è notevole anche nell'ambito dei sistemi embedded.
- **IEEE1394 (FireWire):** FireWire è un marchio commerciale della Apple per una tecnologia progettata tra gli anni 80 e gli anni 90. Il lavoro fu sottoporto alla IEEE e divenne così lo standard 1394. In termini di velocità, questo standard è migliore dell'USB, ed è spesso utilizzato per videocamere digitali o hard disk.

4.3.3 Hardware supportato: I/O

Input e output sono la parte centrale di ogni sistema computerizzato.

Linux supporta un vasto insieme di dispositivi di I/O; alcuni dei seguenti dispositivi di I/O sono supportati nel kernel in due forme: la prima, come driver nativo che gestisce direttamente la connessione con il dispositivo; la seconda, tramite lo strato USB a cui il dispositivo potrebbe essere collegato.

- **Porte seriali:** molti sistemi embedded solo sviluppati e debuggati tramite un collegamento seriale RS232 tra l'host e il target. La semplicità dell'interfaccia RS232 ha incoraggiato il suo vasto utilizzo, anche se la larghezza di banda è spesso limitata rispetto a molti altri mezzi di trasmissione. Ci sono anche molte altre interfacce seriali oltre la RS232, alcune delle quali meno sensibili al rumore e più adatte all'ambiente industriale.

  Essendo RS232 un'interfaccia hardware, il kernel non ha bisogno di un supporto. Spesso, il kernel include i driver del chip che effettua la comunicazione, lo Universal Asynchronous Receiver-Transmitters (UARTs). Gli UARTs variano da un'architettura all'altra, anche se alcuni modelli possono essere utilizzati su più architetture.
- **Modem:** i sistemi embedded che utilizzano modem per chiamare un centro dati sono abbastanza comuni. Sistemi d'allarme e sistemi di monitoraggio remoto sono esempi di sistemi embedded che utilizzano modem. I modem in Linux sono visti come delle porte seriali, e sono controllati dallo stesso driver delle porte seriali native UARTs. Questo discorso non vale per i WinModem, che contengono solo una parte dell'harware

presente nei modem reali, e per il cui utilizzo sono nati svariati progetti.

- **Acquisizione dati:** l'acquisizione dati è alla base di un qualsiasi sistema di automazione. Ogni laboratorio scientifico è pieno di dispositivi di acquisizione dati collegati, in un modo o nell'altro, a computer che elaborano ed analizzano dati. Non esiste un'interfaccia standard in Unix o in altri sistemi operativi per dispositivi di acquisizione dati. Comedi, il *Linux control and measurement devide interface*, è il pacchetto principale per interfacciarsi con hardware DAQ. Esso contiene driver per numerose schede DAQ.
- **Controllo di processo:** come per DAQ, il controllo di processo è alla base di ogni sistema di automazione. Linux può essere utilizzato in molti modi per il controllo di processo; ad esempio, si possono utilizzare porte seriali o parallele, oppure schede particolari connesse ai computer per mezzo di bus.
- **Home Automation:** la home automation è un dominio molto vasto. A differenza di altri campi in cui è utilizzato Linux, qui non esiste un progetto open source principale, ma ci sono alcuni progetti attivi sviluppati in modo indipendente; inoltre, non esistono driver per la domotica implementati direttamente nel kernel, ma le componenti software sono tutte parti dei vari pacchetti distribuiti dai progetti sopraccitati.
- **Keyboard:** la maggior parte dei sistemi embedded non prevede l'uso di tastiera. Recentemente, però, alcuni sistemi orientati al

web o al consumatore possono avere una sorta di tastiera annessa. L'input da tastiera, in ogni caso, è considerato dal kernel come l'input da un terminale.

- **Mouse:** i sistemi embedded che prevedono un'interfaccia utente spesso forniscono interazione basata sul tocco (touch screen). L'input generato dal tocco su un'area di schermo è trattata allo stesso modo dell'input tradizionale di un mouse di una workstation. In questo senso, molti sistemi embedded hanno un mouse.
- **Display:** luci intermittenti, LED e display LCD sono i dispositivi di visualizzazione tradizionali di un sistema embedded. Con l'incremento dell'utilizzo di sistemi embedded nella vita quotidiana, si sente l'esigenza di rimpiazzare questi mezzi obsoleti con interfacce più ricche. Se non serve altro, Linux mette a disposizione delle console che possono visualizzare solo testo; in molti altri casi, però, c'è bisogno di utilizzare interfacce grafiche più elaborate, che includano anche una sorta di sistema basato su finestre. Il modo più semplice per fornire un'interfaccia grafica con Linux è naturalmente il X Window System.
- **Suono:** Unix non è nato come sistema operativo multimediale. Ciononostante, per Linux, sono nati una serie di progetti per supportare il suono. In Linux, il dispositivo principale per il suono è */dev/dsp*. Esistono due progetti principali che forniscono sia supporto per l'hardware, sia le API per programmare le

schede audio. Il primo è *Open Sound System* (OSS), sulla cui architettura sono basati molti driver per schede audio presenti nelle releases del kernel precedenti alla 2.5. Il secondo progetto audio è *Advanced Linux Sound Architecture* (ALSA), i cui obiettivi sono di fornire un pacchetto di driver completamente modularizzato, e offrire un ambiente superiore ad OSS, sia in termini di API che di gestione dell'hardware.

- **Stampanti:** il supporto delle stampanti in Unix è spesso sorpassato rispetto al supporto fornito da molti altri sistemi operativi, ed il supporto fornito da Linux è fortemente basato sul sistema di stampa di Unix. Ad oggi esistono cinque pacchetti di gestione delle stampanti per Linux: LPD, PDQ, LPRng, CUPS e PPR.

4.3.4 Hardware supportato: storage

Tutti i sistemi embedded hanno bisogno almeno di una forma di memoria di massa, per l'inizializzazione del processo di boot. La maggior parte dei sistemi, inclusi i sistemi embedded Linux, continuano ad usare lo stesso dispositivo di memorizzazione iniziale per il resto delle loro operazioni, per eseguire il codice o per accedere ai dati.

- **Memory Technology Devices:** nella terminologia Linux, MTDs includono tutti i dispositivi di memoria, come ROM, RAM, flash e M-Systems'DiskOnChip (DOC). Per evitare il più possibile di avere tools differenti per diverse tecnologie, e fornire capacità

comuni tra le varie tecnologie, il kernel di Linux include il sottosistema MTD. Esso fornisce uno strato uniforme e unificato per abilitare una combinazione di driver per MTD a basso livello, tramite un'interfaccia ad alto livello.

- **ATA-ATAPI (IDE):** *AT Attachment* (ATA) fu sviluppato nel 1986 da tre compagnie: Imprimis, Western Digital e Compaq. Inizialmente fu utilizzato solo sui Compaq, ma divenne presto popolare. Nel 1994, ATA divenne uno standard ANSI. Sulla stessa scia, fu sviluppato *ATA Packet Interface* (ATAPI), che permette l'accesso a CD-ROM e a dispositivi a nastro tramite l'interfaccia ATA utilizzando comandi simili a quelli SCSI. Nei sistemi embedded Linux, i dischi IDE sono solitamente configurati come nelle workstation o nei server. Tipicamente, il disco contiene il bootloader, il file system radice ed una possibile partizione di swap. Il supporto di Linux all'interfaccia ATA è abbastanza esteso e maturo.

4.3.5 Hardware supportato: networking

Un numero sempre maggiore di sistemi embedded utilizza dispositivi di rete. Da questi dispositivi spesso ci si aspettano servizi di rete simili a quelli dei moderni server. Fortunatamente, Linux sfrutta abbastanza bene i dispositivi di rete, dal momento che esso stesso è spesso utilizzato nei server.

- **Ethernet:** inizialmente sviluppato nel centro di ricerca della Xerox, Ethernet è oggi l'adattatore di rete più utilizzato, più

documentato e meno costoso. La sua velocità è aumentata nel tempo. Linux supporta sia dispositivi Ethernet 10/100Mbps che Gigabit Ethernet.

- **IrDA:** la *Infrared Data Association* fu creata nel 1993 da 50 compagnie con lo scopo di creare e promuovere uno standard per l'interconnessione infrarossi a basso costo. Oggi hardware e software per IrDA è presente in molti dispositivi, compresi PDA, telefoni cellulari, stampanti e fotocamere digitali. Esistono due tipi di protocolli principali nelle specifiche IrDA: obbligatori e opzionali. Un dispositivo deve implementare almeno un protocollo obbligatorio per poter comunicare con altri dispositivi IrDA. Un protocollo obbligatorio comprende il livello di segnalazione fisica (IrPHY), il protocollo d'accesso al mezzo (IrLAP) e il protocollo di gestione del collegamento (IrLMP). I dispositivi IrDA possono scambiare dati fino a frequenze di 4Mbps entro un metro di distanza. Linux supporta tutti i protocolli obbligatori e molti dei protocolli opzionali.
- **IEEE 802.11 (Wireless):** il gruppo di lavoro 802.11 fu messo in piedi dalla IEEE nel 1990. Il primo standard 802.11 fu pubblicato nel 1997 ed è stato mantenuto ed aggiornato da allora sempre dallo stesso gruppo. Lo standard fornisce comunicazioni wireless tra computer utilizzando le frequenze 2.4GHz (802.11b/g) e 5GHz (802.11a). Linux ha un supporto esteso per hardware compatibile con lo standard 802.11b/g. Dal momento che molte operazioni dei dispositivi sono simili a quelle

dei dispositivi Ethernet, il kernel non necessita di alcun sottosistema addizionale.

- **Bluetooth:** il Bluetooth fu sviluppato da Ericsson con l'aiuto della Intel e fu introdotto nel 1994. Oggi un vasto numero di dispositivi come PDA e telefoni cellulari sfruttano questa tecnologia. Il Bluetooth lavora a frequenze di 2.4GHz. Linux ha pochi stack Bluetooth, tra cui si possono ricordare BlueZ, OpenBT, Affix e BlueDrekar.

4.3.6 Setup del kernel

Il kernel è il componente software principale di tutti i sistemi Linux. Se il kernel non supporta un dispositivo, questo non funzionerà fintantoché il kernel non sarà modificato. Per mettere a punto il kernel per un sistema embedded bisogna seguire, in linea di massima, la seguente procedura:

- **Selezionare un kernel:** nonostante ci sia una solo principale repository per il kernel, http://www.kernel.org/, le versioni disponibili da questo sito non sono sempre i più appropriati per tutte le architetture supportate da Linux. Per avere un kernel funzionante su un sistema embedded, bisogna ottenere una delle versioni messe a disposizione dal team di sviluppo della casa costruttrice dell'architettura hardware relativa. Una volta trovato il repository adatto all'hardware in possesso, bisogna selezionare una versione del kernel. Questa è un'importante decisione, poiché bisogna far attenzione al supporto hardware

messo a disposizione dal kernel stesso.

- **Configurare il kernel:** la configurazione è il passo iniziale per la compilazione del kernel. Esistono molte configurazioni ed opzioni tra cui scegliere. Durante la configurazione è possibile selezionare le opzioni da includere nel kernel, come, ad esempio, le scelte su MTD, sul networking, sul supporto SCSI, sul supporto ATA, ed altri. Per effettuare la configurazione del kernel, si possono utilizzare i seguenti metodi: *make config*, che fornisce un'interfaccia lineare per effettuare la configurazione; *make menuconfig*, che visualizza un menù di configurazione su terminale; *make xconfig*, che visualizza un menu di configurazione basato sul sistema X Window. Il processo di configurazione termina con la creazione del file *.config* nella directory radice dei sorgenti del kernel, contenente la configurazione risultante.
- **Compilare il kernel:** la compilazione del kernel coinvolge due passi: la compilazione dell'immagine del kernel e la compilazione dei moduli. Per compilare l'immagine del kernel (ovviamente mediante il tool *make*), bisogna impartire il comando *make zImage* da shell (non sono necessari permessi da amministratore); la compilazione può necessitare l'assegnazione di due variabili: ARCH (architettura) e CROSS_COMPILE (parametro di cross-compilazione). Se la compilazione dell'immagine del kernel è andata a buon fine, bisogna procedere alla compilazione dei moduli; questa operazione è

innescata dal comando *make modules*; allo stesso modo della compilazione dell'immagine del kernel, anche per la compilazione dei moduli è possibile passare le due variabili ARCH e CROSS_COMPILE. Se anche la compilazione dei moduli del kernel è andata a buon fine, si può passare al prossimo passo.

- **Installare il kernel:** dopo aver compilato l'immagine del kernel con i relativi moduli, bisogna copiarli sul target per essere utilizzati. Conviene utilizzare directory separate sia per kernel di versioni diverse, sia per kernel con stessa versione, ma differenti configurazioni. Per ogni configurazione del kernel compilato, bisogna copiare quattro file: l'immagine del kernel non compressa (*vmlinux*), l'immagine compressa (*zImage*), il file della mappa dei simboli (*System.map*) ed il file di configurazione (*.config*). Inoltre, bisogna installare i moduli del kernel; questo passo richiede il comando *make modules_install*, che (per default) installa i moduli compilati nella directory */lib/modules*. Essendo in un ambiente di cross-sviluppo, bisogna suggerire al tool make il percorso in cui copiare effettivamente i moduli in questione (la variabile da settare è *INSTALL_MOD_PATH*).

  A questo punto, il kernel è pronto per essere utilizzato.

4.3.7 Contenuto del Filesystem radice

Una delle ultime operazioni effettuate dal kernel di Linux durante lo

startup del sistema è il montaggio del filesystem radice.

Ogni directory a livello più alto del filesystem radice ha uno specifico utilizzo e scopo. Alcune di queste sono utili solo in sistemi multiutente. Per questo motivo, nella maggior parte dei sistemi embedded si possono non seguire alla lettera le linee guida per la creazione di un filesystem radice. Le regole "ufficiali" da seguire per creare un filesystem radice sono contenute nel *Filesystem Hierarchy Standard* (FHS).

In linea generale, l'alberatura radice deve essere così composta:

- *bin*: file binari essenziali
- *boot*: file statici utilizzati dal bootloader
- *dev*: dispositivi ed altri file speciali
- *etc*: file di configurazione del sistema
- *home*: directory home degli utenti, comprese le directory di servizi quali FTP
- *lib*: librerie fondamentali, come le librerie C e moduli del kernel
- *mnt*: punti di montaggio per filesystem temporanei
- *opt*: pacchetti software opzionali
- *proc*: filesystem virtuale per informazioni su processi e kernel
- *root*: directory home per l'utente root
- *sbin*: file binari di sistema
- *tmp*: file temporanei
- *usr*: gerarchia secondaria contenente la maggior parte delle applicazioni e dei documenti utili a molti utenti
- *var*: file variabili utilizzati da demoni e programmi di utilità

Le directory utilizzate in ambiente multiutente (come */home*, */mnt*, */opt* e */root*) possono essere omesse, se non risultano utili allo scopo prefissato. Le directory */var* e */usr* hanno un'alberatura predefinita, proprio come il filesystem radice.

Ovviamente, il filesystem radice può essere formattato con uno dei formati disponibili; i filesystem utilizzati da Linux, di solito, sono *ext2* o *ext3* (uguale a *ext2*, ma con supporto al journaling). I sistemi embedded, però, possono avere necessità particolari rispetto alle workstation o ai server: potremmo volere un filesystem non scrivibile, ad accesso molto veloce; oppure potremmo aver bisogno di un filesystem scrivibile, che eviti la perdita dei dati anche a costo di essere dispendioso nei tempi di scrittura.

4.3.8 Scelta del bootloader

Nonostante il bootloader venga eseguito per un tempo molto corto durante lo startup del sistema ed è principalmente responsabile del caricamento del kernel, esso è un componente molto importante del sistema. La configurazione del bootloader è un passo comune e necessario per tutti i sistemi Linux. Esso è un processo particolare anche per i sistemi embedded, poiché i bootloader utilizzati in alcuni sistemi possono essere completamente differenti da quelli utilizzati nei sistemi comuni (workstation e server) o, anche se sono gli stessi, possono essere configurati ed utilizzati in maniera molto differente. Poiché le architetture hardware differiscono in maniera significativa tra loro, e poiché anche le schede basate sulla stessa architettura sono

molto differenti tra loro, la scelta e la configurazione del bootloader dipende fortemente dall'hardware utilizzato.

Segue una rassegna dei bootloader più utilizzati, accompagnati dalla descrizione delle piattaforme hardware supportate.

- **LILO:** LILO sta per *LInux LOader*, ed è stato introdotto molto presto nella storia di Linux. Ad oggi è mantenuto da John Coffman. Il LILO è un bootloader molto ben documentato, ed il pacchetto include un manuale utente e dei manuali interni. Questo bootloader supporta solo l'architettura x86, per la quale, insieme al GRUB, è il bootloader di riferimento.
- **GRUB:** il *GRand Unified Bootloader* è il principale bootloader per progetti GNU. Il GRUB fu inizialmente scritto da Erich Boleyn con l'intento di creare un bootloader appropriato per quello che poi sarebbe stato lo GNU Mach. Sul sito ufficiale si può trovare il manuale del GRUB, che descrive in maniera esaustiva l'uso del pacchetto. Uno degli aspetti rilevanti del GRUB, utili durante lo sviluppo di sistemi embedded, è la sua capacità di effettuare il boot tramite la rete, utilizzando i protocolli TFTP, BOOTP o DHCP. Come per il LILO, l'unica architettura supportata è la x86.
- **ROLO:** il *ROmable LOader* fu scritto ed è tuttora mantenuto da Robert Kaiser, della Sysgo Gmbh. ROLO può effettuare il boot di Linux direttamente dalla ROM, senza utilizzare alcun BIOS. Nonostante il pacchetto contenga già una piccola documentazione, in rete si può trovare un articolo scritto da Vipin Malik, per utilizzare ROLO nei sistemi embedded.

- **loadlin:** loadlin è una utility DOS per il caricamento del kernel di Linux, mantenuta da Hans Lermen. Anche se si dovrebbe evitare la realizzazione di un sistema che richieda di caricare DOS per eseguire le istruzioni del bootloader, ci sono casi in cui questa applicazione può risultare molto utile. L'unica architettura supportata è la x86.
- **EtherBoot:** questo bootloader, come si può intuire dal nome, supporta il caricamento di sistemi senza dischi, mediante la rete. EtherBoot è stato usato in molti ambienti, compresi i terminali X, i router ed i cluster. Il pacchetto, compreso di una completa documentazione, si può trovare all'indirizzo *http://etherboot.sourceforge.net/*. L'unica architettura supportata è la x86.
- **LinuxBIOS:** questo bootloader rimpiazza completamente il BIOS, ed effettua il boot da una ROM all'avvio del sistema. LinuxBIOS fu sviluppato come parte di una ricerca sui sistemi cluster, condotta alla Los Alamos National Laboratory. Il pacchetto e la documentazione sono disponibili all'indirizzo *http://www.linuxbios.org*. L'unica piattaforma supportata è la x86.
- **Compaq's bootldr:** nonostante inizialmente fu sviluppato solo per i Compaq iPAQ, questo bootloader supporta correntemente Assabet Intel e Jornada 720 HP. Nonostante il Compaq's bootldr sia limitato riguardo al supporto hardware, esso fornisce un insieme molto ricco di comandi, ed è capace di caricare il kernel

direttamente da partizioni JFFS2 su MTD. L'architettura supportata è quella ARM.

- **blob:** blob fu introdotto come bootloader per il progetto LART. Sin dalla sua introduzione, blob è stato portato su molti sistemi basati su architettura ARM, compresi Assabet della Intel e schede Brutus, Shannon e Nesa. Come il Compaq's bootldr, blob può caricare il kernel direttamente da partizioni JFFS2 su MTD.
- **PMON:** *Prom MONitor* fu scritto da Phil Bunce per supportare le schede MIPS della LSI LOGIC. Questo bootloader è distribuito sotto una licenza che non prevede garanzia, e che permette la ridistribuzione senza alcuna restrizione. Ad oggi sono poche le schede supportate, quindi PMON può essere considerato un buon punto di partenza, ma prima di poterlo utilizzare c'è probabilmente bisogno di un porting.
- **sh-boot:** questo bootloader è sviluppato come parte del progetto Linux SH su SourceForge. Sfortunatamente, sh-boot non è aggiornato da un po' di tempo, per cui bisogna valutare bene se convenga utilizzarlo sui propri sistemi. L'unica architettura supportata è la SuperH.
- **U-Boot:** lo *Universal Bootloader* è probabilmente il bootloader open source più ricco, più flessibile e maggiormente sviluppato disponibile ad oggi. Esso è mantenuto da Wolfgang Denk della DENX Software Engineering, ed è sviluppato da un gran numero di sviluppatori. U-Boot è basato sui progetti PPCBoot e ARMBoot. PPCBoot era a sua volta basato sui sorgenti 8xxrom, mentre

ARMBoot era un porting del PPCBoot su architettura ARM, eseguito dalla Sysgo Gmbh. U-Boot supporta circa 100 schede basate su PPC, più di una dozzina di schede basate su ARM ed una manciata di schede basate su architettura x86. Dato il grande successo di questo bootloader, si prevede che gli sviluppatori continueranno il porting verso altre architetture.

