

Appunti di Fondamenti di Comunicazioni e Internet

Canale 1 · Reti · Prof.ssa Francesca Cuomo

Capitolo 5: Strato di collegamento

Autore: Tommaso C.

Appunti tratti dalle lezioni in aula e dal materiale del corso

Argomenti: Collegamenti punto a punto e condivisi · Servizi dello strato 2
Frame, header e trailer · Delimitazione con il flag
Bit stuffing e byte stuffing
Errori sui bit · Codeword, FEC e ARQ · Parità singola

Indice

1	Lo strato di collegamento	1
2	Il frame	3
3	Framing e delimitazione	3
3.1	Bit stuffing	4
3.2	Byte stuffing	5
4	Rivelazione e correzione degli errori	6
4.1	Controllo di parità singola	7
5	Esercizi	8
6	Soluzioni	9

1 Lo strato di collegamento

Definizione – Collegamento e frame

Un **collegamento** (*link*) è il canale di comunicazione che unisce due nodi adiacenti lungo un cammino: host e router, due router, un dispositivo e un access point, un access point e un router. Può essere cablato o wireless. Lo **strato di collegamento** (*data link*, strato 2) trasporta i pacchetti lungo un singolo collegamento; la sua unità di dati è il **frame**.

Lo strato 2 è sezione per sezione (Capitolo 4): c'è in ogni sistema attraversato che ne svolge le funzioni. Un dispositivo che lavora solo sul segnale, come un **amplificatore**, ha soltanto lo strato fisico.

Definizione – Collegamento punto a punto e condiviso

- Un collegamento è **punto a punto** se unisce soltanto due sistemi, come un cavo fra due dispositivi.
- È **condiviso** se lo stesso mezzo è usato da più utenti, come il canale radio di un access point Wi-Fi o di una stazione radio base cellulare.

Solo sul collegamento condiviso serve il sottostrato **MAC** (*Medium Access Control*), che regola l'accesso degli utenti al mezzo.

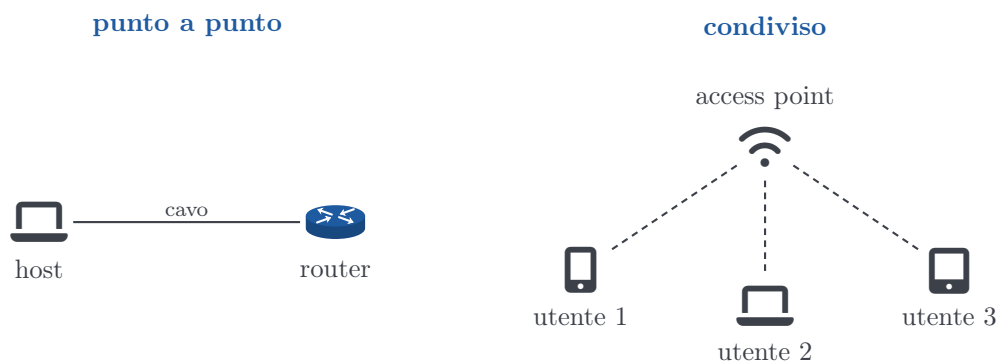


Figura 1: Un collegamento punto a punto unisce due soli sistemi; su un collegamento condiviso più utenti usano lo stesso mezzo e serve il sottostrato MAC.

Definizione – Servizi dello strato di collegamento

Uno strato di collegamento può offrire:

1. il **framing**: incapsula il pacchetto dello strato di rete in un frame;
2. l'**accesso al mezzo**, con il sottostrato MAC, se il collegamento è condiviso;
3. la **rivelazione e correzione degli errori** e la **consegna affidabile**: accorgersi dei bit sbagliati, correggerli o far ritrasmettere i frame, e riconoscere i frame arrivati in ordine diverso da quello di partenza;
4. il **controllo di flusso**: impedisce al trasmettitore di inviare frame più in fretta di quanto il ricevente riesca a elaborarli;
5. la gestione del **duplex**: in **half duplex** si trasmette e si riceve a turno, in **full duplex** contemporaneamente.

Il controllo di flusso serve perché lo strato 2 non si limita a ricevere: deve elaborare ogni frame e poi passarne il contenuto allo strato di rete, in un sistema terminale, o rilanciarlo verso il collegamento successivo, in un nodo intermedio. Il duplex lo gestisce lo strato 2 perché quasi ogni sistema, collegato via cavo o via Wi-Fi, sullo stesso mezzo trasmette e riceve.

Osservazione – Non tutti i protocolli offrono tutti i servizi

I protocolli di strato 2 nascono dagli standard degli enti di standardizzazione, e ogni standard sceglie quali servizi offrire. Un protocollo può fare a meno, per esempio, della consegna affidabile o del controllo di flusso: **Ethernet** non offre la consegna affidabile.

