CAPITOLO 3

CONCETTI GENERALI SULL'ALGORITMO ROHC

3.1 Robust Header Compression (ROHC)

Sono ormai diversi anni che la comunità scientifica internazionale ha cominciato ad interessarsi agli *algoritmi di compressione dell'header*, ma solo in tempi molto recenti tale interesse ha subito un deciso incremento. In particolare, si può far risalire ai primi mesi dell'anno 1999 l'inizio di numerose e concrete attività di ricerca volte ad individuare un *algoritmo di compressione dell'header IP per sistemi wireless*, destinato principalmente (ma non esclusivamente) ai futuri sistemi radiomobili basati su IP (UMTS).

Particolare attenzione a questa problematica è stata dedicata dall' **IETF** (Internet Engineering Task Force), tanto da portare, nel febbraio 2000, alla nascita di un apposito gruppo di lavoro, denominato **ROHC** [29].

3.1.1 Cenni storici sulla nascita dell'algoritmo ROHC

Il gruppo ROHC si pose inizialmente l'obbiettivo di individuare un algoritmo di compressione dell'header specifico per applicazioni real-time e arrivò a definire due distinti schemi di compressione: **ACE** (Adaptive header ComprEssion for real time multimedia) [30,31] e **ROCCO** (Robust Checksum-based header COmpression) [32,33].

Col passare del tempo, però, l'obbiettivo del ROHC WG si è esteso, puntando a definire un algoritmo di compressione del tutto generale, valido cioè per qualunque tipo di traffico implementabile, su piattaforma TCP/IP, in un sistema wireless.

Verso la fine di giugno 2000 è stata così proposta la prima versione dello schema di compressione denominato **ROHC** (*Robust Header Compression*), verso il quale si è concentrata l'attenzione dell'intero gruppo.

Nei mesi successivi, si è poi assistito a rapidi mutamenti nell'ambito dell'intera struttura organizzativa del workgroup: il 30 novembre 2000 sono state rimosse le draft relative agli schermi ACE e ROCCO, ritenute troppo specifiche e di scarsa utilità per i nuovi propositi del gruppo; nel giro di poco più di un anno, inoltre, sono state presentare ben 9 diverse versioni della draft ROHC (ognuna delle quali ha apportato modifiche spesso significative rispetto alla versione precedente), fino al documento finale [3], presentato in tempi recentissimi (luglio 2001).

3.1.2 Concetti generali sulla compressione ROHC

L'algoritmo ROHC è in grado di gestire ogni tipo di flusso di pacchetti che faccia riferimento ad una qualsiasi pila protocollare nell'ambito dell'**architettura TCP/IP**. Nella sua versione originale, tuttavia, esso privilegia i flussi RTP/UDP/IP, essendo questi più idonei a gestire servizi di tipo real-time, per i quali le esigenze di compressione sono ritenute più forti.

La compressione sfrutta la *ridondanza* esistente tra i campi dell'header sia all'interno di uno stesso pacchetto sia soprattutto tra pacchetti consecutivi appartenenti allo stesso flusso. Ai fini della compressione, ognuno degli header gestiti dall'algoritmo viene suddiviso in tre parti (figura 1.3):

- la **parte statica** è costituita dai campi dell'header che si mantengono con alta probabilità costanti durante tutta la connessione. In condizioni ideali, tali campi possono essere inviati solo con il primo pacchetto della connessione;
- la **parte dinamica** è costituita dall'insieme dei campi che variano, in modo più o meno frequente, durante la singola connessione e che quindi devono essere trasmessi, almeno in linea teorica, con ogni pacchetto;
- la **parte ridondante** è infine costituita da quei campi i cui valori possono essere ricavati attraverso la conoscenza della parte statica e dinamica nonché attraverso i campi degli header dei livelli inferiori a quello di rete. Per questo motivo, tali campi possono non essere mai trasmessi nel corso nell'intera connessione.

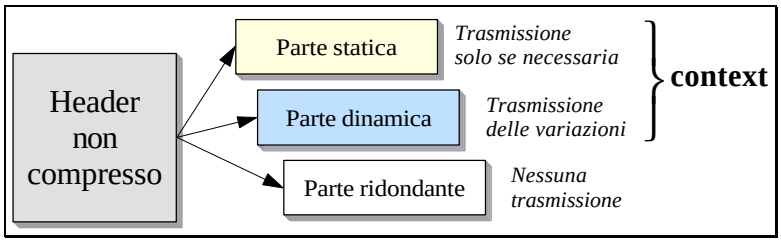

Figura 1.3 - Suddivisione, nello schema ROHC, dell'header di un pacchetto IP

L'insieme delle parti statiche e dinamiche, unito ad una serie di informazioni utili addizionali, prende il nome di **context**. Ad ogni connessione è associato dunque un context: esso contiene l'informazione utile alla compressione

dell'header (nella sezione di trasmissione) ed alla sua corretta decompressione (nella sezione di ricezione).

Ogni context è identificato da un numero, detto **CID** (*Context Identifier*), che gli viene associato durante la fase di negoziazione della connessione. In questo modo, sullo stesso "link" tra compressore e decompressore è possibile gestire più di un flusso. Il *formato* del primo pacchetto inviato dal compressore consente al decompressore di registrare il CID associato alla connessione cui il pacchetto stesso appartiene.

Nella fase iniziale della connessione, il rapporto di compressione è quasi nullo, dato che l'header viene "alleggerito" della sola parte ridondante e contestualmente "appesantito" da informazioni aggiuntive necessarie per inizializzare il decompressore. Successivamente, il compressore provvede ad aumentare il rapporto di compressione, inviando, con opportuni *formati compressi*, solo quei campi dinamici che subiscono variazioni tra un pacchetto ed il successivo.

Inoltre, al fine di migliorare ulteriormente il rapporto di compressione, al posto di inviare per intero i campi dinamici, si inviano le sole *variazioni*, opportunamente codificate.

Il decompressore conserva memoria del context, per tutta la durata della connessione cui è associato, in una tabella indicizzata dal CID; tale tabella viene aggiornata ad ogni pacchetto ricevuto e decompresso con successo.

Su uno stesso link si possono avere più flussi contemporaneamente attivi, con caratteristiche anche molto diverse tra loro. Al fine di ottimizzare la compressione di tali flussi, è necessario definire un insieme di *regole di compressione*, specifiche per ciascun flusso: tale insieme di regole prende il nome di **profilo di compressione**.

Il profilo di compressione definisce le regole di sintassi con cui si devono interpretare i tipi di pacchetti ROHC ricevuti nell'ambito di un particolare context.

