Algoritmo di compressione ROHC per connessioni TCP

di Sandro Petrizzelli

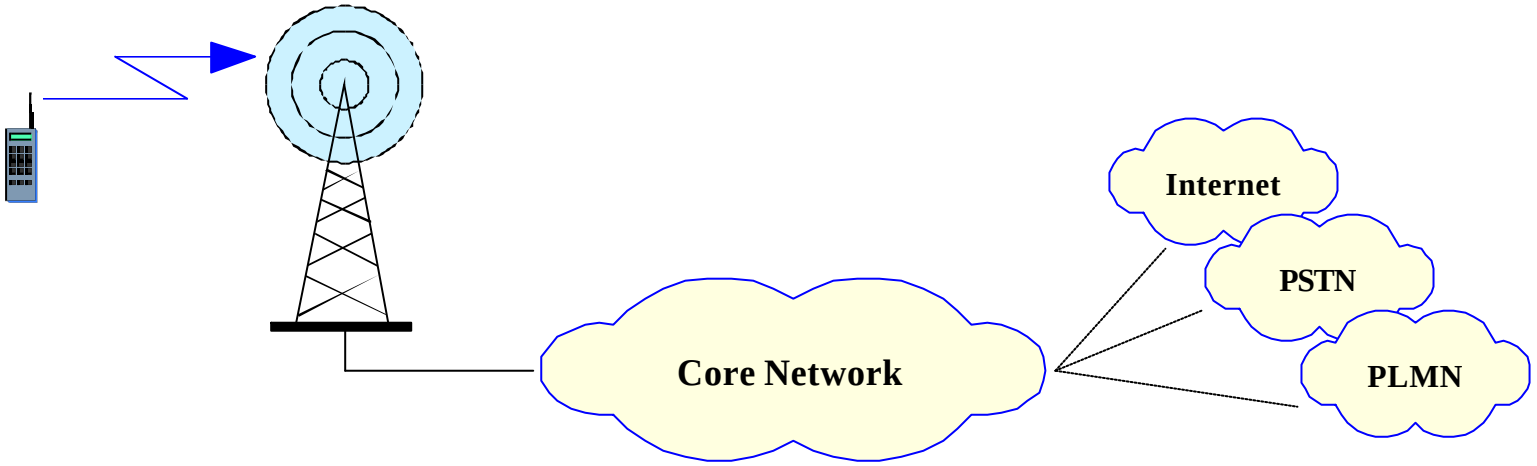
(sandry@iol.it)

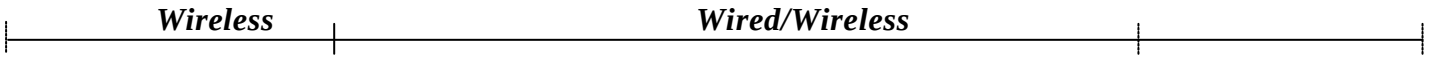
Politecnico di Bari

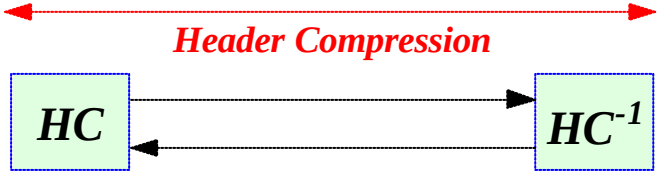
Dip. Elettronica ed Elettrotecnica

Compressione dell'header

Scenario tipico per una rete radiomobile (solo uplink)







Il **punto di decompressione** deve trovarsi non oltre il primo nodo di rete che si occupa dell'instradamento dei pacchetti IP

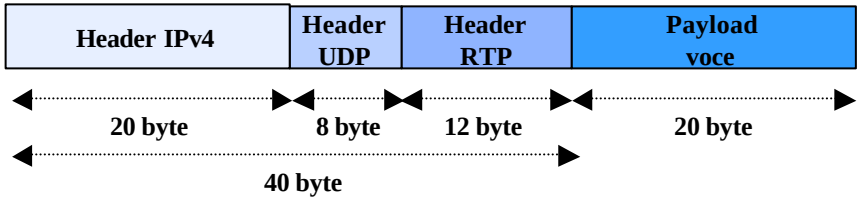
Compressione dell'header

Perché comprimere l'header?

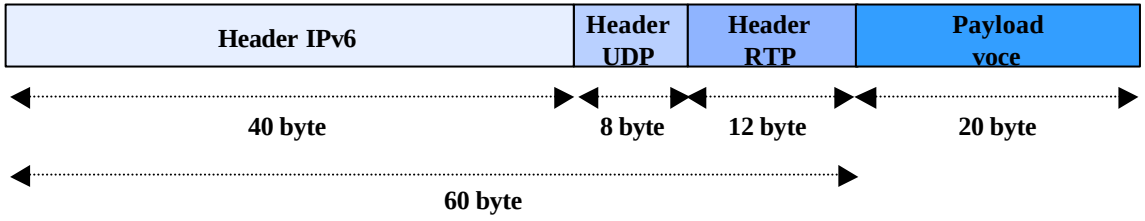
1. **Per risparmiare banda**

Esempio di servizio voce su IP (f_s=50 frame/s, coding: 8 kbit/s)

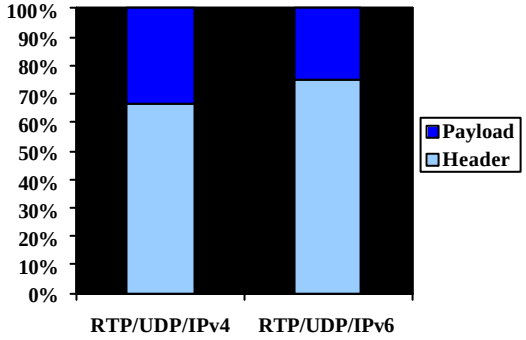
Stack RTP/UDP/IPv4



Stack RTP/UDP/IPv6



Distribuzione della banda



Compressione dell'header

Perché comprimere l'header?

2. Per migliorare il FER (Frame Error Rate)

A parità di caratteristiche di rumore (BER - Bit Error Rate), un canale che utilizza la compressione dell'header trasmette meno bit e quindi presenta un tasso d'errore sui pacchetti inferiore.

$$BER ? x$$

$$FER ? (P ? H) \cdot 8 \cdot BER$$

$$FER_C ? (P ? C) \cdot 8 \cdot BER$$

$$? FER\% ? \frac{FER ? FER_C}{FER} ? \frac{H ? C}{H ? P}$$

P il payload (in byte);

H la dimensione dell' header (in byte);

C la dimensione dell' header compresso (in byte);

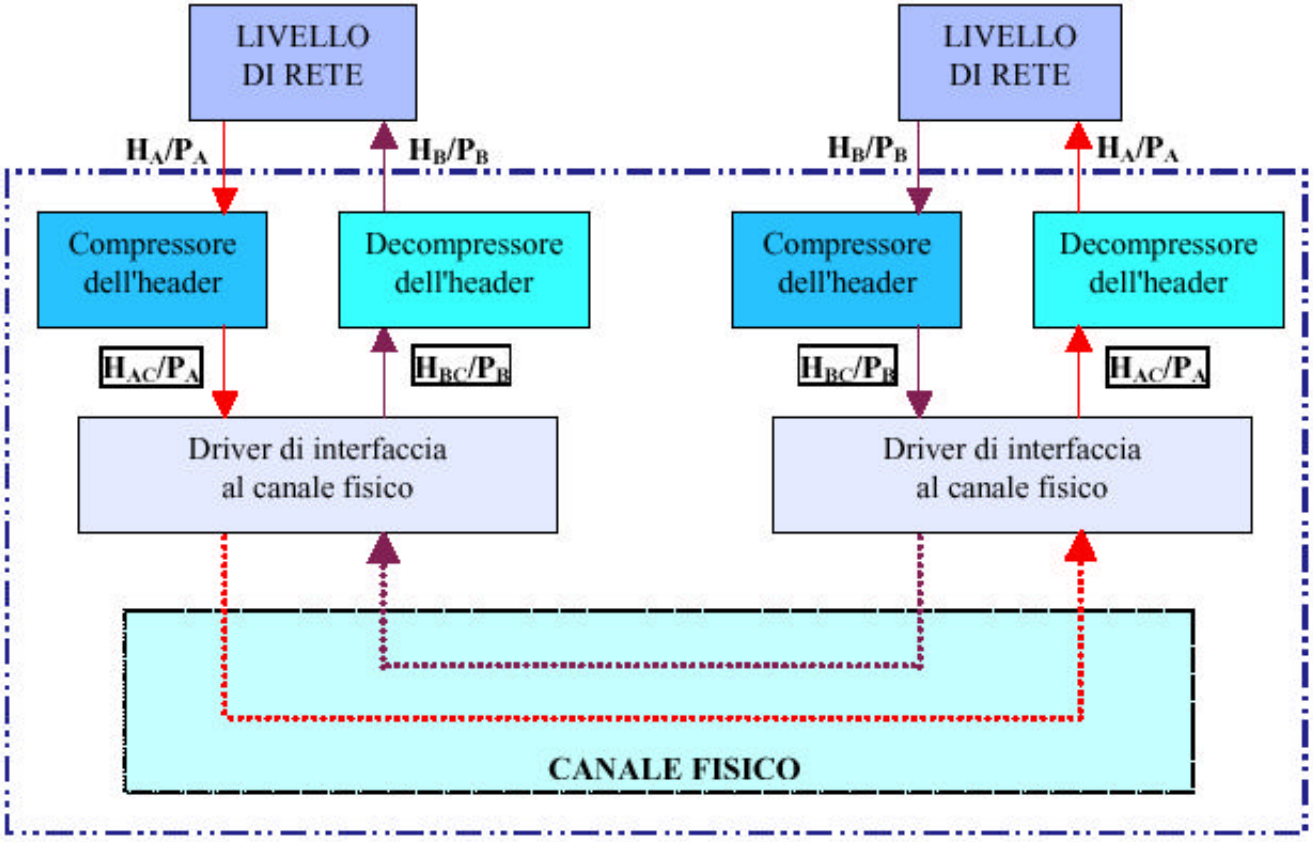
FER ? (P ? H) ·8 ·BER;

FER_C è il FER in presenza di compressio ne dell' header;

? FER% è il migliorame nto del FER.

Compressione dell'header

Architettura TCP/IP con unità di compressione/decompressione



P_A=Payload generato dal processo A

H_A=Pila di header fino al livello rete

H_A/ P_A=pacchetto di livello rete

H_{AC}/ P_A=pacchetto con header compresso, da inviare sul canale

Delimita la sezione entro cui gli header, fino al livello rete, non esplicano alcuna funzione; è la "regione di comprimibilità" dell'header

Compressione dell'header

Schemi di compressione precedenti il ROHC

- **V. Jacobson, "Compressing TCP/IP headers for low-speed serial links", RFC 1144, February 1990.**
- **M. Degermark, B. Nordgren, and S. Pink, "IP Header Compression", RFC 2507, February 1999.**
- **S. Casner, V. Jacobson, "Compressing IP/UDP/RTP Headers for low-speed serial links", RFC 2508, February 1999.**

Questi algoritmi di compressione dell'header non risultano di valida applicazione ad un canale radio per due ragioni principali:

1. il **BER** (Bit Error Rate) di un canale radio è di diversi ordini di grandezza superiore a quello di un cavo;
2. il **RTT** (Round Trip Time) è di circa 200ms.

