Politecnico di Bari

Facoltà di Ingegneria Elettronica

Dipartimento di Elettronica ed Elettrotecnica

Corso di *Reti di Telecomunicazioni*

Compressione dell' header sulla tratta radio di reti IP

Aspetti generali

di Sandro Petrizzelli

Indice

Introduzione

Il problema della **compressione dell'header dei pacchetti nelle reti IP** è relativamente recente: in passato sono state infatti sporadicamente introdotte delle proposte in merito, ma nessuna di esse ha mai riscontrato un grande successo, essenzialmente perché nessuno riteneva che, nelle tradizionali **reti via cavo**, fosse necessario porsi un problema di questo tipo.

Attualmente, invece, lo scenario è decisamente cambiato, essenzialmente per due fondamentali novità:

- l'esplosione delle **reti wireless**, le quali, con i futuri *sistemi radiomobili di terza generazione* (leggi **UMTS** in Europa), troveranno la loro massima espressione;
- l'attuale certezza che tali nuove reti forniranno **servizi multimediali real time** (voce, video, audio in generale) basati sulla **tecnologia IP**.

In particolare, il problema è particolarmente sentito per la **trasmissione della voce** sulle *reti wireless basate su tecnologia IP* (**VoIP**, *Voice Over IP*), tra le quali rientrerà anche il sistema UMTS, sia pure non nelle prime versioni ([1]).

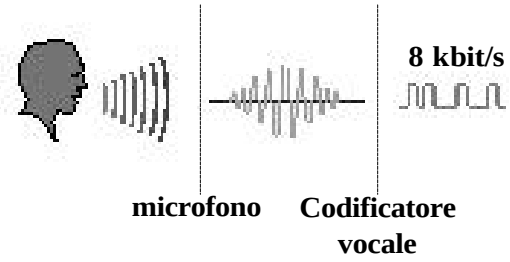
Questo documento illustra gli aspetti essenziali del "problema".

La voce nelle reti radiomobili basate su IP

Il sistema **UMTS**, così come i precedenti sistemi GPRS e GSM, è un sistema di tipo digitale, nel senso che tutti i "segnali d'utente", in qualunque punto della rete, sono costituiti da **flussi di bit**; nel caso delle **conversazioni vocali**, è il *terminale mobile* (o, più volgarmente, il "telefonino") ad operare l'iniziale conversione A→D, campionando e quantizzando opportunamente il segnale elettrico analogico prodotto dal microfono (in risposta alle onde di pressione generate dal parlatore); in tal modo, il terminale mobile "sforna" un flusso di bit che, nel caso dell'UMTS, potrà avere, a seconda di una serie di parametri che non interessano in questa sede, un bit rate

[1] La cosiddetta Release 1999 del sistema UMTS, che dovrebbe diventare operativa tra il 2002 ed il 2003, prevede che i servizi voce utilizzino ancora una **architettura di rete a commutazione di circuito** ereditata dall'attuale sistema GSM; al contrario, per le successive release del sistema, si prevede che anche i servizi voce utilizzano una **architettura di rete a commutazione di pacchetto** (basata proprio su IP), ereditata da quella attualmente usata dal sistema GPRS per la sola trasmissione dati.

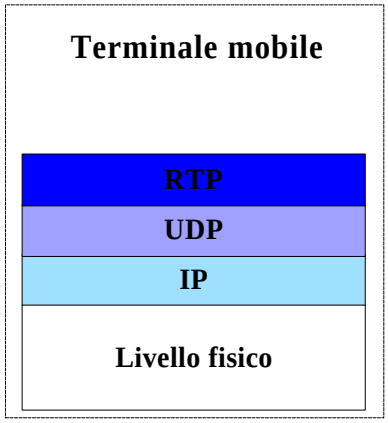
variabile da un massimo di 13 kbit/s ad un minimo di 4 kbit/s (mentre invece il GSM produce 13 kbit/s costanti). Mediamente, si ipotizza un bit rate di **8 kbit/s**.



Conversione analogico-digitale della voce in un terminale mobile

I bit prodotti dalla sorgente (che in questo caso è il *codificatore vocale* del terminale mobile) vengono riuniti in gruppi, denominati in vario modo a seconda del "contesto": *pacchetti*, *frame*, *segmenti*, *datagrammi* e altro. Noi usiamo il termine **pacchetti**, avendo cura però di specificare a quale "livello" o "protocollo" (ad esempio TCP o IP o altro) essi fanno riferimento.

In particolare, così come vedremo dettagliatamente nel seguito, ipotizziamo che il terminale mobile funzioni secondo una *pila di protocolli*, tipica di una **architettura TCP/IP** ([2]), fatta nel modo seguente:



Stack protocollare di un terminale mobile basato su IP

Il livello protocollare più alto (*livello application*) è quello che per primo gestisce i *pacchetti di bit di voce* ([3]). Il protocollo utilizzato è **RTP** (*Real-Time Transport*

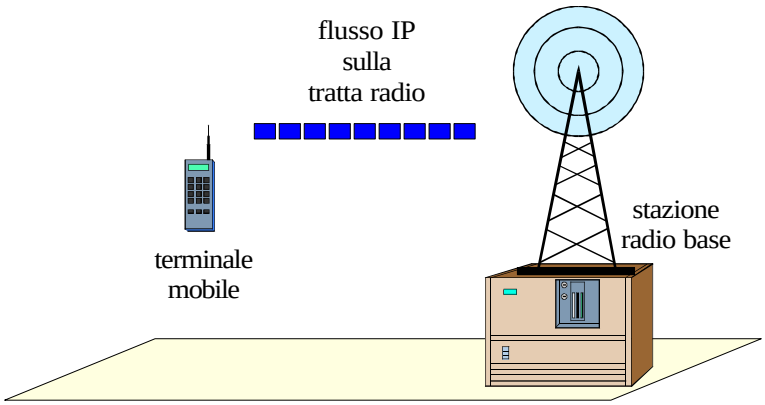
[2] Di questo parleremo più avanti
[3] Al di sopra di esso, infatti, troviamo il "livello" del codificatore vocale che genera i bit di voce.