In definitiva, il compressore ed il decompressore usano, per svolgere i propri compiti, un context ed un profilo di compressione associati a ciascuna connessione attiva (figura 2.3).

I concetti di "context" e di "profilo" conferiscono al ROHC quella flessibilità necessaria per qualunque schema di compressione dell'header: durante la fase di negoziazione della connessione, il compressore verifica quali profili si possono implementare (ossia quali profili sono "permessi" dal decompressore) ed utilizza quello ritenuto ottimale per ciascuna connessione gestita. Non solo, ma l'estensione dello schema ROHC risulta particolarmente

agevole: basta definire un nuovo profilo di compressione che rispetti le "linee guida" indicate in [3].

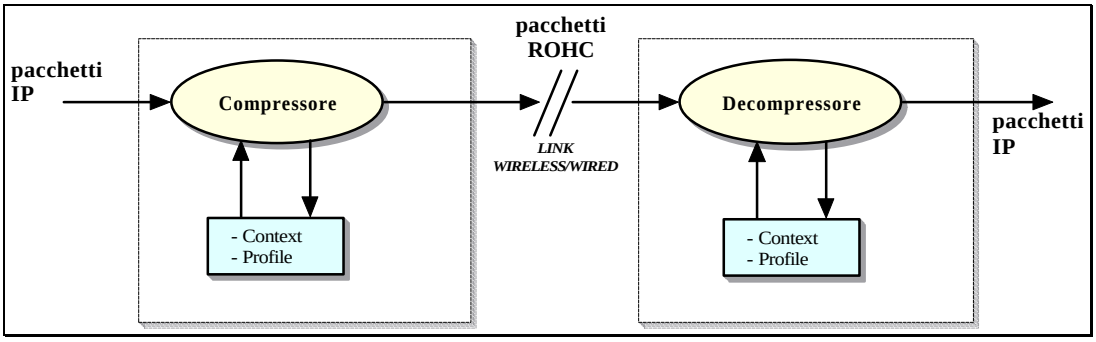

Figura 2.3 - Schematizzazione del processo di compressione e decompressione con algoritmo ROHC

3.1.2.1 Profili di compressione nella versione originale del ROHC

Nella sua versione originale, il ROHC prevede solo quattro distinti profili di compressione:

- **profilo 1 - RTP/UDP/IP**: per la compressione di flussi di tipo RTP/UDP/IP. E' il profilo fondamentale dell'algoritmo ROHC, essendo rivolto ad una pila di protocolli tipica delle applicazioni real-time; esso viene anche espressamente indicato, dal ROHC workgroup, come insieme di "linee guida" cui deve necessariamente adeguarsi ogni altro profilo di compressione da integrare nello schema ROHC;
- **profilo 2 - UDP/IP**: per la compressione di flussi di tipo UDP/IP cui non si voglia o non si possa applicare il profilo precedente;
- **profilo 3 - ESP/IP**: per la compressione dei flussi protetti mediante il protocollo di sicurezza ESP (*Encapsulating Security Payload*) [34];
- **profilo 0 - Flussi non compressi**: per la gestione di tutti i flussi non inclusi nelle categorie precedenti, compresi i flussi che utilizzano TCP a livello di trasporto, nonché per la gestione di flussi che non si desidera vengano compressi.

*La variante **ROHC+**, proposta dall'ing. Vito Squeo [35], include un nuovo profilo di compressione, specifico per la compressione di flussi TCP/IP (profilo*

4). Il perfezionamento e l'analisi (tramite simulazioni) delle prestazioni di questo nuovo profilo costituiscono l'oggetto di questa tesi.

3.1.2.2 Canale di feedback

L'algoritmo ROHC, per poter esplicare meglio la sua funzione, prevede l'utilizzo di periodici *pacchetti di feedback* inviati dal decompressore al compressore, con frequenza variabile a seconda di alcuni parametri operativi che saranno illustrati in seguito.

Per non peggiorare eccessivamente la richiesta di banda, è tuttavia opportuno individuare una modalità efficiente di implementazione del feedback, la quale non può che dipendere strettamente dalle caratteristiche del livello fisico. Per il momento, la scelta definitiva dell'IETF è stata rimandata ad un documento specifico da introdurre in futuro. Ad ogni modo, vengono indicate solo due alternative plausibili:

- uso di un *canale di feedback dedicato*, riservato all'invio dei soli pacchetti di feedback;
- uso di un *canale di feedback virtuale*, sul quale vengano fatti transitare sia i pacchetti di forward sia quelli di feedback. In questo caso, al fine di minimizzare l'occupazione di banda, si possono adottare due distinte soluzioni:
    - *interleaving*: i pacchetti di feedback vengono inviati, con periodicità fissa e nota, ogni *n* pacchetti di forward;
    - *piggybacking*: l'informazione di feedback viene inviata come estensione del payload del generico pacchetto di forward.

La scelta di un canale di feedback dedicato appare senz'altro quella più affidabile, ma richiede anche una occupazione maggiore di banda, per cui viene scartata a favore dell'uso di un **canale di feedback virtuale**. Inoltre, la tecnica dell'interleaving rischierebbe di non far inviare i pacchetti di feedback nelle situazione in cui se ne avrebbe la necessità, per cui viene scartata a favore del **piggybacking**.

Questa soluzione ottimizza sia l'uso della banda sia anche il funzionamento degli organi di compressione/decompressione (dato che il feedback viene inviato ogni volta che se ne presenta la necessità), ma peggiora l'affidabilità, in

quanto la perdita di un pacchetto comporta la perdita sia dell'informazione di forward sia dell'eventuale feedback ad essa accodata.

Nella figura 3.3 viene schematizzata, in modo completo, l'architettura del sistema compreso tra i due **punti di compressione**, con dettaglio proprio sul piggybacking:

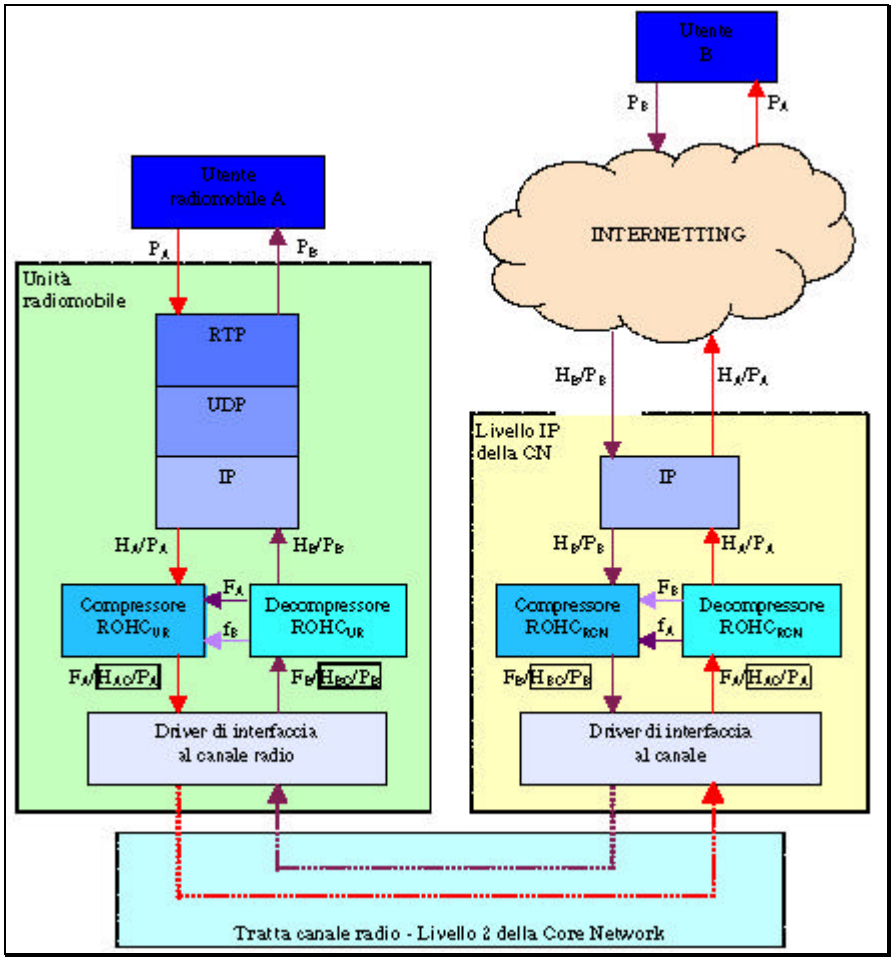

Figura 3.3 - Architettura della tratta radio con dettaglio sul piggybacking: i pacchetti in uscita dal decompressore contengono il pacchetto di forward compresso (header compresso H_C + payload P) cui viene aggiunto il pacchetto di feedback (F)

3.1.2.3 Parametri di stato

Lo schema ROHC prevede che gli organi di compressione e decompressione facciano uso di un set di **parametri di stato** che caratterizzano sia il canale sia il terminale radiomobile:

- **MAX_CID**: numero intero non negativo che indica il massimo valore che il compressore può usare per il CID;
- **LARGE_CIDS**: parametro booleano che, se settato, indica l'uso di *large CID* (7 o 14 bit riservati per il CID); se invece non è settato, indica l'uso di *small CID* (da 0 a 4 bit); insieme al precedente parametro, vincola il numero di connessioni contemporanee che possono usufruire della compressione;
- **PROFILES**: insieme di numeri interi, ciascuno rappresentativo di un profilo di compressione che il decompressore è in grado di gestire. Il compressore è obbligato ad usare solo i profili di compressione appartenenti a questo insieme;
- **FEEDBACK_FOR**: numero intero rappresentativo del tipo di feedback adottato;
- **SDU** (*Service Data Unit*): dimensione massima del pacchetto in entrata al compressore ed in uscita dal decompressore;
- **MRRU** (*Maximum Reconstructed Reception Unit*): massima dimensione dell'unità ricostruita sulla base del protocollo di frammentazione ROHC. Se vale 0, il decompressore non supporta la frammentazione.

L'uso di questi parametri consente allo schema di compressione di mantenere una sufficiente insensibilità alle escursioni parametriche del canale radio (requisito della *robustezza*) e di garantire una elevata compatibilità tra una vasta gamma di terminali radiomobili (requisito della *flessibilità*),

Il valore dei parametri di stato deve essere negoziato prima di iniziare una connessione, sfruttando i meccanismi forniti dai protocolli di livello inferiore [36]. Inoltre, si ritiene utile poter ri-negoziare gli ultimi tre nel corso della connessione: ad esempio, si può pensare di testare periodicamente il canale e la rete in generale, misurando rumorosità, livello di congestione e disponibilità di risorse in ricezione, in modo da modulare i valori dei predetti parametri durante la connessione.

3.2 Classificazione dei campi dell'header

Prima ancora di descrivere i dettagli del profilo di compressione proposto in questa tesi (capitolo 4), è fondamentale illustrare il significato ed i modelli tipici di variazione dei campi degli header su cui tale profilo deve agire.

3.2.1 Criteri generali di classificazione

Il ROHC workgroup ha stabilito di separare i campi dell'header di un qualsiasi protocollo nelle seguenti 5 classi fondamentali:

- **INFERRED** (*ridondanti*): contengono valori che possono essere dedotti da valori di altri campi (come ad esempio la dimensione del frame di livello fisico che ingloba il pacchetto di livello rete) e quindi non vengono mai trasmessi nel corso della connessione;
- **STATIC** (*statici*): sono i campi che ci si attende rimangano costanti per tutta la vita di un dato flusso. In assenza di variazioni e/o di errori, tali campi possono essere trasmessi una sola volta, all'inizio della connessione;
- **STATIC-DEF** (*statici e caratteristici*): si tratta sempre di campi statici, con in più la caratteristica di individuare un flusso di pacchetti, nel senso che tutti i pacchetti di uno stesso flusso hanno sicuramente gli stessi valori dei campi STATIC-DEF;
- **STATIC-KNOWN** (*statici e ben noti*): anche questi campi sono statici e, in più, ci si aspetta che abbiano valori ben precisi e noti a priori, per cui non necessitano di essere trasmessi (come i campi ridondanti);
- **CHANGING** (*dinamici*): questi campi assumono valori variabili nel corso di una stessa connessione, seguendo "modelli di variazione" non necessariamente costanti e non necessariamente noti a priori: possono variare in modo casuale o in modo lineare, possono assumere solo valori all'interno di un limitato intervallo oppure valori qualsiasi e altro ancora. In linea di massima, tali campi devono necessariamente essere trasmessi con ogni pacchetto (sia pure opportunamente codificati).

Nei prossimi paragrafi verrà illustrata la classificazione che il ROHC workgroup, in base alle definizioni appena fornite, ha fatto dei campi degli

header IPv4 ed IPv6. In seguito, si proporrà la classificazione per i campi dell'header TCP introdotta dal profilo TCP/IP oggetto di questa tesi.

3.2.2 Campi dell'header IPv4

Un datagramma IPv4 è costituito da un header e da un payload. A sua volta, l'header ha una parte fissa di 20 byte e una parte, opzionale, di lunghezza variabile. La figura 4.3 illustra la struttura dell' header IPv4 di base.