Algoritmo ROHC
Riferimenti principali

- ROHC – RObust Header Compression

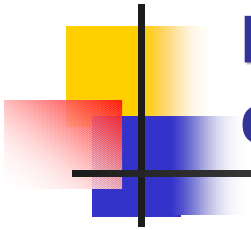
- Responsabili del progetto in IETF (Robust Header Compression Working Group):
  - Carsten Bormann (cabo@tzi.org)
  - Mikael Degermark (micke@cs.arizona.edu)

- Riferimenti:
  - **RFC 3095** (luglio 2001) – ROHC: Framework and four profiles (RTP, UDP, ESP, and uncompressed)
  - **RFC 3096** (luglio 2001) - Requirements for robust IP/UDP/RTP header compression
  - draft-ietf-rohc-over-xxx-yy.txt (xxx=ppp, yy=02)

  - Workgroup IETF: http://www.ietf.org/html.charters/rohc-charter.html
  - Mailing list: http://www.cdt.luth.se/rohc/

Algoritmo ROHC

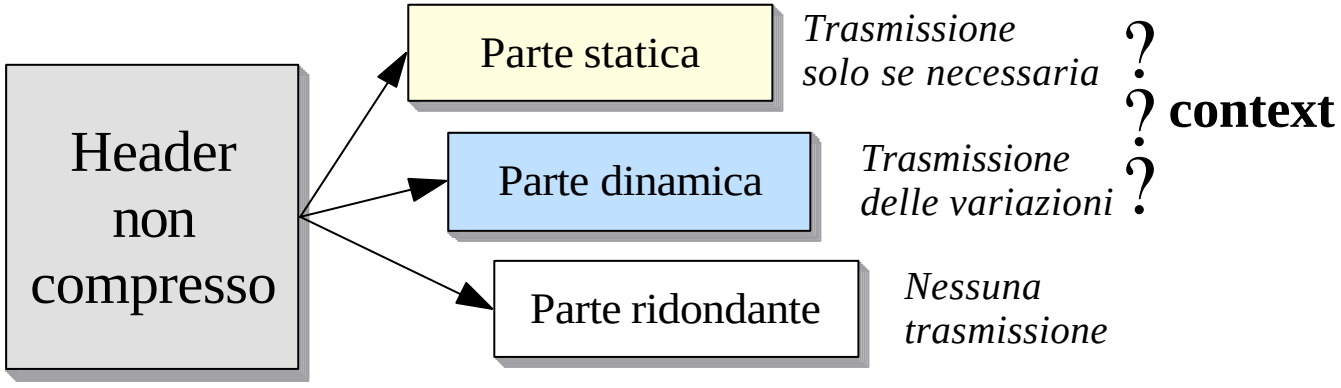
Requisiti fondamentali per uno schema di compressione dell'header



- **Efficienza:** minima dimensione dei pacchetti trasmessi
  - minima propagazione delle perdite introdotta dalle procedure di decompressione;
  - minimo ritardo di computazione introdotto dalle procedure di compressione/decompressione
- **Robustezza** nei confronti delle perdite (specialmente su link con elevato BER ed elevato RTT)
  - Idealmente, dovrebbe essere possibile effettuare la compressione su link unidirezionali
- **Decompressione coerente**: quando un header viene compresso e poi decompresso, il risultato deve essere semanticamente uguale all'originale, altrimenti il pacchetto deve essere scartato;
- **Flessibilità**: capacità di gestire qualsiasi pila protocollare, anche futura, basata sul modello TCP/IP, con particolare attenzione ai servizi offerti dal sistema UMTS. Quindi deve essere in grado di gestire:
  - *IPv4* e *IPv6*:
  - *Mobile IP*: tutte le problematiche relative alla mobilità;
  - *Riordinamento dei pacchetti*: i pacchetti potrebbero giungere al decompressore in ordine diverso da quello di trasmissione;
  - *Frammentazione dei pacchetti*: la trasmissione di pacchetti troppo grandi può comportare la monopolizzazione del canale, con conseguente incremento dei livelli di congestione e degradazione della qualità dei servizi. La frammentazione peggiora l'overhead, per cui bisogna introdurre un ulteriore protocollo di frammentazione efficiente a livello del compressore.

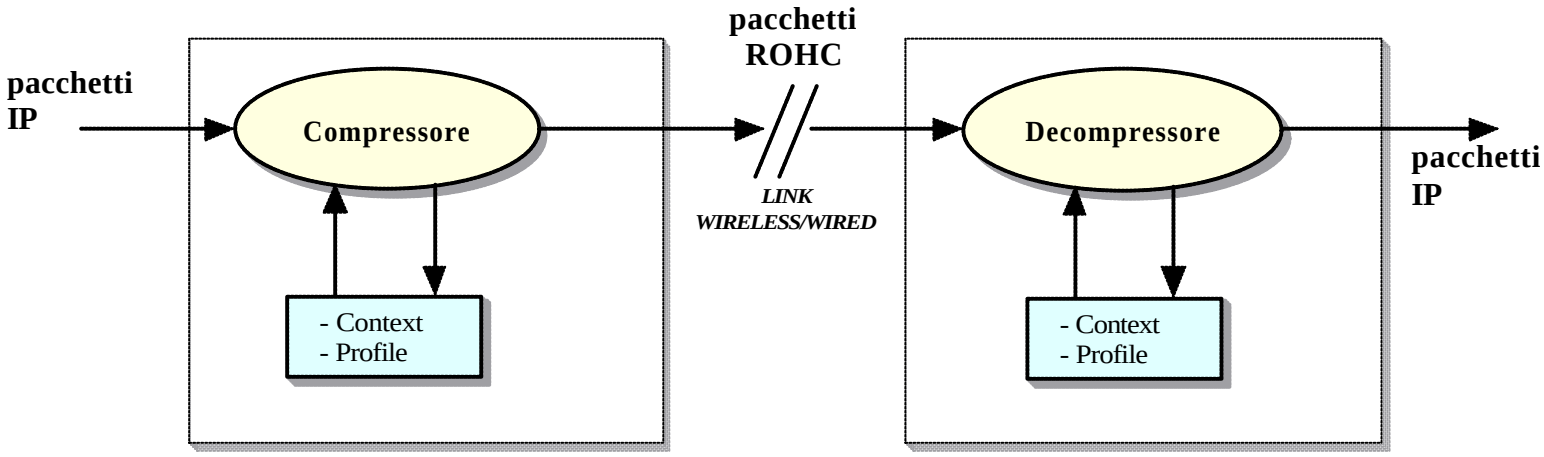
Algoritmo ROHC
Generalità

- 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, ma privilegia, nella sua versione originale, i flussi RTP/UDP/IP.
- La compressione sfrutta la ridondanza esistente tra i campi dell'header sia nel singolo pacchetto sia soprattutto tra pacchetti dello stesso flusso
- Ai fini della compressione, ognuno degli header gestiti dall'algoritmo viene suddiviso in tre parti: statica, dinamica, ridondante



Algoritmo ROHC
Sezioni di compressione e decompressione

Il **context** contiene l'informazione utile alla compressione dell'header (sezione di trasmissione) ed alla sua corretta decompressione (sezione di ricezione)



- 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 (flusso)

- Ogni context è identificato da un numero detto **CID** (Context IDentifier) che gli viene associato durante la fase di negoziazione della connessione.

Algoritmo ROHC

Profili di compressione nella versione iniziale

- **Profilo 1 - RTP/UDP/IP**
  Per la compressione di flussi di tipo RTP/UDP/IP.

- **Profilo 2 - UDP/IP**
  Per la compressione di flussi di tipo UDP/IP, ossia per tutti quei flussi che non utilizzano il protocollo RTP o che non si desidera vengano gestiti dal profilo precedente.

- **Profilo 3 - ESP/IP**
  Per la compressione dei flussi protetti mediante il protocollo di sicurezza dei dati ESP (Encapsulating Security Payload).

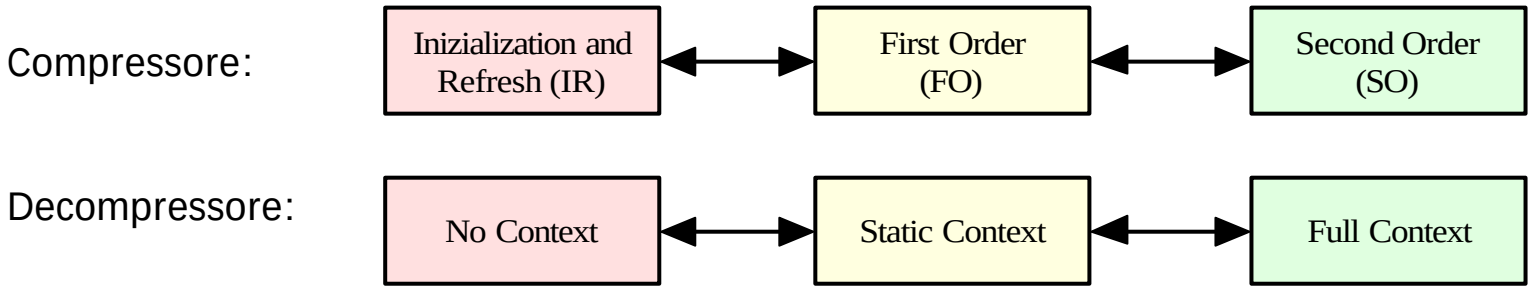
- **Profilo 0 - Flussi non compressi**
  Per la gestione di tutti gli altri flussi, compresi i flussi che utilizzano TCP a livello di trasporto.

