

## Reti di calcolatori

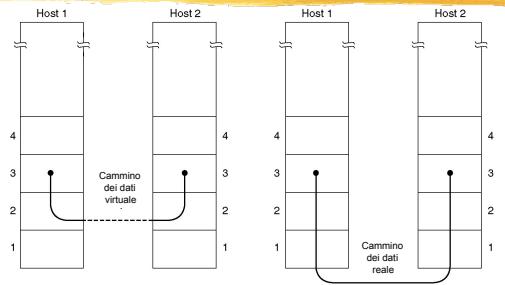
### Livello Data Link

Prof.ssa Simonetta Balsamo  
Dipartimento di Informatica  
Università Ca' Foscari di Venezia  
balsamo@dsi.unive.it  
<http://www.dsi.unive.it/~reti>

Livello data-link

S.Balsamo 2010

R4.1



Livello data-link

S.Balsamo 2010

R4.3

## Livello Data Link - scopo

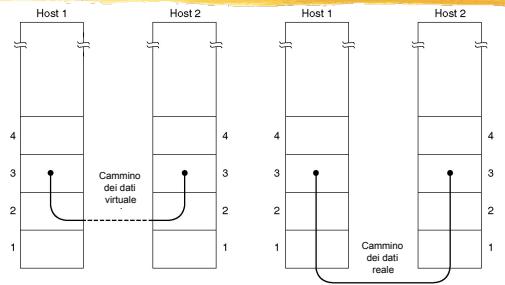
- Offrire una comunicazione affidabile ed efficiente a due macchine **adiacenti**
- Si comporta come un "tubo digitale", cioè i bit partono e arrivano nello stesso ordine
  - Ci sono errori e disturbi occasionali
  - Il canale ha un data rate finito
  - C'è un ritardo nella propagazione
- Compiti:
  - Offrire **servizi** al livello network con un'interfaccia ben definita;
  - Determinare come i bit del livello fisico sono raggruppati in **frame** (**framing**)
  - Gestire gli **errori** di trasmissione
  - Regolare il **flusso** della trasmissione fra sorgente e destinatario

Livello data-link

S.Balsamo 2010

R4.2

### Livello Data Link



Livello data-link

S.Balsamo 2010

R4.3

## Livello Data Link - servizi

1/2

- Trasferire dati dal livello network della macchina di origine al livello network della macchina di destinazione
- **connectionless non confermato**
  - si mandano frame **indipendenti**
  - i frame non vengono confermati
  - **non si stabilisce una connessione**
  - i frame persi non si recuperano (in questo livello)
  - è appropriato per
    - canali con tasso d'errore molto basso
    - traffico real-time (es. voce)
    - le LAN, nelle quali, in effetti, è molto comune

Livello data-link

S.Balsamo 2010

R4.4

## Livello Data Link - servizi

2/2

### Connectionless confermato

- I frame vengono confermati
- Se la conferma non arriva, il mittente può rispedire il frame
- E' utile su canali non affidabili ([sistemi wireless](#))
- Nota 1:** La perdita di un [ack](#) può causare la trasmissione di più copie dello stesso frame
- Nota 2:** La conferma a questo livello è un'ottimizzazione

### Connection oriented confermato

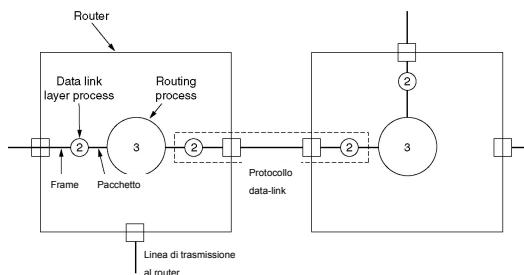
- Più sofisticato
- Prevede tre fasi: apertura connessione, invio dati, chiusura connessione
- Garantisce che ogni frame sia ricevuto esattamente una volta e nell'ordine giusto
- Fornisce al livello network un [flusso di bit affidabile](#)