Tra le altre cose, U-Boot può effettuare il boot del kernel tramite TFTP, da un disco IDE o SCSI e da un DOC. Inoltre è incluso il supporto di sola lettura per partizioni JFFS2. Oltre ad avere un insieme di comandi esteso, questo bootloader è molto ben documentato. Il README incluso nel pacchetto fornisce una discussione approfondita sull'uso dell'U-Boot. Inoltre, la directory *doc* dei sorgenti del pacchetto include informazioni su ogni istruzione extra richiesta da determinate schede. Infine, in rete è possibile trovare il *DENX PPCBoot and Linux Guide*, un documento scritto da Wolfgang che fornisce molti esempi pratici dell'uso di PPCBoot con Linux su schede TQM8xxL.

Il workspace del progetto U-Boot si trova all'indirizzo *http://sourceforge.net/projects/u-boot*.

- **RedBoot:** RedBoot è il bootloader di nuova generazione della Red Hat, che rimpiazza CygMon e GDB con un firmware che supporta una grande quantità di hardware. Anche se Red Hat non sviluppa più eCos, il sistema operativo su cui si basa RedBoot, la licenza di eCos è stata modificata in GPL, ed il sistema operativo è ora mantenuto da alcuni sviluppatori di Red

Hat; sia eCos sia RedBoot sono nelle mani di questo gruppo.

Nonostante la dipendenza di RedBoot da eCos, il bootloader rimane una utility molto potente. Ad esempio, esso è l'unico bootloader opensource che supporta tutte le architetture presentate al paragrafo 4.3.1, e numerose schede basate su di esse. Inoltre, il pacchetto RedBoot è molto ben documentato, ed include *RedBoot User's Guide*, che fornisce esempi reali dell'utilizzo del bootloader su dozzine di sistemi diversi.

Il bootloader utilizzato dalla scheda su cui viene sviluppata questa tesi è proprio il RedBoot; per quanto riguarda la sua configurazione, si può vedere il paragrafo 7.5.

4.4 Strumenti di sviluppo

Proprio come per lo sviluppo su workstation o server, gli sviluppatori di sistemi embedded hanno bisogno di compilatori, linkers, interpreti, ambienti di sviluppo integrati ed altri strumenti di sviluppo. Questi strumenti sono differenti, però, poiché essi vengono eseguiti su una piattaforma diversa di quella per cui si sviluppa. Infatti, questi strumenti sono spesso chiamati tool di sviluppo cross-piattaforma.

4.4.1 Workspace

La prima cosa da fare quando si inizia a sviluppare per un sistema embedded è organizzare il workspace per il progetto. In particolare, c'è bisogno di creare un'alberatura simile a quella proposta di seguito.

- *bootldr*: il bootloader (o i bootloader) per il target
- *build-tools*: i pacchetti e le directory utili per creare la toolchain di sviluppo cross-piattaforma
- *debug*: gli strumenti di debugging, con i relativi pacchetti
- *doc*: tutta la documentazione necessaria per il progetto
- *images*: le immagini binarie del bootloader, del kernel e del filesystem, pronte per essere utilizzate sul target
- *kernel*: le diverse versioni del kernel proposte per il sistema da sviluppare
- *project*: il proprio codice per il progetto
- *rootfs*: il filesystem di root, così come sarà visto dal kernel sul target a runtime
- *sysapps*: le applicazioni di sistema richieste per la macchina target
- *tmp*: una directory temporanea per provare file temporanei
- *tools*: la toolchain di sviluppo cross-piattaforma completa, e le librerie C

Ovviamente, ognuna delle directory sopraccitate contengono una propria alberatura.

Il workspace da creare, solitamente, è posizionato all'interno di una sottodirectory della */home*, magari condivisa tra tutti gli sviluppatori del gruppo, oppure all'interno della directory */opt*; in ogni caso, si sconsiglia l'utilizzo di cartelle di sistema.

4.4.2 Toolchain di svilluppo cross-piattaforma GNU

Essendo in ambiente Linux, viene spontaneo pensare allo sviluppo in linguaggio C.

La toolchain di cui si ha bisogno per mettere insieme le applicazioni cross-sviluppate per qualsiasi target include alcune utility binarie, il compilatore C e le librerie C. La toolchain si può trovare in rete all'indirizzo ftp://ftp.gnu.org/gnu/ o qualsiasi suo mirror. La scelta delle versioni, la configurazione e la compilazione degli strumenti necessari è un compito complesso e molto delicato, a causa delle dipendenze tra i vari pacchetti; ciò è vero, poiché i componenti della toolchain sono sviluppati e rilasciati in maniera indipendente l'uno dall'altro.

Dopo aver scelto le versioni dei pacchetti da utilizzare, bisogna preparare i file di intestazione del kernel, per poi procedere alla configurazione, compilazione ed installazione della toolchain.

In particolare, il pacchetto *binutils*, dopo essere stato installato, fornirà le seguenti applicazioni, utili allo sviluppo:

- *as*: l'assembler GNU
- *ld*: il linker GNU
- *gasp*: il pre-processore dell'assembler GNU
- *ar*: crea e gestisce il contenuto di archivi
- *nm*: visualizza la lista dei simboli di un file oggetto
- *objcopy*: copia e traduce i file oggetto
- *objdump*: visualizza informazioni sul contenuto dei file oggetto
- *ranlib*: genera un indice dei contenuti di un archivio

- *size*: visualizza una lista delle misure delle sezioni di un file oggetto
- *strings*: stampa a video le stringhe di caratteri stampabili di un file oggetto
- *strip*: visualizza i simboli di un file oggetto
- *c++filt*: converte le etichette assembly di basso livello risultanti da funzioni c++ "sovraccariche" nei rispettivi nomi a livello utente
- *addr2line*: converte gli indirizzi in numeri di linea all'interno del file sorgente originale

Dopo la compilazione e l'installazione delle utility, bisogna installare il compilatore C GNU (*gcc – GNU C Compiler*); una sua prima installazione prevede il solo supporto per il linguaggio C. Successivamente, dopo aver configurato ed installato anche le librerie C GNU (*glibc – GNU Libraries C*), si potrà procedere alla ricompilazione del gcc con supporto per C++.

A questo punto, la toolchain è completamente installata; bisogna solo provvedere all'esportazione di alcune variabili d'ambiente, necessarie per istruire i tool sull'architettura del target.

4.4.3 Librerie C alternative

Considerati i vincoli ed i limiti dei sistemi embedded, la grandezza delle librerie GNU C standard rende queste stesse delle pessime candidate all'utilizzo sul target. Al posto loro, è possibile utilizzare delle librerie che abbiano funzionalità simili alle librerie GNU C, ma siano

relativamente piccole.

A tale scopo, nel corso degli anni sono state implementate numerose librerie, tra le quali sono degne di nota *uClibc* e *diet libc*.

Il primo pacchetto ha origine dal progetto uClinux, che fornisce un sistema Linux in grado di essere eseguito su processori senza MMU. Queste librerie, però, sono diventate un progetto a parte, e supportano un numero di processori che possono o meno essere dotati di MMU. Le librerie uClibc possono essere utilizzate su tutte le architetture viste fin'ora, poiché includono un loader di librerie condivise nativo per ogni architettura; se un loader non è implementato per una certa piattaforma, allora saranno caricate le librerie glibc al posto delle uClibc. Le librerie uClibc forniscono la maggior parte delle funzionalità delle librerie GNU C, anche se non rispondono esattamente allo standard.

Il progetto diet libc, invece, fu iniziato ed è ancora mantenuto da Felix von Leitner, il quale ha scritto le librerie a partire da zero. L'obiettivo principale di questo pacchetto è di minimizzare lo spazio ed ottimizzare le performance. L'impronta di queste librerie e la velocità del codice sono paragonabili alle libc. Purtroppo, però, le diet libc supportano solo le architetture ARM, MIPS, x86 e PPC. Inoltre, queste librerie non sono dinamiche, bensì statiche. È da notare che le diet libc sono rilasciate sotto licenza GPL, al contrario di altre librerie, come le uClibc, rilasciate sotto LGPL.

4.4.4 Altri linguaggi di programmazione

Per programmare sistemi embedded Linux, però, l'unica alternativa non è il linguaggio C, anche se forse sarebbe la scelta più logica.

Uno dei linguaggi più popolari al momento (anche per la sua nota portabilità), è il Java, introdotto dalla Sun nel 1995. I progetti più interessanti in questo senso sono il *Blackdown Project*, che fornisce il JDK (Java Development Kit) ed il JRE (Java Runtime Environment), la *Kaffe Java Virtual Machine* ed il *GNU Java Compiler* (gjc), un'estensione del gcc che gestisce sia codice Java che byte code Java.

Un altro linguaggio di programmazione utilizzabile in sistemi embedded è il Perl, introdotto da Larry Wall nel 1987. Il punto di forza del Perl è che il suo compilatore, le sue utility e le sue librerie sono distribuite come open source. Essendo un linguaggio interpretato, per utilizzare il Perl su una macchina target c'è bisogno di un interprete. Sono incluse nel pacchetto del Perl, infatti, due versioni dell'interprete per la cross-compilazione: il Microperl ed il Miniperl.

Simile al Perl c'è il Python, altro linguaggio che, però, nel pacchetto non include alcun supporto alla cross-compilazione. In ogni caso, si può trovare in rete una patch a tale scopo.

Altri linguaggi di programmazione supportati da Linux, ed utili per lo sviluppo di sistemi embedded potrebbero essere Ada, Forth, Lisp o FORTRAN.

4.4.5 Ambienti di sviluppo integrati

Molti ambienti di sviluppo integrati, meglio conosciuti come *IDE*

(Integrated Development Environment), sono disponibili sotto Linux. Essi non sono utilizzati necessariamente per cross-sviluppare, ma sono molto spesso adoperati per lo sviluppo di codice nativo.

Gli IDE più conosciuti sono i seguenti:

- *Anjuta*: supporta i linguaggi Ada, bash, C, C++, Java, make, Perl e Python
- *Eclipse*: supporta i linguaggi C, C++ e Java
- *Glimmer*: supporta i linguaggi Ada, bash, C, C++, Java, make, Perl, Python e assembly x86
- *KDevelop*: supporta i linguaggi C, C++ e Java
- *NetBeans*: supporta i linguaggi C++ e Java
- *Source Navigator*: supporta i linguaggi C, C++, Java, Tcl, Python, FORTRAN e COBOL

4.4.6 Emulatori di terminale

Il modo più comune per comunicare con un sistema embedded è quello di utilizzare un programma emulatore di terminale, collegando l'host al target mediante una porta seriale RS232.

Nonostante Linux metta a disposizione alcuni emulatori di terminale, non tutti sono soddisfacenti per ogni scopo. Ad esempio, sono noti dei problemi riguardanti lo scambio di file tra il programma *minicom* ed U-Boot.

I programmi emulatori di terminale più conosciuti sono *minicom*, *cu* e *kermit*.

Prima di utilizzare uno dei suddetti programmi per comunicare con la

macchina target, bisogna assicurarsi di avere i diritti appropriati per accedere al dispositivo di porta seriale; infatti, bisogna accertarsi di avere diritti di lettura e scrittura sul file */dev/ttyS0* (per trasmettere e ricevere dati), e su */var/lock* (per bloccare l'accesso alla porta seriale).

4.5 Configurazione della rete

Come già detto in precedenza, un sistema embedded può aver bisogno di accedere alla rete per portare a termine i propri compiti. Ad esempio, per la realizzazione di un gateway VoIP, la macchina target deve poter comunicare con l'ambiente esterno per instradare le chiamate telefoniche verso altri gateway o proxy. Un esempio ancora più lampante è la realizzazione di un router, il cui compito principale è la gestione degli adattatori e dei servizi di rete, nonché l'attuazione degli algoritmi di instradamento dei pacchetti.

Proprio per questo motivo, Linux è sempre più utilizzato nei sistemi embedded che fanno largo uso della rete (è nota l'affidabilità relativa alla gestione delle reti di questo sistema operativo).

I servizi di rete più comuni che possono essere abilitati in un sistema Linux sono i seguenti:

- *SNMP*: SNMP è l'acronimo di *Simple Network Management Protocol* (semplice protocollo di gestione di rete); esso permette la gestione remota dei dispositivi su reti TCP/IP. Oltre ai dispositivi di rete (come router e switch), quasi tutti i sistemi che utilizzano la rete fanno uso di questo servizio. SNMP

permette il monitoraggio remoto ed automatico del target; in altre parole, non c'è bisogno di verificare manualmente il corretto funzionamento del sistema, ma questo viene periodicamente interrogato da un gestore, eseguito su un altro sistema. Il sistema target può anche essere configurato in modo da trasmettere dati riguardanti eventuali fallimenti hardware o software al manager remoto.

- *telnet*: il protocollo Telnet è uno dei modi più semplici per autenticarsi in un sistema remoto. Per abilitare il login remoto tramite telnet, c'è bisogno che sul target sia installato e configurato il demone di Telnet; i demoni disponibili per sistemi Linux embedded sono due: *telnetd* (che è parte del pacchetto netkit) e *utelnetd*. In termini di spazio, l'eseguibile generato da utelnetd è più piccolo rispetto a quello del telnetd. Inoltre, utelentd non ha bisogno di un super-server, al contrario del suo antagonista. L'inconveniente del protocollo Telnet è che non prevede alcuna forma di crittografia del flusso dati scambiato in rete (è ormai ritenuto un protocollo non sicuro per l'utilizzo in Internet).
- *SSH*: SSH sta per *Secure SHell*, ovvero shell sicura; al contrario del protocollo di autenticazione appena visto, SSH prevede una forma di crittografia a chiave pubblica, il che rende le comunicazioni di gran lunga più sicure rispetto al telnet. Essendo uno standard IETF, esistono varie implementazioni, alcune delle quali sono prodotti proprietari. Il progetto open source principale

è *OpenSSH*; nonostante esistano altre implementazioni open source di SSH, esse risultano molto ostiche da cross-compilare.

- *HTTP*: uno dei trend odierni è l'inclusione di web server all'interno dei sistemi embedded che sfruttano le infrastrutture di rete. Aggiungere server HTTP può risultare utile per abilitare l'amministrazione remota o per la visualizzazione di dati. Nonostante Apache sia il web server open source più utilizzato al mondo, questa non è sicuramente la scelta ottimale per un sistema embedded, principalmente perché è molto difficile cross-compilare, e tende ad essere piuttosto grande in termini di spazio. Esistono, però, altri server HTTP open source molto più adatti allo scopo. In particolare *Boa* e *thttpd* sono server molto piccoli, leggeri e veloci, e sono perfetti per sistemi Linux embedded. Non esistono differenze rilevanti tra i due pacchetti: entrambi supportano lo scripting CGI, e la grandezza degli eseguibili è comparabile; l'unica vera differenza è nelle licenze: Boa è distribuito sotto GPL, mentre thttpd è distribuito sotto una licenza simile alla BSD.
- *DHCP*: DHCP è l'acronimo di *Dynamic Host Configuration Protocol*, e permette ad una macchina la configurazione automatica della rete. Per configurazione automatica della rete si intende l'assegnazione dell'indirizzo IP della macchina, della maschera di sottorete, ed eventualmente del gateway e dei server DNS. Un sistema Linux embedded può essere facilmente utilizzato come server DHCP; in maniera duale, un sistema

embedded può aver bisogno di ricevere la configurazione di rete da parte di un server DHCP remoto. Oltre al pacchetto standard utilizzato in molte distribuzioni, esiste un altro pacchetto open source che prevede sia client che server DHCP, ed è un ottimo candidato per essere utilizzato in sistemi embedded: *udhcp*.

Capitolo 5

PIATTAFORMA DI SVILUPPO

La piattaforma hardware di riferimento nello sviluppo della tesi è la IXDPG425 della Intel©, un gateway di rete che fornisce tutte le funzionalità necessarie per un gateway residenziale. La piattaforma è equipaggiata con il processore di rete IXP425, memoria SDRAM, flash, uno switch Ethernet a quattro porte per rete LAN, una porta addizionale per rete WAN, quattro interfacce per linee telefoniche, un'interfaccia miniPCI per connessione a rete WLAN 802.11, un'interfaccia ADSL e due porte USB 2.0. Il processore dell'IXDPG425 è l'Intel XScale© Core.

L'architettura software di riferimento è la Intel© IXP400 Software v1.4 release, che offre pieno supporto alla linea di prodotti Intel© IXP42x. Questo software è costituito da quattro elementi: Intel© IXP400 Software v1.4 library, documentazione completa e codice sorgente dei componenti della libreria, immagini del microcode dei NPE e codelets di esempio. Per l'utilizzo del DSP, inoltre, è stato aggiunto un ulteriore

modulo software, Intel© IXP400 Digital Signal Processing (DSP) Software v2.6.2, con relativa codelet di esempio.

5.1 Piattaforma hardware

Il gateway di rete Intel© IXDPG425 è costituito da:

- processore di rete Intel© IXP425
- SDRAM (32 Mbyte) – 48LC8M16A (x2)
- memoria flash (16 Mbyte) – E28F128J3x-150
- switch Ethernet a 5 porte con 4 porte per rete LAN e 1 porta per rete WAN – Realtek RTL8305SB-VD
- interfaccia UTOPIA ADSL
- interfaccia voce a 4 porte – Silicon Labs 3201KS (x4); 3201KT (x4)
- connettore miniPCI per rete WLAN 802.11
- porta seriale
- 2 porte high-speed USB 2.0
- 1 porta USB 1.1

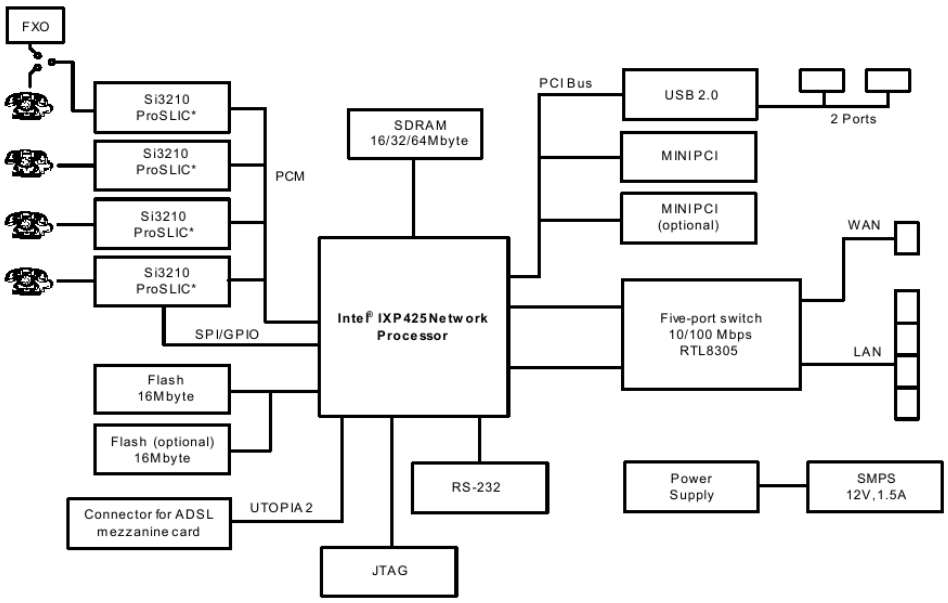


FIGURA 5.1: Diagramma a blocchi dell'hardware

In *Figura 5.1* si può vedere il diagramma dei blocchi hardware costituenti la piattaforma, con i relativi connettori per i dispositivi esterni. In *Figura 5.2* si può vedere una fotografia della piattaforma Intel© IXDPG425.