Algoritmo ROHC
Principi cardine per la compressione ROHC

- **Campi dinamici**
  - La maggior parte dei campi dell'header può essere compressa in quanto essi, nell'ambito di uno stesso flusso, non cambiano mai oppure cambiano solo raramente da pacchetto a pacchetto
  - Solo pochi campi necessitano di meccanismi di codifica più complicati
- **Campo "leader"**
  - In ogni flusso deve esistere un campo dinamico, detto **campo leader**, sulla base del quale ottimizzare la compressione
  - Esso è caratterizzato da una variazione costante fra pacchetti consecutivi ($?x=1$)
- **"Regolarizzazione" del flusso**
  - Il compressore monitora il campo leader ed i campi dinamici ad esso connessi
  - Quando le variazioni dei campi dinamici, tra pacchetti consecutivi, risultano proporzionali alle variazioni del campo leader, il flusso si dice **regolarizzato** e si invia soltanto la variazione del campo leader.
  - Così si ottimizzano sia l'**efficienza** di compressione sia la **robustezza**

Algoritmo ROHC
Macchine a stati

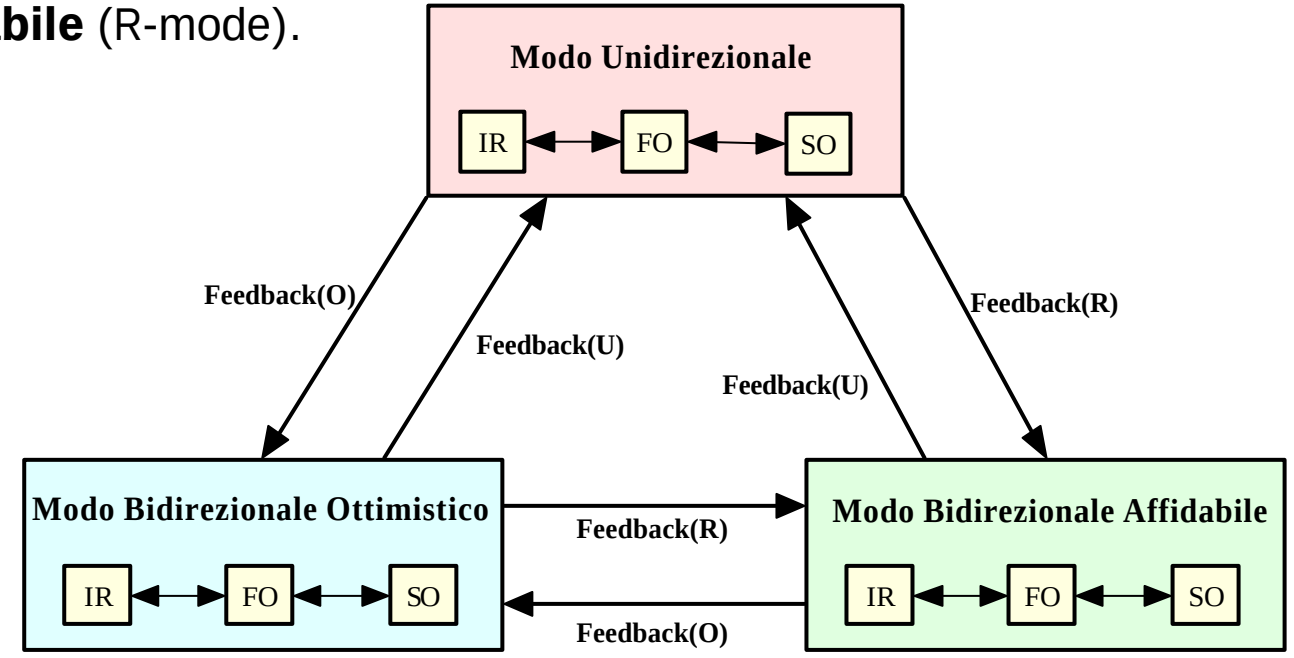
- La compressione dell'header con l'algoritmo ROHC può essere vista come l'interazione tra due **macchine a stati**, ciascuna con 3 stati



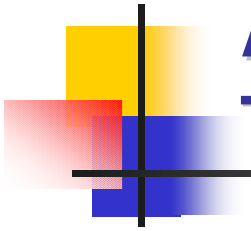
- Entrambe le macchine partono sempre nello stato più "basso" di funzionamento (minima compressione) e gradualmente si spostano verso gli stati a maggiore compressione.
- Le transizioni di stato nelle due macchine non necessariamente sono sincronizzate.
  - Il compressore tende a permanere nello stato più "alto", ma è costretto a transitare temporaneamente verso stati inferiori in corrispondenza di determinati *eventi*.
  - Il decompressore ha transizioni verso stati inferiori solo in presenza di errori nel context. In *condizioni ideali*, rimane sempre nello stato più "alto"

Algoritmo ROHC
Modalità operative

- **Unidirezionale** (U-mode)
- **Bidirezionale Ottimistico** (O-mode)
- **Bidirezionale Affidabile** (R-mode).



Ciascun modo operativo contempla tutti gli stati operativi e regola in modo diverso le transizioni tra uno stato e l'altro nonché le operazioni da compiere in ciascuno stato.



Algoritmo ROHC
Transizioni tra le modalità operative

- La compressione comincia sempre in modo unidirezionale. Eventuali **transizioni** ad uno dei modi bidirezionali sono possibili solo quando il decompressore dà la sua "disponibilità".
- Il decompressore decide qual'è la modalità operativa più idonea a soddisfare i requisiti richiesti dall'applicazione, sulla base di:
  - fattori di tipo *deterministico*: disponibilità di un canale di feedback, rapporto di compressione desiderato, ecc.;
  - fattori di tipo *aleatorio*: livello di congestione della rete, rumorosità del canale, ecc.
- Il progetto di questa parte del software di compressione deve basarsi su una analisi dettagliata delle caratteristiche dei **servizi** offerti dal sistema di telecomunicazioni preso in considerazione (GSM, GPRS, UMTS, Internet, altro): tale analisi deve portare a stabilire delle **soglie** dei parametri fisici che determinano la scelta della modalità più idonea

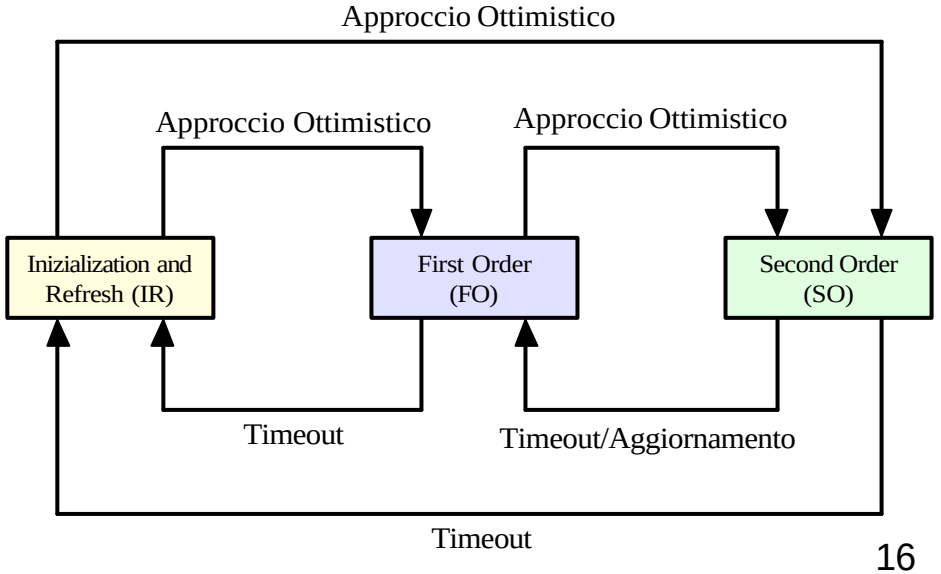
Algoritmo ROHC
Modo unidirezionale: stati di compressione

- Le transizioni all'indietro sono ottenute solo alla scadenza dei periodici timeout (aggiornamenti) nonché in presenza di irregolarità nel flusso
- La compressione risulta meno efficiente (timeout di aggiornamento) ed ha una probabilità di propagazione delle perdite più alta (mancanza di feedback) rispetto ai due modi bidirezionali

Approccio ottimistico: il compressore "transita" in avanti solo quando è sufficientemente sicuro che il decompressore abbia ricevuto le informazioni necessarie per decomprimere un pacchetto con maggiore compressione

Timeout: periodicamente il compressore deve aggiornare lo stato del decompressore, per prevenire perdite di sincronizzazione

- Slow start
- Header Refresh

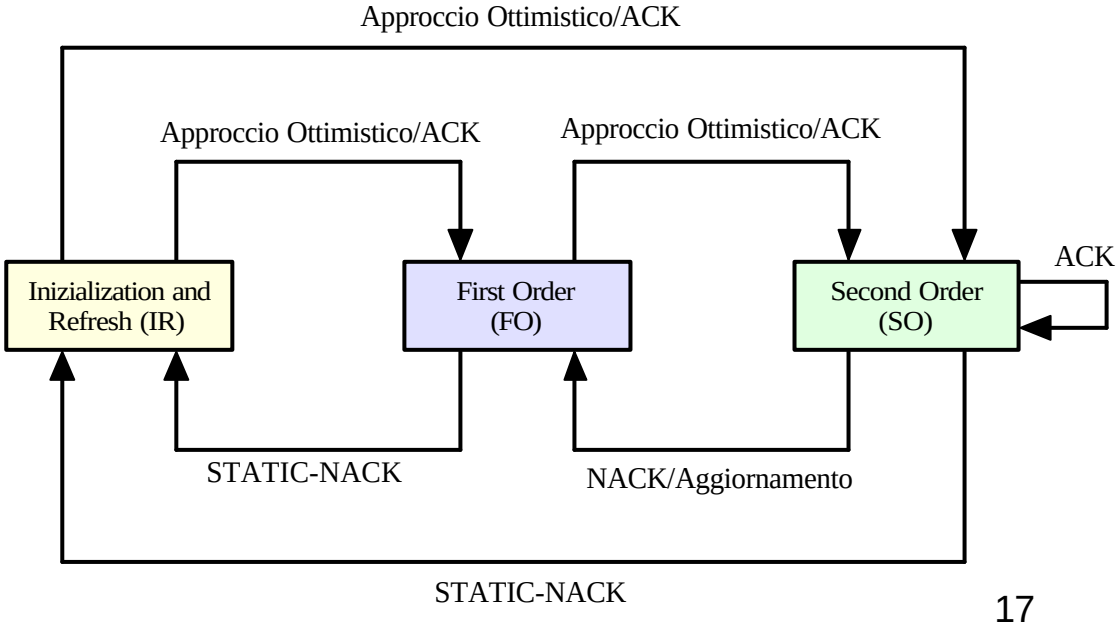


Algoritmo ROHC
Modo ottimistico: stati di compressione

- Il canale di feedback serve per richieste di aggiornamento del context e (opzionalmente) per ACK o NACK di avvenuto aggiornamento (ottimizzazione dell'approccio ottimistico)
- Non ci sono aggiornamenti periodici del context (timeout), data la presenza del feedback

- *Transizioni in avanti*: certezza del riferimento (ACK) e approccio ottimistico
- *Transizioni all'indietro*: aggiornamenti (irregolarità nel flusso) e NACK

L'intento è di massimizzare l'**efficienza di compressione** con un uso solo saltuario del canale di feedback