Livello data-link

S.Balsamo 2010

R4.5

## Livello Data Link



Livello data-link

S.Balsamo 2010

R4.6

## Livello Data Link - funzionamento

### In trasmissione:

- Spezza** il flusso di bit in arrivo dal livello 3 in una serie di **frame** calcola un'apposita funzione ([checksum](#)) per ogni frame
- Inserisce il [checksum](#) nel frame
- Consegna** il frame al livello 1, che lo spedisce come sequenza di bit

### In ricezione:

- Riceve** una sequenza di bit dal livello 1
- Ricostruisce** i frame
- Per ogni frame ricalcola il [checksum](#)
  - Se è uguale a quello contenuto nel frame questo viene accettato, altrimenti viene considerato errato e scartato

Livello data-link

S.Balsamo 2010

R4.7

## Livello Data Link - framing

1/4

### 1. Conteggio di caratteri

- Un campo nell'header, indica quanti caratteri ci sono del frame.
- Se durante la trasmissione si rovina tale campo, diventa impossibile ritrovare l'inizio del prossimo frame e di conseguenza anche quello dei successivi

### 2. Delimitatori e byte stuffing

- Ogni frame inizia e finisce con una particolare sequenza di caratteri ASCII
- Una scelta diffusa è la seguente:
  - Inizio frame: **DLE** (Data Link Escape), **STX** (Start of TeXt)
  - Fine frame: **DLE**, **ETX** (End of TeXt)
- In caso di errore si ricerca DLE
- Problema:** nella trasmissione di dati binari, il byte corrispondente alla codifica di DLE può apparire dentro il frame, confondendo la trasmissione

Livello data-link

S.Balsamo 2010

R4.8

## Livello Data Link - framing 2/4

- Soluzione: Il livello 2 sorgente aggiunge davanti a tale byte un altro DLE, per cui in arrivo solo i singoli DLE segnano i confini dei frame. Il livello 2 destinazione rimuove i DLE aggiunti dentro ai dati prima di consegnarli al livello 3 (**character stuffing**)
- La tecnica character stuffing è legata alla codifica ASCII ad 8 bit, non va bene per codifiche più moderne (es. UNICODE)
- Alternativa: una tecnica che permette codifiche diverse di carattere

Livello data-link

S.Balsamo 2010

R4.9

## Livello Data Link - framing 3/4

### 3. Delimitatori e *bit stuffing*

- Ogni frame inizia e finisce con una specifica sequenza di bit (**bit pattern**), ad es. 0111110 chiamata un **flag byte (delimitatore)**
- Un problema analogo al caso precedente, quindi:
  - **in trasmissione:** Ogni volta che il livello due incontra nei dati da trasmettere 5 bit consecutivi uguali a 1 inserisce uno 0 aggiuntivo
  - **in ricezione:** Quando nei dati ricevuti compaiono 5 bit uguali a 1, si rimuove lo 0 che li segue

Alla sorgente (a) 0 1 1 0 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 | 1 0 0 1 0

Trasmessa (b) 0 1 1 0 1 1 1 1 | 0 1 1 1 1 0 | 1 1 1 1 | 0 1 0 0 1 0  
Stuffed bits

A destinazione (c) 0 1 1 0 | 1 1 1 1 1 | 1 1 1 1 1 | 1 1 1 1 | 1 0 0 1 0

Livello data-link

S.Balsamo 2010

R4.10

## Livello Data Link - framing 4/4

### 4. Violazione della codifica

- In molte reti (soprattutto LAN) si codificano i bit al livello fisico con una certa ridondanza. Ad esempio:
  - Il valore 1 di un bit di dati è codificato con la coppia **high/low** di bit fisici
  - Il valore 0 di un bit di dati è codificato con la coppia **low/high** di bit fisici
- Le coppie **low/low** ed **high/high** non sono utilizzate, quindi possono essere usate per delimitare i frame