Protocol), ossia il protocollo di livello *application* ormai divenuto standard delle reti IP per i servizi multimediali di tipo real time (tra i quali appunto la telefonia).

I **pacchetti RTP** (costruiti aggiungendo, ai bit di voce, un **header RTP**) vengono inseriti nei **pacchetti UDP** (*User Datagram Protocol*) e questi ultimi nei **pacchetti IP**, secondo il noto meccanismo dell'**imbustamento**:

Struttura di un pacchetto RTP/UDP/IP

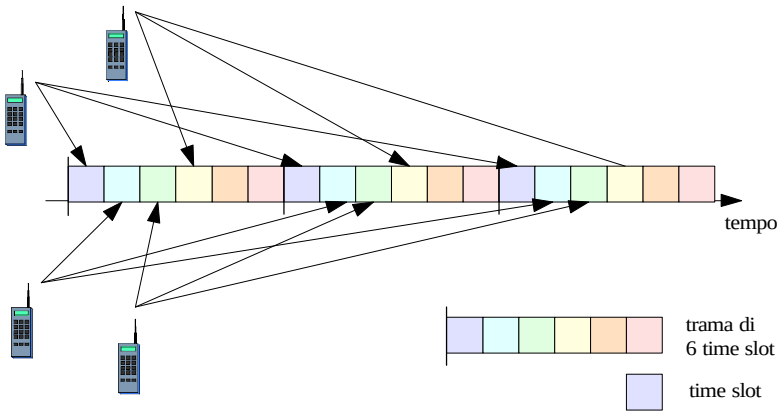
I pacchetti che ci interessano sono proprio i pacchetti IP, per cui affermiamo che, dal nostro punto di vista, la "comunicazione" tra il terminale mobile ed il resto della rete radiomobile avviene tramite la trasmissione di pacchetti IP ([4]):



Collegamento wireless tra un terminale mobile ed una stazione radio base basati entrambi su una pila IP

Un'altra caratteristica del sistema UMTS, in comune con il GSM e il GPRS, è quella di attribuire, a ciascun terminale, la facoltà di trasmettere i propri pacchetti non con continuità, ma solo all'interno di brevi intervalli di tempo (detti **time slot**) che si susseguono periodicamente nel tempo, secondo una tecnica nota come **TDM** (*Time Division Multiplexing*). La figura seguente mostra un caso semplice in cui i time slot vengono raggruppati a gruppi di 6:

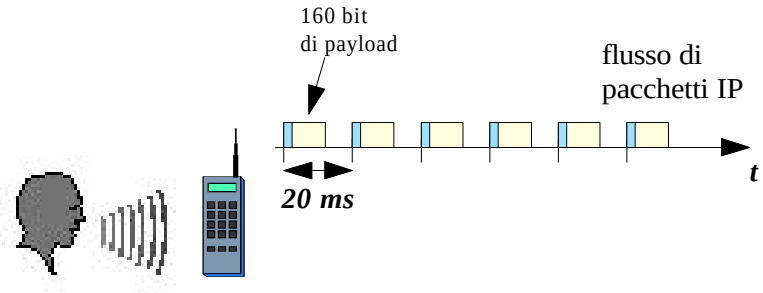
[4] In realtà, come è noto, anche i pacchetti IP subiscono l'imbustamento, in quanto vanno a costituire il payload (carico utile) dei frame di livello fisico, che costituiscono le "unità" realmente trasmesse sul canale di comunicazione. Tuttavia, per la nostra discussione, è sufficiente fermarsi alla trasmissione di pacchetti IP, tralasciando quindi ulteriori suddivisioni dovute a livelli inferiori rispetto al livello network (quello appunto di IP).



Schematizzazione di un sistema TDM basato su trame da 6 time slot

Un gruppo di time slot prende il nome di **trama**. Le specifiche del sistema UMTS in merito alla durata delle trame ed al numero di time slot per trama non sono ancora chiare: ogni trama dovrebbe essere composta da **15** o **30** time slot e dovrebbe durate, rispettivamente, 10 ms o 20 ms. L'ipotesi più accreditata è quella dei **20 ms**, per cui su di essa basiamo i nostri discorsi.

Dire che la singola trama dura 20 ms significa dire che il singolo time slot si ripete ogni 20 ms e quindi che il terminale mobile è costretto ad inviare un **pacchetto voce** ogni 20 ms: dato allora che il bit rate medio è di 8 kbit/s, deduciamo che ogni 20 ms è necessario inviare un pacchetto voce (del singolo utente) di **160 bit**, cioè 20 byte.