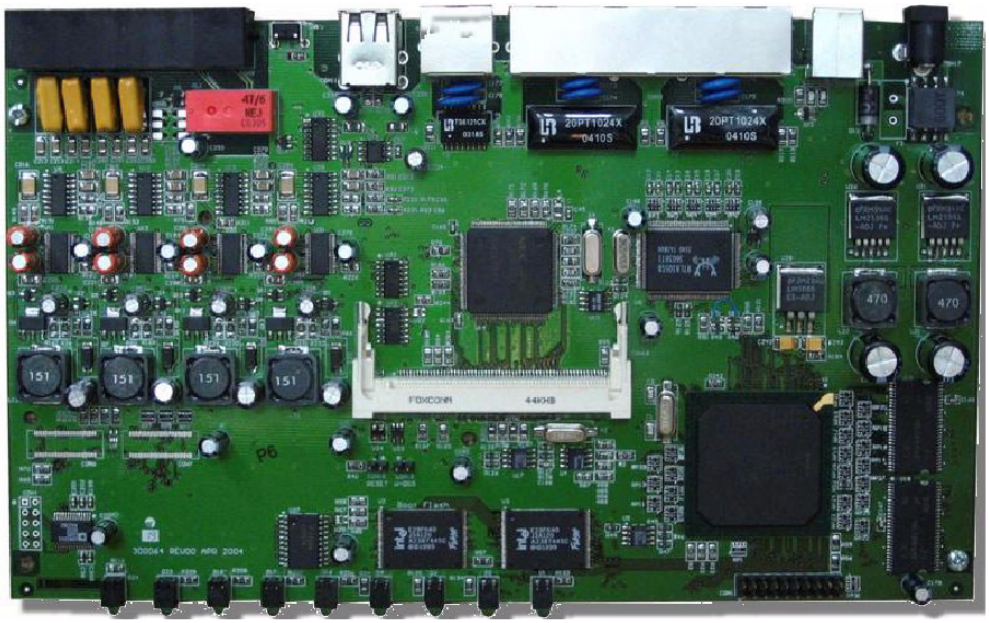


FIGURA 5.2: Scheda Intel IXDPG425

5.1.1 Intel© IXP425 Network Processor

Il processore di rete IXP425 può operare a frequenze di clock di 266, 400 o 533 MHz. L'IXDPG425 è la piattaforma di riferimento per l'IXP425. Un cristallo esterno a 33.33 MHz funge da segnale di clock per il processore di rete.

Le caratteristiche principali dell'IXP425 sono:

- processore Intel XScale Core, utilizzabile alla frequenza di sistema di 133 MHz (x2, x3 o x4)
- 3 NPE per il processamento di pacchetti/frame di rete
- 2 MAC IEEE 802.3 full-duplex a 10/100 Mbps
- SDRAM dedicata utilizzabile a 133 MHz; l'accesso ai banchi avviene 32 bits alla volta.
- bus di espansione
- supporto DSP per DSPs Texas Instruments
- interfaccia UTOPIA-2
- 2 bus seriali TDM high-speed: HSS-0 e HSS-1
- 2 UART, ciascuno supporta 921 Kbaud
- bus PCI 2.2
- 16 GPIO
- controllore di dispositivo USB 1.1

5.1.2 Flash/Boot ROM

La boot ROM dell'IXDPG425 è la Intel StrataFlash® E28F128J3x-150 di 16 Mbyte, connessa tramite il bus di espansione. La E28F128J3x supporta la common flash interface (CFI).

Sulla IXDPG425 ci sono due dispositivi flash: Flash 0, la flash di boot; Flash 1, non correntemente utilizzata. La IXDPG425 può eseguire il boot solo dalla Flash 0. La size del trasferimento dati (8 o 16 bit) è configurata sulla scheda tramite il jumper del bus d'espansione all'indirizzo 0.

5.1.3 Porta WAN ADSL

Le caratteristiche principali dell'interfaccia ADSL sono:

- Control Interface (CRTLE) – bus d'espansione
- Data Interface – UTOPIA Level 2
- Chip Select – EXP_CS4_N
- Ready Signal – On IO_WAIT_N
- Interrupt – GPIO 16 Mbyte
- Clock – esterno, 33Mhz

L'interfaccia WAN ADSL è fornita da una scheda mezzanine inserita nel socket ADSL. Il chipset ADSL si interfaccia con l'host, tramite il bus d'espansione, per scopi di controllo e inizializzazione.

5.1.4 Interfaccia PCI

L'IXDPG425 possiede un'interfaccia PCI (solo supporto a 33 MHz). Le periferiche PCI includono un controllore USB 2.0 e un socket miniPCI. Opzionalmente può essere installata un secondo socket miniPCI.

5.1.5 Interfacce Ethernet

Il processore di rete IXP425 ha due interfacce Ethernet 10/100 base T, implementate da un coprocessore Ethernet all'interno degli NPE. I coprocessori forniscono supporto per l'interfaccia MII verso le PHY. Essi supportano sia modo di operare full-duplex che half-duplex, e contengono anche due code FIFO a 256 byte, una per la trasmissione dei dati, l'altra per la ricezione dei dati.

Sulla piattaforma è utilizzato il componente Realtek RTL8305-SBVD, il quale contiene cinque trasmettitori a livello fisico e cinque unità MAC, con uno switch a livello Data Link.

5.1.6 Porte voce

L'interfaccia di linea è implementata utilizzando quattro componenti Silicon Laboratories ProSLIC Si3210, che integrano compensatori lineari A-law e u-law, decodificatori DTMF e generatori dual-tone per una singola linea telefonica. I componenti Si3210 sono connessi al processore di rete IXP425 tramite i bus SPI e PCM. I dati voce sono ricevuti/inviati da/verso il codec tramite l'interfaccia seriale PCM in maniera sincrona.

Un quinto connettore telefonico è fornito per permettere l'utilizzo di una convenzionale linea telefonica di scambio.

Il PCM lavora sotto il controllo del firmware NPE, il quale viene caricato in fase di startup per configurare le porte HSS dell'IXDPG425.

La decodifica dei segnali DTMF è implementata negli SLIC, e la sequenza di chiamata è supportata dal processore IXP425 e dal driver

di voce.

5.1.7 Porte USB

L'interfaccia USB 1.1 utilizza il controllore USB del processore di rete IXP425, e può lavorare solo ad una velocità di 12 Mbps.

Le due interfacce USB 2.0 sono sviluppate attorno all'adattatore NEC uPD720101.

5.1.8 Porta seriale

Il processore di rete IXP425 fornisce due porte seriali I/O asincrone dedicate (UART0 e UART1). Esse rispettano la specifica 16550, con flusso controllato e potenziato grazie a buffer di trasmissione e ricezione di 64 byte. Sulla IXDPG425 è utilizzata solo la UART0.

5.2 Piattaforma software

5.2.1 Intel© IXP400 Software v1.4

Intel© IXP400 Software v1.4 è una collezione di componenti software specifici per la linea di prodotti IXP42x e le loro schede di riferimento. Il diagramma a blocchi del software è riportato in *Figura 5.3*.

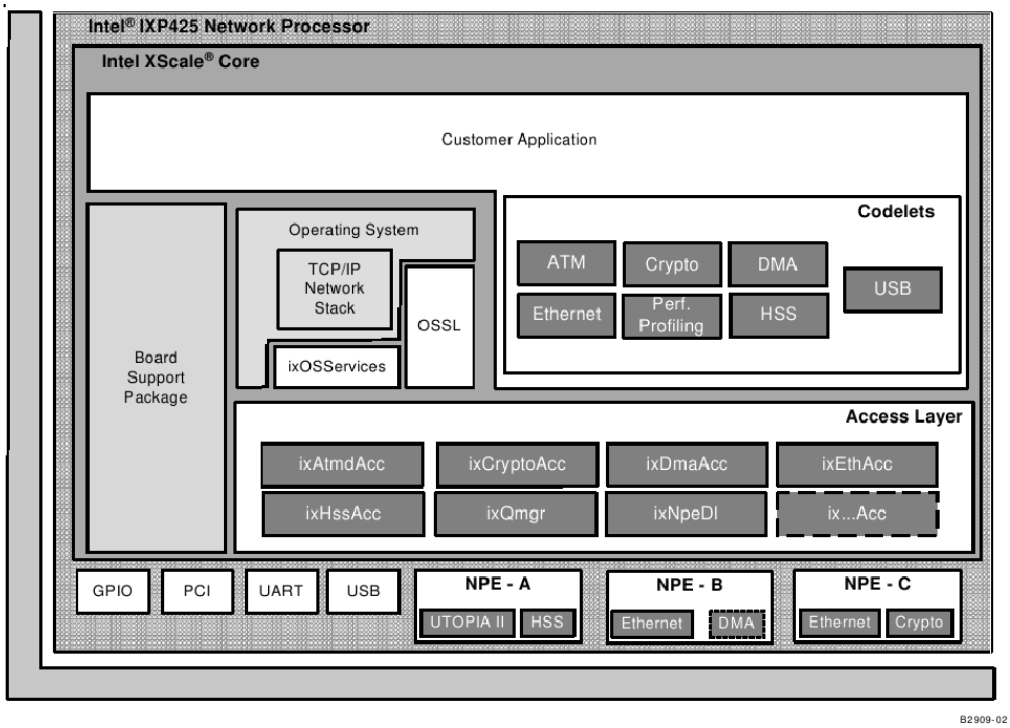


FIGURA 5.3: Diagramma a blocchi del software

Intel© IXP400 Software v1.4 si compone dei seguenti blocchi:

- NPE microcode, che consiste di uno o più file di istruzioni al NPE che implementano le funzionalità del NPE. Gli NPE sono processori RISC, embedded nel processore principale, e circondati da più coprocessori. I coprocessori forniscono specifici servizi hardware (come elaborazione Ethernet e interfacce MAC, crittografia, ecc). I file di istruzione NPE sono inclusi nella libreria della release 1.4, che include il downloader NPE, il quale fornisce la selezione della versione del codice ed il caricamento dei servizi. Sono fornite una varietà di immagini di codice per NPE, che abilitano diverse combinazioni di servizi.
- L'Access Layer (strato d'accesso) fornisce un'interfaccia software che dà il codice d'accesso alle funzioni fondamentali dell'IXP425.

Questo strato è composto da un insieme di componenti software (Access Layer Components), che il client può usare per configurare, controllare e comunicare con l'hardware. Specificatamente, la maggior parte dei componenti d'accesso forniscono una API interface alle specifiche funzionalità del Network Processor Engine. Nello specifico, le Access Layer Components principali sono:

1. ixAtmdAcc: ATM access-driver; questa componente fornisce un'astrazione del cell-processing hardware ATM sull'IXP425 e, quindi, mette a disposizione un'interfaccia per l'accesso ad ATM.
2. ixCryptoAcc: Security API; questa componente fornisce l'accesso al Security Hardware Accelerator, e mette a disposizione un supporto per l'autenticazione e servizi di crittografia/decrittografia necessari nelle applicazioni che utilizzano IPSec. In questo modo, i client possono alleggerire il carico d'elaborazione del core XScale, demandando l'applicazione degli algoritmi di cifratura e autenticazione ai coprocessori specializzati. Inoltre, ixCryptoAcc fornisce l'accesso ai servizi di crittografia per applicazioni Wireless Equivalent Privacy (WEP).
3. ixDmaAcc: DMA Access Driver; questa componente fornisce le funzionalità necessarie per effettuare trasferimenti DMA tra periferiche collegate ai bus AHB.
4. ixEthAcc: Ethernet Access Driver; questa componente

fornisce l'accesso ai dispositivi MAC Ethernet, per il controllo e la gestione dei pacchetti lato rete.

5. ixHssAcc: High Speed Serial Access Driver; questa componente fornisce alle applicazioni l'accesso ai coprocessori HSS e HDLC (High-Level Data Link Control).
6. ixNpeDl: NPE Downloader component; questa componente è stand-alone, e fornisce le funzioni per il caricamento delle immagini di microcode agli NPE A, NPE B e NPE C. La ixNpeDl contiene le librerie di default con le immagini di microcode; essa, inoltre, permette di fornire agli NPE immagini di microcode personalizzate, da utilizzare al posto di quelle di default.
7. ixQmgr: è una collezione di servizi responsabili della configurazione della Advanced High-Performance Bus Queue Manager (AQM). Questa componente è anche responsabile della gestione dei messaggi tra gli NPE e il software del core XScale. Le componenti che utilizzano ixQmgr possono registrare delle funzioni di callback per la gestione dei messaggi.

- Le Codelets, applicazioni d'esempio che utilizzano le Access Layer Components e le astrazioni messe a disposizione dal sistema operativo. Le codelets, anche se non sono esempi esaustivi delle funzionalità messe a disposizione degli sviluppatori, forniscono un buon punto di partenza per iniziare a sviluppare codice per analisi prestazionale, o anche per

applicazioni funzionali da mettere in commercio. Le codelets principali sono:

1. ATM Codelet (IxAtmCodelet), mostra un esempio di implementazione di un driver ATM funzionante, ed utilizza le Access Layer Components ixAtmdAcc, ixAtmm, ixAtmSch;
2. Crypto Access Codelet (IxCryptoAccCodelet), mostra come utilizzare la componente ixCryptoAcc. Esemplifica l'inizializzazione degli NPE e del Queue Manager, e l'utilizzo di operazioni di crittografia (3DES, AES, ARC4), decrittografia e autenticazione (SHA1, MD5). Sono inoltre dimostrati sia servizi IPSec che WEP.
3. DMA Access Codelet (IxDmaAccCodelet), esegue trasferimenti DMA per diversi modi di trasferimento, modi di indirizzamento e capacità.
4. Ethernet Access Codelet (IxEthAccCodelet), mostra sia servizi sul controllo, sia servizi sulla gestione.
5. HSS Access Codelet (IxHssAccCodelet), testa i servizi di pacchettizzazione e di selezione dei canali. La codelet opera una trasmissione dati e verifica che i dati ricevuti siano gli stessi di quelli trasmessi.

- Operating System Abstraction Layers (strati di astrazione del sistema operativo). Nell'architettura Intel© IXP400 Software v1.4 sono presenti due livelli di astrazione:

  1. Il livello OsServices fornisce un ristretto insieme di servizi

del sistema operativo. Tutte le altre componenti di accesso utilizzano questo modulo per risolvere le proprie dipendenze dal sistema operativo. Il livello OsServices rende disponibili i propri servizi anche alle codelets ed allo strato software applicativo.

2. L'Operating System Service Layer (OSSL) definisce un'interfaccia estesa e più potente per i servizi del sistema operativo.

Tra le funzionalità fornite dal livello OsServices e dal livello OSSL c'è una certa sovrapposizione ma, dove possibile, si preferiscono utilizzare i servizi del livello OsServices.

5.2.2 Intel© IXP400 DSP Software v2.6.2

L'Intel© IXP400 DSP Software v2.6.2 è un modulo software che fornisce le funzionalità di base per il processamento di contenuti multimediali, utilizzabili nelle applicazioni per gateway VoIP, con obiettivo gli IADs (Integrated Access Devices) di nuova generazione. In particolar modo, il software può essere visto come uno strato di elaborazione che possiede le seguenti caratteristiche:

- Supporto per i codec G.729ab-G.729a, G.711 u-law e A-law con pacchettizzazione a 10ms, G.711 Annex 2, G.723.1 a 5.3 e 6.3 Kbps, G.722, G.726 a 16, 24, 32 e 40 Kbps
- Packet loss concealment (PLC – occultamento dei pacchetti persi) per G.711, G.726 e G.722
- Interfaccia PCM configurabile in modalità wideband o

narrowband

- Commutazione del tipo di codifica "al volo"
- Commutazione automatica del tipo di decodifica in accordo ai pacchetti RTP in ricezione
- Supporto "multiple frames per packet". Il numero massimo di frame per pacchetto è:
    - 6 per G.711 e G.722
    - 8 per G.723
    - 9 per G.726 a 40Kbps
    - 12 per G.726 a 32Kbps
    - 16 per G.726 a 24Kbps
    - 24 per G.726 a 16Kbps e G.729
- Modifica dinamica dei frames per packet "al volo"
- Routing dinamico degli streams audio tra le risorse
- Supporto per il controllo del guadagno automatico (AGC)
- Supporto per il controllo automatico del livello (ALC)
- Cancellazione dell'eco (EC – Echo Cancellation)
- Generazione e rilevazione di toni DTMF
- Ricezione di cifre DTMF
- Rilevazione di toni del fax
- Capacità di generazione di toni modulati
- Rilevazione e generazione di toni specificati dall'utente
- Generazione di toni per chiamate US, Cina e Giappone
- Algoritmi per la gestione del jitter
- Mixer audio per three-way call e piccole conferenze

- Player audio per voce e musica d'attesa
- Commutazione TDM a bassa latenza

Una volta compilato il codice sorgente, viene generato il codice oggetto "ixp400_codelets_dspEng.o", inseribile come modulo del kernel, e dal quale bisogna creare un nodo da utilizzare come driver (major number uguale a 253), "ixDspCodeletModule".

5.2.2.1 Architettura dell'Intel© IXP400 DSP Software v2.6.2

Il software è implementato come modulo indipendente, con i propri processi e ambiente di esecuzione. L'Intel© IXP400 DSP Software v2.6.2 fornisce tre interfacce all'applicazione, per comunicare informazioni di controllo, dati PCM e pacchetti voce. Inoltre, esso consiste di alcune risorse di elaborazione dati multimediali (Media Processing Resource – MPR), che possono essere indirizzate indipendentemente dall'applicazione; ogni risorsa possiede le proprie funzioni d'elaborazione, controllate dai messaggi.

La *Figura 5.4* mostra l'architettura dell'intero modulo.

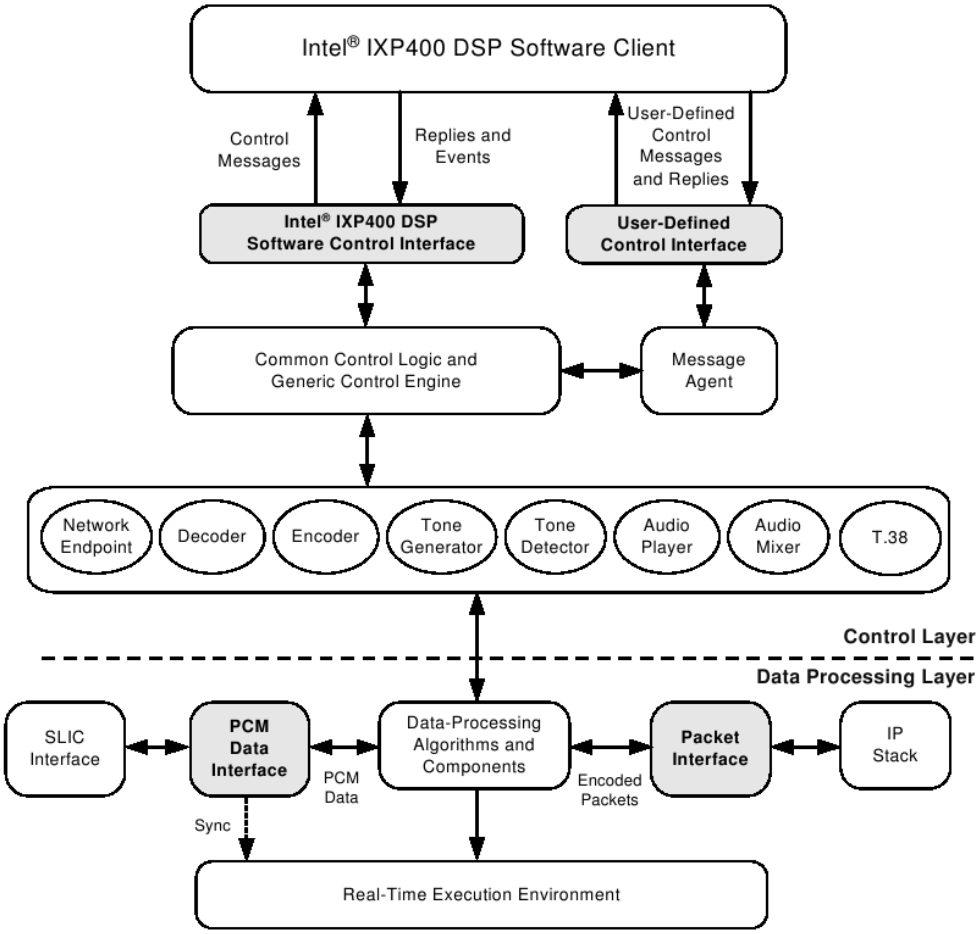


FIGURA 5.4: Architettura dell'Intel IXP400 DSP Software

Control Interface

Le applicazioni comunicano con il DSP fondamentalmente attraverso l'interfaccia di controllo, definita da un insieme di funzioni, messaggi e macro.