Livello data-link

S.Balsamo 2010

R4.11

## Livello Data Link - gestione errori 1/9

- Gli errori sono dovuti in generale a:

- Rumore di fondo
- Disturbi (ad es. fulmini) improvvisi
- Interferenze (ad es. motori elettrici)

- Tipi di errori

- **Perdita** di bit
- **Modifica** del valore

- Due approcci al trattamento degli errori:

- Includere sufficiente informazione aggiuntiva in modo da ricostruire il messaggio originario (**correzione dell'errore**)
- Includere meno informazione aggiuntiva, in modo da rilevare un errore, senza necessariamente correggerlo (**rilevazione dell'errore**)

Livello data-link

S.Balsamo 2010

R4.12

## Livello Data Link - gestione errori 2/9

- Rilevazione dell'errore:
  - Bit aggiunti dal mittente
  - Rilevati e analizzati dal destinatario
- Overhead
  - di gestione
  - di trasmissione
- Garanzia statistica

Livello data-link

S.Balsamo 2010

R4.13

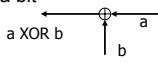
## Livello Data Link - gestione errori 3/9

- Tecniche
  - Bit di parità:
    - aggiunge un bit
    - tollera errori di un solo bit
  - Checksum:
    - Tratta i dati come interi
    - Mittente: calcola e invia una somma aritmetica
    - Overhead: checksum
    - Destinatario: calcola e controlla
    - Gestisce errori multipli
    - Non tutti

Esempio:	Dato in binario	checksum	Dato in binario	checksum
(Il bit sempre sbagliato)	0001	1	0011	3
	0010	2	0000	0
	0011	3	0001	1
	0001	1	0011	3
	totale	7	totale	7

Livello data-link S.Balsamo 2010 R4.14

## Livello Data Link - gestione errori 4/9

- Un frame (a parte i delimitatori) consiste di  $n = m + r$  bit, dove:
  - $m$  bit costituiscono il messaggio vero e proprio
  - $r$  bit sono ridondanti, e sono detti **redundant bit** (o **check bit**)
- Una sequenza di  $n$  bit fatta in tal modo si dice **codeword**, o **parola di codice**
- Date due qualunque parole di codice, ad es. **1000 1001** e **1011 0001** è possibile determinare il numero di bit che in esse differiscono (3 nell'esempio) tramite un semplice XOR fatto bit a bit
 
- Tale numero si dice la **distanza di Hamming** delle due codeword

Livello data-link

S.Balsamo 2010

R4.15

## Livello Data Link - gestione errori 5/9

- Se due **codeword** hanno una distanza di Hamming uguale a  $d$ , ci vogliono  $d$  errori su singoli bit per trasformare l'una nell'altra
- Un insieme prefissato di **codeword** costituisce un **codice (code)**
- La **distanza di Hamming di un codice** è il minimo delle distanze di Hamming fra tutte le possibili coppie di codeword del codice
- In particolare:
  - Per rilevare  $d$  errori serve un codice con distanza di Hamming ( $d+1$ ). Qualunque combinazione di  $d$  errori non riesce a trasformare un **codeword** valido in un altro **codeword** valido
  - Per correggere  $d$  errori, serve un codice di Hamming con distanza ( $2d+1$ ). Una parola con  $d$  errori è più vicina all'originale che a qualunque altra parola

Livello data-link S.Balsamo 2010 R4.16

## Livello Data Link - gestione errori 6/9

- Cyclic Redundancy Code (CRC, polynomial code) considerano le stringhe di bit come rappresentazioni di polinomi a coefficienti 0 e 1 (un numero ad  $m$  bit corrisponde ad un polinomio di grado  $m-1$ )
- L'aritmetica polinomiale è modulo 2
  - Addizione e sottrazione sono equivalenti a XOR
  - La divisione è calcolata attraverso la sottrazione modulo 2
- Mittente e destinatario concordano un polinomio generatore  $G(x)$ , con bit più significativo e meno significativo uguali ad 1
- Sia  $r$  il grado del polinomio  $G(x)$ 
  - Esempio:  $r=16$   $G(x) = x^{16} + x^{15} + x^2 + 1$
  - $r$  = numero di bit ridondanti