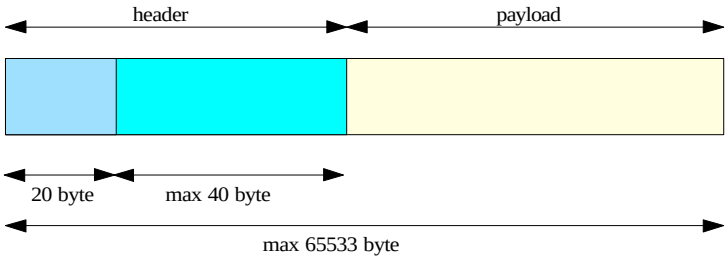


Ritmo di trasmissione di pacchetti IP da parte di un terminale mobile impegnato in una tipica conversazione telefonica

Naturalmente, questi 160 bit costituiscono il cosiddetto **payload** (o *carico utile*) del *pacchetto di livello application*, che poi coincide con il payload del pacchetto IP. Per arrivare al pacchetto IP completo da trasmettere, in base al meccanismo dell'*imbustamento* accennato prima, dovremo aggiungere l' **header** (o *intestazione* o *preambolo*) dei protocolli RTP, UDP ed infine IP stesso. In particolare, l' **header IP** è

quello che contiene tutte quelle informazioni necessarie affinché il pacchetto possa essere correttamente instradato nella rete verso la sua destinazione ([5]), mentre invece gli header UDP e RTP servono ad una opportuna gestione del flusso tra sorgente e destinazione, secondo le caratteristiche tipiche di questo tipo di protocolli.

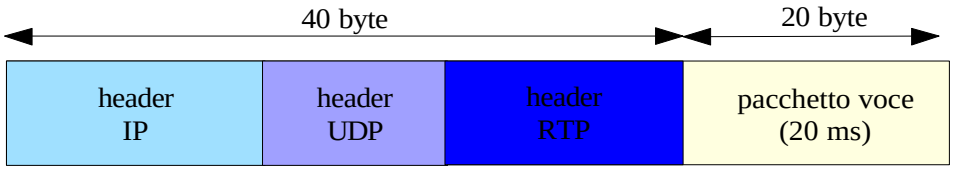
Nel caso del **protocollo IP versione 4** (quello attualmente in uso, anche se a breve si prevede il passaggio alla **versione 6**), ci sono dei precisi vincoli circa la lunghezza dei pacchetti:



Struttura generale di un pacchetto IP: header (parte fissa + parte opzionale) + payload

- la massima lunghezza totale prevista (che in realtà non viene mai adottata) è di 65535 byte ([6]);
- la massima lunghezza dell' header è di 60 byte, dove 20 byte sono fissi mentre i restanti 40 sono opzionali.

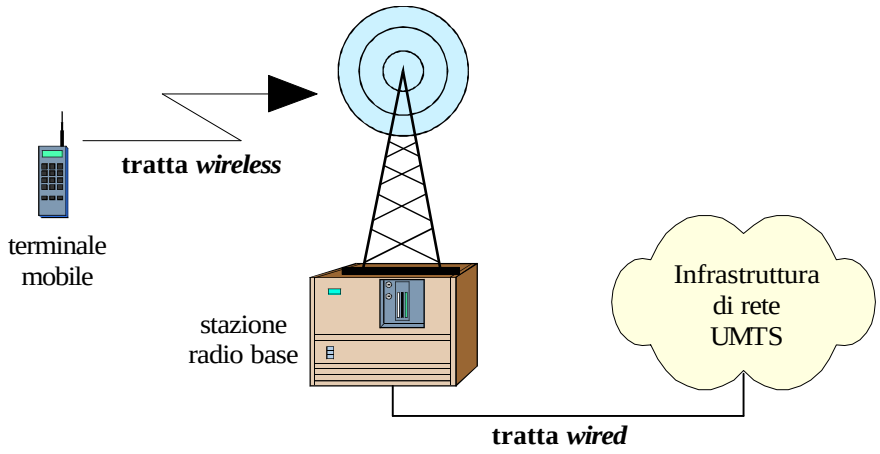
Supponendo allora di trascurare la parte opzionale dell' header IP nonché quella opzionale dell' **header RTP**, il terminale mobile si troverebbe ad inviare, ogni 20 ms, pacchetti IP in cui **40 byte** sono di header (20 byte IP + 8 byte di UDP + 12 byte di RTP) e solo 20 contengono il payload.



Struttura generale di un "pacchetto IP voce"

[5] Si ricordi, infatti, che, nelle reti TCP/IP, uno dei compiti fondamentali del livello rete (il quale usa appunto l'IP) è proprio l'instradamento.
[6] Nelle tradizionali reti IP via cavo, si adotta generalmente una lunghezza 1500 byte;

L'invio di pacchetti in cui l'intestazione rappresenta i 2/3 del totale è un evidente spreco di banda da parte del sistema, tanto maggiore se si pensa che proprio la banda è la risorsa più preziosa per un sistema radiomobile, essendo fortemente limitata e costosa. La limitazione di banda, e la conseguente inefficienza dovuta ai pacchetti prima descritti, è ovviamente solo sulla cosiddetta **tratta radio**, ossia sul "percorso" via radio che i pacchetti devono compiere dal terminale mobile fino alla **stazione radio base** (BTS) su cui esso è appoggiato:



Schematizzazione dei collegamenti wireless e dei collegamenti wired per una tipica rete radiomobile cellulare

Una volta giunti alla stazione radio base, i pacchetti "percorrono" la rete non più via radio, ma su *collegamenti via cavo ad alta velocità* (a 64 kbit/s), per cui effettivamente lo spreco di risorse viene meno ed ha quindi meno senso porsi problemi di efficienza.