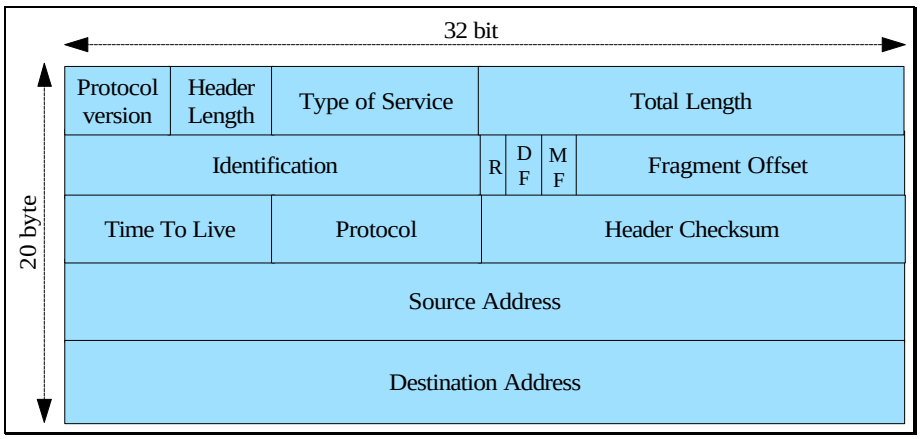

Figura 4.3 - Header IPv4 di base

Dal punto di vista della compressione, il ROHC workgroup ha inteso classificare i vari campi come riportato nella tabella 1.3.

Tabella 1.3 – Classificazione dei campi dell'header di base IPv4

Campo	Dimensione (bit)	Classe
Protocol Version	4	STATIC
Header Length	4	STATIC-KNOWN
Type of Service	8	CHANGING
Packet Length	16	INFERRED
Identification	16	CHANGING
Reserved Flag	1	STATIC-KNOWN
Don't Fragment Flag	1	STATIC
More Fragment Flag	1	STATIC-KNOWN
Fragment Offset	13	STATIC-KNOWN
Time to Live	8	CHANGING
Protocol	8	STATIC
Header Chacksum	16	INFERRED
Source Address	32	STATIC-DEF
Destination Address	32	STATIC-DEF

Segue una descrizione del significato dei vari campi e dei motivi della relativa classificazione:

- *Protocol Version*: indica la versione di IP utilizzata (attualmente 4 o 6). I pacchetti con differenti valori in questo campo devono essere trattati da diverse pile IP. Tutti i pacchetti di uno stesso flusso devono quindi avere lo stesso valore di questo campo, per cui esso è classificato di tipo STATIC;
- *Header Length*: dato che la lunghezza dell'header IPv4 può non essere costante per la presenza dei campi opzionali, questo campo la riporta esplicitamente, in termini di parole da 32 bit (il valore massimo è dunque 15·4 byte). In assenza dei campi opzionali la lunghezza rimane costante e ben nota (20 byte). In ogni caso, il ROHC assume che le informazioni sulla lunghezza dei pacchetti (sia complessiva sia del solo header) vengano fornite dai livelli inferiori, per cui il campo Header Length viene classificato STATIC-KNOWN;
- *Type of Service*: consente all'host mittente di comunicare alla rete il *tipo di servizio* richiesto, sulla base di una serie di combinazioni di *affidabilità* e *velocità* ([1]); nel corso di una stessa connessione, generalmente tale campo rimane invariato in quanto rimangono invariati i requisiti richiesti dal mittente; tuttavia, non si possono escludere variazioni occasionali, il che impone di classificare questo campo di tipo CHANGING;
- *Packet Length*: contiene la lunghezza l'intero pacchetto IP (header+payload); l'algoritmo di compressione presuppone che le informazioni sulla lunghezza del pacchetto siano fornite dal livello fisico, per cui questo campo è classificato INFERRED;
- *Identification*: consente all'host destinazione di determinare a quale pacchetto appartiene un dato frammento, in quanto tutti e soli i frammenti di uno stesso pacchetto hanno lo stesso valore di questo campo; esso viene dunque considerato di tipo CHANGING. Maggiori considerazioni su questo campo saranno fatte nel paragrafo 3.2.5;

[1] Ad esempio, per la voce digitalizzata si richiede la massima velocità di trasmissione, mentre invece per il trasferimento file si richiede soprattutto una trasmissione priva di errori. Il campo è allora diviso in *sottocampi*, in modo da esprimere le combinazioni desiderate dalla sorgente

- *Flag*: l'header IPv4 contiene 3 flag; il *Reserved Flag* vale sempre 0 ed è quindi classificato STATIC-KNOWN; il flag *Don't Fragment* è usato dalla sorgente IP per indicare se i suoi datagrammi possono essere frammentati (valore 0) o meno (valore 1) ([2]), per cui è necessariamente costante per tutti i pacchetti di uno stesso flusso, il che lo fa classificare come STATIC. Il flag *More Fragment* serve infine ad indicare se il frammento è l'ultimo (MF=0) del pacchetto cui appartiene oppure no (MF=1). Nelle reti miste wireless-wired, la frammentazione non risulta mai necessaria (dato che i pacchetti sono generalmente di piccole dimensioni per le applicazioni considerate), per cui questo flag è atteso sempre di valore 0 e quindi classificato STATIC-KNOWN;
- *Fragment Offset*: indica in quale posizione del pacchetto originale (espressa come offset, in unità di 64 bit, rispetto al primo byte) si trova il frammento attuale, in modo da consentire al ricevente di ricostruire il pacchetto originale a partire dai singoli frammenti in cui è stato scomposto, rispettando l'ordine corretto ([3]). Sotto l'ipotesi che non risulti mai necessaria la frammentazione dei pacchetti, si assume che questo campo contenga sempre valore 0, per cui è classificato STATIC-KNOWN;
- *Time to Live*: è semplicemente un contatore (inizializzato al valore 255) che viene decrementato a ogni *salto* (*hop*), in modo da contare i salti compiuti dal pacchetto nel suo percorso verso la destinazione ([4]) e da eliminare il pacchetto quando si ritiene che sia ancora in rete solo per errore (contatore nullo). In ogni nodo di rete che riceve il pacchetto, il valore di tale campo dipende dunque dalla "strada" precedentemente percorsa, per cui il campo è classificato CHANGING;
- *Protocol*: indica il protocollo di livello di trasporto cui i dati devono essere consegnati; esistono diverse possibilità (*TCP*, *UDP*, ecc.), secondo la numerazione universale definita in [37]. Questo campo ha dunque lo stesso valore per tutti i pacchetti di uno stesso flusso, per cui è classificato STATIC;

[2] Se la frammentazione si rende necessaria ma risulta DF=1, il datagramma IP viene scartato e viene trasmesso un messaggio di errore (ICMP).
[3] Essendo lungo 13 bit ed essendo 64 kB la dimensione massima di un datagramma IP, questo campo impone che i frammenti siano necessariamente di lunghezza multipla di 64 bit.
[4] In realtà, il decremento è funzione anche del tempo di attesa in coda al router se questo supera un certo limite [2].