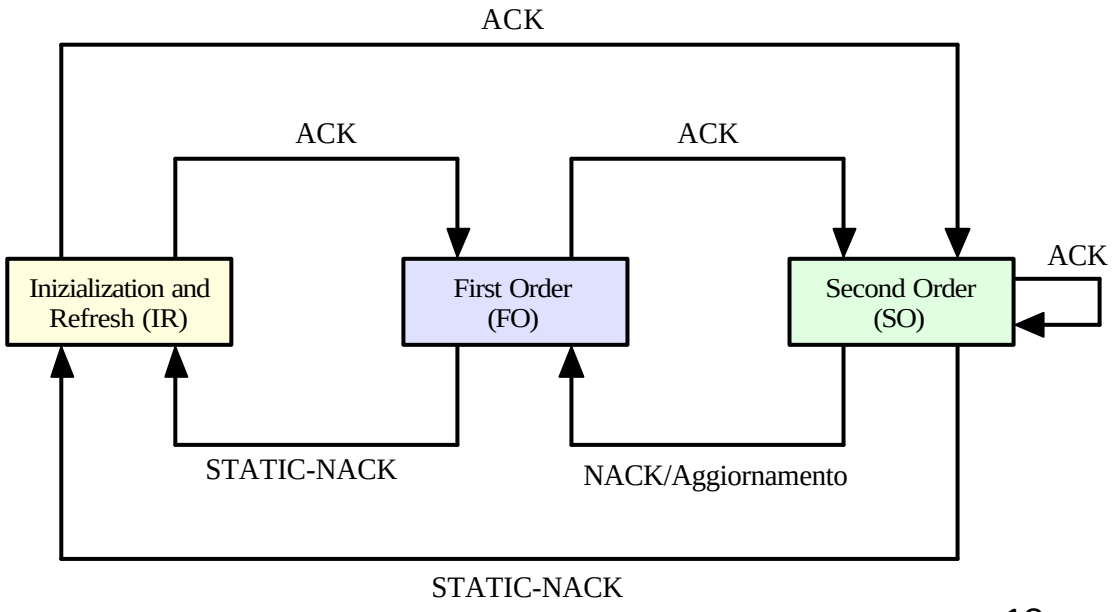


Algoritmo ROHC
Modo affidabile: stati di compressione

- Uso più intensivo, rispetto al modo ottimistico, del canale di feedback
- Le informazioni di feedback sono inviate per confermare tutti gli aggiornamenti del context e tutti gli aggiornamenti del campo riportante il numero di sequenza ROHC.

- *Transizioni in avanti*: certezza del riferimento (ACK)
- *Transizioni all'indietro*: aggiornamenti (irregolarità nel flusso) e NACK

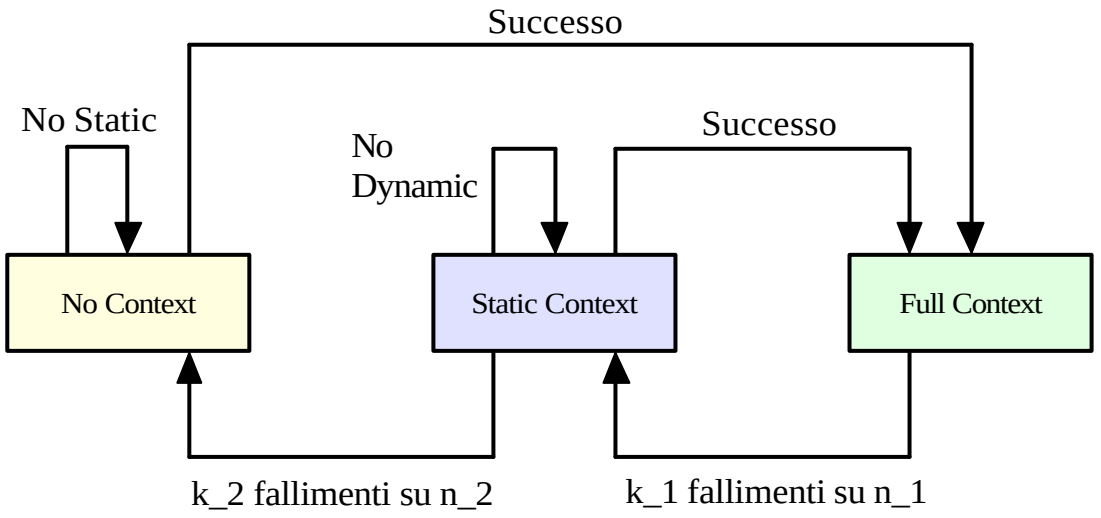
L'intento è di massimizzare la **robustezza** dell'algoritmo nei confronti della propagazione delle perdite e degli errori.

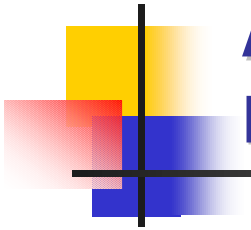


Algoritmo ROHC
Stati di decompressione (modi U, O e R)

- Una decompressione conclusa con successo comporta sempre un passaggio del compressore nello stato Full Context.
- Una serie ripetuta di decompressioni sbagliate comporta una transizione all'indietro.
- Il decompressore tenta una decompressione solo quando l'insieme delle informazioni contenuto nel pacchetto arrivato e nel context lo consentono

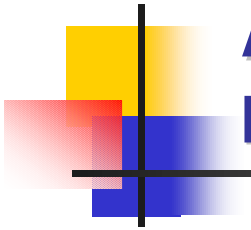
N.B. Gli stati di decompressione e la logica delle transizioni di stato sono gli stessi per tutte le modalità operative. Le differenze sono invece rappresentate nella logica di decompressione da adottare e nella logica del feedback da inviare.





Algoritmo ROHC
Modalità di feedback

- L'algoritmo di compressione necessita di periodici pacchetti di riscontro (**feedback**) dal decompressore, inviati con frequenza variabile a seconda della modalità operativa
- Per non peggiorare eccessivamente la richiesta di banda, i pacchetti di feedback non possono essere inviati come autonomi (dotati di proprio header)
- Di conseguenza, si ritiene utile adottare un **canale di feedback virtuale** in cui le informazioni di feedback vengano "accodate" ai pacchetti di forward (**piggybacking**)
- Questa soluzione ottimizza l'uso della banda, ma peggiora l'affidabilità (rispetto ad esempio ad un *canale di feedback dedicato*)



Algoritmo ROHC
Parametri di stato

Per garantire sufficiente insensibilità alle escursioni parametriche del canale radio (**robustezza**) e elevata compatibilità tra una vasta gamma di terminali radiomobili (**flessibilità**), il ROHC introduce un set di *parametri di stato* che caratterizzano il canale e il terminale radiomobile:

- **MAX_CID**: massimo valore del CID
- **LARGE_CIDS** : uso di large CID (7 o 14 bit) o small CID (0 o 4 bit)
- **PROFILES** : elenco profili supportati dal decompressore
- **FEEDBACK_FOR** : tipo di feedback adottato (canale virtuale con piggybacking)
- **SDU** (*Service Data Unit*) : massima dimensione pacchetto in uscita dal decompressore
- **MRRU** (*Maximum Reconstructed Reception Unit)*

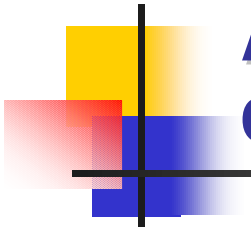
Il valore di tali parametri deve essere negoziato prima di iniziare una connessione. Inoltre, si ritiene utile poter ri-negoziare gli ultimi tre nel corso della connessione

Algoritmo ROHC
Profilo di compressione oggetto delle simulazioni

- Profilo 4 - TCP/IP

  Per la compressione di flussi di tipo TCP/IP, introdotto dall' ing. Vito Squeo

- **Requisiti:**
  - La logica di compressione ed i formati dei pacchetti ROHC devono seguire le linee guida dello schema ROHC originale (profilo RTP/UDP/IP)
  - Il profilo di compressione si deve adeguare alle caratteristiche dei flussi TCP
  - Il profilo di compressione deve ottemperare ai requisiti fondamentali per un algoritmo di compressione dell'header: efficienza, robustezza, flessibilità, coerenza della decompressione



Algoritmo ROHC
Caratteristiche distintive dei flussi TCP

- Il protocollo TCP fornisce un **servizio orientato alla connessione ed affidabile** (e quindi bidirezionale)
- L'instaurazione di una connessione avviene tramite lo schema **three-way handshake**
- Il TCP mittente suddivide il flusso dei dati in ingresso in **segmenti** e li invia al ricevente corredati ciascuno di un **numero di sequenza** ed altre informazioni di controllo (header)
- Il TCP ricevente, nella versione più semplice, invia al mittente un **segmento ACK** per ogni segmento dati correttamente ricevuto, corredandolo del numero di sequenza del prossimo byte del segmento che aspetta di ricevere
- Tramite opportuni **timeout**, il mittente si preoccupa di inviare nuovamente i segmenti per i quali non ha ricevuto riscontro. La durata del timeout e la modalità di ritrasmissione vengono stabilite tramite un **algoritmo di controllo della congestione**
- Le versioni del TCP attualmente utilizzate prevedono "perfezionamenti" più o meno complessi alle caratteristiche di base, con particolare riferimento al controllo della congestione (Slow Start, Fast Retransmit, ...) ed ai meccanismi di ritrasmissione (il Selective ACK, ...)

Algoritmo ROHC
Caratteristiche del profilo TCP/IP

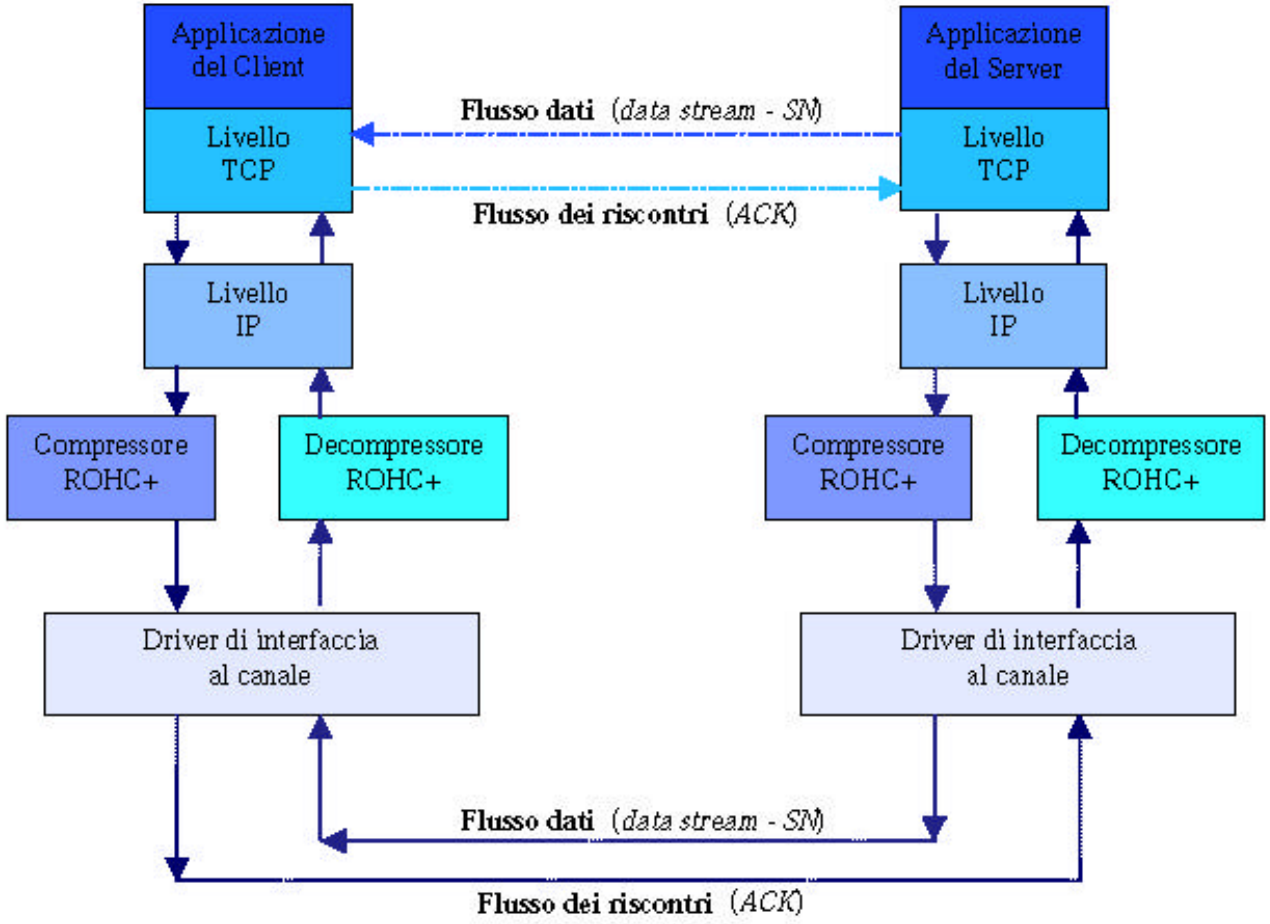
- Ogni connessione TCP è costituita da due flussi, in direzione opposta:
  - Data stream (server ? client)
  - ACK stream (server ? client)