Definizione – Scheda di rete

Lo strato 2 e lo strato fisico di un sistema sono realizzati dalla **scheda di rete** (*Network Interface Card*, **NIC**), una combinazione di hardware e software: la parte hardware è a contatto con il mezzo fisico, la parte software esegue il protocollo di collegamento.

Un sistema ha una scheda per ogni tecnologia con cui si collega. Un portatile può averne per Ethernet, Wi-Fi, Bluetooth e NFC; uno smartphone non ha Ethernet ma ha una scheda per la rete cellulare 4G o 5G. Anche una chiavetta di rete esterna è una NIC, con strato fisico e strato 2.

2 Il frame

Definizione – Frame, header e trailer

Lo strato 2 riceve un pacchetto dallo strato di rete e lo incapsula in un **frame**: il pacchetto è la **SDU**, a cui lo strato aggiunge la propria **PCI**. A differenza degli strati superiori, la PCI di strato 2 può stare in parte all'inizio del frame, l'**header**, e in parte alla fine, il **trailer**.

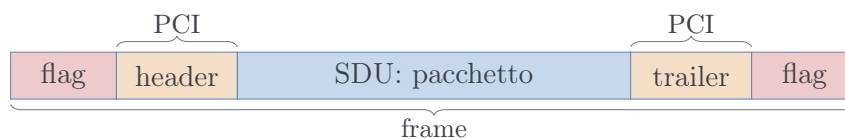


Figura 2: Struttura di un frame: la PCI dello strato 2 sta in parte nell'header e in parte nel trailer; i flag segnano inizio e fine (Sezione 3).

Nella PCI stanno le informazioni che servono ai servizi del protocollo: per esempio i bit per rivelare gli errori. Il frame passa fisicamente dallo strato 1 del trasmettitore allo strato 1 del ricevente; le due entità di strato 2 comunicano logicamente. Se il protocollo prevede la consegna affidabile, lo scambio può richiedere più ritrasmissioni prima che il pacchetto venga consegnato allo strato di rete.

3 Framing e delimitazione

Definizione – Delimitazione con il flag

Se i frame hanno **lunghezza variabile**, il ricevente deve riconoscere senza ambiguità dove ciascuno inizia e finisce: è la **delimitazione**. Il trasmettitore mette all'inizio e alla fine del frame una sequenza nota, il **flag**; il ricevente esamina il flusso di bit e riconosce i flag di apertura e di chiusura. Un flag diffuso è il byte

$$\text{flag} = 01111110.$$

Con frame di **lunghezza fissa**, concordata in anticipo fra trasmettitore e ricevente, la delimitazione può non servire: basta contare i bit, per esempio frame sempre da 250 bit.

Osservazione – Simulazione del flag

I dati dell'utente sono bit qualsiasi: dentro la SDU può comparire proprio il byte 01111110. Il ricevente lo scambiarebbe per il flag di chiusura e taglierebbe il frame. Per evitarlo, il trasmettitore modifica i dati con lo **stuffing** (riempimento) e il ricevente li ripristina con il **destuffing**.

Lo stuffing obbliga lo strato 2 a leggere i bit della SDU, che di norma tratta come un contenuto da trasportare senza guardarlo (Capitolo 4). Esistono due tecniche, bit stuffing e byte stuffing: ogni protocollo ne usa una sola, mai entrambe.

3.1 Bit stuffing

Definizione – Bit stuffing e destuffing

1. **Stuffing:** il trasmettitore scorre i bit del frame e inserisce uno 0 dopo ogni sequenza di **cinque 1** consecutivi, qualunque sia il bit successivo. Solo dopo aggiunge i flag.
2. **Destuffing:** il ricevente conta gli 1 consecutivi. Dopo cinque 1, se il bit seguente è 0 è un bit di stuffing e lo elimina; se è 1, la sequenza è un flag.

Dopo lo stuffing il contenuto del frame non ha mai sei 1 di fila: sei 1 consecutivi compaiono solo nel flag, e il ricevente non si confonde.

Esempio – Bit stuffing

Sequenza da trasmettere, 32 bit:

10111111111110110111110011111100

Il trasmettitore inserisce uno 0 dopo ogni gruppo di cinque 1; la sequenza trasmessa, prima dei flag, ha 36 bit:

101111101111101011011111000111110100

Il ricevente, ogni volta che conta cinque 1 seguiti da uno 0, elimina lo 0 e ritrova la sequenza originale.