- *Header Checksum*: protegge i singoli nodi dal processare un header corrotto, calcolando, sull'header originale, una somma di controllo ed imponendone la verifica su ogni nodo. Quando quasi tutta l'informazione dell'header è stata compressa, non c'è bisogno di avere questa checksum addizionale, che tra l'altro viene ricalcolata in ciascun nodo a seguito delle variazioni del campo Time to Live. Di conseguenza, essa può essere rigenerata in sede di decompressione [38], per cui questo campo è classificato INFERRED;
- *Source Address* e *Destination Address*: contengono gli indirizzi IP dell'host sorgente e dell'host destinazione; sono quindi costanti per l'intera connessione e, inoltre, fanno parte della definizione del flusso. Sono perciò classificati STATIC-DEF.

La tabella 2.3 riepiloga la dimensione totale dei campi di ciascuna classe. Si può notare la massiccia presenza di campi statici e ridondanti rispetto a quelli dinamici, il che consente buoni livelli di compressione in tutti i casi in cui solo questi ultimi devono essere trasmessi. Viceversa, appare ridotta la dimensione totale dei campi ridondanti, il che significa che, nelle fasi iniziali della compressione (in cui avviene sostanzialmente solo l'eliminazione dei campi ridondanti), l'efficienza raggiungibile è necessariamente limitata.

Tabella 2.3 – Lunghezza totale dei campi dell'header di base IPv4 suddivisi in base alla classe di appartenenza

Classe	**Totale bit**
INFERRED	32
STATIC	13
STATIC-KNOWN	19
STATIC-DEF	64
CHANGING	32

3.2.3 Campi dell'header IPv6

Il formato del basic header IPv6 è riportato in figura 5.3.

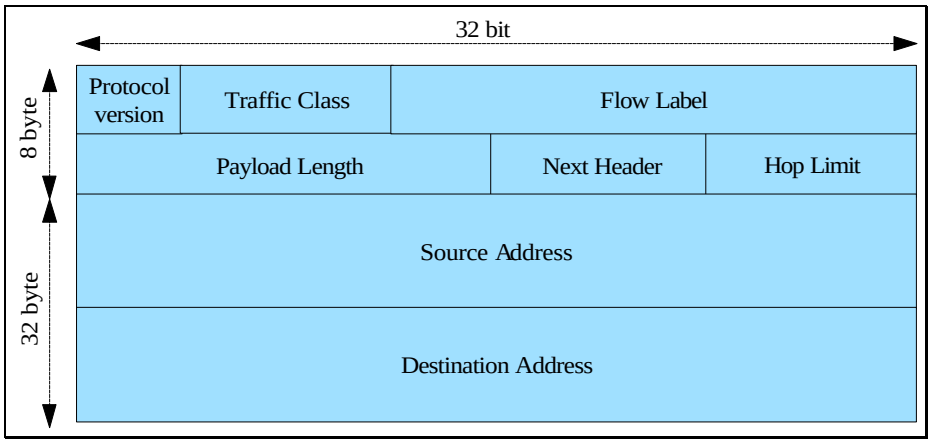

Figura 5.3 - Header IPv6 di base

Dal punto di vista della compressione, il ROHC workgroup ha inteso classificare i vari campi come riportato nella tabella 3.3.

Tabella 3.3 – Classificazione dei campi dell'header IPv6 di base

Campo	Dimensione (bit)	Classe
Protocol Version	4	STATIC
Traffic Class	8	CHANGING
Flow Label	20	STATIC-DEF
Payload Length	16	INFERRED
Next Header	8	STATIC
Hop Limit	8	CHANGING
Source Address	128	STATIC-DEF
Destination Address	128	STATIC-DEF

Segue la descrizione dei vari campi ed i motivi della relativa classificazione:

- *Protocol Version*: come in IPv4;
- *Traffic Class*: stesse considerazioni del campo IPv4 Type of Service;
- *Flow label*: riporta una etichetta che individua i flussi dei pacchetti in base alle particolari caratteristiche richieste dal servizio (ad esempio QoS, categorie real time, ecc.). Si tratta di un campo decisamente ancora sperimentale [25], che i router IPv6 dovrebbero utilizzare per gestire

l'instradamento dei pacchetti in maniera differenziata. Se non viene usato, esso deve contenere valore 0; altrimenti, tutti i pacchetti appartenenti ad uno stesso flusso devono avere lo stesso valore in questo campo, che quindi diventa uno dei campi identificativi del flusso. Per queste ragioni, il campo è classificato STATIC-DEF;

- *Payload Length*: indica la lunghezza del payload del pacchetto IP. L'algoritmo di compressione presuppone che le informazioni circa la lunghezza del pacchetto siano fornite dal livello fisico, per cui questo campo è classificato INFERRED;
- *Next Header*: in uno stesso pacchetto IPv6, possono essere presenti diversi header (l'*header di base* e, opzionalmente, i cosiddetti *extension header*); ognuno ha un proprio campo Next Header, che indica il tipo di header successivo, sia esso di livello rete (extension header IP) o di livello di trasporto (UDP,TCP). Il campo è codificato esattamente come il campo Protocol di IPv4. In questa tesi, non si considerano gli *extension header IPv6*, per cui anche questo campo viene considerato STATIC;
- *Hop Limit*: stesse considerazioni del campo IPv4 Time To Live;
- *Source address* e *Destination Address*: stesse considerazioni dei corrispondenti campi nell'header IPv4.

La tabella 4.3 riepiloga la dimensione totale dei campi di ciascuna classe. Rispetto a quanto osservato per IPv4 (tabella 2.3), si nota decisamente una migliore "predisposizione alla compressione" di IPv6 rispetto ad IPv4, in quanto i campi dinamici occupano solo 2 byte, a fronte di 36 byte di campi statici. Sono invece "corti" i campi ridondanti (solo 2 byte), per cui, così come osservato per IPv4, le fasi iniziali della compressione non possono fornire prestazioni particolarmente elevate.

Tabella 4.3 – Lunghezza totale dei campi dell'header di base IPv6 suddivisi in base alla classe di appartenenza

Classe	Totale bit
INFERRED	16
STATIC-KNOWN	12
STATIC-DEF	276
CHANGING	16

3.2.4 Campi dell'header TCP

L'header di un segmento TCP comincia sempre con una parte fissa da 20 byte ed una parte opzionale di lunghezza variabile (multipla di parole da 32 bit). La parte fissa ha la struttura riportata in figura 6.3.