  Entrambi questi flussi devono essere compressi, in modo indipendente uno dall'altro per ottimizzare il rapporto di compressione

- Tutti i profili di compressione ROHC si basano sull'individuazione di un "campo leader" (il Sequence Number per RTP), sul quale basare la trasmissione dei pacchetti con massimo compressione. Nel caso delle connessioni TCP, tale campo non è presente, per cui è stato appositamente introdotto: **SNR** (Sequence Number ROHC, 2 byte).
  - Il campo SNR viene generato dal compressore, ad un valore iniziale casuale, e viene incrementato di 1 per ogni pacchetto trasmesso.

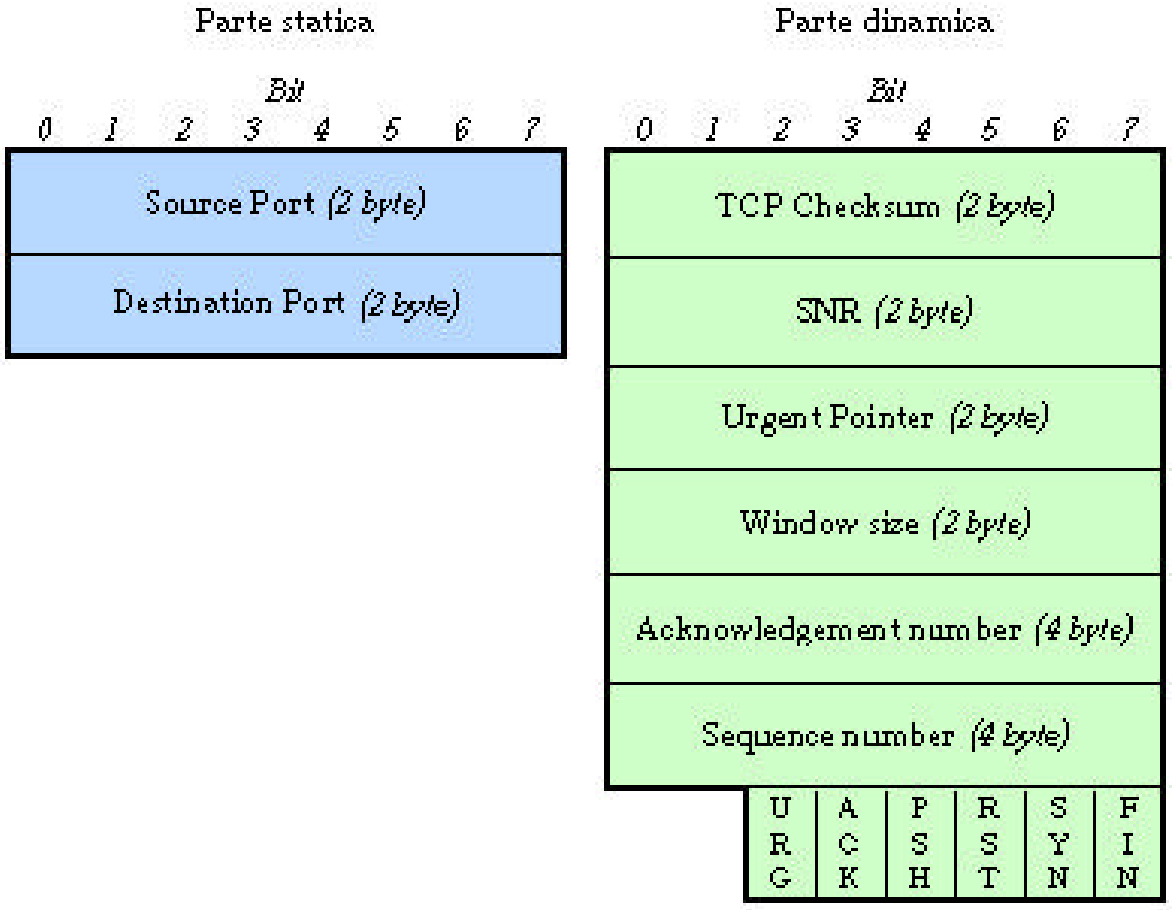
Algoritmo ROHC
TCP: Data stream e ACK stream

- Ciascun flusso TCP è caratterizzato dalla presenza di due diversi "campi chiave" per la compressione: **TCP Sequence Number** per il data stream e **ACK number** per l' ACK stream.
- Il compressore monitora le variazioni di tali campi e deduce quando il flusso si è regolarizzato oppure quando ha perso la regolarità



Algoritmo ROHC

Suddivisione dei campi dell'header TCP



Campi ridondanti:

Header Length

Reserved

Campo extra:

Sequence Number ROHC (SNR)

Algoritmo ROHC

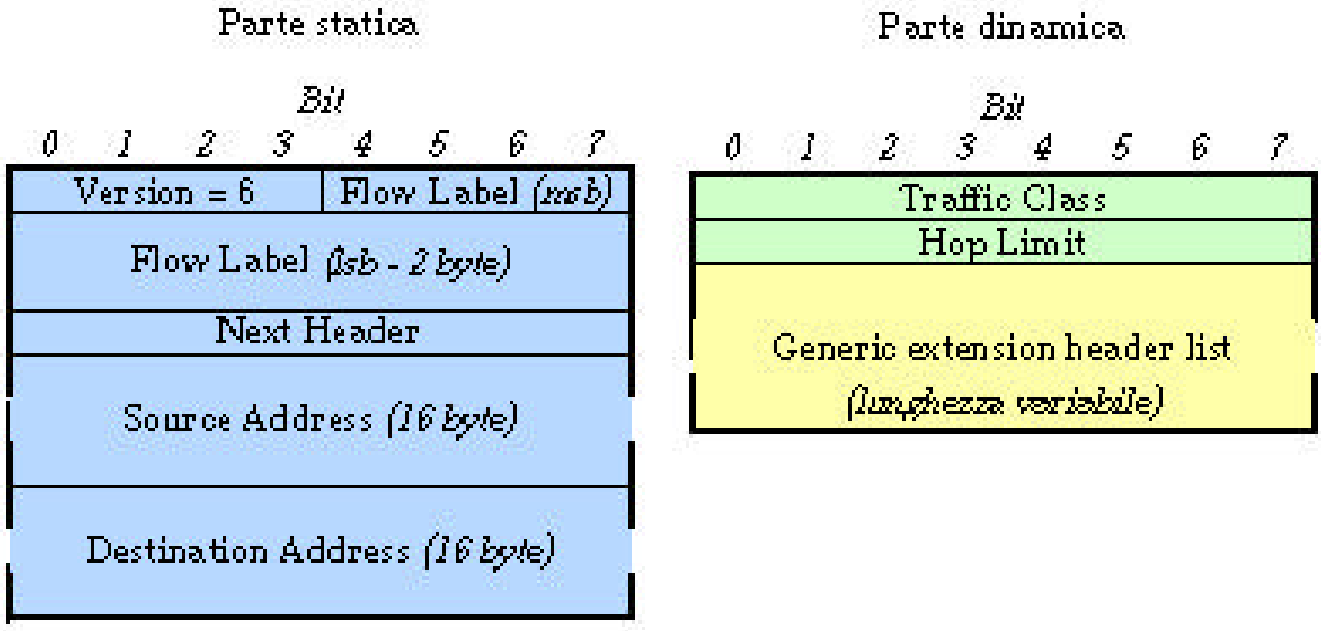
Suddivisione dei campi dell'header IPv6

Campi ridondanti:

Payload Length

Campo extra:

Generic Extension Header List



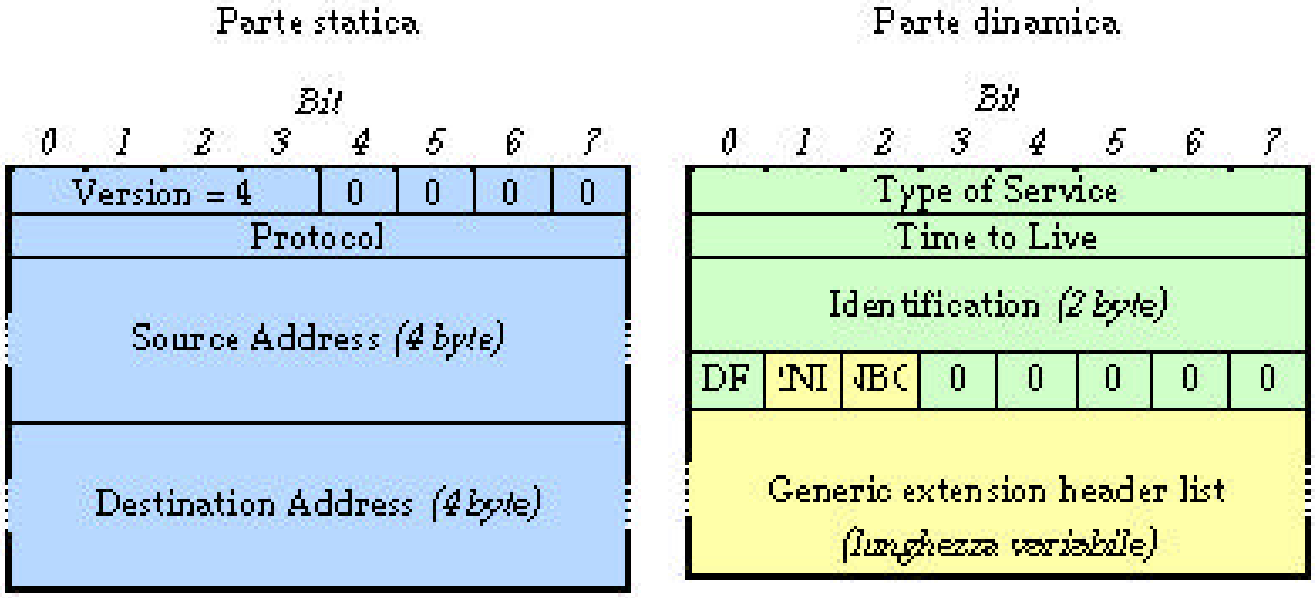
Algoritmo ROHC
Suddivisione dei campi dell'header IPv4