Osservazione – Costi del bit stuffing

- **Overhead:** aggiunge bit di controllo, e li mette dentro i dati dell'utente.
- **Elaborazione:** per formare il frame bisogna leggere tutta la sequenza, un bit alla volta; un processore moderno lo fa in fretta, ma è tempo speso.

Le regole di stuffing e destuffing fanno parte del protocollo: trasmettitore e ricevente devono conoscerle entrambi.

3.2 Byte stuffing

Definizione – Byte stuffing e destuffing

Si usa un secondo byte speciale, il **control escape**:

$$\text{ESC} = 01111101.$$

1. **Stuffing**: il trasmettitore legge il frame **un byte alla volta** e inserisce un ESC davanti a ogni byte uguale al flag o a ESC. Solo dopo aggiunge i flag.
2. **Destuffing**: il ricevente legge un byte alla volta. Se legge un ESC, lo elimina e prende il byte seguente come dato, anche se è un flag o un ESC. Un flag non preceduto da ESC delimita il frame.

Esempio – Byte stuffing

La SDU di tre byte

01000001 01111110 01111101

contiene un byte uguale al flag e uno uguale a ESC. Il frame trasmesso è

flag 01000001 **ESC** 01111110 **ESC** 01111101 flag

Il ricevente legge il flag di apertura e il primo byte; poi un ESC, che elimina, prendendo 01111110 come dato; poi un altro ESC, che elimina, prendendo 01111101 come dato; infine un flag non preceduto da ESC, che chiude il frame.

	Bit stuffing	Byte stuffing
Unità letta	un bit alla volta	un byte alla volta
Che cosa inserisce	uno 0 dopo cinque 1	un ESC davanti a flag ed ESC
Overhead	minore (un bit per inserimento)	maggiore (un byte per inserimento)
Elaborazione	maggiore	minore
Esempio d'uso		PPP (<i>Point-to-Point Protocol</i>)

Tabella 1: Bit stuffing e byte stuffing a confronto.

Quale tecnica convenga dipende da che cosa pesa di più: l'overhead, a favore del bit stuffing, o il tempo di elaborazione, a favore del byte stuffing. L'Esercizio 1 applica le due tecniche alla stessa SDU e ne confronta i costi.

4 Rivelazione e correzione degli errori

Definizione – Errore sul bit

Un **errore** sul bit (*flip*) si ha quando un bit trasmesso come 0 viene ricevuto come 1, o viceversa.

Il bit cambia perché cambia il segnale che lo rappresenta: un'**interferenza** che si sovrappone può far leggere 1 al posto di 0; un'**attenuazione** può rendere troppo debole il segnale di un 1, che viene letto 0. Gli errori sono rari in fibra ottica e frequenti nelle comunicazioni wireless, tanto più quanto più c'è interferenza.

Definizione – Codeword

Per controllare gli errori, il trasmettitore passa i k bit da proteggere a un **codificatore**, che aggiunge $n - k$ bit di **ridondanza** e forma una **parola di codice** (*codeword*) di n bit; trasmette la codeword al posto dei dati originali. Il ricevente controlla se ciò che riceve è una codeword **valida**: se non lo è, rivela un errore.

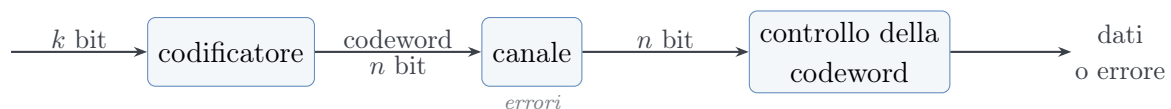


Figura 3: Controllo d'errore: il codificatore aggiunge $n - k$ bit di ridondanza; il ricevente verifica se la sequenza ricevuta è una codeword valida.

Osservazione – Errori non rivelati

Il canale può trasformare la codeword trasmessa in un'altra codeword valida. In questo caso il ricevente non si accorge dell'errore: è un **errore non rivelato**.

Definizione – FEC e ARQ

Il controllo d'errore segue uno di due approcci:

- **FEC** (*Forward Error Correction*): la ridondanza aggiunta in avanti, prima della trasmissione, è tale che il ricevente possa non solo rivelare gli errori ma anche **correggerli** da solo;
- **ARQ** (*Automatic Repeat Request*, o *error detection and retransmission*): il ricevente **rivela** gli errori e il frame errato viene **ritrasmesso**.

4.1 Controllo di parità singola

Definizione – Codice a parità singola

Il **codice a parità singola** aggiunge ai k bit informativi $b_1, b_2, \dots, b_k$ un solo **bit di parità**, in fondo, scelto in modo che la codeword contenga un numero **pari** di 1:

$$b_{k+1} = (b_1 + b_2 + \dots + b_k) \bmod 2.$$

Il ricevente conta gli 1 della codeword: se sono dispari, rivela un errore.