Esistono due code di messaggi, una per i messaggi in arrivo (in-bound messages) ed una per i messaggi in spedizione (out-bound messages). La gestione di queste due code è affidata a due funzioni, xMsgSend() e xMsgReceive(). La ragione dell'isolamento del software DSP

dall'applicativo tramite le code di messaggi è quella di evitare l'accesso a funzioni interne di controllo da parte di più processi dell'applicativo stesso.

Il software DSP invia risposte o eventi all'applicazione mediante la coda di messaggi in uscita. L'applicazione può recuperare i messaggi tramite la funzione bloccante xMsgReceive(). Una terza funzione di controllo, xMsgWrite(), permette all'applicazione di scrivere messaggi esterni direttamente nella coda di uscita. Il meccanismo di gestione delle code è raffigurato in *Figura 5.5*.

A causa della limitatezza della lunghezza delle code, queste possono andare in overflow ed i messaggi potrebbero andare persi se l'applicazione continuasse ad inviare messaggi senza recuperarli.

Inoltre, è utilizzato un meccanismo di spedizione dei messaggi basato sulla copia. Ciò significa che viene passata una copia effettiva del messaggio dal mittente al destinatario, piuttosto che un puntatore al messaggio stesso. In questo modo si evita di allocare memoria dinamica e, dal momento in cui non c'è condivisione di memoria tra il software DSP e l'applicazione, questa può riutilizzare lo spazio allocato per un messaggio per qualsiasi scopo immediatamente dopo che il messaggio è stato utilizzato.

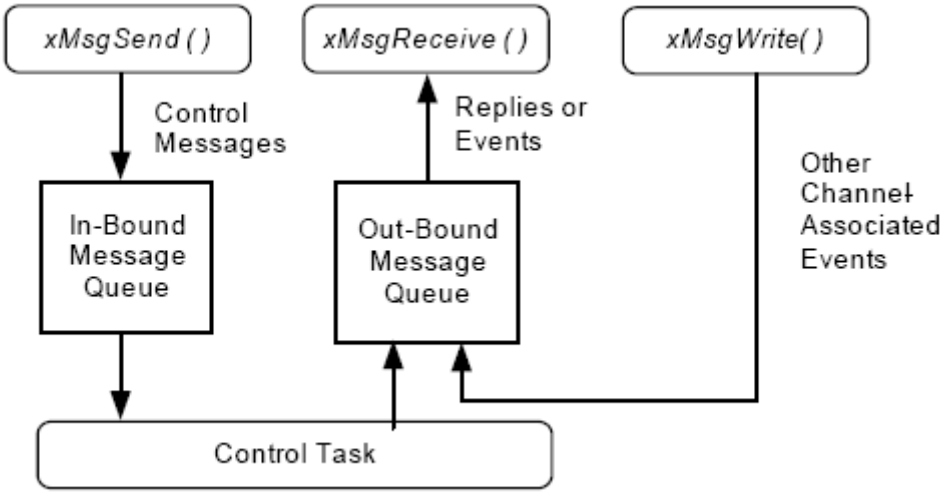


FIGURA 5.5: Gestione delle code dei messaggi

PCM Data Interface

I dati PCM rappresentano lo stream di dati audio tra il software DSP e l'interfaccia telefonica. Il modo più semplice per la loro gestione è utilizzare il bus TDM dell'Intel IXP425, anche chiamato HSS. Al contrario dell'interfaccia di rete, l'interfaccia HSS è integrata nel software DSP, il che permette un trasferimento più efficiente di dati PCM in real-time.

Dal punto di vista dell'applicazione, HSS può essere vista come hardware da configurare specificatamente per comunicare con l'interfaccia esterna ad essa connessa. Una volta configurata, l'applicativo non è più coinvolto. La configurazione avviene mediante la specifica del formato dei segnali presentati al TDM, inclusi la frequenza di clock, il time slot, la sincronizzazione dei frame, l'endian, ecc.

Il funzionamento dell'interfaccia dati PCM è rappresentato in *Figura*

5.6.

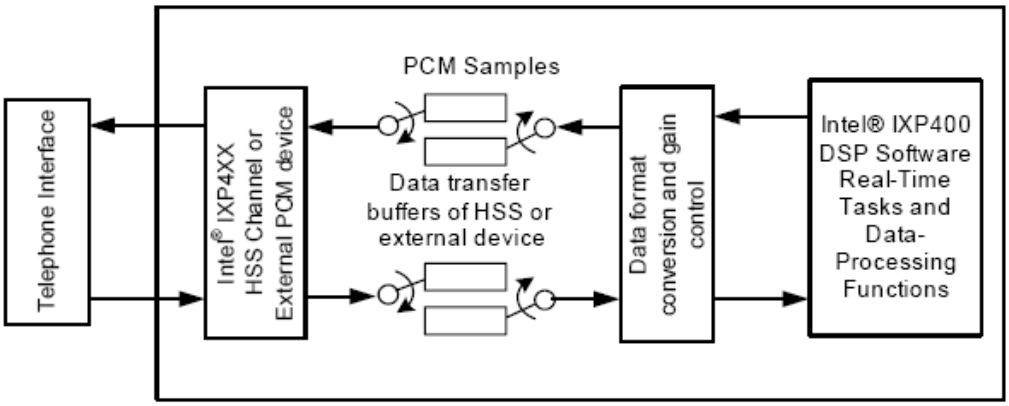


FIGURA 5.6: Funzionamento dell'interfaccia PCM

Packet Interface

Rispetto all'interfaccia dati PCM, l'interfaccia dei pacchetti è un protocollo software puro che definisce il modo in cui i pacchetti di dati audio codificato sono scambiati tra il software DSP e l'interfaccia IP.

Esistono due funzioni ed un formato dei pacchetti coinvolti in questa interfaccia, come mostrato in *Figura 5.7*. Il software DSP definisce il formato dei pacchetti e fornisce la funzione per la loro ricezione. L'applicativo utente, invece, è responsabile della funzione di trasmissione.

In ingresso (pacchetti entranti lato rete), l'interfaccia IP converte ogni pacchetto VoIP che riceve in un pacchetto comprensibile al software DSP, poi chiama la xPacketReceive() per spedirlo al DSP. L'applicativo utente necessita di decodificare i pacchetti IP entranti, per spedire il payload RTP al software DSP con gli appropriati formato

dell'intestazione, timestamp e canale. La funzione xPacketReceive() copia il pacchetto nel jitter buffer senza alcuna ulteriore elaborazione. Questo meccanismo copy-based permette al chiamante della funzione di liberare o riutilizzare il buffer contenente il pacchetto per qualsiasi scopo.

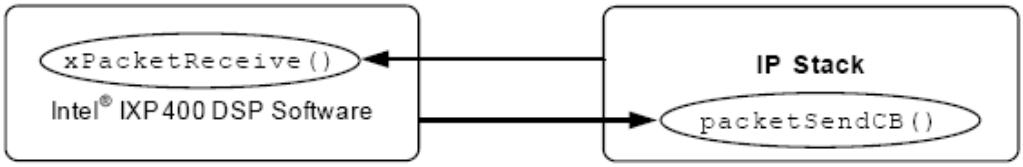


FIGURA 5.7: Meccanismo di comunicazione tra rete e DSP

Media Processing Resources - MPRs

Esistono nove risorse che collaborano insieme per fornire le elaborazioni necessarie ad un gateway VoIP. La comunicazione di informazioni di controllo avviene attraverso messaggi predefiniti, di cui alcuni comuni a tutte le risorse, altri unici e specifici per una particolare risorsa.

Le componenti per l'elaborazione multimediale sono le seguenti:

- Network Endpoint: questa componente è l'unità d'elaborazione dati che connette l'interfaccia HSS al resto delle componenti MPR; fornisce funzionalità di compressione e decompressione A-law e u-law, filtro passa-alto, cancellazione dell'eco in ricezione, e funzioni supplementari quali timer e flash hook detection; è sensibile ad eventi di modifica dello stato di hook e timer

expired.

I canali del Network Endpoint possono essere configurati in modalità narrowband o wideband durante l'inizializzazione.

Il filtro passa-alto è applicato all'audio in ingresso dall'interfaccia HSS per rimuovere il rumore a bassa frequenza e salvaguardare altri algoritmi dalla dannosa corrente di bias; il filtro utilizza un taglio a 3dB alla frequenza di 270Hz (narrowband) o 150 Hz (wideband), e non può essere disabilitato.

La cancellazione dell'eco è la funzione più significativa di questa componente. EC elimina l'eco generato dall'interfaccia telefonica locale, in modo che l'altro partecipante alla conversazione non udirà eco; in altre parole, il beneficiario dell'Echo Cancellation è il partecipante remoto.

- Decoder: questa componente riceve i pacchetti audio codificati dall'interfaccia IP, e li converte in stream audio verso l'interfaccia HSS; il decoder fornisce funzionalità di decodifica, controllo automatico del livello e/o controllo del volume, generazione di rumore di fondo (Comfort Noise Generator – CNG) e compensazione del jitter; è sensibile ad eventi di pacchetti persi o corrotti, e modifica del tipo di payload RTP.

  L'algoritmo PLC (Packet Loss Conceilment) utilizza i dati precedenti per riparare i frames persi, ma non può riparare grosse quantità di dati persi.

  Il jitter buffer regola il flusso di dati proveniente dalla rete verso l'interfaccia HSS. Ciò è necessario affinché la codifica audio

avvenga in maniera real-time, utilizzando il protocollo RTP. Questo significa che i pacchetti possono essere ritardati, disordinati, duplicati o persi. Il jitter buffer monitora le condizioni della rete verificando alcuni parametri dell'header RTP dei pacchetti.

- Encoder: la funzione primaria di questa componente è quella di codificare e pacchettizzare i dati audio provenienti dall'interfaccia HSS, e spedirli lato rete. Esistono due elementi del controllo automatico del guadagno: AGC in uscita e ALC in ingresso; uno solo dei due dovrebbe essere attivo, relativamente alle funzioni di controllo del guadagno implementate dal partecipante remoto. Tipicamente, le chiamate VoIP utilizzano ALC.

  Un'altra funzione importante dell'encoder è la rilevazione del suono: l'algoritmo VAD riesce a distinguere il parlato dal silenzio (rumore di fondo); durante i periodi di silenzio, l'encoder manda solo pacchetti molto più piccoli, contenenti parametri di rumore (ciò riduce il traffico di rete).

  Come il decoder, l'encoder è sensibile ad eventi di pacchetto corrotto e modifica del tipo di payload RTP.

- Tone Generator: questa componente è responsabile della generazione di toni a singola o doppia frequenza e toni modulati in ampiezza. Possiede un insieme di toni predefiniti, ma possono essere aggiunti toni definiti dall'utente, o combinati insieme i segnali in un unico tono. Il Tone Generator è una risorsa

narrowband, e non può produrre frequenze superiori a 4KHz.

- Tone Detector: come il Tone Generator, questa componente è una risorsa narrowband, e può rilevare toni a singola o doppia frequenza, con un range di frequenze che va dai 300Hz ai 3.5KHz, tramite un analizzatore FFT. Oltre ai toni predefiniti, si possono aggiungere nuove tabelle per la rilevazione di segnali definiti dall'utente. Per la rilevazione di toni a frequenza doppia, è necessario che le frequenze differiscano di almeno 200Hz.
- Audio Player: questa componente riproduce i dati audio pre-registrati; possono essere registrati permanentemente fino a 32 suoni in formato G.711 A/u-law o G.729, e caricati in memoria in formato grezzo, senza VAD o altre informazioni extra.
- Audio Mixer: questa componente miscelare gli streams audio per conferenze. L'utilizzo primario è per le chiamate three-way. Miscelare troppi streams può dare luogo a perdita di qualità della voce, come rumore di fondo "built-up" o volumi non bilanciati.
- T.38 Fax: fornisce la funzionalità di gateway per fax real-time tra l'interfaccia TDM e la rete IP.
- Message Agent: non fornisce vere e proprie funzionalità per l'elaborazione multimediale dei dati, ma permette la conversione dei messaggi definiti dall'utente e l'esecuzione del controllo che ne consegue.

Capitolo 6

TOOLCHAIN

Per l'integrazione del modulo krtpd nell'architettura hardware/software di cui al Capitolo 5, sono stati utilizzati i seguenti software:

- Wireshark per la cattura dei pacchetti, in particolare per controllare se il sistema riusciva ad inviare pacchetti RTP a macchine remote, e per verificare i parametri prestazionali più interessanti.
- MontaVista Linux 3.1, con relativa toolchain di cross-compilazione per architettura ARM.
- Source Navigator, per navigare attraverso tutti i sorgenti, compresi quelli del kernel e della piattaforma software fornita dalla Intel.
- Client NX per il controllo di una macchina da remoto.
- Minicom 2.0, per emulazione di un terminale sull'hardware.

6.1 Wireshark

Wireshark (versione successiva di Ethereal) è un analizzatore di rete open source, rilasciato sotto licenza GNU GPL, disponibile per diverse piattaforme e sistemi operativi. Per ammissione degli stessi autori, è un prodotto non ancora completo, ma rappresenta uno strumento essenziale per ogni amministratore di rete, ed il fatto che il programma occupi, da diversi anni, una posizione di rilievo all'interno della lista dei principali strumenti di sicurezza, stilata da insicure.org, ne è sicuramente la riprova.

Wireshark è un programma per l'analisi del traffico di rete, che agisce fino al livello due del modello ISO-OSI (collegamento dati), riuscendo a riconoscere all'interno di questo una serie di protocolli al livello tre e quattro del modello ISO-OSI (rete). In particolare, individua correttamente molti protocolli collegati a IPv4 e IPv6.

È pensato principalmente per accumulare il traffico intercettato, allo scopo di consentirne un'analisi dettagliata in un momento successivo; nello stesso modo è predisposto per accedere ad informazioni di questo genere accumulate da programmi diversi, così come è in grado di esportare i propri dati in formati diversi. Wireshark consente anche una visualizzazione in tempo reale del traffico in corso. In questo senso, Wireshark è un ottimo strumento didattico per lo studio delle reti. Wireshark viene usato normalmente attraverso il sistema grafico X e deve funzionare con i privilegi dell' utente root, per poter accedere direttamente all' interfaccia di rete da sondare. L' eseguibile da avviare è *wireshark*.

Si mostra di seguito il funzionamento di Wireshark in modo interattivo, senza l' uso di opzioni nella riga di comando.

La finestra principale di Wireshark si divide in tre parti: in quella superiore appare l'elenco dei pacchetti intercettati con una descrizione essenziale del loro contenuto; selezionando un pacchetto nella parte superiore, in quella centrale appare un elenco ad albero di componenti del pacchetto stesso; selezionando una voce nell' elenco del riquadro centrale, appare, in quello inferiore, evidenziata, la porzione di pacchetto che lo riguarda (*Figura 6.1*).

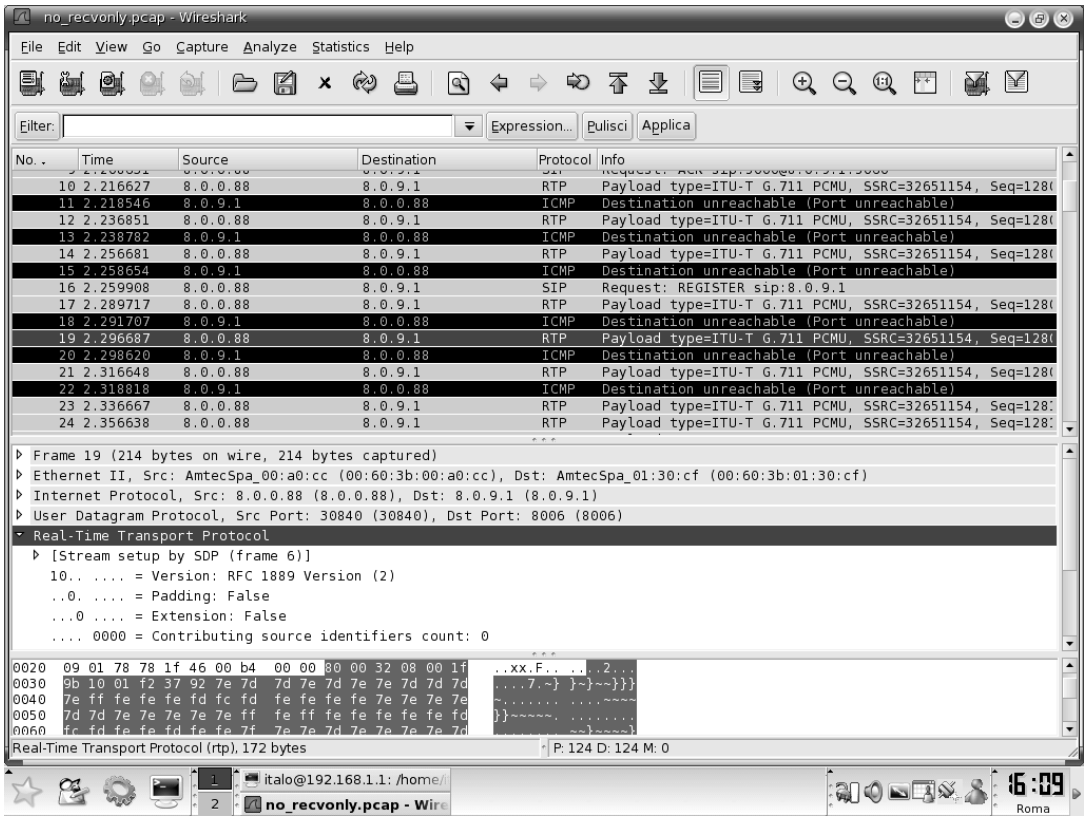


FIGURA 6.1: Cattura di pacchetti da parte di Wireshark

Durante lo sviluppo della tesi, Wireshark è risultato molto utile per l'analisi statistica dei parametri prestazionali dei flussi RTP. In *Figura 6.2* è possibile vedere alcuni dati relativi ai suddetti stream.

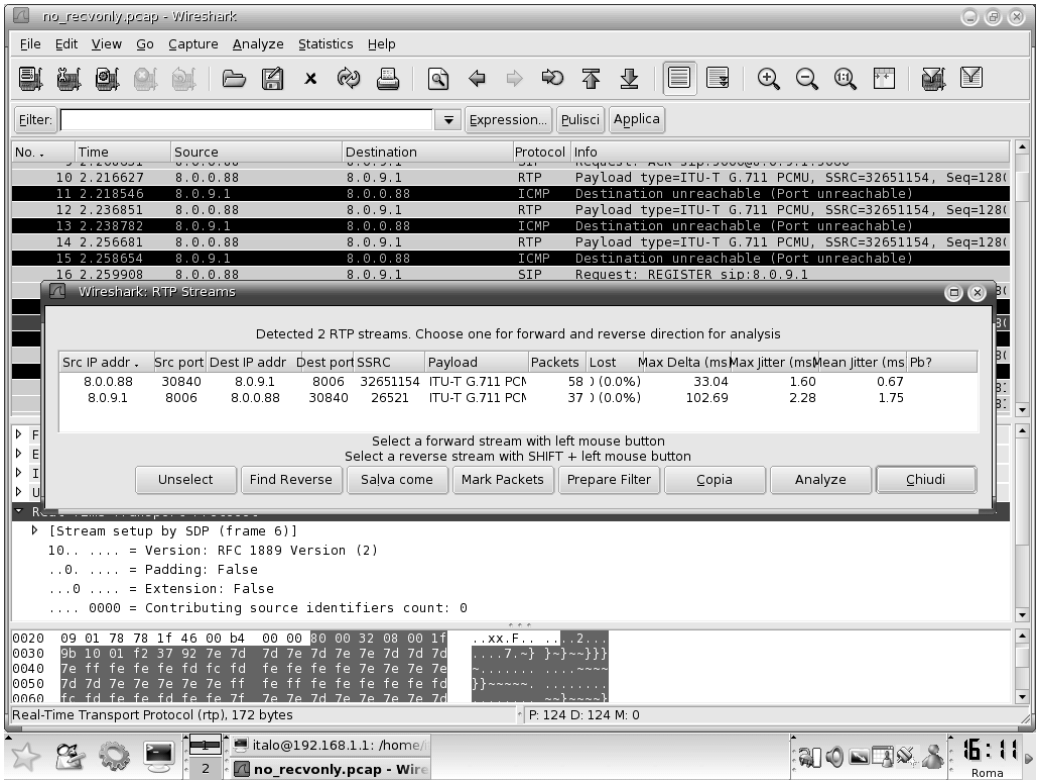


FIGURA 6.2: Analisi di flussi di pacchetti RTP

6.2 MontaVista Linux 3.1

MontaVista Linux 3.1 è una delle principali distribuzioni Linux utilizzate nei sistemi embedded. In particolare, la versione del kernel della distribuzione è 2.4.20.