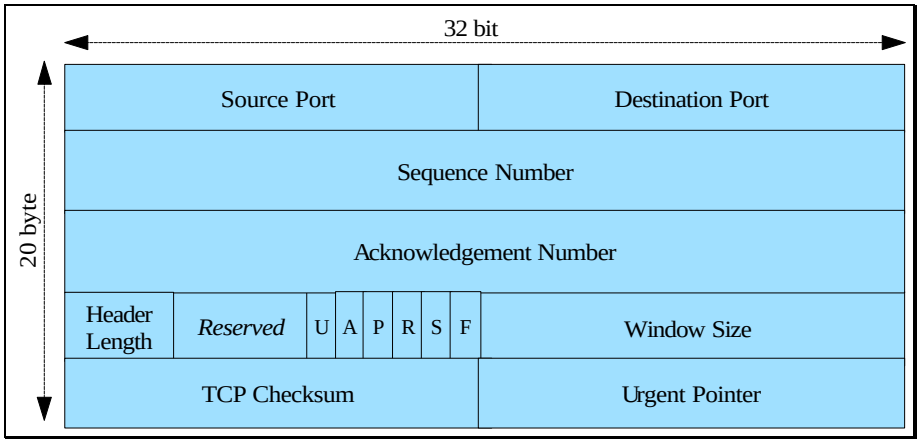

Figura 6.3 - Header TCP di base

Sono possibili segmenti senza payload, che sono usati per gli ACK ed i messaggi di controllo.

In questa tesi, si ritiene di poter classificare i vari campi, dal punto di vista della compressione, nel modo indicato nella tabella 5.3.

Tabella 5.3 – Classificazione dei campi dell'header TCP di base

Campo	Dimensione (bit)	Classe
Source Port	16	STATIC-DEF
Destination Port	16	STATIC-DEF
Header Length	4	STATIC-KNOWN
Reserved	4	INFERRED
Flag U,A,P,R,S,F	6	CHANGING
Window Size	16	CHANGING
TCP Checksum	16	CHANGING
Urgent Pointer	16	CHANGING
Sequence Number	32	CHANGING
ACK Number	32	CHANGING

Segue la descrizione dei vari campi, corredata dai motivi che hanno spinto ad adottare la classificazione appena citata:

- *Source Port* e *Destination Port*: identificano gli end point della connessione ([5]) e si mantengono perciò sempre costanti; essendo inoltre univoci per ciascuna connessione, sono classificati STATIC-DEF;
- *Header Length*: esprime la lunghezza dell'header TCP in termini di parole da 32 bit; è necessario in quanto possono essere presenti campi opzionali, che rendono variabile la dimensione complessiva dell'header; tuttavia, si adotta ancora una volta l'ipotesi per cui le informazioni circa la lunghezza dei pacchetti vengano fornite dai livelli inferiori, per cui il campo Header Length viene classificato STATIC-KNOWN;
- *Reserved*: comprende 6 bit non utilizzati, da cui appunto la classificazione come INFERRED;
- *Flag*: sono presenti 6 flag, tutti considerati di tipo CHANGING; essi hanno le seguenti funzioni:
    - *URG*: vale 1 se il campo *Urgent Pointer* viene usato e 0 altrimenti;
    - *ACK*: vale 1 se l'ACK number è valido (cioè se il segmento convoglia anche un ACK) e 0 altrimenti; se ACK=0, il segmento non contiene un ACK e quindi il campo ACK number viene ignorato;
    - *PSH*: se settato ad 1, indica la presenza di "dati urgenti", che cioè dovranno essere consegnati all'applicazione destinataria senza aspettare che il buffer TCP si riempia;
    - *RST*: se settato ad 1, indica la richiesta del mittente di reset della connessione, a seguito di "problemi" come ad esempio una connessione diventata instabile oppure l'arrivo di un segmento non valido o altro;
    - *SYN*: viene usato per creare una connessione TCP; la richiesta di connessione è caratterizzata da SYN=1 e ACK=0 (*segmento SYN*); la risposta a tale segmento è caratterizzata poi da SYN=1 e ACK=1 (*segmento SYN+ACK*);
    - *FIN*: viene usato per rilasciare una connessione, in quanto, se posto ad 1 (*segmento FIN*), specifica che il mittente non ha ulteriori dati da spedire;

[5] Assieme ai corrispondenti numeri IP, sono gli identificatori dei due estremi della connessione TCP

- *Window size*: contribuisce al controllo di flusso e indica la capacità residua del buffer di ingresso del mittente del segmento, al fine di rallentare il tasso di emissione dei pacchetti da parte del mittente. Viene classificato CHANGING;
- *TCP Checksum*: contiene la somma di controllo necessaria per verificare l'eventuale presenza di errori di trasmissione; il calcolo include l'intero segmento TCP ed anche uno "pseudoheader", costituito dagli indirizzi IP sorgente e destinazione, dal campo IP Protocol e da un campo recante la lunghezza complessiva del segmento TCP. Questo campo varia in modo impredicibile da pacchetto a pacchetto e svolge inoltre un ruolo fondamentale per la rilevazione degli errori, per cui è classificato di tipo CHANGING;
- *Urgent Pointer*: è un "puntatore ai dati urgenti" e viene usato per indicare uno scostamento in byte, a partire dal numero di sequenza attuale, dove possono essere trovati i dati considerati urgenti. Si tratta di un campo di tipo CHANGING, anche se le implementazioni attuali ne fanno un uso assolutamente saltuario;
- *Sequence Number*: contiene il numero di sequenza del primo byte contenuto nel payload del segmento ([6]); insieme al campo Acknowledgement Number descritto dopo, esercita la massima influenza sul rapporto di compressione. E' classificato di tipo CHANGING;
- *Acknowledgement Number*: contiene il numero di sequenza del prossimo byte atteso dal mittente del segmento ed è anch'esso classificato di tipo CHANGING.

La tabella 6.3 riepiloga la dimensione totale dei campi di ciascuna classe.

Tabella 6.3 – Lunghezza totale dei campi dell'header di base TCP suddivisi in base alla classe di appartenenza

Classe	Totale bit
INFERRED	4
STATIC-KNOWN	4
STATIC-DEF	32
CHANGING	118

[6] Si spiega così la proprietà del TCP secondo cui ogni byte del flusso TCP è numerato: in realtà, ciascun segmento TCP porta con sé il numero di sequenza del primo byte contenuto, per cui i numeri di sequenza dei byte successivi (appartenenti allo stesso segmento) sono dedotti di conseguenza.