Livello data-link

S.Balsamo 2010

R4.17

## Livello Data Link - gestione errori 7/9

- Il mittente appende in coda al frame un checksum in modo che il polinomio corrispondente (di grado  $m+r-1$ ) sia divisibile per  $G(x) - x^r M(x)$
- |         |                        |         |     |
|---------|------------------------|---------|-----|
| DLE STX | dati con byte stuffing | DLE ETX | CRC |
|---------|------------------------|---------|-----|
- Il ricevente ricevuto il frame e checksum, divide il tutto per  $G(x)$  e se il risultato non è 0 c'è stato un errore

Livello data-link

S.Balsamo 2010

R4.18

## Livello Data Link - gestione errori 8/9

- Mittente
  - Dato il frame di  $m$  bit  $M(x)$  e il polinomio generatore  $G(x)$  di grado  $r$
  - Aggiunge  $r$  bit meno significativi a valore 0 nei  $\Rightarrow$  il frame di dimensione  $m+r$  corrisponde a  $x^r M(x)$
  - Divide  $x^r M(x)$  per  $G(x)$  con divisione modulo 2
  - Sottrae il resto (al massimo di  $r$  bit) da  $x^r M(x)$  con sottrazione modulo 2
  - Il risultato è inserito nel frame come checksum
- Ricevente
  - Ricevuto il frame completo di checksum, lo divide per  $G(x)$
  - se il risultato non è 0 c'è stato un errore
- Rileva errori che non corrispondono a polinomi che contengono  $G(x)$

Livello data-link

S.Balsamo 2010

R4.19

## Livello Data Link - gestione errori 9/9

- Un codice polinomiale con  $r$  bit rileva tutti gli errori singoli e doppi, tutti gli errori di  $x$  bit,  $x$  dispari, tutti i burst di errori di lunghezza  $\leq r$
- Tra i polinomi sono diventati standard internazionali:
  - CRC-12:  $x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
  - CRC-16:  $x^{16} + x^{15} + x^2 + 1$
  - CRC-CCITT:  $x^{16} + x^{12} + x^5 + 1$
- Usato per IEEE 802:
  - $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$
- Un checksum a 16 bit corregge: errori singoli e doppi, di numero dispari di bit, burst di lunghezza  $\leq 16$ , il 99.997% di burst lunghi 17, e il 99.998% di burst lunghi 18 (sotto ipotesi di uniformità)

Livello data-link

S.Balsamo 2010

R4.20

## Livello Data Link - gestione sequenza 1/3

- Bisogna informare il mittente se i frame spediti sono anche arrivati al destinatario, con o senza errori:
  - **Servizi connectionless non confermati:** non c'è bisogno di alcuna conferma
  - **Servizi connectionless confermati:** sono arrivati tutti e senza errori?
  - **Servizi connection oriented confermati:** sono arrivati tutti, senza errori e nell'ordine giusto?

Livello data-link

S.Balsamo 2010

R4.21

## Livello Data Link - gestione sequenza 2/3

- L'**acknowledgement**, è un messaggio inviato dal destinatario al mittente per informarlo che:
  - Il frame è arrivato correttamente (**positive ack**)
  - Il frame è errato (**negative ack**)
- **Frame dati** indica un frame che trasporta informazioni generate nel colloquio fra le peer entity
- **Frame di ack** indica un frame il scopo è trasportare un acknowledgement

Livello data-link

S.Balsamo 2010

R4.22

## Livello Data Link - gestione sequenza 3/3

- **Problema:** Un frame può perdere e il mittente rimanere bloccato in attesa di un ack che non arriverà mai
  - **Soluzione:** Time-out del mittente per la ricezione dell'ack, ritrasmissione
- **Problema:** Se si perde l'ack, il destinatario può trovarsi più copie del frame
  - **Soluzione:** Il mittente numera la sequenza all'interno di ogni frame dati