MontaVista è una distribuzione snella, che prevede il supporto per architettura ARM (ciò di cui avevo bisogno per utilizzare la macchina

della Intel IXDPG425). Il pacchetto software è corredato di una toolchain di cross-compilazione per questa piattaforma, che è stata utilizzata per produrre tutti gli eseguibili testati sul gateway. In *Figura 6.3* è possibile vedere la schermata di login della distribuzione MontaVista, eseguita sul sistema embedded.

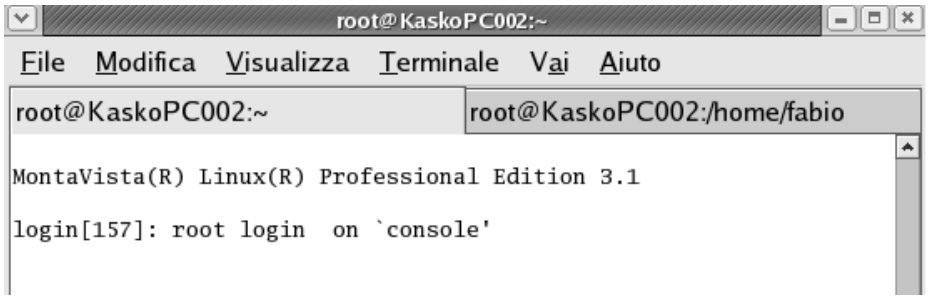


FIGURA 6.3: MontaVista Linux 3.1

6.3 Source Navigator

Source Navigator è un tool per l'analisi del codice sorgente (panoramica in *Figura 6.4*). Con esso si può modificare il codice, mostrare le relazioni tra classi, funzioni e membri, e mostrare gli alberi delle chiamate di una funzione. E' possibile anche eseguire il build del progetto, sia tramite il makefile che utilizzando Source Navigator stesso per generarne uno. Questo tool funziona tramite la Insight GUI interface per GDB; supporta i linguaggi C, C++, Java, Tcl, FORTRAN e COBOL, e fornisce un SDK per scrivere i propri parser.

Source Navigator ha come usi principali:

- Analizzare come un cambiamento avrà effetto sul codice di moduli esterni.
- Trovare qualsiasi linea di codice dove una determinata funzione viene chiamata.
- Trovare qualsiasi file che include uno specifico header file.
- Trovare un stringa data in tutti i file sorgenti tramite il tool grep.

Source Navigator viene rilasciato sotto la licenza GNU.

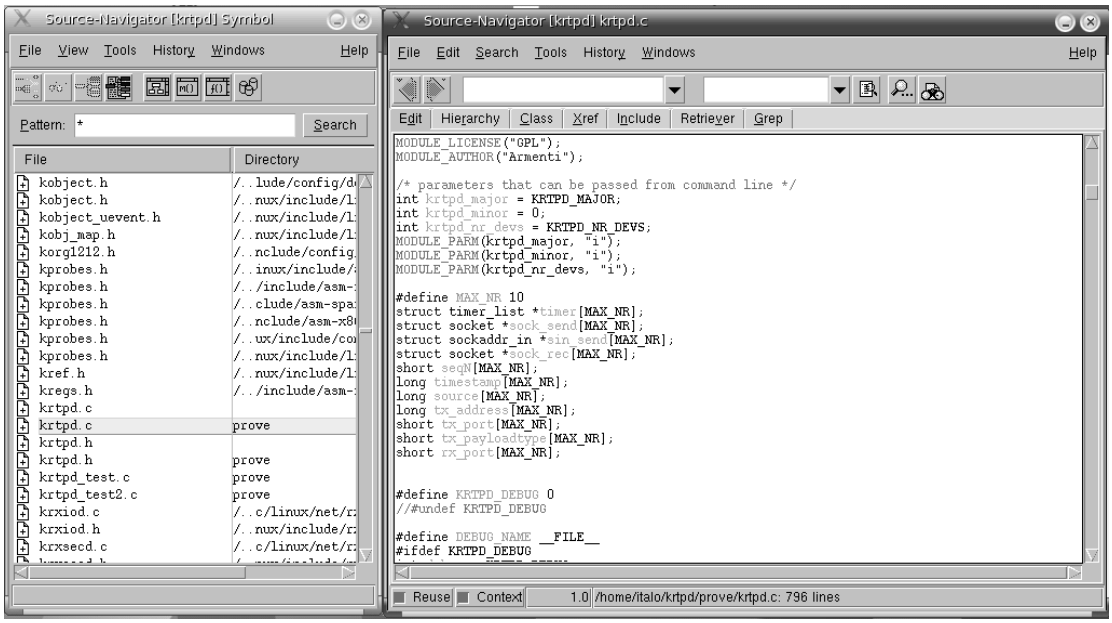


FIGURA 6.4: Panoramica di Source Navigator (a sinistra tutti i sorgenti del programma e del kernel di Linux, a destra uno dei file)

6.4 NoMachine NX

NoMachine NX è un Terminal Server ed un Remote Access basato su di una architettura che nasce da tecnologie proprietarie integrate con protocolli e standard aperti. Questa include l'eccezionale compressione, la durata della sessione e la gestione delle risorse del sistema X-Window, e l'integrazione con audio non particolarmente potente, capacità di stampa e condivisione risorse di Unix.

NX è una suite di applicazioni client/server per l'esecuzione remota di desktop e applicazioni Linux e Windows su sistemi thin, compresi i device mobili. Il server NX gira su piattaforme Linux e Sun Solaris: il backend assicura una completa gestione dell'utenza e delle connessioni. All'utente è garantita la massima flessibilità: può connettersi al server NX e può accedere a dati e applicazioni in ogni luogo, da qualsiasi dispositivo e sistema operativo, indipendentemente dal tipo di connessione.

Come si può facilmente intuire il programma NX Server è stato installato sulla macchina da gestire da remoto, mentre il programma NX Client è stato installato sulla mia macchina. La *Figura 6.5* illustra l'utilizzo di NX Client.

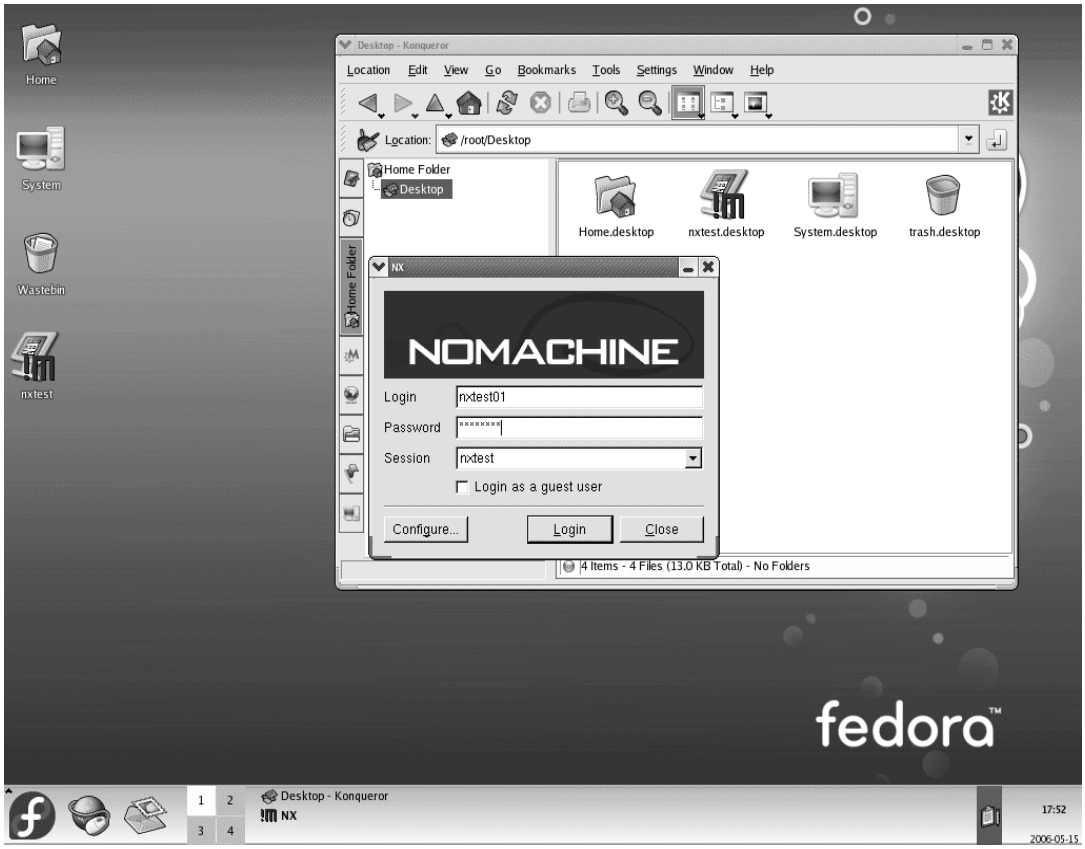


FIGURA 6.5: Login al computer remoto

6.5 Minicom 2.0

Minicom è un emulatore di terminale, utilizzato per visualizzare i dati scambiati con il gateway tramite la porta seriale RS232. Alcuni dei parametri di configurazione sono visibili in *Figura 6.6*.

Tramite Minicom è possibile, per esempio, effettuare una chiamata tramite modem connesso alla porta seriale, e vederne l'output a video.

Per una descrizione più approfondita della configurazione di Minicom per la comunicazione con il gateway Intel IXDPG425, vedere il paragrafo 7.5.

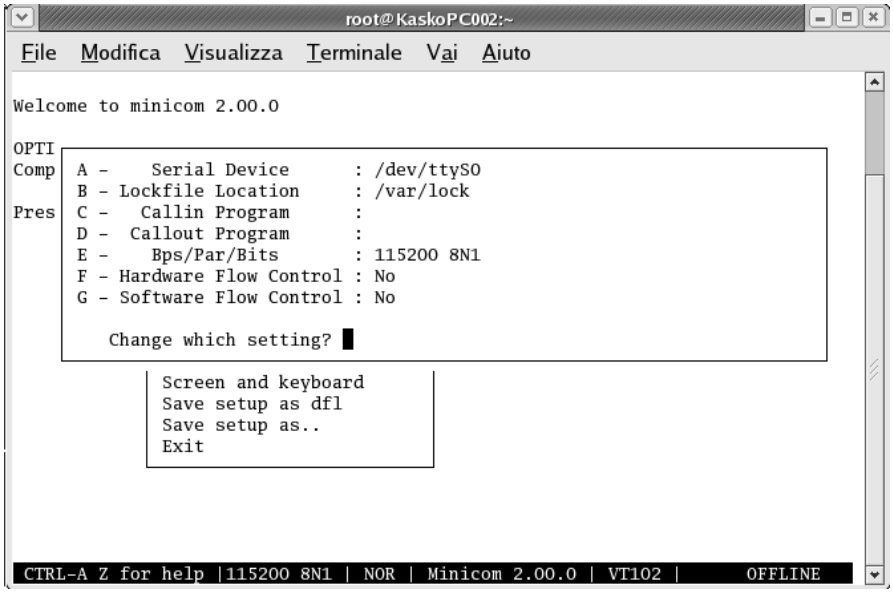


FIGURA 6.6: Alcune opzioni di Minicom

Capitolo 7

INTEGRAZIONE KRTPD – IXP400 DSP SOFTWARE

7.1 Premessa

Per l'ottenimento degli obiettivi preposti, è stato utilizzato il driver a caratteri krtpd per Linux (kernel 2.6), sviluppato nella precedente tesi di Italo Armenti dal titolo "Sviluppo di uno stack RTP in Kernel Space per applicazioni VoIP time critical in ambiente Linux". Questo modulo software si interpone tra il modulo DSP (IXP400 DSP Software nella fattispecie) ed i Network Layers, in modo da garantire una comunicazione di uno stream dati, rispettando le specifiche del protocollo RTP.

La motivazione di questo adattamento proviene dal fatto che, solitamente, le applicazioni in Linux sono eseguite in user-space, mentre il software del DSP è eseguito in kernel-space. Ciò significa che i pacchetti in arrivo dalla rete (kernel-space) risalgono a livello applicativo (user-space), il quale li indirizza verso il driver DSP (kernel-

space); analogamente, i campioni audio provenienti dal DSP sono spediti all'applicativo utente, il quale li indirizza verso la rete. Questo comportamento è visibile in *Figura 7.1*.

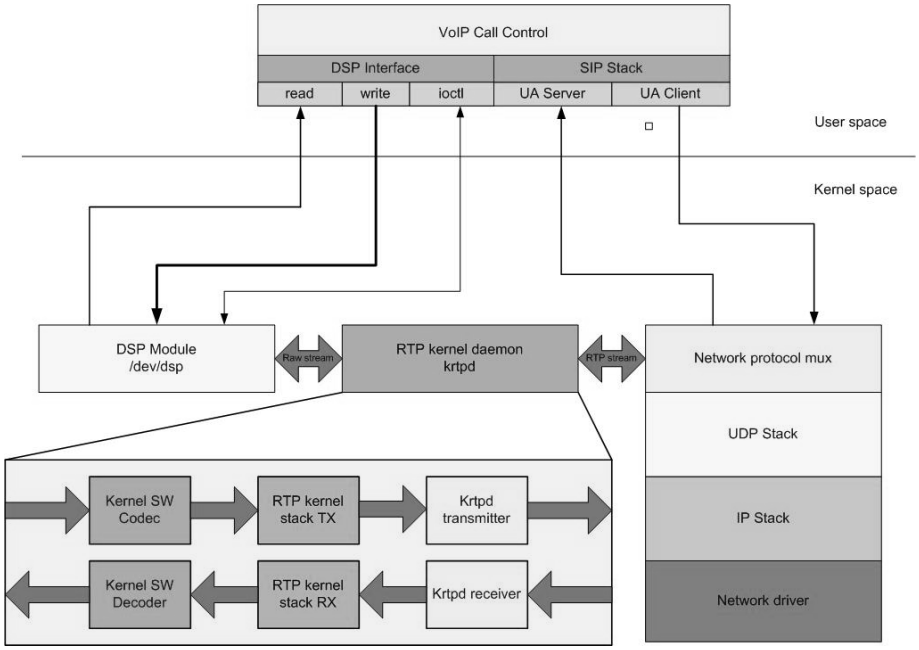


FIGURA 7.1: Flusso dei dati rete/DSP in un applicativo VoIP e relativo ruolo del krtpd

Interporre il driver krtpd significa far rimanere lo stream di dati da trasmettere/ricevere al livello kernel-space, incrementando sensibilmente le prestazioni dell'intero software.

7.2 Panoramica sul "krtpd"

Il modulo krtpd è un driver a caratteri, implementato in linguaggio C, per il sistema operativo Linux (kernel 2.6).

Le funzioni più importanti del software sono krtpd_threadsend, krtpd_threadreceive, krtpd_sendfile, krtpd_sendto, krtpd_continuasend, krtpd_receivefrom e krtpd_ioctl.

La funzione krtpd_threadsend ha il compito di creare un nuovo kernel thread, associato alla funzione krtpd_sendfile. La krtpd_sendfile implementa il comportamento di trasmissione del modulo; essa crea un primo pacchetto RTP, con marker pari a 1, sincronizzatore di sorgente scelto casualmente, timestamp e sequence number pari a 0. Il payload di questo pacchetto è riempito con una serie di "A", mentre il tipo di payload, porta e indirizzo di destinazione sono parametri passati da user-space. Una volta spedito il pacchetto attraverso la krtpd_sendto, la funzione inizializza un timer che, dopo 30ms, richiama la funzione krtpd_continuasend. Questa funzione è analoga alla krtpd_sendfile, ma i pacchetti RTP che crea hanno marker pari a 0, timestamp e sequence number adeguati al numero di pacchetto inviato, e payload riempito con una serie di "5". Una volta spedito il pacchetto tramite la krtpd_sendto, la funzione inizializza un timer che, dopo 30ms, richiama la krtpd_continuasend.

La funzione krtpd_threadreceive, analogamente alla krtpd_threadsend, è responsabile della creazione di un kernel thread associato alla funzione krtpd_receivefrom. Questa, a sua volta, si pone in ascolto sulla porta di ricezione UDP (parametro passato da user-space) e, per

ogni pacchetto in arrivo, ne visualizza in output i parametri RTP fondamentali.

La funzione krtpd_ioctl, invece, è responsabile dell'I/O da/verso user-space; tramite questa routine, infatti, è possibile settare tipo di payload coinvolto nella comunicazione, porta UDP e indirizzo IP di destinazione dello stream e porta UDP di ricezione; inoltre, questa funzione mette a disposizione la possibilità di iniziare o terminare una trasmissione/ricezione, attraverso opportune chiamate alle funzioni sopra descritte.

7.3 Porting del krtpd verso il kernel 2.4

Le piattaforme Intel IXP400 Software v1.4 e Intel IXP400 DSP Software v2.6.2 sono implementate secondo le specifiche del kernel 2.4; ciò ha reso necessario il porting del driver krtpd dal kernel 2.6 al kernel 2.4.

In particolare, le modifiche apportate sono state le seguenti:

- nella sezione "Include" è stato eliminato il codice relativo alle inclusioni delle librerie "linux/cdev.h", "asm/unistd.h" e "linux/moduleparam.h";
- nella sezione "Include" è stata aggiunta l'inclusione della libreria "linux/devfs_fs_kernel.h";
- nella funzione "krtpd_ioctl" è stato modificato il controllo relativo all'accesso a zone di memoria proibite;

- è stata modificata la definizione della struttura "krtpd_fops", indispensabile per rendere il modulo un driver a caratteri;
- nella "krtpd_init" è stata eliminata la creazione dei nodi di device (l'operazione viene effettuata manualmente, tramite script);
- è stato modificato il prototipo della funzione "krtpd_cleanup", ed eliminata la "unregister_chrdev_region" nella stessa funzione.

Dopo aver effettuato le suddette modifiche, il driver è stato ricompilato con successo su di una macchina (architettura i586) con kernel 2.4.20, e verificato il corretto funzionamento con un applicativo di prova.

7.4 Preparazione di kernel e RAM disk image

Il primo passo necessario per l'utilizzo della IXDPG425 è stata la cross-compilazione di un kernel da far eseguire all'avvio, unitamente alla preparazione di un'immagine di RAM disk ad-hoc.

7.4.1 Cross-compilazione del kernel

L'architettura software utilizzata è implementata per un kernel 2.4 di Linux; in particolare, per lo sviluppo della tesi, è stata utilizzata la distribuzione MontaVista Linux 3.1, con kernel 2.4.20.

Prima della cross-compilazione, è stato necessario configurare il kernel nel seguente modo:

- nella sezione "System Type" è stato selezionato IXP4XX come ARM system type, e sono stati abilitati il supporto per schede IXDPG425 e la compilazione delle Access Library IXP425;

- nella sezione "General Setup" è stata indicata la generazione di un kernel big-endian;
- nella sezione "Memory Technology Devices (MTD)", sottosezione "RAM/ROM/Flash chip drivers", è stato abilitato il supporto per chips flash Intel/Sharp; nella sottosezione "Mapping drivers for chip access", è stato indicato "CFI Flash device mapped on IXP4XX systems";
- nella sezione "Block devices" sono stati abilitati il supporto a RAM disk e a init-RAM disk, con una size del RAM disk di default pari a 25600;
- nella sezione "Network device support", sottosezione "Ethernet 10/100Mbit", è stato abilitato supporto a Ethernet Device IXP425; nella sottosezione "Ethernet 1000Mbit", è stato abilitato il supporto a Ethernet Gigabit per Intel.

Quindi è stato avviato il processo di cross-compilazione, che ha generato un'immagine compressa del kernel (successivamente rinominata "zImage.ixdpg") ed i moduli necessari al funzionamento della piattaforma IXDPG425 ("ixp400.o", "ixp400_dsr.o" e "ixp400_codelets_dspEng.o").

Infine, è stata cross-compilata l'applicazione demo per l'utilizzo del DSP ("IxDspCodeletApp").

7.4.2 Generazione del RAM disk image

L'IXDPG425 non è dotata di dispositivi di memoria di massa e, per poter disporre di un'alberatura dalla quale caricare il software di cui abbiamo bisogno, si necessita la creazione di un RAM disk.

Il RAM disk deve contenere le directories ed i files necessari per l'utilizzo del sistema; in particolare, l'alberatura creata è quella tipica dei sistemi unix-like.