3.2.5 Modelli di variazione dei campi dinamici TCP/IP

Al fine di progettare un efficiente meccanismo di compressione dell'header, bisogna analizzare dettagliatamente i "modelli di variazione" dei campi dei vari protocolli coinvolti. La maggiore attenzione va chiaramente dedicata ai campi classificati di tipo CHANGING, considerando che, invece, quelli di tipo STATIC possono essere trasmessi solo saltuariamente nel corso di una stessa connessione.

I campi di tipo CHANGING vengono ulteriormente classificati secondo le seguenti cinque sottoclassi:

- *STATIC*: questi campi sono definiti di tipo CHANGING in linea generale e adesso prendono l'appellativo STATIC per indicare che la loro variazione è costante nel corso di uno stesso flusso ([7]);
- *SEMI-STATIC*: questi campi sono quasi sempre di tipo STATIC; tuttavia, occasionalmente i loro modelli di variazione subiscono dei cambiamenti per poi riprendere i modelli originali dopo un certo numero di pacchetti;
- *RARELY-CHANGING*: questi campi cambiano valore, ma in modo abbastanza occasionale;
- *ALTERNATING*: questi campi cambiano valore in modo più o meno frequente, ma sempre nell'ambito di un numero ridotto di possibili valori;
- *IRREGULAR*: questi sono gli unici campi per i quali non risulta possibile individuare un determinato e utile modello di variazione.

Una volta effettuata la classificazione sulla base di queste 5 categorie, si può ulteriormente perfezionare l'analisi individuando altre informazioni utili alla compressione. Ad esempio, per i campi CHANGING classificati STATIC o SEMI-STATIC, l'incremento potrebbe o meno essere noto a priori. Inoltre, per i campi che non sono di tipo IRREGULAR, si potrebbe eventualmente individuare un range limitato di variazione, contenuto nel massimo range possibile (determinato dal numero di bit da cui è costituito il campo).

[7] Non bisogna confondere i campi CHANGING STATIC appena definiti con quelli STATIC definiti in precedenza: questi ultimi, infatti, non cambiano mai valore, al contrario degli altri, di cui si parlerà meglio più avanti.

La tabella 7.3 applica tutte queste considerazioni ([8]). Nei prossimi paragrafi verranno meglio giustificate le informazioni riportate nella tabella e si procederà ad illustrare i metodi di trasmissione ritenuti ottimali, nel profilo qui proposto, per ciascun campo.

Tabella 7.3 – Riepilogo della sotto-classificazione dei campi CHANGING, con dettaglio sui modelli di variazione previsti

Campo	Codifica (Valore/Delta)	Sottoclasse	Modello di variazione
Ipv4 Identification			
Sequenziale	Delta	STATIC	Noto
Sequenziale con salti	Delta	RARELY-CHANGING	Limitato
Casuale	Valore	IRREGULAR	Impredicibile
Ipv4 TOS / Ipv6 Traffic Class	Valore	RARELY-CHANGING	Impredicibile
Ipv4 TimeToLive / Ipv6 HopLimit	Valore	ALTERNATING	Limitato
TCP Checksum	Valore	IRREGULAR	Impredicibile
TCP Rohc Sequence Number	Delta	STATIC	Noto
TCP Urgent Pointer	Valore	RARELY-CHANGING	Impredicibile
TCP Window Size	Delta	RARELY-CHANGING	Limitato
TCP Sequence Number	Delta	SEMI-STATIC	Limitato
TCP ACK Number	Delta	SEMI-STATIC	Limitato
TCP Flag U,A,P,R,S,F	Valore	RARELY-CHANGING	Limitato

La seconda colonna della tabella 7.3 indica se sia necessario trasmettere un dato campo a valore pieno (*valore*) oppure codificando in qualche modo le sue variazioni (*delta*).

Campo IPv4 Identification

Il campo *IPv4 Identification* (d'ora in poi **ID**) merita una serie di considerazioni aggiuntive rispetto agli altri campi dell'header TCP/IP.

Questo campo viene usato per identificare quali frammenti costituiscono un datagramma, in modo che in ricezione quest'ultimo possa essere correttamente riassemblato. Le specifiche di IPv4 non indicano però con esattezza come assegnare valori a questo campo, il che rappresenta ovviamente una complicazione per quanto riguarda l'implementazione del processo di compressione.

[8] Si fa notare che la tabella non considera variazioni dovute a *perdite* o *riordino di pacchetti* prima del punto di compressione.

Il ROHC Workgroup ha allora considerato tre diverse modalità di assegnazione dei valori al campo ID:

- *modo sequenziale*: i valori di ID vengono assegnati in base ad un contatore diverso per ciascun flusso di pacchetti in uscita dalla sorgente, in modo che, all'interno di uno stesso flusso, il campo venga incrementato di uno per ogni pacchetto in uscita. In questo modo, la variazione di ID è costante e nota a priori (*delta=1*);
- *modo sequential jump*: come nel caso precedente, ma con la differenza di usare un unico contatore per tutte le connessioni; questo comporta che, quando la sorgente sta gestendo più di un flusso contemporaneamente, il valore del campo ID possa subire variazioni, da un pacchetto al successivo nello stesso flusso, maggiori di uno: l'incremento sarà determinato, in generale, dal numero di flussi in uscita e dalla frequenza di emissione dei pacchetti di ciascun flusso ed assumerà valori in un intervallo tendenzialmente ristretto di possibilità;
- *modo random*: in quest'ultimo caso, la sorgente assegna i valori del campo ID usando un generatore di numeri pseudo-casuale.

Appare evidente che, con il *modo random*, non esiste correlazione tra valori relativi a pacchetti consecutivi e quindi non si può far altro che trasmettere il campo senza compressione, occupando stabilmente 2 byte nell'header del pacchetto compresso. Invece, relativamente alla politica sequenziale di assegnazione, si ritiene opportuno fare una serie di considerazioni:

- al fine di garantire sufficiente sicurezza, in presenza di più flussi che avvengono tra la stessa coppia di nodi e che usano lo stesso protocollo di trasporto (ad esempio TCP), è opportuno che pacchetti appartenenti a flussi diversi abbiano anche valori sempre diversi del campo ID. Allora, se si vuole garantire comunque questo requisito e, allo stesso tempo, si vuole ancora conservare l'assegnazione sequenziale di ID, è necessario fare in modo che i diversi flussi usino sottoinsiemi diversi di valori di ID, in modo da evitare "collisioni". E' possibile pensare a varie soluzioni implementative, ad esempio introducendo "salti" (jump) occasionali della numerazione, col risultato di rendere ovviamente la politica di assegnazione non più perfettamente sequenziale. Pur non essendo particolarmente vantaggiosa ai fini della compressione dell'header, questa

scelta appare comunque un buon compromesso tra le esigenze di sicurezza e quelle di riduzione dell'overhead;

- in generale, comunque, è opportuno che il compressore sappia a priori quale politica di assegnazione di ID venga usata dalla sorgente IP e possa scegliere la tecnica di compressione ottimale. In caso contrario, se cioè il compressore dovesse prevedere un'unica tecnica di codifica, l'eventuale "disadattamento" tra essa e la politica di assegnazione potrebbe determinare un overhead persino superiore ai due byte per la trasmissione del campo as-is;
- i "problemi" legati al campo ID scompaiono se si usa IPv6, in quanto questa versione di IP non prevede un simile campo. Si ritiene che questo sia un ulteriore punto a favore di IPv6 rispetto ad IPv4.