Livello data-link

S.Balsamo 2010

R4.23

## Livello Data Link - gestione flusso 1/2

- Alcune assunzioni per il **mittente**:
  - Nei livelli fisico, data link e network, ci sono processi (**HW** o **SW**) indipendenti, che comunicano fra loro
  - Quando il **SW** di livello data link riceve un pacchetto dal livello network, lo incapsula in un **header** ed un **trailer** contenenti **informazioni di controllo**; quindi vengono calcolati il **checksum** e i **delimitatori**
  - Il frame viene passato al livello sottostante, che trasmette il tutto

Livello data-link

S.Balsamo 2010

R4.24

## Livello Data Link - gestione flusso 2/2

- In ricezione:
  - L'HW di livello 2 identifica i **delimitatori**, estraе il frame, ricalcola il **checksum**:
    - Se sbagliato, il SW data link viene informato dell'errore; altrimenti il SW data link riceve il frame (**senza più checksum**)
  - Il SW data link, quando riceve un frame, esamina le informazioni di controllo:
    - Se è tutto OK consegna **solo il pacchetto**, al livello network

Livello data-link

S.Balsamo 2010

R4.25

## Livello Data Link - frame

- E' diviso in due campi, il campo **header** e il campo **info**
- Nel campo **header** si possono individuare altri sottocampi:
  - **Kind**: Serve a distinguere il tipo di frame (contiene dati, di controllo, ecc.)
  - **Seq**: Contiene il numero progressivo del frame
  - **Ack**: Contiene informazioni legate all'acknowledgement
- Il campo **info** contiene un pacchetto di livello network completo (comprendente le informazioni di controllo di tale livello)

Livello data-link

S.Balsamo 2010

R4.26

## Livello Data Link - protocolli elementari

- **Protocollo 1**: per canale simplex senza vincoli
- **Protocollo 2**: per canale simplex **stop and wait**
- **Protocollo 3**: per canale simplex con canale rumoroso

Livello data-link

S.Balsamo 2010

R4.27

## Livello Data Link protocollo **heaven** 1/2

- Per canale **simplex**, basato sulle ipotesi (non realistiche):
  - I frame dati vengono trasmessi in una sola direzione
  - Le peer entity di livello network sono **sempre pronte**
    - non devono mai attendere per inviare o ricevere al/dal livello data link
  - Si ignora il tempo di elaborazione del livello data link
  - C'è **spazio infinito** per il buffering nel ricevitore
  - Il canale fisico non fa **mai errori**
- Assumiamo che il livello rete si sia sempre pronto a trasmettere e a ricevere

Livello data-link

S.Balsamo 2010

R4.28

## Livello Data Link - protocollo **heaven** 2/2

### ■ PROTOCOLLO 1 - canale simplex senza errori, spazio infinito

■ Mittente (loop infinito):

- 1) attendi un pacchetto dal livello network;
- 2) costruisci un frame dati;
- 3) passa il frame al livello fisico;
- 4) torna ad 1).

■ Destinatario (loop infinito):

- 1) attendi evento:  
\* arriva frame da livello fisico;
- 2) estraia pacchetto;
- 3) lo passa al livello network;
- 4) torna ad 1).

Livello data-link

S.Balsamo 2010

R4.29

## Livello Data Link protocollo **stop and wait** 1/2

■ Per canale simplex, con le stesse ipotesi precedenti eccetto:

- Esiste un **buffer** del destinatario con spazio **finito**
- Il mittente deve essere opportunamente rallentato
- Non con ritardi prefissati (caso pessimo)
- Il destinatario invia una esplicita autorizzazione all'invio del prossimo frame
- il mittente attende OK dal destinatario

Livello data-link

S.Balsamo 2010

R4.30

## Livello Data Link protocollo **stop and wait** 2/2

### ■ PROTOCOLLO 2 - canale simplex senza errori, spazio finito

■ Mittente (loop infinito):

- 1) attendi un pacchetto dal livello network;
- 2) costruisci un frame dati;
- 3) passa il frame al livello fisico;
- 4) attende evento:  
\* arriva frame di ack (vuoto);
- 5) torna ad 1).