Campi ridondanti:

IHL
Total Length
MF flag
Fragment Offset
Header checksum
Options
Padding

Campi extra:

*Flag **RND** e **NBO***
Generic Extension Header List



Algoritmo ROHC
Codifica dei campi dinamici: LSB e W-LSB

- Per migliorare il rapporto di compressione, vengono trasmesse solo le variazioni dei campi dinamici
- La tecnica di base utilizzata è quella cosiddetta **LSB (Last Significant Bit)**: dato il valore da trasmettere, ne vengono inviati solo i bit meno significativi che sono cambiati rispetto al valore del pacchetto precedente
- Nella versione base, la tecnica LSB è tutt'altro che robusta nei confronti delle perdite di pacchetti. E' stata perciò ideata la variante cosiddetta **W-LSB (Window-Based LSB)** : il compressore basa le compressioni solo su un valore di riferimento per il quale ha ricevuto un riscontro esplicito o implicito (U-mode) dal decompressore
- I campi codificati W-LSB sono:
  - TCP SNR
  - TCP Sequence Number (data stream) o TCP Acknoledgement Number (ACK stream)
  - TCP Window Size
  - IPv4 Identification

Algoritmo ROHC

Flessibilità dei formati dei pacchetti ROHC

L'algoritmo è progettato in modo da generare sempre pacchetti con la minima dimensione possibile, data la disponibilità di molteplici formati :

Pacchetti di inizializzazione o aggiornamento del context →
- Formato IR
- Formato IR-DYN

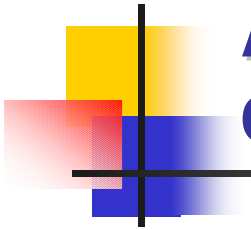
Pacchetti compressi →
- tipo 0 (5 byte)
- tipo 1 (6 byte)
- tipo 2 (7 byte)

+ **estensioni**:
- ext 0 (1 byte)
- ext 1 (2 byte)
- ext 2 (3 byte)
- ext 3 (variabile)
- ext X0 (da 1 a 4 byte)

In totale, ci sono 73 "formati" diversi tra cui il compressore deve scegliere ad ogni trasmissione

Pacchetti di feedback →
- tipo feedback-1 (4 byte)
- tipo feedback 2 (minimo 5 byte)

+ **opzioni**

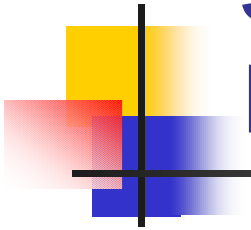


Algoritmo ROHC
Controllo degli errori

Il requisito della **robustezza** impone che la propagazione degli errori venga minimizzata tramite tentativi di riparazione locale del context. Nel caso dei flussi TCP, non conta tanto la rapidità degli schemi di riparazione (flussi real-time), quanto l'ottimizzazione dell'occupazione della banda.

Sono previsti tre distinti meccanismi per il controllo degli errori:

- **Codice CRC**: solo i pacchetti decompressi che verificano il CRC (3, 7 o 8 bit) possono aggiornare il context ed essere consegnati ai livelli superiori
  - *Algoritmo di riparazione locale del context*: ipotizza che il fallimento del CRC sia dovuto ad errori nella decodifica W-LSB (wrap-around) oppure ad un context non valido (errori residui nei precedenti pacchetti) e applica i relativi algoritmi di correzione
- **Algoritmo TWICE**: il decompressore verifica la Checksum TCP del pacchetto appena decompresso; in caso negativo, tenta una o due "riparazioni" su tutti i campi dinamici; se la checksum fallisce ancora, il pacchetto viene scartato
- **Meccanismo dei riscontri intrinseco del TCP**

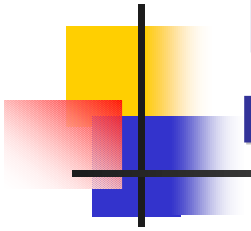


Strumento di simulazione: Network Simulator

- Simulatore di reti IP, nato nel 1989 come variante del simulatore *REAL*
- Dello sviluppo di NS si occupano i ricercatori del **VINT Project**, che vede la collaborazione di UC Berkeley, LBL, USC/ISI, e Xerox PARC, e che è supportato da DARPA (Defense Advanced Research Projects Agency)
- Il *core* del simulatore è scritto in C++ (meno intuitivo ma più rapido nell'esecuzione)
- L'interfaccia con l'utente è invece in OTcl (più intuitivo ma meno rapido nell'esecuzione)
- Il simulatore è basato sulla programmazione ad oggetti
- Sono possibili almeno quattro modalità di visualizzazione dei risultati:
  - Diagrammi cartesiani (XGRAPH)
  - Animazioni grafiche (NAM)
  - Estrazione "manuale" delle statistiche dai "trace files"
  - Inserimento di breakpoint e contatori nel codice sorgente
- Versione adottata in laboratorio: 2.1b7a su piattaforma Linux Mandrake 8.0
- Riferimenti:
  - Sito ufficiale: http://www.isi.edu/nsnam/ns/
  - Mailing list ufficiale: http://www.isi.edu/nsnam/ns/ns-lists.html

Network Simulator

Elementi cardine delle simulazioni

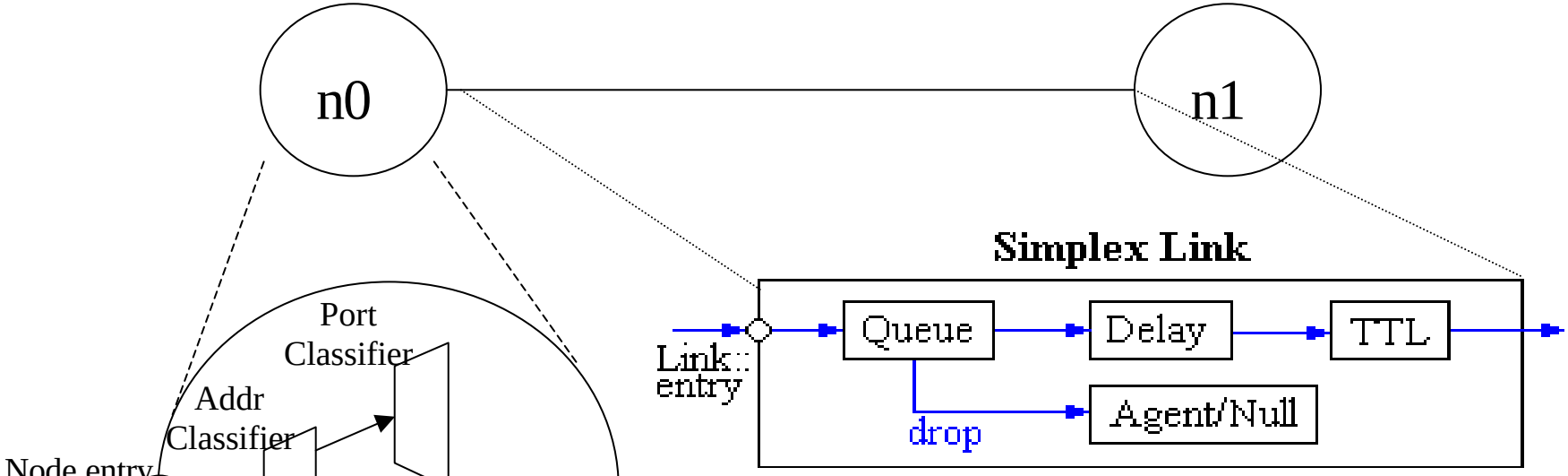


- Nodi
- Link
- Agenti
- Applicazioni
- Pacchetti
- Moduli di errore

Network Simulator

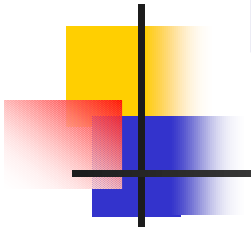
Topologia di rete: i “nodi” e i “link”

I **nodi** servono a “posizionare”, nella topologia di rete, tutte le entità per la “generazione” e l’ ”elaborazione” dei pacchetti



Nodo di tipo “unicast”

- Collegano i nodi della topologia.
- Possono essere unidirezionali o bidirezionali.
- Richiedono che ne vengano specificati almeno la **banda**, il **ritardo** e la **disciplina di coda** in ingresso.



Network Simulator

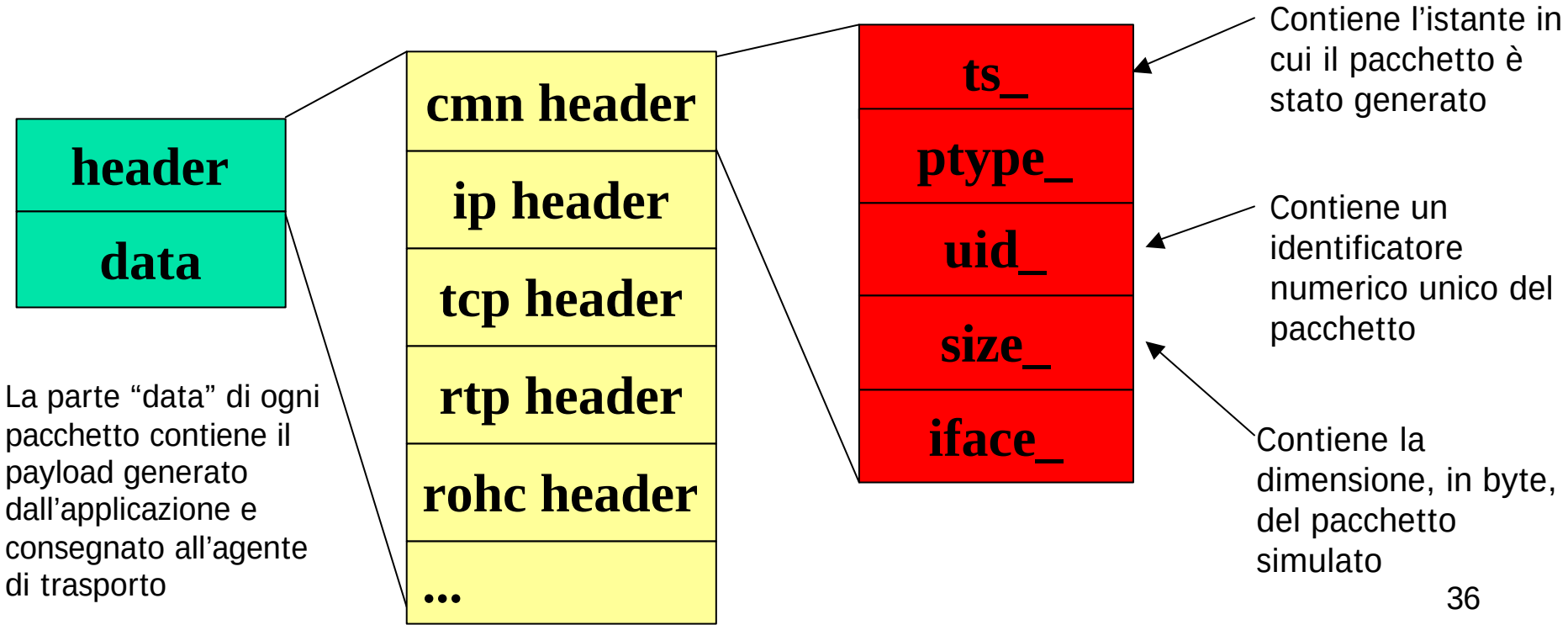
Gli “agenti” e le “applicazioni”