3.3 Strategie di compressione per il profilo TCP/IP

Sulla base delle considerazioni dei paragrafi precedenti, si può procedere a stabilire i metodi con cui codificare e trasmettere i singoli campi degli header TCP e IP. In questa tesi sono state individuate sei diverse possibilità, di seguito descritte.

Nessun invio

I campi che presentano valori costanti e ben noti a priori (campi di tipo STATIC-KNOWN) non devono essere mai inviati. Analogo discorso vale per i campi ridondanti (INFERRED). In totale, quindi, non devono mai essere trasmessi i seguenti campi:

```
- IPv6 Payload Length
- IPv4 Header Length
- IPv4 Reserved Flag
- IPv4 More Fragment (MF) Flag
- IPv4 Fragment Offset
- TCP Header Length
- TCP Reserved
- IPv4 Header Checksum
- IPv4 Total Length
```

Trasmissione solo all'inizio

I campi che contengono valori costanti (ma non noti a priori) durante la vita di un flusso di pacchetti devono essere inviati, almeno teoricamente, solo una volta (campi STATIC), con il primo pacchetto della connessione. Si tratta dei seguenti campi:

- IPv6 Version
- IPv6 Source Address
- IPv6 Destination Address
- IPv6 Flow Label
- IPv4 Version
- IPv4 Source Address
- IPv4 Destination Address
- IPv4 Don't Fragment (DF) Flag
- IPv4 Network Byte Order (NBO) Flag
- IPv4 Random (RND) Flag
- TCP Source Port
- TCP Destination Port

Trasmissione solo iniziale, con possibilità di aggiornamento

I campi che cambiano solo occasionalmente in uno stesso flusso (campi RARELY-CHANGING) devono essere trasmessi inizialmente e deve inoltre essere previsto un modo efficiente per aggiornarli in caso di variazioni, trasmettendoli a valore pieno, eventualmente insieme alle relative variazioni.

I campi di questo tipo sono i seguenti:

- IPv6 Next Header
- IPv6 Traffic Class
- IPv6 Hop Limit
- IPv4 Protocol
- IPv4 Type Of Service (TOS)
- IPv4 Time To Live (TTL)
- TCP Window Size
- TCP Urgent Pointer
- TCP flag URG, ACK, PSH, RST, SYN, FIN

Tecniche di aggiornamento e invio as-is

I campi critici, dal punto di vista della compressione, sono quelli che variano sempre, in modo più o meno regolare, tra un pacchetto e l'altro di uno stesso flusso (CAMPI CHANGING di tipo SEMI-STATIC): per essi si deve prevedere un efficiente meccanismo di aggiornamento (codificando opportunamente e trasmettendo solo le variazioni) e ci si deve anche preparare ad inviarli così come sono (*as-is*) in talune condizioni. Si tratta dei seguenti campi:

- `TCP Sequence Number`
- `TCP ACK Number`
- `IPv4 Identification` `(sequenziale con contatore unico)`

E' importante osservare una cosa: il campo TCP Sequence Number varia da pacchetto a pacchetto, ma il "delta" non necessariamente è costante, essendo vincolato alla dimensione dei segmenti TCP generati dal mittente ed alle ritrasmissioni; questo complica la codifica di tale campo e suggerisce di includere il "delta" nel context del decompressore, considerandolo come campo di tipo RARELY CHANGING: esso deve essere inizializzato all'inizio del flusso e deve essere aggiornato, in modo efficiente, in presenza di variazioni. Analoghe considerazioni valgono anche per il campo TCP ACK Number, le cui variazioni sono ancora più "instabili", essendo legate non solo alle variazioni del campo Sequence Number dei segmenti ricevuti, ma anche all'ordine con cui tali segmenti arrivano, alle perdite ed ai meccanismi usati dal ricevente per la generazione degli ACK.

Trasmissione continua

I campi più facili da comprimere sono quelli che si comportano come dei contatori (campi CHANGING di tipo STATIC), nel senso che hanno delle variazioni fisse (tipicamente unitarie) da pacchetto a pacchetto. In questo caso, infatti, l'unico requisito da garantire è che il decompressore sia robusto nei confronti delle eventuali perdite di pacchetti, ottenendo allo stesso tempo un buon livello di compressione.

Nel profilo proposto in questa tesi, ci sono due campi appartenenti a questa categoria:

- IPv4 Identification (non sequenziale)
- ROHC Sequence Number

Il campo **ROHC Sequence Number** viene appositamente introdotto dal profilo di compressione proposto in questa tesi, come si vedrà nel prossimo capitolo, al fine di raggiungere un elevato rapporto di compressione. Esso viene gestito interamente dal compressore, il quale lo incrementa di 1 per ogni pacchetto ricevuto dalla sorgente IP nell'ambito di ciascun flusso.

La stessa regolarità si ha anche per il campo IPv4 Identification, a patto che la sorgente IP utilizzi un contatore separato per ciascun flusso attivo. Se invece la sorgente IP usa un unico contatore per tutti i flussi, allora, in presenza di più flussi, sono molto più probabili i periodi in cui la regolarità è assente (variazione non costante e maggiore di 1): in questi casi, non si può codificare il campo come un contatore, per cui valgono le stesse considerazioni del paragrafo precedente.

Trasmissione integrale in tutti i pacchetti

I campi che assumono valori del tutto casuali in ciascun pacchetto (campi CHANGING di tipo IRREGULAR) devono necessariamente essere trasmessi così come sono in tutti i pacchetti, determinando un inevitabile degradazione del rapporto di compressione.

Si tratta dei seguenti campi:

- IPv4 Identification (random)
- TCP Checksum

Tali campi influiscono chiaramente in modo "pesante" sull'efficienza di compressione ottenibile, imponendo un overhead medio costante pari alla loro lunghezza (2 byte).