■ Destinatario (loop infinito):

- 1) attendi evento:  
\* arriva frame dati da livello fisico;
- 2) estraia il pacchetto;
- 3) consegna il pacchetto al livello network;
- 4) invia un frame di ack (vuoto) al mittente;
- 5) torna ad 1).

Livello data-link

S.Balsamo 2010

R4.31

## Livello Data Link protocollo con **canale rumoroso** 1/8

■ Per canale simplex, con le stesse ipotesi del protocollo stop and wait eccetto che il **canale è soggetto ad errori**

■ Aggiungendo al protocollo precedente un timer per il mittente si ottiene che:

- il mittente invia un frame e fa partire un **timer**  
se non arriva l'ack entro il tempo **ripete** l'invio
- il destinatario invia un ack quando riceve un frame senza errori

■ C'è **possibile duplicazione** in caso di perdita dell'ack!

■ Il mittente aspetta un ack prima di trasmettere il prossimo frame

- PAR (Positive Ack with Retransmission)
- ARQ (Automatic Repeat Request)

■ Uso di variabili per memorizzare il numero di sequenza

- n\_seq numero di frame da inviare (mittente)
- n\_exp numero di frame atteso (destinatario)

Livello data-link

S.Balsamo 2010

R4.32

## Livello Data Link protocollo con canale rumoroso 2/8

**PROTOCOLLO 3 - canale simplex con errori, spazio finito**

Mittente (loop infinito; [seq] rappresenta il campo seq di un frame):

- 0)  $n\_seq = 1;$
- 1)  $n\_seq = 1 - n\_seq;$
- 2) attendi un pacchetto dal livello network;
- 3) costruisci frame dati e copia  $n\_seq$  in [seq];
- 4) passa il frame dati al livello fisico;
- 5) resetta il timer(seq);
- 6) attendi un evento:  
 \* timer scaduto: torna a 4)  
 \* arriva frame di ack (vuoto) non valido: torna a 4)  
 \* arriva frame di ack (vuoto) valido:  
 se  $ack == n\_seq$   
 ferma il timer(ack)  
 torna ad 1)

Livello data-link

S.Balsamo 2010

R4.33

## Livello Data Link protocollo con canale rumoroso 3/8

**PROTOCOLLO 3 - canale simplex con errori, spazio finito**

Destinatario (loop infinito; [seq] rappresenta il campo seq di un frame):

- 0)  $n\_exp = 0;$
- 1) attendi evento;  
 \* arriva frame dati valido da livello fisico:
- 2) se ( $[seq] == n\_exp$ )  
 2.1) estrai pacchetto  
 2.2) consegna al livello network  
 2.3)  $n\_exp = 1 - n\_exp$
- 3)  $ack = 1 - n\_exp;$
- 4) invia frame di ack (vuoto)
- 5) torna ad 1)  
 \* arriva frame non valido: torna ad 1)

Livello data-link

S.Balsamo 2010

R4.34

## Livello Data Link protocollo con canale rumoroso 4/8

- Il **mittente** etichetta i frame dati con la sequenza ...0,1,0,1..., ma passa all'etichetta e frame successivi solo quando arriva un ack; finché ciò non succede, continua a ritrasmettere lo stesso frame
- Il **ricevente** invia un ack di conferma per tutti i frame dati privi di errori, ma consegna al livello network solo quelli giusti, e cioè etichettati secondo la sequenza ...0,1,0,1....