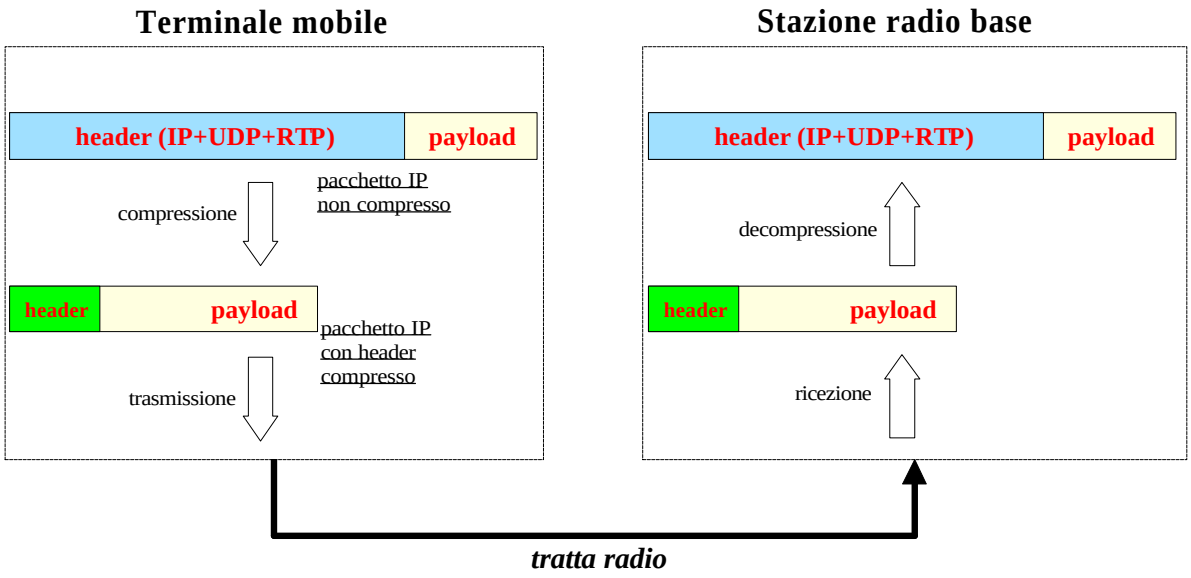
Vale anche, ovviamente, il discorso inverso, relativo cioè ai pacchetti voce che dalla stazione radio base devono giungere al terminale nel corso di una normale conversazione telefonica: anche in questo caso, è opportuno trovare il modo di ottimizzare l'efficienza del sistema.

In definitiva, quindi, tutto si riconduce a *capire se ci sia un modo per aumentare l'efficienza dei protocolli di trasmissione dei pacchetti tra il terminale mobile e la stazione radio base*. La prima strada che viene in mente è proprio quella delle **tecniche di compressione**. Considerando che i "bit di voce", ossia quelli "sfornati" dal codificatore vocale del terminale mobile, hanno già subito tutti i possibili ed opportuni meccanismi di riduzione di ridondanza, non avrebbe alcun senso tentare di effettuare ulteriori compressioni, per cui l'unica possibilità è quella

di tentare la compressione delle "informazioni aggiuntive" che accompagnano tali bit nel loro transito sulla rete. Si tratta appunto delle informazioni contenute negli **header** dei pacchetti scambiati.

Se si trovasse il modo di ridurre gli attuali 40 byte (sempre nell'ipotesi di trascurare la parte opzionale dei vari protocolli, che non sempre per il traffico real time risulta inutile) a pochi byte (ad esempio 4 o anche meno), allora si potrebbero conseguire notevoli miglioramenti di efficienza, in quanto i byte di payload tornerebbero ad essere una parte predominante rispetto a quelli di intestazione.

Si deve dunque attuare un meccanismo del tipo schematizzato nella figura seguente:



Schematizzazione del processo di trasmissione di un pacchetto IP con header compresso

L'eventuale pacchetto voce generato dal terminale mobile (risp. giunto alla stazione radio base) viene prima sottoposto alla compressione dell' header e poi inviato sulla tratta radio, tramite l'opportuna **interfaccia radio** (o *interfaccia di livello fisico* in generale). Una volta giunto alla stazione radio base (risp. al terminale) viene prima decompresso e poi trattato in modo tradizionale (estrazione del payload RTP nel caso del terminale mobile oppure instradamento nel caso della stazione radio base).

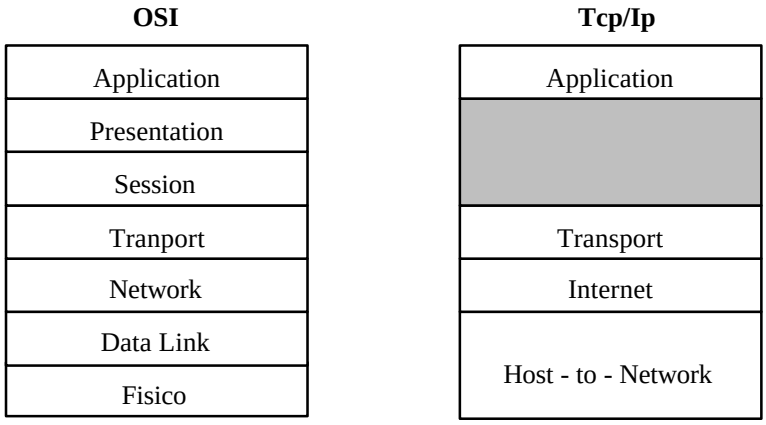
Architettura protocollare

Date le premesse del precedente paragrafo, possiamo ora seguire un approccio al problema di tipo "protocollare", descrivendo prima l'architettura tipica utilizzata dagli attuali sistemi (o di prossimo utilizzo) e poi l'architettura che si è pensato di introdurre con la compressione dell' header.