Per generare un'immagine compressa di RAM disk, bisogna innanzitutto allocare spazio per un file su disco delle dimensioni del RAM disk. Questa operazione è semplice, e attuabile con il comando "dd if=/dev/zero of=ramdisk bs=1024k count=DIM"; in questo modo sarà generato un file, chiamato "ramdisk", e di dimensione DIM megabyte. Per formattare il RAM disk con file system ext2 bisogna lanciare il comando "/sbin/mke2fs -F -m0 ramdisk". A questo punto è possibile montare in loop il RAM disk (vuoto) appena generato su una qualsiasi directory, per poi crearvi la propria alberatura. Successivamente bisogna smontare il RAM disk e comprimerlo; è possibile, ad esempio, lanciare il comando "gzip –v9 ramdisk", che creerà l'effettiva immagine compressa del RAM disk nel file "ramdisk.gz".

Nella fattispecie, l'immagine compressa del RAM disk si chiama "new_ramdisk.gz", ed in prima istanza contiene alcuni files di sistema, oltre ad una directory "/test", con uno script di inizializzazione "start" ed i moduli appena cross-compilati "ixp400.o", "ixp400_dsr.o" e

"ixp400_codelets_dspEng.o", nonché l'applicativo demo "IxDspCodeletApp".

7.5 Inizializzazione e configurazione IXDPG425

Il gateway Intel® IXDPG425 è preinstallato con il boot loader RedBoot, il quale permette, mediante uno script di avvio, il caricamento del kernel e dell'immagine del RAM disk da utilizzare.

Dopo aver collegato la porta seriale dell'IXDPG425 ad un pc, bisogna utilizzare un emulatore di terminale per comunicare con il gateway. Nella fattispecie, è stata avviata una sessione minicom. I parametri da utilizzare per la comunicazione sono i seguenti:

- Bits per secondo: 115.200
- Data bits: 8
- Nessuna parità
- Bits di stop: 1
- Nessun controllo del flusso

In *Figura 7.2* è possibile vedere l'effettiva configurazione dei parametri di comunicazione, durante una sessione minicom.

```
root@KaskoPC002:~
File  Modifica  Visualizza  Terminale  Vai  Aiuto

Welcome to minicom 2.00.0

OPTI
Comp  A -    Serial Device      : /dev/ttyS0
      B - Lockfile Location     : /var/lock
Pres  C -   Callin Program      :
      D -  Callout Program      :
      E -    Bps/Par/Bits       : 115200 8N1
      F - Hardware Flow Control : No
      G - Software Flow Control : No

         Change which setting?

              Screen and keyboard
              Save setup as dfl
              Save setup as..
              Exit

 CTRL-A Z for help |115200 8N1 | NOR | Minicom 2.00.0 | VT102 |      OFFLINE
```

FIGURA 7.2: Parametri di configurazione di Minicom

Avviando la IXDPG425, il boot loader viene automaticamente eseguito, e nel terminale dovrebbero apparire le stringhe del processo di bootstrap (*Figura 7.3*).

```
root@KaskoPC002:/home/fabio
File  Modifica  Visualizza  Terminale  Vai  Aiuto

Welcome to minicom 2.00.0

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Sep 12 2003, 17:27:02.

Press CTRL-A Z for help on special keys

Ethernet eth0: MAC address 00:00:84:00:50:02
IP: 192.168.1.159/255.255.255.0, Gateway: 192.168.1.1
Default server: 192.168.1.151, DNS server IP: 0.0.0.0

RedBoot(tm) bootstrap and debug environment [ROM]
Red Hat certified release, version 1.94 - built 16:46:14, Oct 12 2004

Platform: IXDPG425 Development Platform (XScale) BE
Copyright (C) 2000, 2001, 2002, 2003, 2004 Red Hat, Inc.

RAM: 0x00000000-0x02000000, 0x0001f158-0x01fd0000 available
FLASH: 0x50000000 - 0x51000000, 128 blocks of 0x00020000 bytes each.
== Executing boot script in 5.000 seconds - enter ^C to abort
^C
RedBoot>
```

FIGURA 7.3: Processo di bootstrap del RedBoot

Il RedBoot mette a disposizione una shell minimale per la configurazione dell'avvio del gateway. Tramite l'istruzione "fconfig -l" è possibile visualizzare la corrente configurazione del boot loader, mentre la sola "fconfig" permette la riconfigurazione dell'avvio. In *Figura 7.4* è possibile vedere lo script avviato dal boot loader, nonché altri parametri di configurazione.

```
root@KaskoPC002:/home/fabio
File  Modifica  Visualizza  Terminale  Vai  Aiuto
RedBoot> fconfig -l
Run script at boot: true
Boot script:
.. load -r -v -b 0x01008000 zImage.ixdpg
.. load -r -v -b 0x01100000 new_ramdisk_fabio.gz
.. exec -c "console=ttyS0,115200 root=/dev/ram rw ip=off initrd=0x01100000,10M"0

Boot script timeout (1000ms resolution): 5
Use BOOTP for network configuration: false
Gateway IP address: 192.168.1.1
Local IP address: 192.168.1.159
Local IP address mask: 255.255.255.0
Default server IP address: 192.168.1.151
Console baud rate: 115200
DNS server IP address: 0.0.0.0
GDB connection port: 9000
Force console for special debug messages: false
Network debug at boot time: false
Default network device: npe_eth0
Network hardware address [MAC] for NPE eth0: 0x00:0x00:0x84:0x00:0x50:0x02
Network hardware address [MAC] for NPE eth1: 0x00:0x00:0x84:0x28:0x50:0x02
RedBoot>
RedBoot>
RedBoot>
```

FIGURA 7.4: Script di configurazione del RedBoot

In particolare, la stringa "load -r -v -b 0x01008000 zImage.ixdpg" indica al boot loader di caricare all'indirizzo di memoria 0x01008000 l'immagine del kernel precedentemente compilato. L'istruzione "load -r -v -b 0x01100000 new_ramdisk.gz", analogamente, indica di caricare all'indirizzo 0x01100000 l'immagine del RAM disk da utilizzare. Infine, la stringa "exec -c "console=ttyS0,115200 root=/dev/ram rw ip=off initrd=0x01100000,10M" 0x01008000" fa saltare l'esecuzione all'istruzione il cui indirizzo è 0x01008000 (il kernel precedentemente

caricato), indicando come init RAM disk l'intervallo di memoria a partire da 0x01100000, di dimensione 10Mb.

Gli altri parametri di avvio indicano il tempo di timeout prima dell'esecuzione dello script, l'indirizzo IP del gateway di rete, l'indirizzo IP locale, la maschera di rete e l'indirizzo IP del server di default (da cui caricare i files richiesti nello script tramite TFTP).

Effettuata la configurazione del RedBoot, e verificata la corretta esecuzione del kernel, il terminale presenterà la tipica schermata di login di Linux: la IXDPG425 è pronta all'utilizzo.

Per constatare il corretto funzionamento del sistema, è stato cross-compilato ed aggiunto al RAM disk un "hello world" di prova, la cui esecuzione è visibile in *Figura 7.5*.

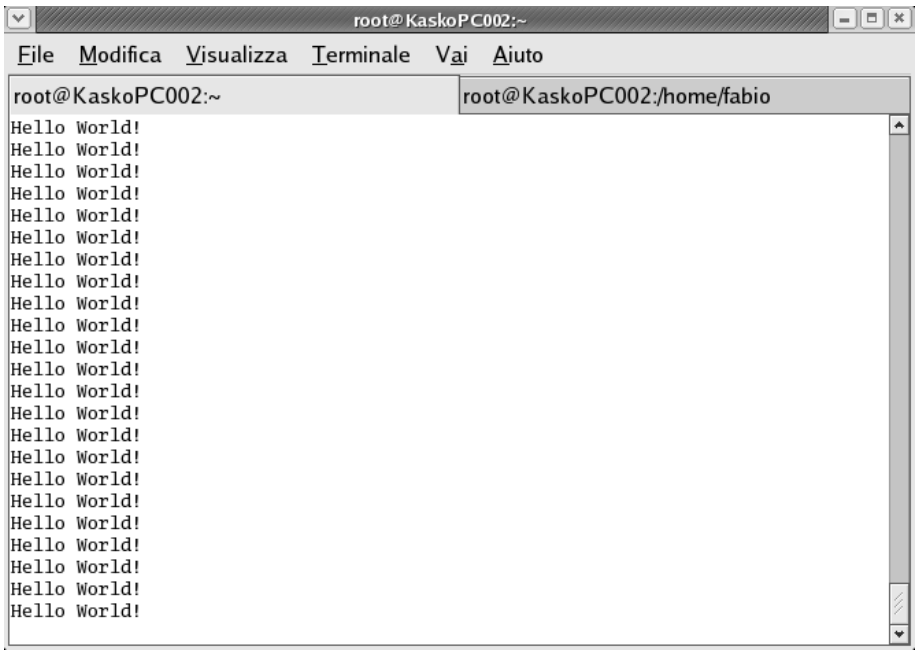


FIGURA 7.5: Esecuzione di "Hello World!" su Intel IXDPG425

7.6 Adattamento krtpd

Il processo di adattamento del driver krtpd alla piattaforma Intel IXP400 Software si è sviluppato in due istanze: la prima ha visto la modifica del thread di trasmissione, la seconda quella del thread in ricezione.

Il meccanismo di trasmissione/ricezione di stream audio utilizzato dalla piattaforma software prevede la registrazione di due funzioni di callback; per l'invio di pacchetti lato rete è stata individuata la funzione ixDspCodeletEthRtpSend in IxDspCodeletEthIf.c mentre, per la ricezione lato rete, viene utilizzata la funzione xPacketReceive, disponibile solo come modulo oggetto. Questa situazione è descritta dalla *Figura 7.6*.

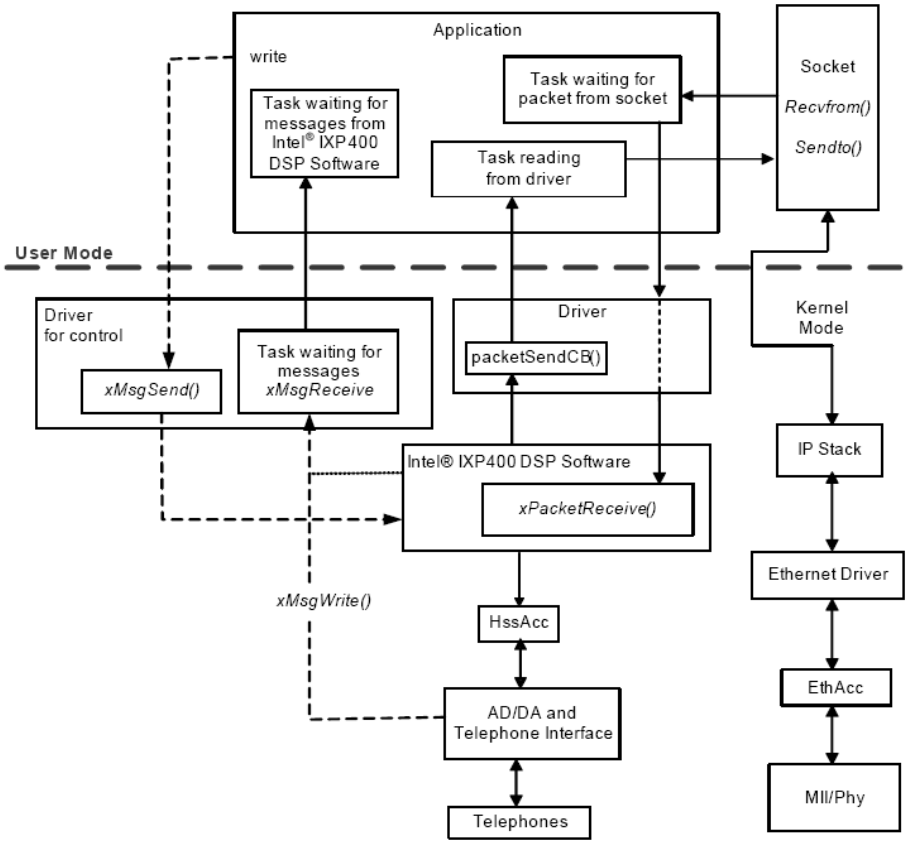


FIGURA 7.6: Flusso dati rete/DSP nella piattaforma software utilizzata

7.6.1 Strutture dati e funzioni di supporto

Prima di illustrare le modifiche apportate al modulo krtpd per l'adattamento al modulo del DSP, è bene notificare la creazione di alcune strutture dati e funzioni di supporto.

È stata introdotta, in *krtpd.h*, una struttura per il mantenimento dei dati relativi ad una sessione RTP; la dichiarazione è la seguente:

```
struct rtpSession
{
        short seqN;
        long timestamp;
        long source;
        short tx_port;
```

```
        unsigned long tx_address;
        short payloadtype;
        short rx_port;
        int rxStatus;
        struct socket *sock_send;
        struct sockaddr_in *sin_send;
        struct socket *sock_rec;
        struct sockaddr_in *sin_rec;
};
```

seqN rappresenta il sequence number; source è la sorgente di sincronizzazione; *tx_port* e *rx_port* sono le porte UDP di trasmissione e ricezione; *payloadtype* è il tipo di payload (dipendente dal codec utilizzato nella comunicazione); *rxStatus* rappresenta lo stato della ricezione, il cui valore può essere *KRTPD_IDLE* o *KRTPD_REC*, rispettivamente se il thread di ricezione è in stato di riposo oppure è attivo; *sock_send* e *sin_send* sono socket e indirizzo di trasmissione, mentre *sock_rec* e *sin_rec* sono socket ed indirizzo di ricezione.

In *krtpd.c*, quindi, è stata allocata un vettore di *rtpSession* tramite la dichiarazione *struct rtpSession rtps[MAX_NR]*; (MAX_NR è chiaramente il numero massimo di sessioni istanziabili).

Per avere informazioni adeguate nella gestione del payload, è stato necessario definire il tipo di payload e la dimensione, relativamente ai codec di possibile utilizzo, vale a dire G.711 u-law e A-law, G.722, G.723.1, G.726, G.729.

Infine, è stata definita la struttura *XPacket_t*, utilizzata dal modulo del DSP, nel seguente modo:

```
typedef struct{
   unsigned char channelID;
   unsigned char payloadType;
   unsigned int mediaType:4;
   unsigned int payloadLen:12;
   unsigned int timeStamp;
```

```
    unsigned char payload[XMAX_PAYLOAD_SIZE];
 #ifdef __GNUC__
} __attribute__ ((packed)) XPacket_t;
#endif
#ifdef __ghs__
} XPacket_t;
#pragma pack()
#endif
```

La sessione RTP è inizializzata da user-space tramite chiamate alla *krtpd_ioctl*; la ioctl, a seconda della funzionalità richiesta, configurerà le porte di ricezione/trasmissione, il tipo di payload e l'indirizzo di trasmissione, oppure invocherà la funzione *krtpd_initRtpSession*, responsabile dell'allocazione e inizializzazione dei socket di trasmissione e ricezione, nonché dell'inizializzazione del sequence number a 0, e della creazione casuale della sorgente di sincronizzazione.

La gestione del tipo di payload necessita di due funzioni; la prima, con prototipo *static int segSize(int pType)*, restituisce la dimensione del payload del codec stabilito da *pType* (payload type). La seconda funzione, dal prototipo *static inline int pktSize(int pType)*, restituisce la dimensione di un pacchetto RTP, cioè la somma tra la dimensione dell'header RTP e la dimensione del payload; il parametro passato è, analogamente alla *segSize*, il tipo di payload.

7.6.2 Trasmissione

Per quanto riguarda la trasmissione dei pacchetti lato rete, è evidente che non c'è più bisogno di un thread che generi e invii lo stream dati, ma di una funzione di trasmissione, resa visibile al software del DSP, e

richiamata ogni qualvolta il DSP genera un pacchetto voce. Di conseguenza, sono state eliminate le funzioni *krtpd_threadsend* e *krtpd_stopthreadsend*, ed è stata modificata la *krtpd_sendfile*. In particolare, il prototipo di questa funzione è *static inline int krtpd_sendfile(unsigned short minor, XPacket_t* packet)*, dove *minor* individua un device (che faremo corrispondere ad una linea telefonica), e *packet* è un puntatore ad un pacchetto voce definito dal DSP, la cui struttuta è raffigurata in *Figura 7.7*.

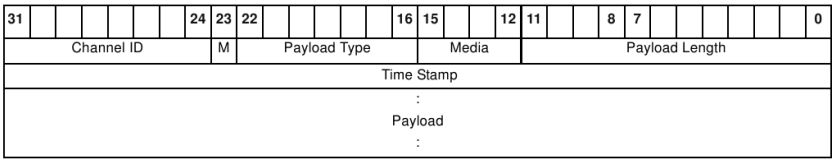


FIGURA 7.7: Struttura del pacchetto XPacket_t

In effetti, la struttura XPacket_t è uno pseudo pacchetto RTP in arrivo dal DSP.

Il primo compito della krtpd_sendfile è quello di generare un pacchetto RTP, unendo le informazioni provenienti dall'XPacket_t (tipo di payload, dimensione del payload, payload effettivo e timestamp) con alcuni parametri mantenuti all'interno della struttura *rtpSession rtps* (sorgente di sincronizzazione e sequence number), relativi al minor. Successivamente, il pacchetto appena preparato viene spedito in rete tramite la chiamata di sistema *sock_sendmsg*. Infine, viene liberata la memoria allocata per il pacchetto, e viene incrementato il sequence number della sessione RTP.

Per rendere visibile la funzione krtpd_sendfile al modulo del DSP è stato necessario esportare la funzione stessa come simbolo nel kernel, aggiungendo al modulo krtpd le istruzioni al preprocessore C

```
#ifndef EXPORT_SYMTAB
#define EXPORT_SYMTAB
#endif
```

e la macro *EXPORT_SYMBOL(krtpd_sendfile).*

Nella codelet del DSP è stato necessario modificare la funzione di callback, il cui prototipo è *IX_STATUS ixDspCodeletEthRtpSend(UINT16 channel, XPacket_t *buffer)*; *channel* è un intero che rappresenta la linea telefonica, mentre *buffer* è un puntatore al pacchetto voce.

La *ixDspCodeletEthRtpSend*, originariamente, effettuava dei controlli sulla natura del payload in arrivo (tono, voce, T.38 o evento), costruiva un pacchetto RTP sugli strati UDP, IP ed Ethernet, aggiornava alcune variabili di sessione e chiamava la funzione *ixEthAccPortTxFrameSubmit*, con lo scopo di inviare il pacchetto verso la porta di destinazione. Inoltre, la *ixDspCodeletEthRtpSend* eliminava il pacchetto appena trasmesso dalla coda, per evitare l'overflow.

Per poter utilizzare la krtpd_sendfile appena esportata, è stato necessario eliminare la chiamata alla *ixEthAccPortTxFrameSubmit*, sostituendola, appunto, con la chiamata alla funzione del krtpd. Non essendo più utilizzato il meccanismo di accodamento dei pacchetti, è anche stata eliminata l'inizializzazione delle code, commentando la chiamata alla *ixQMgrInit* in *IxDspCodeletInit.c*.

7.6.3 Ricezione

Per quanto riguarda la ricezione dei pacchetti dalla rete, è stato necessario modificare la funzione *krtpd_receivefrom*; questa è lanciata come thread e, se la ricezione è abilitata, viene allocata memoria per il pacchetto in arrivo (la dimensione dipende dal codec utilizzato nella comunicazione) e per un XPacket_t. Appena pronto, viene ricevuto dalla rete un pacchetto, tramite la chiamata di sistema *sock_recvmsg*. Successivamente, vengono copiati nell'XPacket_t i campi relativi al canale (linea telefonica), tipo di payload, dimensione del payload, timestamp e payload effettivo, ottenuti dal pacchetto appena arrivato. L'XPacket_t viene così passato come parametro alla funzione *xPacketReceive*, la quale lo trasferirà nel Jitter Buffer dell'IXDPG425. Infine, la funzione dealloca lo spazio relativo al pacchetto ed all'XPacket_t. Il tutto si ripete in un ciclo, fintantoché la ricezione è abilitata.

Per disabilitare la ricezione, bisogna invocare la funzione *krtpd_stopthreadreceive*, la quale scrive *KRTPD_IDLE* nella variabile *rxStatus* della sessione RTP e si assicura che il socket non sia più in ricezione, per poi rilasciarlo.