Proposizione – Che cosa rivela la parità singola

- Rivela ogni configurazione con un numero **dispari** di bit errati.
- Non rivela alcuna configurazione con un numero **pari** di bit errati.
- Non dice **quale** bit è errato, quindi non permette di correggerlo.

Dimostrazione

1. Ogni bit errato cambia di uno il numero di 1 della codeword, quindi ne cambia la parità.
2. Con m bit errati la parità cambia m volte: risulta dispari, e l'errore è rivelato, se e solo se m è dispari.
3. Un errore singolo in una qualsiasi delle $k+1$ posizioni produce lo stesso effetto, un numero dispari di 1: il ricevente non può distinguere la posizione.

□

Esempio – Errore singolo e doppio

I 7 bit informativi 0101100 contengono tre 1, quindi il bit di parità è 1 e la codeword è 01011001.

Caso	Ricevuto	N. di 1	Esito
nessun errore	01011001	4	accettata, corretta
errore nel bit 3	01 <u>1</u> 11001	5	errore rivelato
errori nei bit 3 e 5	01 <u>1</u> 1 <u>0</u> 001	4	accettata, ma errata

L'errore può colpire anche il bit di parità: il risultato non cambia, perché il ricevente conta gli 1 di tutta la codeword. Il codice costa un solo bit di overhead ed è semplicissimo, ma è poco robusto: in tutti i casi con un numero pari di errori dichiara corretto un frame errato.

Riepilogo

- **Collegamento:** unisce due nodi adiacenti; punto a punto o condiviso, e solo il condiviso richiede il sottostrato MAC.
- **Servizi:** framing, accesso al mezzo, rivelazione e correzione degli errori, controllo di flusso, half o full duplex; non tutti i protocolli li offrono.
- **Frame:** SDU (il pacchetto) più PCI, divisa fra header e trailer.
- **Flag:** 01111110 all'inizio e alla fine del frame; ESC = 01111101.
- **Bit stuffing:** uno 0 dopo cinque 1; meno overhead, più elaborazione.
- **Byte stuffing:** un ESC davanti a ogni flag o ESC nei dati; più overhead, meno elaborazione.
- **FEC e ARQ:** la codeword porta ridondanza; FEC corregge al ricevente, ARQ rivela e fa ritrasmettere.
- **Parità singola:** $b_{k+1} = (b_1 + \dots + b_k) \bmod 2$; rivela gli errori dispari, non quelli pari, e non li localizza.

5 Esercizi

Esercizio 1 – Framing di sei byte con le due tecniche

Lo strato 2 deve formare il frame per la SDU di sei byte

01000001 01111110 00110011 01111101 01010101 00001111

con flag 01111110 e control escape ESC = 01111101.

1. Formare il frame con il byte stuffing.
2. Formare il frame con il bit stuffing.
3. Confrontare le due soluzioni: bit aggiunti alla SDU e numero di letture necessarie per formare il frame.

Soluzione a pagina 9 →

6 Soluzioni

Soluzione dell'Esercizio 1 – Framing di sei byte con le due tecniche

1. *Byte stuffing.* Si legge la SDU un byte alla volta: il secondo byte è uguale al flag e il quarto è uguale a ESC, quindi davanti a ciascuno va un ESC. Poi si aggiungono i due flag. Il frame ha 10 byte:

```
01111110 01000001 01111101 01111110 00110011
01111101 01111101 01010101 00001111 01111110
```

Il ricevente toglie il flag iniziale e quello finale e, a ogni ESC, elimina l'ESC e tiene il byte seguente: ritrova i sei byte di partenza.

2. *Bit stuffing.* Si scorrono i 48 bit della SDU. Cinque 1 consecutivi compaiono due volte, nel secondo byte e nel quarto; nessun gruppo di cinque 1 scavalca il confine fra due byte. Si inseriscono quindi due 0:

```
01000001 011111010 00110011 011111001 01010101 00001111
```

Poi si aggiungono i due flag: il frame ha $8 + 50 + 8 = 66$ bit.

3. *Confronto.*

	Byte stuffing	Bit stuffing
aggiunti dallo stuffing	2 byte = 16 bit	2 bit
flag	2 byte = 16 bit	16 bit
overhead totale	32 bit	18 bit
overhead rispetto alla SDU	$32/48 \approx 67\%$	$18/48 = 37,5\%$
letture per formare il frame	6 (una per byte)	48 (una per bit)

Il bit stuffing aggiunge meno overhead, il byte stuffing richiede meno elaborazione.

[← Torna all'Esercizio 1, pagina 8](#)