Il discorso, evidentemente, presuppone che sia il terminale mobile sia la stazione radio base adottino una pila di protocolli della **famiglia TCP/IP**. Una tradizionale pila di protocolli di questa "famiglia" prevede sostanzialmente, al di sopra del *livello fisico* (che si occupa della trasmissione dei bit grezzi sul canale di comunicazione), solo altri 3 livelli:

- un *livello network* impiegante il protocollo IP,
- un *livello transport* impiegante un protocollo da scegliersi, a seconda delle applicazioni, tra UDP e TCP;
- un *livello application* corrispondente appunto alla applicazione in corso di esecuzione (ad esempio un *client di VoIP* appoggiato ad RTP nel nostro caso oppure i più tradizionali *client* appoggiati su HTTP, SMTP, FTP, Telnet e via discorrendo).

La figura seguente illustra questa struttura, comparandola con l'analoga pila protocollare ISO-OSI, altrettanto nota:

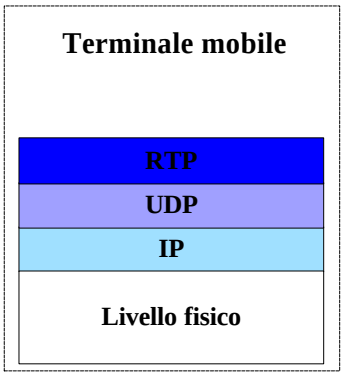


Confronto tra la pila ISO-OSI e la pila TCP/IP

Ciascuno dei livelli genera i propri pacchetti di bit e li consegna al livello inferiore, il quale li inserisce nel payload dei propri pacchetti, secondo il noto meccanismo

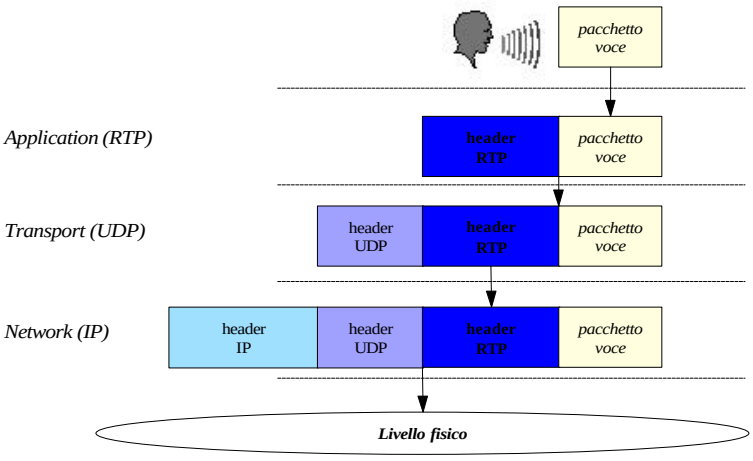
dell'**imbustamento**, compila l' header di propria competenza e cede il tutto al livello inferiore, fino ad arrivare al livello più basso, che si occupa solo della trasmissione dei bit così come gli arrivano dal livello superiore.

In particolare, per i nostri discorsi ipotizziamo che lo stack protocollare del terminale mobile sia il seguente (già mostrato in precedenza):



Stack protocollare di un terminale mobile basato su IP

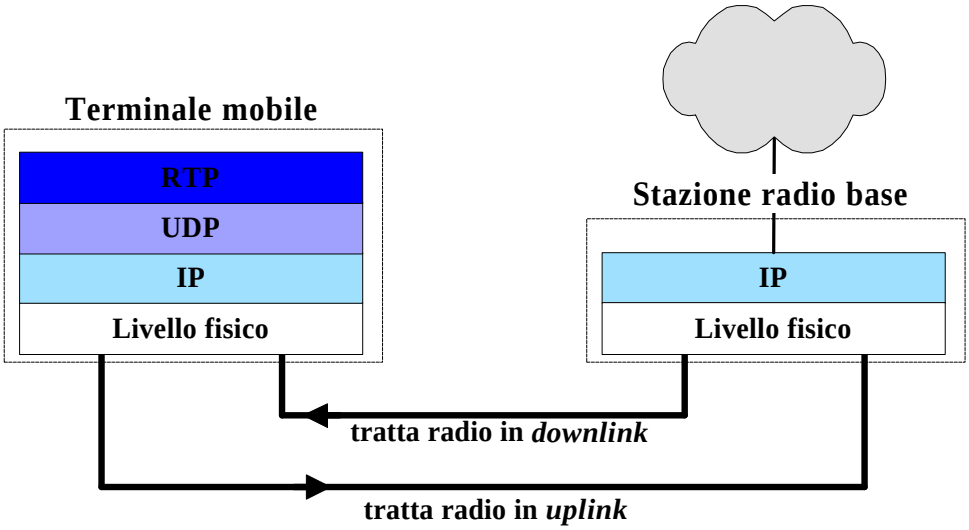
Di conseguenza, il meccanismo di imbustamento che determina i pacchetti IP in trasmissione è quello riportato nella figura seguente:



Schematizzazione del meccanismo di "imbustamento" per una pila RTP/UDP/IP

In ricezione, si ha invece la procedura inversa, per cui i pacchetti, una volta "intercettati" dal livello fisico, vengono passati ai livelli via via superiori, ciascuno dei quali elabora l' header di propria competenza e passa il payload al livello superiore.