7.6.4 Ulteriori modifiche

Un'ulteriore considerazione va fatta sull'accesso alla rete da parte del modulo DSP. La codelet DSP accede alle porte Ethernet in modo diretto, il che provoca un conflitto con il driver di rete, cross-compilato al momento della generazione del kernel. Per ovviare a questo

problema, bisogna eliminare l'inizializzazione dello strato di rete; in particolare, è stato necessario commentare la chiamata alla funzione *ixDspCodeletEthRtpInit* in *IxDspCodeletMain.c*. Così facendo è possibile instradare pacchetti anche al di fuori del gateway (ad esempio, è possibile effettuare una chiamata inter-gateway) attraverso l'interfaccia *ixp0*. L'indirizzo di rete dell'IXDPG425 è stato impostato a 192.168.1.160.

7.7 Test d'utilizzo

A questo punto, è stato reso automatico, all'avvio del gateway, l'inserimento dei moduli *ixp400* (modulo base per il funzionamento del gateway), *ixp425_eth* (driver di rete Ethernet), *ixp400_dsr* (modulo per l'utilizzo delle risorse del DSP), *krtpd* (stack RTP interposto tra rete e DSP) e *ixp400_codelets_dspEng* (codelet che utilizza le risorse del DSP).

Per inizializzare i parametri di comunicazione (numero del device da utilizzare, tipo di payload, indirizzo destinazione, porte di ricezione e di trasmissione) e per lanciare i thread in ricezione, è stato necessario utilizzare un applicativo specifico, *user*, il cui utilizzo è visibile in *Figura 7.9*.

```
root@KaskoPC002:~
File  Modifica  Visualizza  Terminale  Vai  Aiuto
root@KaskoPC002:~        root@KaskoPC002:/home/fabio

MontaVista(R) Linux(R) Professional Edition 3.1

login[157]: root login  on `console'



BusyBox v0.60.3 (2004.01.09-22:53+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

# lsmod
Module                  Size  Used by
ixp400_codelets_dspEng   43388   0 (unused)
krtpd                    6388   0 [ixp400_codelets_dspEng]
ixp400_dsr            1435740   0 [ixp400_codelets_dspEng krtpd]
ixp425_eth              17884   2
ixp400                5103232   0 [ixp400_codelets_dspEng ixp400_d
#
```

FIGURA 7.8: Elenco dei moduli caricati dopo il boot del kernel

```
# ./user 0 127.0.0.1 2004 0 2005
127.0.0.1
device: /dev/krtpd0
Numero IP: 7f000001
Numero tx port: 2004
Numero payloadtype: 0
Numero rx port: 2005

premi un tasto per chiudere:
```

FIGURA 7.9: Esecuzione del programma user

Il primo parametro è il numero di device da utilizzare; il secondo parametro è l'indirizzo di destinazione della trasmissione; il terzo parametro è la porta di trasmissione; il quarto parametro è il tipo di payload; il quinto parametro è la porta di ricezione.

Per testare il corretto funzionamento dell'integrazione tra il modulo krtpd ed il modulo DSP, è stato utilizzato l'applicativo d'esempio fornito

dalla Intel, *IxDspCodeletApp*. Tramite questa demo è possibile effettuare una chiamata telefonica.

L'applicazione, per prima cosa, apre una comunicazione con il driver *ixDspCodeletModule*, attraverso il quale viene lanciata la funzione *demostart*; questa inizialmente chiama la *ixDspCodeletNPEInit*, per inizializzare i Network Processor Engine; successivamente la demostart richiama *ixDspCodeletDspInit* per inizializzare il modulo DSP, poi la *xDspGetResConfig*, per ottenere le informazioni di configurazione del modulo DSP. Infine viene invocata la *ixScInit* per inizializzare il driver Slic, ed effettuare la registrazione delle funzioni di callback. Eseguita la demostart, l'applicazione inizializza i thread necessari alla gestione dei messaggi ed al trasferimento dei pacchetti. L'ultimo passo è la visualizzazione a video del menù utente (*Figura 7.10*).

Dopo aver configurato i parametri con l'applicativo user, utilizzando i devices 0 e 1, e incrociando le porte di trasmissione/ricezione, è stato lanciato l'applicativo demo. La scelta effettuata è stata "7 - Gateway and Fax Bypass Demo". Scelto il codec con cui iniziare la comunicazione, è stata lanciata una chiamata da un canale verso l'altro: la comunicazione dei pacchetti RTP è avvenuta con successo e, almeno qualitativamente, non si sono presentate anomalie durante la conversazione.

```
root@KaskoPC002:~
File  Modifica  Visualizza  Terminale  Vai  Aiuto
root@KaskoPC002:~        root@KaskoPC002:/home/fabio
# ./IxDspCodeletApp
KASKO DEBUG - outMsgQue = 3
Please select telephone interface
        1: Digital T1, 4 channels
        2: Analog Narrow Band, 2 channels
        3: Analog Wide Band, 2 channels
        4: Analog Wide & Narrow Band, 1 + 1 channels
        5: Analog Narrow Band, 4 channels
        -> 2
OSSL: Message queues createdmaxMsg = 64, maxSize = 192 in=4 out=3

----------------------------------------
-      IxDspCodelet Demo Menu          -
----------------------------------------
  0 - Print Menu
  1 - Channel Setup
  2 - Channel Teardown
  3 - Show Channel Parameters
  4 - Show Resource Parameters
  5 - Set a Parameter
  6 - Caller ID Demo
  7 - Gateway and Fax Bypass Demo
  8 - 3-Way Call Demo
  9 - Player Demo
 10 - Diagnostic
 11 - System Test CLI
 12 - Exit
Please select test item -
   OFFLINE                    NOR                    VT102
```

FIGURA 7.10: Menù dell'applicazione demo per gateway

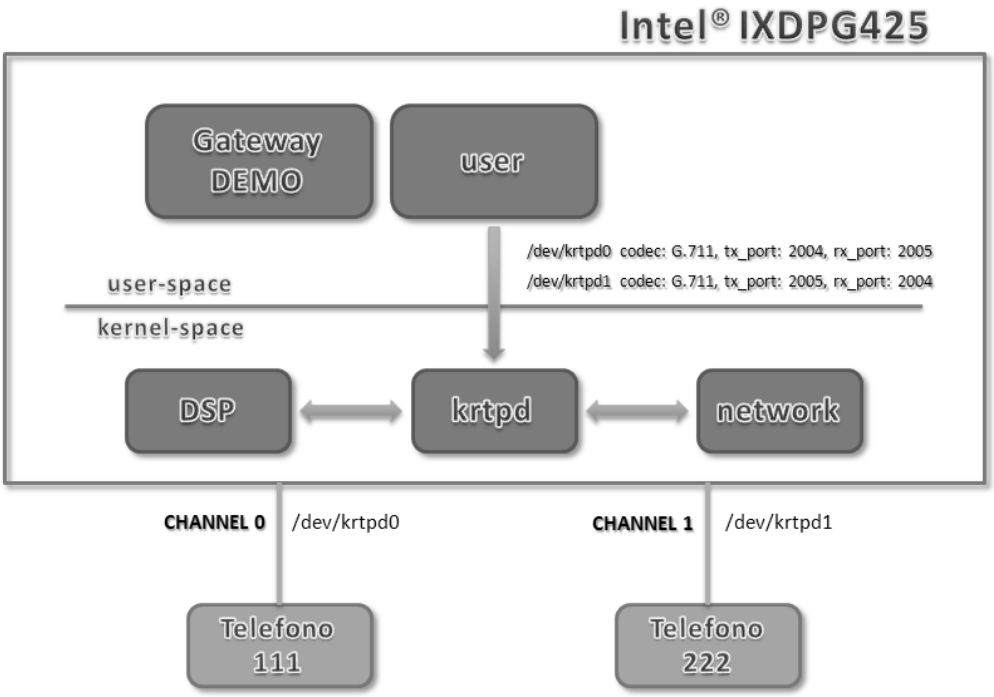


FIGURA 7.11: Configurazione del sistema durante l'esecuzione della demo

È risultato interessante verificare quantitativamente le prestazioni dell'utilizzo del krtpd calato nell'architettura IXDPG425. Allo scopo, utilizzando l'applicativo user, sono stati inizializzati due devices del krtpd *(/dev/krtpd0*, */dev/krtpd1*), impostando gli indirizzi di destinazione della chiamata a 192.168.1.20 (indirizzo IP di un notebook Asus, processore Intel Pentium 4 2.8 GHz, 512 MB di RAM), le porte di destinazione a 2004 (device 0) e 2005 (device 1). Dopo aver lanciato la cattura dei pacchetti dal notebook tramite l'applicazione Wireshark, è stata effettuata una chiamata di circa 100 secondi. I risultati della cattura sono visibili in *Figura 7.12*.

Filter: ip.addr == 192.168.1.160 and not icmp | Expression... | Pulisci | Applica

No.	Time	Source	Destination	Protocol	Info
8	17.168462	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=0, Time=664080, Mark
10	17.168473	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=0, Time=664080, Mark
12	17.173915	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=1, Time=664160
14	17.173958	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=1, Time=664160
16	17.184001	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=2, Time=664240
18	17.184028	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=2, Time=664240
20	17.194093	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=3, Time=664320
21	17.194119	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=3, Time=664320
22	17.204181	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=4, Time=664400
23	17.204211	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=4, Time=664400
24	17.214264	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=5, Time=664480
25	17.214298	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=5, Time=664480
26	17.224361	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=6, Time=664560
27	17.224389	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=6, Time=664560
28	17.234439	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=7, Time=664640
29	17.234465	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=7, Time=664640
30	17.244510	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=8, Time=664720
31	17.244562	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=8, Time=664720
32	17.254612	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=9, Time=664800
33	17.254646	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=9, Time=664800
34	17.264704	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=10, Time=664880
35	17.264730	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=10, Time=664880
36	17.274793	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=11, Time=664960
37	17.274820	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=11, Time=664960
38	17.284897	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=12, Time=665040
39	17.284922	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=12, Time=665040
40	17.294980	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=13, Time=665120
41	17.295026	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=94820933, Seq=13, Time=665120
42	17.305073	192.168.1.160	192.168.1.20	RTP	Payload type=ITU-T G.711 PCMU, SSRC=1338848255, Seq=14, Time=665200

```
▷ Frame 8 (138 bytes on wire, 138 bytes captured)
▷ Ethernet II, Src: Orban_01:2a:fe (00:0e:ec:01:2a:fe), Dst: AsustekC_c7:2b:95 (00:0e:a6:c7:2b:95)
▷ Internet Protocol, Src: 192.168.1.160 (192.168.1.160), Dst: 192.168.1.20 (192.168.1.20)
▷ User Datagram Protocol, Src Port: 1026 (1026), Dst Port: 2004 (2004)
▽ Real-Time Transport Protocol
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0001 = Contributing source identifiers count: 1
    1... .... = Marker: True
    Payload type: ITU-T G.711 PCMU (0)
    Sequence number: 0
    Timestamp: 664080
    Synchronization Source identifier: 1338848255
```

FIGURA 7.12: Risultati della cattura dei flussi RTP da parte di Wireshark

I pacchetti catturati in totale sono stati 20591, 10334 per lo stream destinato alla porta 2004 (identificato da SSRC 1338848255), 10257 per lo stream destinato alla porta 2005 (identificato da SSRC 94820933).

Dall'analisi della *Figura 7.13* si può notare che, durante la trasmissione, non si è verificata la perdita di pacchetti (ciò denota la correttezza dei sequence number). Anche il payload type degli streams (G.711 u-law) è stato riconosciuto a dovere.

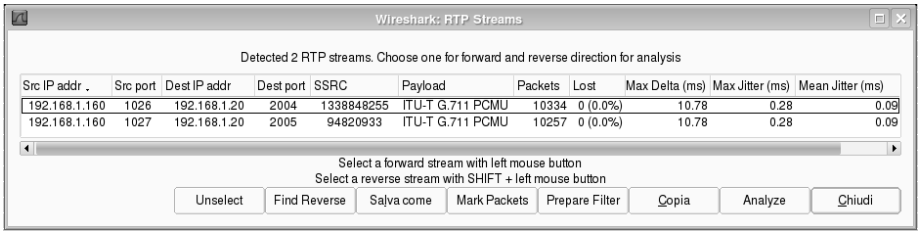
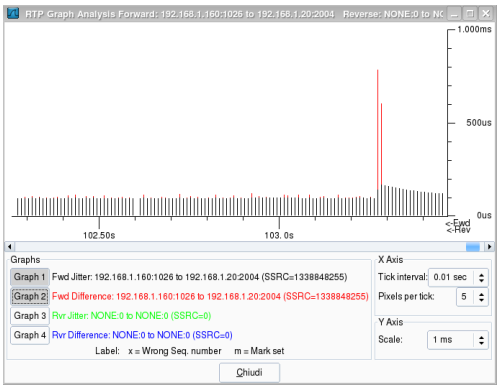


FIGURA 7.13: Sintesi delle informazioni sugli stream RTP catturati

Ma l'analisi sicuramente più interessante riguarda la distanza temporale massima tra un pacchetto e l'altro (Max Delta in ms), che si è attestata per entrambi gli streams a 10.78 ms; questo è un ottimo risultato, considerando che l'IXDPG425, con codec G.711, campiona ogni 10 ms.

Un altro parametro prestazionale da prendere in considerazione è il jitter (o tremolio). Come è stato visto nel paragrafo 2.1.2, il jitter è la differenza tra l'istante in cui un pacchetto è atteso e l'istante in cui effettivamente viene ricevuto; è semplice intuire che, maggiore è il

jitter, peggiore è la qualità della comunicazione. Analizzando il comportamento degli streams catturati, si evince che il jitter massimo è stato di 0.28 ms; questo parametro, però, è fuorviante, poiché può dipendere dalla condizione della rete in un dato momento. Sicuramente è più significativa l'analisi del mean jitter (o jitter medio): la cattura ha rilevato, per entrambi gli streams, un jitter medio di 0.09 ms. In *Figura 7.14* e *Figura 7.15* si può vedere uno stralcio dei flussi trasmessi. Nelle due immagini si evidenzia l'andamento costante del jitter; in particolare, nel primo stream, l'aumento del jitter avvenuto a circa 103.6 secondi è dovuto al carico computazionale che la macchina ha dovuto sopportare per la terminazione della comunicazione (il telefono relativo all'altro stream è stato agganciato).



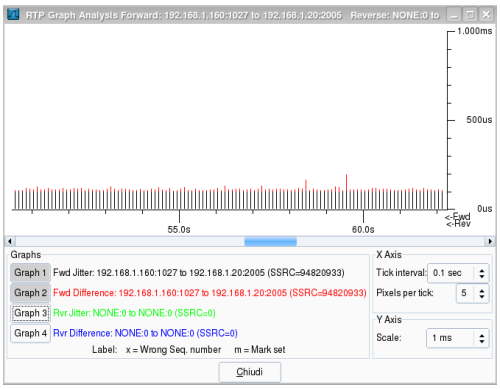


FIGURE 7.14, 7.15: Andamento temporale dei pacchetti scambiati (le due figure sono relative ai due diversi flussi)

7.8 Conclusioni e possibili sviluppi futuri

Il testbed della tesi di cui al paragrafo 7.1 ha dimostrato che, inviando pacchetti RTP da un desktop con processore Intel Pentium 4 a 1.5 GHz con 1GB di memoria RAM, utilizzando ovviamente il driver krtpd, il jitter medio rilevato è stato di 0.1ms; considerata la relativamente bassa potenza di calcolo di un sistema embedded, quale il gateway IXDPG425, i risultati ottenuti sono addirittura migliori rispetto a quelli attesi.

Utilizzando il driver krtpd, inoltre, è stato eliminato l'accesso diretto alle porte Ethernet da parte della codelet DSP, abilitando l'accesso alla rete tramite il driver fornito dalla distribuzione MontaVista Linux 3.1, ixp425_eth.

Le positive conclusioni, a cui si è arrivati dopo lo sviluppo di questa tesi, portano inevitabilmente a pensare all'utilizzo del modulo krtpd all'interno di un applicativo reale, che sfrutti un protocollo di segnalazione quale il SIP; sarebbe interessante calare il driver RTP nel software linphone (telefono per chiamate VoIP OpenSource), oppure nella nota (e più complessa) piattaforma SIP Proxy OpenSource Asterisk.

L'applicazione del krtpd a software reali, dunque, renderebbe sicuramente più efficiente l'utilizzo della banda, migliorando notevolmente la qualità delle comunicazioni, ed aumentando conseguentemente il numero massimo di utenze supportabili.

GLOSSARIO

µ-law: una delle varianti della modulazione PCM a 64 kbit/s. Utilizza una compressione logaritmica per inserire 12 o 13 bit di qualità PCM lineare in 8 bit ed ha lieve vantaggio prestazionale nel rapporto segnale - rumore a basso livello rispetto alla variante a-law.

a-law: una delle varianti della modulazione PCM a 64 kbit/s. Utilizza una compressione logaritmica per inserire 12 o 13 bit di qualità PCM lineare in 8 bit.

ADPCM (Adaptive Differential Pulse Code Modulation): esegue la codifica utilizzando campioni a 4 bit, con una velocità di trasmissione di 32 kbit/s. A differenza di PCM, i 4 bit non codificano direttamente l'ampiezza della voce ma le differenze di ampiezza ed il tasso di cambiamento di tale ampiezza, impiegando una semplice previsione lineare.

ALSA: acronimo di "Advanced Linux Sound Architecture", è un pacchetto utile per il funzionamento del sistema audio sotto Linux.

AppleTalk: era un protocollo molto usato nelle reti Macintosh. Il suo principale scopo era quello di condividere stampati e/o file, e quindi metterli a disposizioni degli utenti che fanno parte della rete.
Ora il protocollo AppleTalk non è più usato, perché si collegano i Mac usando TCP/IP.

ARM: ARM, acronimo di "Advanced RISC Machine" (macchina RISC avanzata), è una famiglia di processori mantenuta e promossa dalla ARM Holdings Ltd.

Assembly: detto anche linguaggio assemblatore (spesso chiamato impropriamente assembler) è, tra i linguaggi di programmazione, quello più vicino al linguaggio macchina vero e proprio.

ATM: è un protocollo di rete a commutazione di cella che incapsula il traffico in celle a lunghezza fissa (53 byte) invece che in pacchetti a lunghezza variabile come nelle reti a commutazione di pacchetto (IP o Ethernet).

AX.25: è un protocollo del livello di collegamento della scala OSI. Derivato dal protocollo X.25 e disegnato per i radioamatori, è utilizzato molto in reti packet radio.

BECN (Frame Relay Backward Explicit Congestion Notification): in una rete Frame Relay, è un bit dell'header che indica alla stazione ricevente di diminuire l'invio di dati.

BSD (**Berkeley Software Distribution**)**:** indica una delle varianti originarie di Unix, sviluppata presso l'università di Berkeley in California e alla base di una delle due famiglie principali di sistemi operativi liberi attualmente più diffusi.

Canali Bearer: in una linea ISDN, i canali Bearer o B sono utilizzati per i dati (che possono essere anche di fonia digitale) ed hanno una banda prefissata di 64 Kbit/s

CAR (Committed Access Rate): algoritmo che permette di impostare azioni di superamento (per esempio scartare pacchetti o contrassegnarli con bassi valori di precedenza IP) o conformità (per trasmettere il pacchetto) del traffico, limitando il traffico in base alla precedenza, all'indirizzo MAC, all'indirizzo IP o ad altri parametri.

CB-WFQ (Class-Based Weighted Fair Queuing): è un tipo di accodamento che presenta tutti i vantaggi del WFQ con la funzionalità aggiuntiva di un supporto granulare per determinate classi di traffico definite dall'amministratore di rete, ad esempio una classe specifica per il traffico vocale.

CELP (Code Excited Linear Prediction): è un algoritmo di codifica vocale.

CID (contesto di sessione): identificatore casuale riferito ad una sessione

CoS (Class of Service): servizio che permette di raggruppare i vari flussi di pacchetti, ognuno dei quali presenta requisiti di latenza e ampiezza di banda differenti.

CQ (Custom Queuing): metodo di accoramento in cui gli utenti possono decidere le priorità delle code.

cRTP: protocollo che consente di comprimere l'intestazione IP/RTP/UDP in 2 o 4 byte.

DAQ: sta per "data acquisition", è uno dei processi per cui possono essere utilizzati sistemi embedded.

DDR (Deficit Round Robin): protocollo secondo il quale i pacchetti IP vengono inviati a code aventi classi di servizio differenti a seconda dei bit di precedenza.