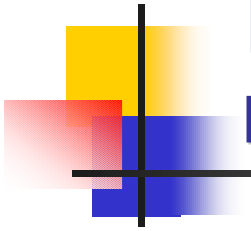
- **AGENTI**
  - Sono le entità che NS usa per l'implementazione dei protocolli ai vari livelli. Implementano sia i protocolli di trasporto (TCP e UDP) sia i protocolli di livello superiore (ad esempio RTP e RTCP); possono inoltre essere usati per l’implementazione di protocolli di livello inferiore ad IP.
  - Costruiscono o leggono o semplicemente elaborano pacchetti
  - Gli agenti collocati idealmente a livello di trasporto (TCP o UDP) guidano le simulazioni, ricevendo i dati dalle applicazioni e gestendone la trasmissione a destinazione.
- **APPLICAZIONI**
  - Sono entità poste idealmente al di sopra del livello di trasporto. Consegnano i propri dati (payload) agli agenti sottostanti, richiedendone la trasmissione a destinazione.
  - Ci sono solo applicazioni trasmittenti, non essendo prevista l’elaborazione del payload ricevuto
  - Ci sono vari tipi di applicazioni:
    - Applicazioni “tradizionali”: FTP, Telnet, HTTP, ...
    - Generatori di traffico: CBR, VBR, ...

Network Simulator

Meccanismi di gestione dei pacchetti

- Gli oggetti della classe "Packet" sono le unità fondamentali di scambio di dati tra gli oggetti di una simulazione.
- Nuovi protocolli possono definire gli header dei propri pacchetti oppure possono estendere gli header di formati preesistenti con campi addizionali.





Network Simulator

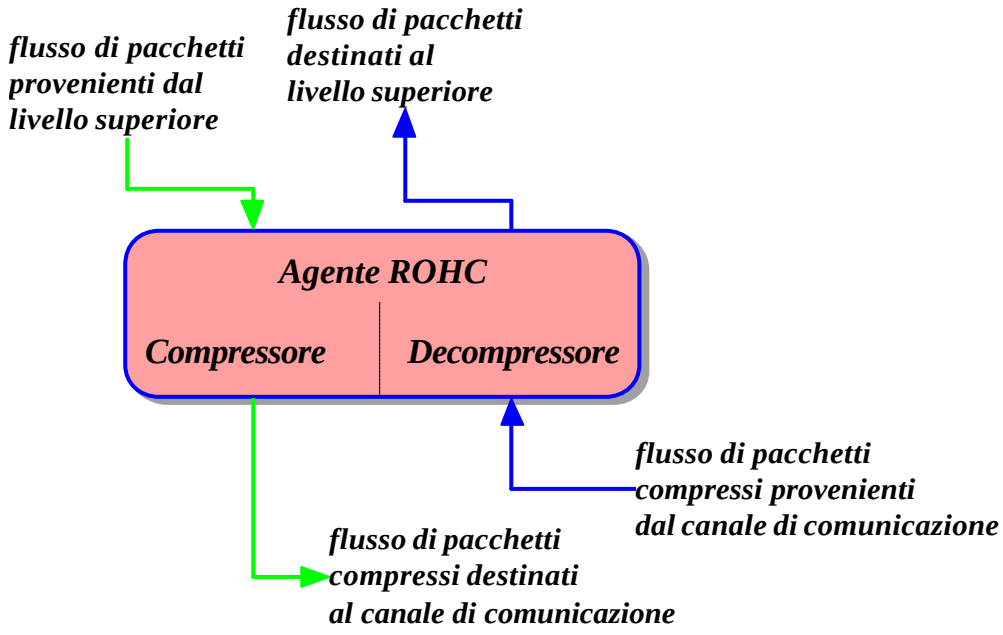
Moduli di errore

- I **moduli di errore** sono gli oggetti che *ns* utilizza per simulare la “non idealità” dei link
- Esistono modelli di errore semplici (tasso d’errore costante) e modelli più complessi (ad esempio markoviano), ciascuno caratterizzato da propri parametri che l’utente deve impostare
- Il simulatore è in grado sia di “cancellare” pacchetti su un link sia di “segnalare” la presenza di uno o più errori su un dato pacchetto: in questo secondo caso, è l’agente ricevente a gestire la situazione d’errore.
  - Eventuali errori su uno o più bit non sono “applicati” al pacchetto, ma solo “segnalati”, tramite l’ “*error flag*” nel *common header*

Simulazioni

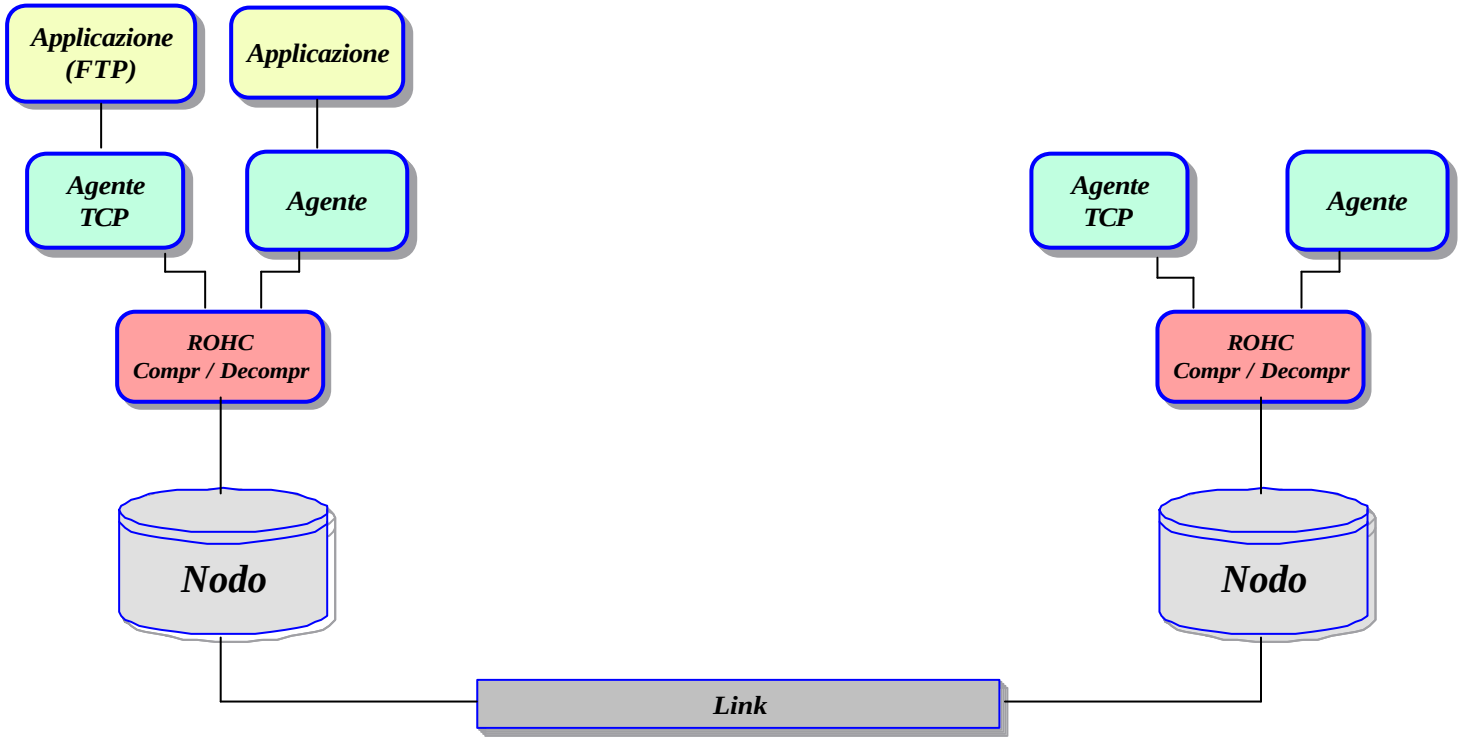
Agenti ROHC: caratteristiche generali

Si è fatta la scelta di implementare compressore e decompressore in un unico modulo (**agente ROHC**), che faccia perciò da interfaccia tra TUTTE le entità di livello superiore (agenti di trasporto) ed il livello inferiore (in prima approssimazione, il livello fisico)



Simulazioni

Topologia di rete con agenti ROHC



Simulazioni

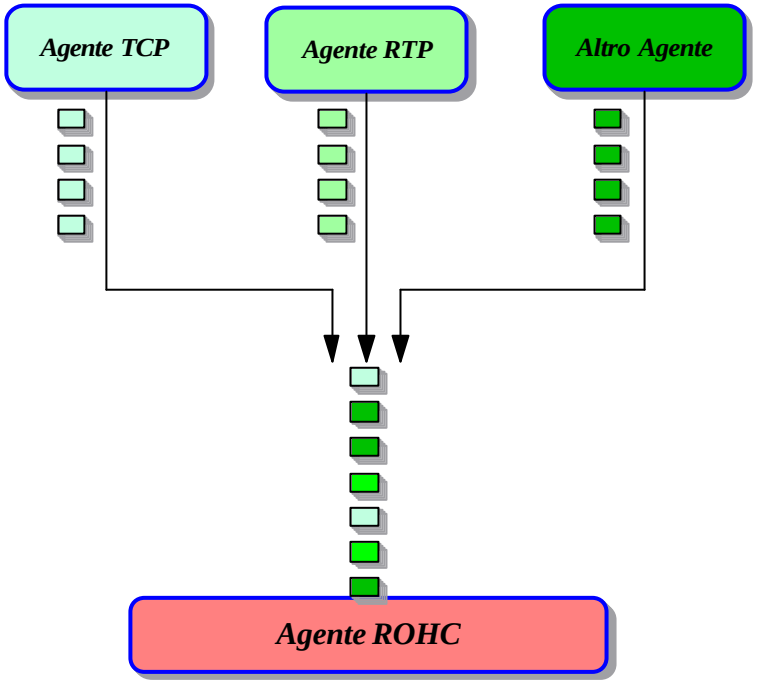
Flussi in ingresso e “tabella delle connessioni”

La capacità di gestire più flussi contemporanei in ingresso viene garantita dall’uso di una “**tabella delle connessioni**”.