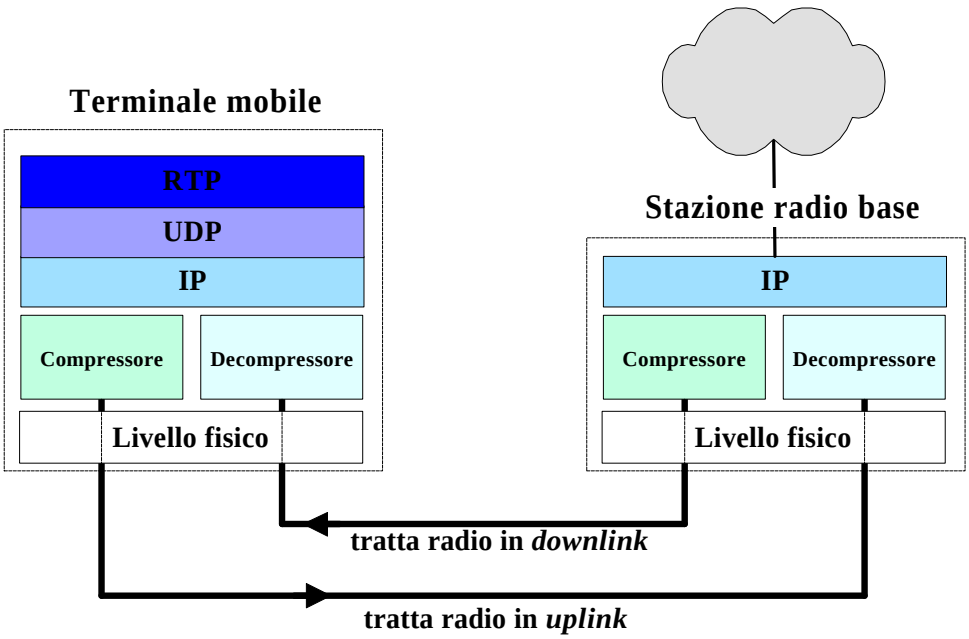
Una "tradizionale" architettura di rete TCP/IP è dunque del tipo riportato nella figura seguente:



Tipica architettura di rete TCP/IP

Si noti la caratteristica per cui la stazione radio base prevede protocolli non oltre il *livello network*: infatti, dal livello di transport in su, tutti i protocolli utilizzati (nel nostro caso UDP e RTP) sono di tipo *end-to-end*, ossia utilizzati solo dalla stazione sorgente e da quella destinazione.

Il nostro scopo è quello di modificare questa architettura, introducendo, tra il livello del protocollo IP ed il livello fisico, degli ***algoritmi di compressione/decompressione*** *degli header*. Si tratta perciò di giungere ad una architettura del tipo seguente:



Architettura di rete TCP/IP modificata con l'introduzione degli organi di compressione e decompressione

E' evidente che sia il terminale mobile sia la stazione radio base devono comprendere sia un organo **compressore** (per i pacchetti da trasmettere) sia un organo **decompressore** (per i pacchetti da ricevere).

Idee alla base della compressione dell' header

Possiamo ora spiegare quali sono le idee sulla base delle quali si è pensato alla possibilità di comprimere l' header dei pacchetti IP.

In generale, una **tecnica di compressione lossless** ([7])consente, a fronte di un certo numero di bit contenente una certa quantità di *informazione*, di trasmettere solo una parte di tali bit senza modificarne il contenuto informativo. Un esempio semplicissimo si presenta quando bisogna trasmettere una *immagine* costituita da uno sfondo bianco e qualche variazione di colore solo in alcuni punti; se si decide di trasmettere tutti i bit corrispondenti ai *pixel* di cui si compone l'immagine ([8]), è abbastanza evidente che si trasmette molto più di quanto sia in realtà necessario: per esempio, in corrispondenza di una sequenza di pixel tutti bianchi, si trasmetterebbe un numero molto alto di bit aventi tutti lo stesso valore (ad esempio tutti 0) ed è noto che tanti bit uguali tra di loro hanno un contenuto informativo estremamente basso ([9]). Le *tradizionali tecniche di compressione*, in una situazione di questo tipo, intervengono in modo molto semplice: anziché inviare, ad esempio, 1000 bit tutti posti a 0, inviano un solo bit posto a 0 e poi una parola, di pochi bit, che indica che i bit consecutivi posti a 0 sono 999. In questo modo, il bit rate, a parità di informazione da trasmettere, viene senz'altro abbassato.

La **tecnica di compressione dell' header** di cui ci occupiamo in questa sede è invece qualcosa di diverso, anche se, ovviamente, il concetto di fondo è sempre quello di ridurre la ridondanza delle informazioni, inviando solo quelle strettamente necessarie. La compressione di cui ci occupiamo prende le mosse dalle seguenti considerazioni fondamentali:

[7] Le tecniche di compressione cosiddette *lossless* sono quelle che forniscono la compressione senza perdita di informazione. Al contrario, quelle definite *lossy* forniscono la compressione tramite un filtraggio di quelle informazioni che sono ritenute non essenziali per gli utenti. In questo contesto, si considerano solo tecniche lossless.

[8] Si ricordi che una immagine è suddivisa in un certo numero di **pixel**, ciascuno dei quali ha un colore codificato da un certo numero di bit: ad esempio, associando un solo bit a ciascun pixel, si potrà solo dire se tale pixel è bianco o nero e cioè si potrà avere solo una immagine in bianco e nero; oppure, associando 3 bit a ciascun pixel, si potranno avere 2^3=8 gradazioni intermedie (le cosiddette *gradazioni di grigio*) tra il bianco ed il nero.

[9] La teoria dell'informazione dice infatti che una informazione è tanto più "importante" quanto meno è ricorrente: se una sorgente, dotata di un certo alfabeto di simboli $X_1,X_2,\ldots,X_N$, emette mille volte il simbolo X_1 e 10 volte il simbolo X_N, è evidente che quest'ultimo ha un contenuto informativo maggiore rispetto ad X1, proprio perché ricorre meno volte.