Debian: è una delle più longeve distribuzioni Linux; è nota per la sua estrema stabilità (per questo motivo è spesso utilizzata per realizzare

server), per la sua moltitudine di pacchetti (circa 18000) e per le sue ostiche procedure di installazione/configurazione. È mantenuta dalla comunità Open Source, ed attualmente l'ultima versione è la 4, meglio conosciuta con il nome "Etch".

DHCP (Dynamic Host Configuration Protocol): è il protocollo usato per assegnare gli indirizzi IP ai calcolatori di una rete.

DNS (Domain Name System): è un servizio utilizzato per la risoluzione di nomi di host in indirizzi IP. In pratica un nome host o un indirizzo internet vengono tradotti in un indirizzo numerico di tipo indirizzi IP. Il servizio permette così di utilizzare i nomi e le parole di uso comune per ricercare ad esempio un sito internet.

DOS (Disk Operating System): un sistema operative sviluppato negli anni 80.

DS-0 (Digital Signal 0): è un segnale digitale che viaggia a 64 kbit/s, corrispondente alla velocità di una canale per la frequenza della voce.

DSL (Digital Subscriber Line): tecnologia che fornisce trasmissione digitale di dati nell'ultimo miglio attraverso i fili del telefono. Tipicamente le velocità (in ricezione e trasmissione) sono per lo più comprese fra 256 kb/s e 24.000 kb/s, a seconda della tecnologia DSL, delle condizioni della linea e del livello di servizio.

DSP (Digital Signal Processor): è un particolare tipo di microprocessore ottimizzato per eseguire, in maniera estremamente efficiente, sequenze di istruzioni molto ricorrenti nel condizionamento di segnali digitalizzati (come ad esempio somme, moltiplicazioni e traslazioni). DSP è anche acronimo di "digital signal processing", esso è un insieme di tecniche, tecnologie, algoritmi implementati da processori che permettono di trattare un segnale continuo dopo che è stato campionato.

E1: circuito di trasmissione che può raggiungere i 2.048 Mbit/s.

ELF (Executable and Linkable Format): è il formato oggi più usato per la memorizzazione di file oggetto, programmi eseguibili e librerie dinamiche, sia su GNU/Linux sia sugli altri principali sistemi operativi.

ETHERNET: è il nome di un protocollo per reti locali, attualmente è il sistema LAN più diffuso.

EXT2: è un filesystem utilizzato su sistemi GNU/Linux.

EXT3: è un file system utilizzato su sistemi GNU/Linux.

FEC (Forward Error Correction): è un parametro che indica quanti dei bit trasmessi vengono utilizzati per correggere eventuali errori in una trasmissione digitale di dati.

FECN (Frame Relay Forward Explicit Congestion Notification): in una rete Frame Relay, è un bit dell'header che indica alla stazione ricevente di diminuire le richieste di dati.

FRF.12: è una specifica che descrive il metodo di frammentazione dei frame Frame Relay in frammenti più piccolo.

FRTS (Frame Relay Traffic Shaping): strumento di controllo che crea un buffer del traffico in eccesso ed attende il successivo intervallo libero per trasmettere i dati.

G.114: è uno standard raccomandato dall'ITU che si occupa dei ritardi accettabili per applicazioni voce.

G.711, G.726, G.728, G.729, G.731.1: codec per la codifica della voce trasportata tramite pacchetti.

GNU (GNU is Not Unix): è un progetto che si basa su una gestione particolare dei diritti d'autore sul software, secondo la definizione di software libero.

GPL: acronimo di "General Public License", è la licenza per eccellenza dei software Open Source; attualmente la GPL è arrivata alla terza versione.

GTS (Generic Traffic Shaping): strumento di controllo che riduce la velocità del traffico sulla base della politica impostata.

H.225, H.245: membro della famiglia di protocolli H.323.

H.323: raccomandazione della ITU che definisce i protocolli per sessioni di comunicazione audio-video su qualsiasi rete a pacchetti.

HID (Human Interface Device): è un tipo di computer device che interagisce e prende input direttamente dall'uomo.

HLT: codice di un'operazione del linguaggio Assembly.

HTTP: acronimo di "Hyper-Text Transfer Protocol", è il protocollo utilizzato dai client e dai server web.

IETF (Internet Engineering Task Force): è una comunità aperta di tecnici, specialisti e ricercatori interessati all'evoluzione tecnica e tecnologica di Internet.

IP (Internet Protocol): è un protocollo di rete a pacchetto, secondo la classificazione ISO/OSI è di livello rete. La versione correntemente usata del protocollo IP è detta anche IPv4.

IPX (Internetwork Packet Exchange): è un protocollo di rete al livello network della scala OSI. L'uso di IPX è in generale declino in quanto il boom di Internet ha reso il TCP/IP praticamente universale.

ISA: acronimo di *"Industry Standard Architecture"*, è un tipo di bus utilizzato nelle piattaforme i386.

ISDN (Integrated Services Digital Network): è un servizio di telefonia digitale. Più specificamente, l'ISDN è un protocollo che descrive l'effettuazione delle chiamate e la relativa terminazione.

ITU-T (International Telecomunication Unit-Telecomunication Standardization Bureau): è il settore della Unione Internazionale delle Telecomunicazioni che si occupa di regolare le telecomunicazioni telefoniche e telegrafiche.

LEC (Local Exchange Carrier): carrier autorizzati a cui è stata commissionata l'erogazione di servizi locali di telecomunicazioni vocali all'interno di un'area predeterminata.

LMI (Local Management Interface): sistema di comunicazione tra router e switch Frame Relay.

LPC (Linear Predicting Coding): è una tecnica di analisi di segnali vocali, ideata per consentire una digitalizzazione di buona qualità con bassi valori di bitrate.

Mach: è un kernel sviluppato dalla Carnegie Mellon University durante l'attività di ricerca sui sistemi operativi paralleli e distribuiti. È uno dei primi microkernel ed attualmente è anche il più famoso, infatti spesso viene utilizzato come campione di paragone con altri microkernel.

Mandriva: è una distribuzione Linux francese, una volta chiamata Mandrake. È molto semplice da utilizzare, anche se notoriamente buggata; è adatta per sistemi desktop, e per chi si avvicina per la prima volta al mondo Linux.

MCML PPP (Multi-Class Multilink Point-to-Point Protocol): meccanismo che consente di svolgere la frammentazione al livello di collegamento OSI senza coinvolgere i livelli superiori.

MIPS (Million Instruction Per Second): è un processore di architettura RISC sviluppato. Le prime architetture MIPS erano realizzate a 32-bit (generalmente 32-bit per registri e data paths), mentre le ultime versioni sono a 64-bit.

MIT (Massachusetts Institute of Technology): è una università statunitense, specializzata nel campo scientifico e tecnologico.

MMU: acronimo di "Memory Management Unit", è l'unità di gestione della memoria presente in alcune architetture di processori.

MOS (Mean Opinion Score): è un metodo per la valutazione soggettiva della qualità della voce. Viene eseguito su un gruppo di persone alla volta che possono valutare la qualità del materiale con un valore compreso tra 1 (cattivo) e 5 (eccellente).

MP – MLQ (MultiPulse, Multilevel Quantization): tecnologia per la codifica della voce.

MTU (Maximum Transmission Unit): indica le dimensioni massime in byte di un pacchetto dati che può essere inviato attraverso un protocollo di comunicazione.

NFS (Network File System): è un file system che consente ai computer di utilizzare la rete per accedere ai dischi remoti come fossero dischi locali. La versione 2 del protocollo originale utilizzava unicamente UDP e prevedeva che i server non dovessero conservare memoria degli accessi degli utenti. Eventuali meccanismi di blocco delle risorse (Lock) dovevano essere implementati esternamente al

protocollo. La versione 3 introdusse il supporto per il TCP per il trasporto delle informazioni. Molte società avevano comunque già esteso l'NFS v2 per supportare il TCP in trasmissione. L'utilizzo del TCP consente di utilizzare l'NFS attraverso una WAN sebbene non sia molto semplice né efficiente.

La versione 4 venne influenzata dall'AFS e includeva miglioramenti nelle prestazioni, aggiungeva un supporto migliorato alla sicurezza e introduceva un protocollo che teneva conto dello stato dei client.

NPE: acronimo di "Network Processor Engine", è il processore di rete presente sulla scheda Intel© IXDPG425.

NTFS (New Technology File System): file system dei sistemi operativi basati su kernel NT. La performance di NTFS è nettamente superiore a quella di FAT e di FAT32. Vi sono stati anche aggiunti i cosiddetti *punti di reparse*, ovvero dei meccanismi che consentono le giunzioni (*hardlink*) tra directory, altrimenti impossibili per la struttura del filesystem.

OSI (Open Systems Interconnection): è un modello costituito da una pila (o stack) di protocolli attraverso i quali viene ridotta la complessità implementativa di un sistema di comunicazione per il networking. In particolare la scala OSI è costituito da strati (o livelli), i cosiddetti layer, che racchiudono uno o più aspetti fra loro correlati della comunicazione fra due nodi di una rete. I layers sono in totale 7 e

vanno dal livello fisico (quello del mezzo fisico, ossia del cavo o delle onde radio) fino al livello delle applicazioni, attraverso cui si realizza la comunicazione di alto livello.

OSS: acronimo di "Open Sound System", è un pacchetto software utile per il funzionamento del sistema audio sotto Linux.

P.861: è una raccomandazione dell'ITU-T che descrive i modi in cui si può determinare oggettivamente la qualità della voce (vedi MOS) utilizzando il metodo di misura PSQM.

PBX (Private Branch eXchange): è una centrale telefonica per uso privato. È principalmente usato nelle aziende per fornire una rete telefonica interna.

PCI (Peripheral Component Interconnect): è l'interfaccia sviluppata da Intel intorno agli anni Novanta per collegare al computer le più svariate periferiche. La larghezza di banda dell'interfaccia PCI è rimasta negli anni ancorata a 133 MBytes/s, generata da una trasmissione dati con frequenza pari a 33MHz a 32bit.

PCM (Pulse-Code Modulation): è una rappresentazione digitale di un segnale analogico dove la grandezza del segnale è provata regolarmente a intervalli definiti e quantizzata in una serie di simboli in codice digitale (di solito binario).

PCMCIA: acronimo di "Personal Computer Memory Card International Association", è un tipo di bus.

POS (Packet Over SONET): protocollo per determinare la priorità del traffico sull'interfaccia SONET.

PowerMac: è il nome di una linea di personal computer prodotta dall'Apple Macintosh basata sui processori PowerPC.

PPP (Point-to-Point Protocol): è un protocollo di rete di Livello datalink, comunemente usato nello stabilire connessioni dirette tra due nodi. Il PPP compone spesso il livello datalink (il livello di collegamento dati) del modello OSI nelle connessioni su circuiti punto-punto sincronizzati e non sincronizzati, soprattutto in ambito WAN.

PSQM (Percentual Speech Quality Measurement): è un algoritmo definito nella raccomandazione P.861 di ITU-T per valutare oggettivamente la qualità della voce (in un range di frequenza 300 - 3400Hz).

PSTN (Public Switched Telephone Network): è la normale rete telefonica analogica per le trasmissioni vocali. Può essere utilizzata per l'invio di dati tramite modem.

PVC: tipo di circuito ATM.

Q.931: è un protocollo sviluppato per le comunicazioni su rete ISDN. Esso permette la predisposizione della chiamata tra i terminali ISDN interessati alla comunicazione sia in fonia, sia in trasmissione/ricezione dati. Questo protocollo appartiene al layer 3 del modello di controllo ISO/OSI.

Il suo compito primario è quello di fornire all'interfaccia del collegamento tra il client (utente della rete) ed il server di rete ISDN le procedure per attivare, mantenere e disattivare le connessioni di rete a commutazione di circuito o a commutazione di pacchetto della comunicazioni fonia-dati in progresso.

QoS (Quality of Service): il termine Qualità di servizio è usato per indicare i parametri usati per caratterizzare la qualità del servizio offerto dalla rete (ad esempio perdita di pacchetti, ritardo), o gli strumenti per ottenere una qualità di servizio desiderata.

RED (Random Early Detection): meccanismo che evita le congestioni nelle reti ad alta velocità.

Red Hat: è una nota distribuzione Linux; inizialmente era possibile trovarla in rete, ora è stato creato un nuovo progetto basato su Red Hat, Fedora; Red Hat, invece, è disponibile sotto pagamento. È una

distribuzione semplice da installare, ed è particolarmente adatta per sistemi desktop.

ReiserFS: è un file system correntemente supportato solo dal kernel Linux, ma potrà essere incluso in altri sistemi operativi nel futuro. Con la versione 2.4.1 di Linux, esso è stato il primo journaling file system ad essere incluso nel kernel vanilla.

RISC (Reduced Instruction Set Computer): indica un'architettura per microprocessori formata da un set di istruzioni contenente istruzioni in grado di eseguire operazioni semplici che possono essere eseguite in tempi simili.

RSVP (Resource Reservation Protocol): protocollo punto-a-punto che consente ai punti terminali della rete di segnalare il tipo di qualità di servizio richiesto da una determinata applicazione.

RTCP (Real Time Control Protocol): protocollo che affianca l'RTP. Serve per controllare, attraverso messaggi di eco e simili, che le prestazioni della rete siano quelle attese.

RTP (Real Time Protocol): è lo standard per la trasmissione del traffico sensibile ai ritardi lungo reti a pacchetti. E' adatto soprattutto per il trasporto di dati audio/video.

RUDP (Reliable User Datagram Protocol): è un protocollo che aumenta l'affidabilità dell'UDP. Questo invia più copie dello stesso pacchetto e consente alla stazione ricevente di eliminare i pacchetti non necessari o ridondanti.

SCSI: acronimo di "Small Computer Systems Interface", è un'interfaccia generica che può essere utilizzata per connettere diversi tipi di periferiche.

Slackware: è una distribuzione GNU/Linux tra le più longeve. Fu creata partendo dalla distribuzione SLS come base, e venne pubblicata per la prima volta il 16 luglio 1993.

SLIC: acronimo di "Subscriber Line Interface Circuit".

SLIP (Serial Line Internet Protocol): è un protocollo di rete di Livello datalink, comunemente usato nello stabilire connessioni dirette tra due nodi. È stato rimpiazzato dal più nuovo protocollo PPP.

SLS (Softlanding Linux System): era una distribuzione GNU/Linux, creata nella metà del 1992. Fu la prima distribuzione ad offrire una collezione di software che contenesse qualcosa in più che il kernel Linux e le utilità di base.

SMB (Server Message Block): è un protocollo usato principalmente per condividere files, stampanti, porte seriali e comunicazioni di varia natura tra diversi nodi di una rete. Esso include anche un meccanismo di comunicazione tra processi autenticata. È soprattutto usato dai sistemi Microsoft Windows.

SMP: è un multiprocessore simmetrico la cui architettura hardware è dotata di più processori che possono accedere equamente a tutta la memoria RAM.

SONET (Synchronous Optical NETwork): infrastruttura di rete basata su segnali ottici multipli. Ha una velocità base di 51,48 Mbit/s.

SPARC (Scalable Processor ARChitecture): è il nome di un'architettura per microprocessore big-endian RISC. SPARC International ha voluto che SPARC fosse un'architettura "aperta" per creare un grande ecosistema per la progettazione, è stato licenziato a vari produttori tra cui Texas Instruments, Cypress Semiconductor e Fujitsu. È risultato che attualmente SPARC è un'archittetura aperta e non proprietaria: ne esiste anche un'implementazione open source completa.

SS7 (Signaling System #7): si tratta di un set di protocolli per la telefonia che vengono utilizzati per gestire la maggior parte delle chiamate telefoniche che avvengono tramite PSTN.

T1: è una tecnologia di comunicazione digitale (anche nota come *Digital Signal 1*: **DS1**) che nasce per scopi telefonici negli anni '60 dove consentiva di instradare fino a 24 comunicazioni telefoniche, digitalizzate ad alta qualità, contemporaneamente. A partire dagli anni '80 la tecnologia è stata utilizzata anche per i collegamenti dati (nello specifico internet) diventando e restando per molto tempo uno dei più veloci canali di connessione sfruttando tecnologie telefoniche, grazie al transfert rate di 1,536 Mb/s.

TCP (Transmission Control Protocol): è uno dei principali elementi della suite di protocolli Internet. I suoi punti di forza sono l'alta affidabilità e robustezza.

TDM (Time Division Multiplexing): tecnica di condivisione di un canale di comunicazione secondo la quale ogni dispositivo ottiene a turno l'uso esclusivo dello stesso per un breve lasso di tempo (tipicamente 125 μs).

ToS (Type of Service): campo dell'intestazione IP che offre la possibilità di raggiungere una determinata classe di servizio (CoS).

UART: acronimo di "Universal Asynchronous Receiver-Transmitters", è un chip che controlla le porte seriali.

UBR (Unspecified Bit Rate): tipo di rete a Bit Rate non specificato.

UDP (User Datagram Protocol): è uno dei principali elementi della suite di protocolli Internet. UDP è un protocollo di trasporto a pacchetto. È usato di solito in combinazione con il protocollo IP. L'UDP ha come caratteristica di essere un protocollo di rete inaffidabile, privo di connessione, ma in compenso molto rapido ed efficiente per le applicazioni "leggere" o time-sensitive.

UDP-Lite (Lightweight UDP): è un nuovo protocollo standardizzato per reti error-prone. Questo offre alle applicazioni come quelle che eseguono streaming dei dati audio/video, di utilizzare le parti buone di un pacchetto parzialmente corrotto.

USB: acronimo di "Universal Serial Bus", è l'interfaccia più usata per le periferiche, grazie alla facilità di utilizzo ed alle alte velocità di scambio dati.

VBR (Variable Bit Rate): tipo di rete a Bit Rate variabile.

WAN (Wide Area Network): è una rete geografica, ovvero una rete informatica usata per connettere insieme più reti locali (*local area network* o LAN) in modo che un utente di una rete possa comunicare con utenti di un'altra rete. Molte WAN sono costruite per una particolare organizzazione e sono private.

WFQ (Weighted Fair Queuing): algoritmo di accoramento che usa più code per distinguere i flussi di diversi tipi di pacchetti, assegnando un'ampiezza di banda uguale per ciascun flusso.

WRED (Weighted Random Early Drop/Detect): meccanismo che combina le funzionalità dell'algoritmo RED e la precedenza IP. Questo segnala ad una stazione TCP di smettere di trasmettere scartando un po' del traffico inviato da essa.

BIBLIOGRAFIA

Jonathan Davison and James Peters, *Fondamenti di Voice Over IP,* I edizione, **Milano, McGraw Hill, 2000.**

RFC 2508; **S. CASNER, V. JACOBSON, CISCO SYSTEMS**. ***Compressing IP/UDP/RTP Headers for Low-Speed Serial Links***, **1999**.

RESOURCE RESERVATION PROTOCOL (RSVP),
www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/rsvp.htm

Augusto Ciuffoletti, *Tecnologie di rete e Internet*, I edizione, **Presso (MI), Jackson Libri, 2002**

http://research.edm.uhasselt.be/jori/thesis/onlinethesis/images/chapter_5/fig_rtpheader.png

Il trasporto dati Real-Time,
www.cs.unibo.it/~scacciag/home_files/teach/RTP-RTCP.pdf

Alessandro Rubini, *I moduli del kernel*,
http://www.linux.it/~rubini/docs/modules26/modules26.html

Jonathan Corbet, Alessandro Rubini e Greg Kroah-Hartman, *Linux Device Drivers*, III edizione, **Sebastopol – CA, O'Reilly Media Inc., 2005**

Jonathan Corbet, Alessandro Rubini e Greg Kroah-Hartman, ***Linux Device Drivers*****,** II edizione, **Sebastopol – CA, O'Reilly Media Inc., 2001**

Italo Armenti, ***Sviluppo di uno stack RTP in Kernel Space per applicazioni VoIP time critical in ambiente Linux*****, 2007**

Karim Yaghmour, ***Building Embedded Linux Systems*****, O'Reilly, 2003**

Intel© IXDPG425 Gateway Reference Platform – User's guide**, 2004**

Intel® IXP400 Digital Signal Processing (DSP) Software Version 2.6.2 - Programmer's Guide**, 2005**

Intel® IXP400 Digital Signal Processing (DSP) Software Version 2.6.2 - API Reference Manual**, 2005**

Intel® IXP400 Software: Codelets - Application Note, 2004

Intel® IXP400 Digital Signal Processing (DSP) Software: Voice Over Internet Protocol - Application Note**, 2004**

The Source Navigator IDE, http://sourcenav.sourceforge.net/

Products, http://www.nomachine.com/products.php