Ad ogni connessione viene associato (tramite il CID) un elemento della tabella, che contiene tutto ciò di cui il compressore ed il decompressore necessitano per svolgere i propri compiti sui pacchetti in ingresso.

Ogni **elemento** della tabella contiene:

- CID;
- Identificatore tipo di connessione;
- Context;
- Contatori;
- Strutture dati per la codifica W-LSB

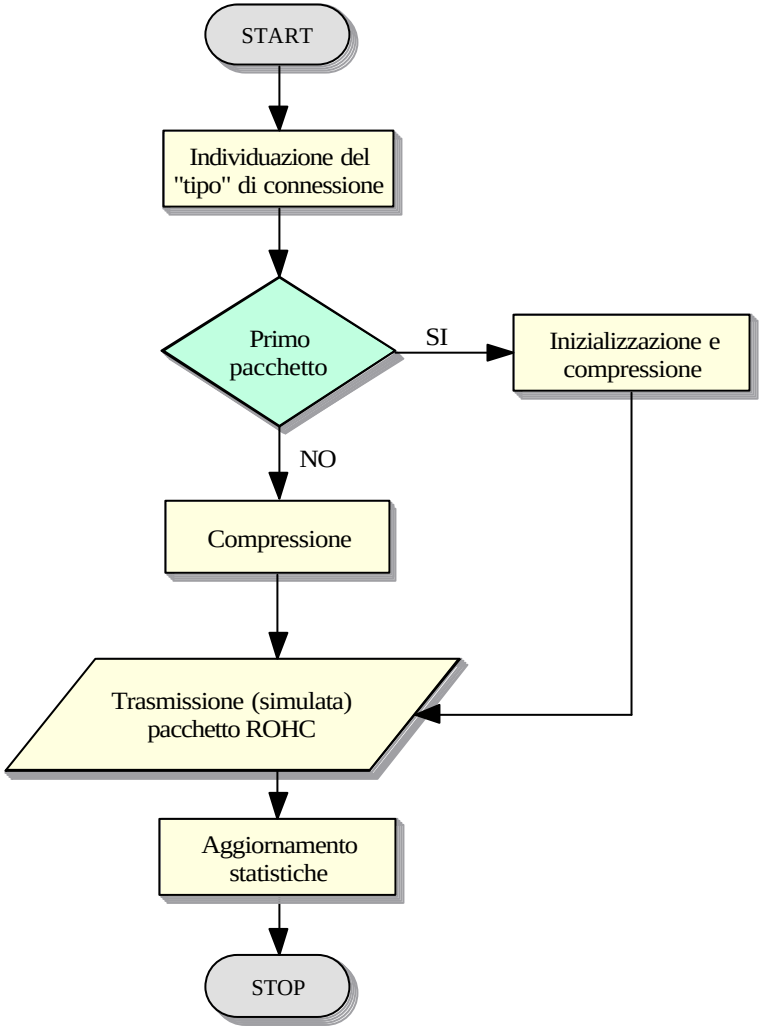


Simulazioni

Compressore: diagramma a blocchi

Il compressore:

- Individua il “tipo” di connessione cui il pacchetto appartiene, sulla base dei campi dell’header IP e dei parametri di configurazione della simulazione
- a seconda che il pacchetto sia il primo o meno della connessione cui appartiene, inizializza o aggiorna le proprie tabelle
- effettua la compressione sulla base del profilo di compressione opportuno
  - l’aspetto più importante e complesso è la scelta del **formato** del pacchetto ROHC da generare
- effettua la trasmissione (simulata) del pacchetto compresso sul mezzo fisico

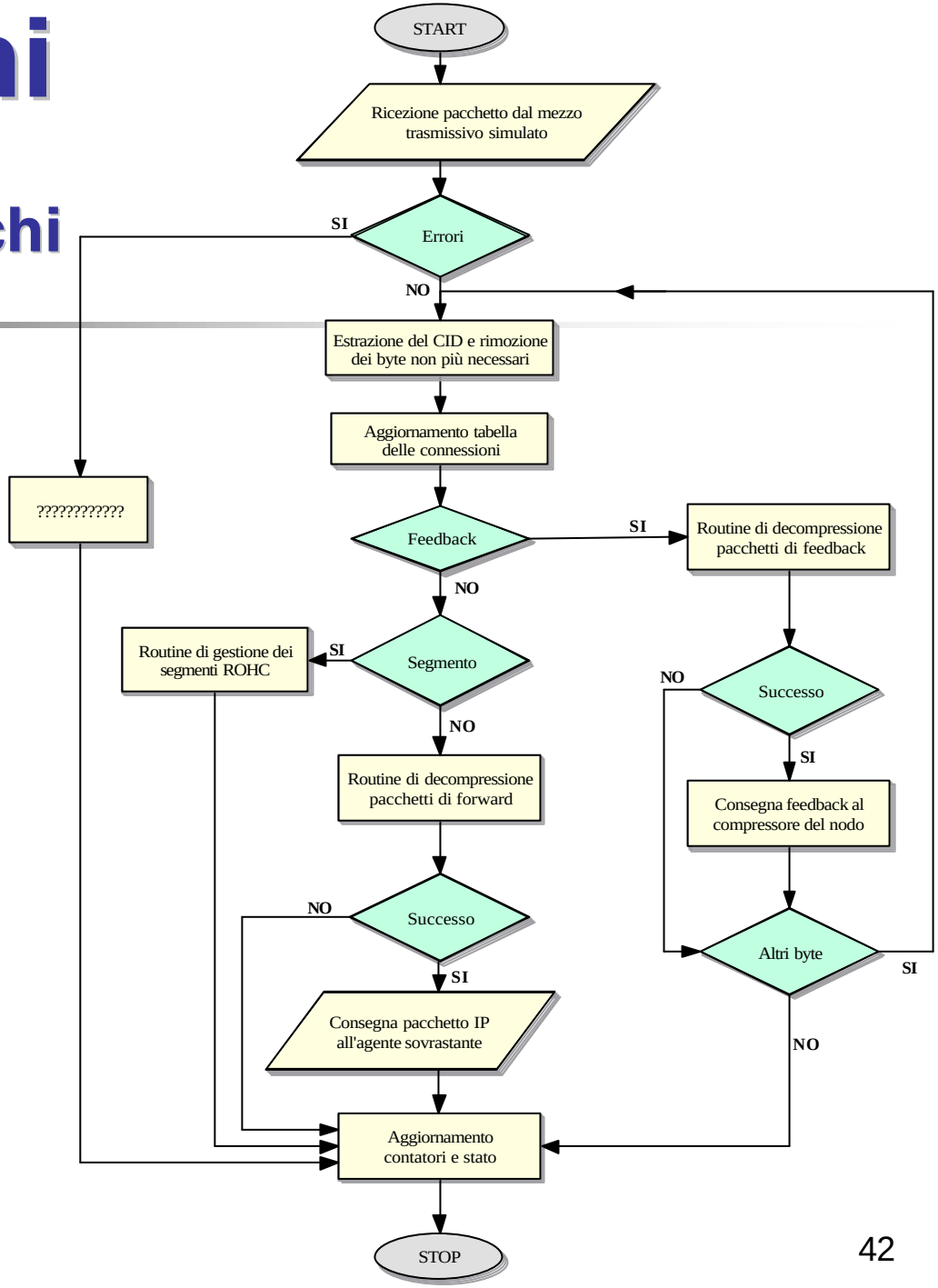


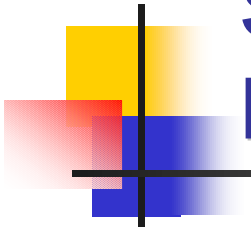
Simulazioni

Decompressore: diagramma a blocchi

Il decompressore:

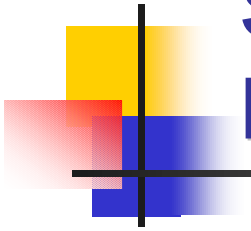
- verifica la presenza di eventuali errori segnalati dai livelli inferiori
- in assenza di errori, estrae il CID per individuare la connessione cui il pacchetto appartiene; con il CID indicizza la tabella delle connessioni
- stabilisce il tipo di pacchetto (forward, feedback, segmento) e lo “consegna” alla opportuna routine di elaborazione
- se la decompressione ha successo, consegna il pacchetto al destinatario (agente sovrastante oppure compressore adiacente), dopodiché aggiorna le statistiche sul proprio operato nonché sulla singola connessione





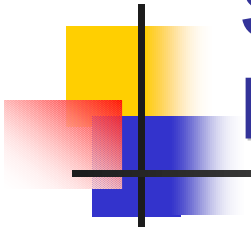
Suddivisione temporale del progetto
Fase 1 – “Preliminari”

- Studio approfondito dell’algoritmo ROHC
- Studio dei meccanismi di gestione dei pacchetti in *ns*
- Estensione degli header TCP e IP predefiniti
- Integrazione degli agenti ROHC come mittenti e riceventi di pacchetti
- Implementazione iniziale dell’algoritmo:
  - modalità di funzionamento unidirezionale
  - struttura di base degli agenti ROHC
  - algoritmo di scelta del formato ottimale per i pacchetti da trasmettere
  - routine per la codifica W-LSB
  - routine per la costruzione dei pacchetti nei formati compressi e nei formati di inizializzazione



Suddivisione temporale del progetto
Fase 2 – Completamento algoritmo

- Completamento dell'implementazione dell'algoritmo ROHC
  - Aspetti generali
    - Gestione feedback
    - Meccanismi di riparazione locale del context
    - CRC e TCP Checksum
  - Modalità bidirezionale ottimistica
  - Modalità bidirezionale affidabile
- Ottimizzazione iniziale dell'algoritmo



Suddivisione temporale del progetto
Fase 3 - Perfezionamenti

- Caratterizzazione dei **link** per le simulazioni, sulla base di eventuali misure e stime sui link reali (UMTS):
  - banda e ritardo
  - modelli di errore
    - caratterizzazione dei modelli di errore predefiniti e/o introduzione di nuovi modelli più specifici
- **Aspetti dinamici** dell'algoritmo di compressione:
  - modalità di monitoraggio dello "stato" del canale radio e della rete
  - individuazione ed implementazione dei legami tra le informazioni di stato ed i parametri di compressione

Suddivisione temporale del progetto
Fase 4 – Simulazioni e misure

- **Scelta ed implementazione dello “scenario di simulazione”:**
  - Topologia della rete (numero di nodi, posizione dei punti di compressione e decompressione, caratteristiche dei link)
  - Versione TCP (NewReno, Westwood, ...) e IP (IPv4 e/o IPv6, tunneling)
  - Applicazioni (FTP, CBR, Telnet, HTTP)
- **Effettuazione delle simulazioni**
- **Analisi delle prestazioni**
  - Guadagno di compressione
  - Efficienza di utilizzo della banda
  - Efficienza dei meccanismi di riparazione locale del context
  - Efficienza delle variazioni dinamiche dei parametri di compressione
  - **Eventuali revisioni dei passi precedenti**

Simulazioni
Possibile scenario di simulazione