- consideriamo tutti i pacchetti che appartengono ad uno stesso *flusso* instauratosi tra due "stazioni" della rete, ad esempio un terminale mobile e la stazione radio base su cui esso è attualmente appoggiato: proprio per il fatto di appartenere ad uno stesso flusso, questi pacchetti presentano header molto simili tra loro; ad esempio, presentano valori, in alcuni campi dell' header, che rimangono costanti da pacchetto a pacchetto (si pensi tipicamente ai campi di indirizzo); allora, diventa inutile trasmettere sempre questi valori, pacchetto per pacchetto: è sufficiente trasmetterli una volta (per il primo pacchetto del flusso in questione) e rendere noto al ricevitore (cioè al decompressore) che, per tutti i pacchetti successivi, potrà usare sempre gli stessi valori. In quest'ottica, quindi, il compito del compressore (in trasmissione) è quello di *filtrare* le informazioni che non è necessario trasmettere, mentre invece quello del decompressore (in ricezione) è quello di *aggregare*, ai pacchetti ricevuti (che possiamo definire **pacchetti con header compresso**), le informazioni di cui sono stati privati e che il decompressore stesso ha precedentemente conservato nella propria memoria;

- ovviamente, non tutti i campi dei vari header contengono valori costanti nell'ambito di uno stesso flusso; ci sono infatti campi i cui valori variano da pacchetto a pacchetto, ma secondo regole ben precise: ad esempio, il campo *sequence number* dell' header TCP viene di volta in volta incrementato della dimensione (espressa in byte) del pacchetto TCP precedente. In campi come questo, quindi, la variazione di valore non solo è predicibile, ma è anche generalmente piccola rispetto al *valore pieno*: questo significa che, anziché trasmettere di volta in volta il valore pieno, si può trasmettere solo la variazione (**tecniche di codifica differenziale**), utilizzando sicuramente un numero minore di bit e quindi ottenendo ancora una volta un risparmio di bit rate. Ovviamente, il decompressore dovrà sapere che, per questo tipo di campi, il valore ricevuto è solo la differenza con il valore relativo al pacchetto precedente, per cui dovrà compiere una banale somma per ottenere il valore effettivo;

- ci sono, infine, quei campi il cui contenuto varia in modo del tutto casuale: questi sono gli unici campi su cui il compressore/decompressore non può compiere operazioni di alcun tipo, per cui vanno trasmessi comunque così come sono ("as is"). Sono dunque i campi che determinando in maniera

preponderante la dimensione dell' header dopo la compressione e quindi anche il bit rate ottenibile.

Tutte queste considerazioni (alle quali si potrebbe aggiungere in realtà molto altro) mostrano dunque la possibilità e l'utilità di effettuare la compressione dell' header dei pacchetti IP che viaggiano tra un terminale mobile ed una stazione radio base, dove per **header** non intendiamo solo l'header IP, ma anche l' header dei protocolli superiori (ad esempio UDP ed RTP) rispetto al livello network: ad esempio, per un pacchetto RTP/UDP/IP, l'header è costituito dalla sequenza dei campi che costituiscono, nell'ordine, l'header IP, l'header UDP e l'header RTP.

Ovviamente, dal punto di vista dell'implementazione pratica degli algoritmi di compressione e decompressione, le considerazioni da fare sono molte di più, specialmente perché bisogna tener conto del problema fondamentale dei collegamenti wireless: infatti, la **rumorosità** di questi collegamenti, quantificata da un **BER** (*Bit Error Rate*) non così basso come nei collegamenti via cavo, determina frequenti **errori** sui pacchetti trasmessi o, addirittura, **perdita** dei pacchetti stessi, il che potrebbe causare problemi non indifferenti al decompressore, che potrebbe non accorgersi (dall'implementazione) degli errori e/o delle perdite e quindi effettuare delle decompressioni errate.

A questo proposito, si può fare subito una distinzione tra **flussi TCP** (che cioè utilizzano a livello transport il protocollo TCP) e **flussi non TCP** (che cioè tipicamente usano UDP, come per esempio il protocollo RTP):

- essendo il **TCP** un protocollo affidabile orientato alla connessione, esso è intrinsecamente "protetto" dalla perdita di pacchetti ([10]) tramite il meccanismo delle **ritrasmissioni**; tuttavia, dato che il TCP usa trasmettere sempre un certo numero di pacchetti successivi (secondo il meccanismo della *sliding window*) prima di accertarsi (tramite il meccanismo degli *ACK*) che tutto sia andato bene, l'eventuale perdita di un pacchetto potrebbe non essere immediatamente avvertita dal decompressore in ricezione, il che potrebbe provocare un errato comportamento del decompressore stesso fino al momento in cui gli errori non vengono *riparati* dal TCP;

[10] Per "perdita" di un pacchetto possiamo intendere sia il fatto che esso si perda del tutto sul collegamento (cosa non infrequente sui collegamenti via radio), e quindi non giunga a destinazione, sia più semplicemente che, una volta giunto in ricezione, il meccanismo della checksum di livello fisico rilevi degli errori sui bit e scarti il pacchetto senza passarlo ai livelli superiori.

- più "pericolosa" è la situazione per quei flussi che si appoggiano su **UDP** o che, in generale, non prevedono alcun meccanismo di ritrasmissione: in questi casi, la presenza di eventuali errori sui pacchetti giunti (o non giunti affatto) avrà senz'altro ripercussioni anche su tutti i pacchetti successivi, a meno di non prevedere opportuni meccanismi che, nel corso della connessione, si preoccupino di ripristinare, a prescindere dalla presenza o meno di problemi, il corretto funzionamento del decompressore.

Nonostante queste affermazioni possano apparire poco "familiari" per chi non conosca i dettagli di implementazione degli algoritmi di compressione/decompressione, esse dovrebbero comunque dare una idea del fatto che le problematiche da affrontare in questo contesto siano in realtà davvero notevoli. Del resto, un semplicissimo esempio può chiarire le idee:

- consideriamo un flusso di pacchetti per il quale sia stato individuato un particolare campo dell' header suscettibile di *codifica differenziale*; questo significa, come già detto, che, dopo aver trasmesso una volta il valore pieno di tale campo, il compressore, per ogni pacchetto successivo, invia solo la variazione di valore rispetto al pacchetto precedente; sarà poi compito del decompressore "ricostruire" (o "decomprimere") il valore corretto, facendo una banale somma tra il valore arrivato e quello precedentemente conservato in memoria. Ad esempio, se il valore iniziale pieno giunto al decompressore era X ed il pacchetto successivo reca il valore Y, il decompressore deduce che il valore corretto è X+Y, per cui ricostruisce (cioè decomprime) il pacchetto e, contemporaneamente, sostituisce il valore in memoria X con il nuovo valore X+Y;
- supponiamo adesso che il pacchetto ancora successivo (sarebbe il terzo del flusso), recante un determinato valore Z (sempre incrementale), si perda sul collegamento ed il decompressore non sappia nulla di tale perdita; quando arriva il pacchetto successivo (il quarto), recante un valore U nello stesso campo, il decompressore "pensa" che il valore corretto sia adesso X+Y+U ed invece commette un errore, in quanto il valore corretto (tenendo conto del pacchetto perso) sarebbe X+Y+Z+U;
- a questo punto, se il flusso in questione è basato sul TCP, c'è la possibilità che il *TCP ricevente* riconosca (tramite il meccanismo della *checksum*)

l'errato valore ricostruito dal decompressore, nel qual caso il decompressore viene avvertito che c'è stato un problema e viene quindi fermato fin quando il *TCP mittente*, non avendo ricevuto la conferma del terzo pacchetto, riprende proprio da esso la trasmissione ([11]), ripristinando così la correttezza del flusso;

- se invece il flusso è basato su UDP, allora l'errore potrebbe anche non essere rilevato, nel qual caso il decompressore continuerà a lavorare come se niente fosse, sbagliando i valori di quel campo per tutti i successivi pacchetti; per evitare questo, si deve prevedere un qualche *meccanismo riparatore*: ad esempio, si può fare in modo che, periodicamente, dopo un certo numero di pacchetti inviati dal trasmettitore in forma compressa, venga comunque inviato un **pacchetto non compresso**: il decompressore, avvisato di questo, è così in grado di ripristinare periodicamente il valore corretto nella propria memoria e quindi di riprendere a funzionare correttamente. La differenza con il flusso TCP è che, comunque, con l'UDP risultano inevitabili gli errori su alcuni pacchetti successivi a quello andato perso.

Algoritmo ROHC

Quelle esposte nei precedenti paragrafi sono dunque le linee guida del "problema". Come si è detto, le proposte di compressione dell'header di pacchetti IP sono state poche nel tempo: la prima proposta di una certa importanza risale al 1990 ([12]), dopodiché la questione è stata quasi del tutto ignorata fino al 1999, quando sono state introdotte due nuove proposte di una certa concretezza: la prima ([13]) era limitata a pacchetti con header IPv4, IPv6, TCP e UDP, mentre la seconda ([14]) passò ad includere anche l'uso del protocollo RTP, che costituisce lo standard ormai più utilizzato per i servizi multimediali (inclusa quindi la telefonia) sulle reti IP ed è quindi uno dei maggiori potenziali "fruitori" dei meccanismi di compressione.

Il limite grosso, però, di queste proposte è quello di essere riferite tipicamente a collegamenti via cavo (nonostante siano comunque presenti, in tutti i documenti

[11] Il protocollo TCP usa, infatti, in presenza di perdita di un pacchetto, il meccanismo detto go-back-n, per cui la trasmissione, una volta individuato il pacchetto perso, riprende proprio da quel pacchetto e quindi comporta l'eventuale ritrasmissione anche dei successivi pacchetti che sono già stati trasmessi.
[12] RFC 1144: V. Jacobson, "Compressing TCP/IP headers for low-speed serial links", Febbraio 1990
[13] RFC 2507, M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression", February 1999.
[14] RFC 2508, S. Casner, V. Jacobson, "Compressing IP/UDP/RTP Headers for low-speed serial links", February 1999.

citati, una serie di "indicazioni" circa le applicazioni su collegamenti wireless): le differenze fondamentali tra i link via cavo e quelli wireless risiedono nel tasso d'errore (BER) e nel ritardo (RTT), che sono entrambi generalmente molto più bassi (specialmente il BER) sul cavo che non via radio. Di conseguenza, le prestazioni degli algoritmi proposti non sono sicuramente adeguate ai requisiti richiesti dai collegamenti wireless.

E' stata allora fatta una nuova ed interessante proposta esplicitamente per i collegamenti wireless: si tratta del cosiddetto algoritmo di **RObust Header Compression** ([15]), sul quale sarà concentrato il mio lavoro di tesi. Il mio compito è quello di implementare concretamente l'algoritmo ROHC (o meglio una sua recentissima evoluzione) nel programma di simulazione **Network Simulator**, al fine di esaminare concretamente le prestazioni di tale algoritmo (con particolare riguardo a quello che succede in presenza di perdita di pacchetti) e, eventualmente, di proporre modifiche per l'incremento di tali prestazioni.

Autore: **Sandro Petrizzelli**
e-mail: sandry@iol.it
sito personale: http://users.iol.it/sandry

[15] Bormann , Internet Draft (IETF) Rohc-RTP-09, February 2001