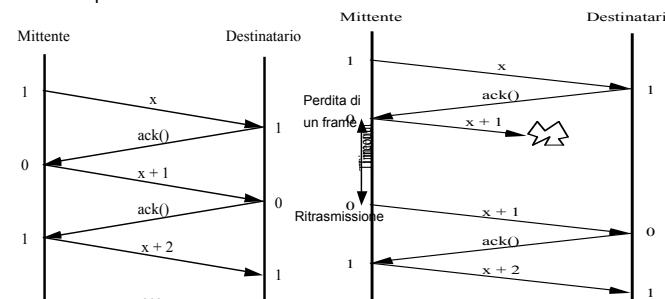
Livello data-link

S.Balsamo 2010

R4.35

## Livello Data Link protocollo con canale rumoroso 5/8

Esempi di trasmissione:

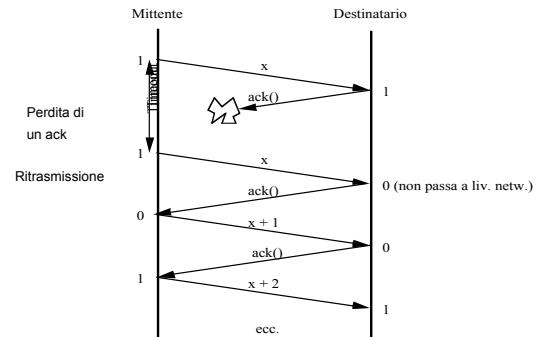


Livello data-link

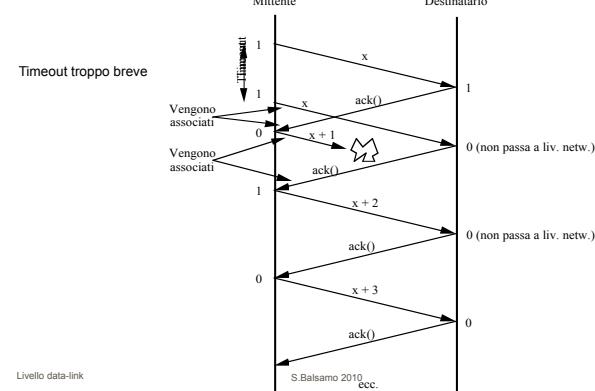
S.Balsamo 2010

R4.36

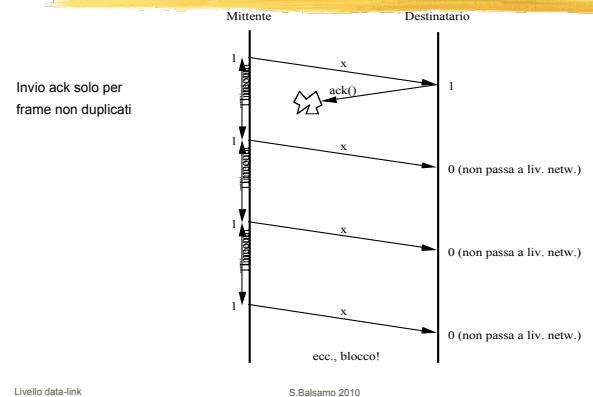
### Livello Data Link protocollo con canale rumoroso 6/8



### Livello Data Link protocollo con canale rumoroso 7/8



### Livello Data Link protocollo con canale rumoroso 8/8



### Livello Data Link protocolli a finestra scorrevole

- Protocollo a finestra scorrevole **di un bit**
- Protocollo a finestra scorrevole **basato su go-back-N**
- Protocollo a finestra scorrevole **basato su ripetizione selettiva**

Livello data-link S.Balsamo 2010 R4.40

## Livello Data Link protocollo a finestra scorrevole

1/6

■ Comunicazione sia fra A e B, si avrà che:

- Nella direzione da A a B viaggiano
  - i frame dati inviati da A a B e
  - i frame di ack inviati da A a B (in risposta ai frame dati inviati da B ad A)
- Nella direzione da B a A viaggiano
  - i frame dati inviati da B a A e
  - i frame di ack inviati da B a A (in risposta ai frame dati inviati da A ad B)
- Il campo **kind** serve a distinguere fra i due tipi di frame, dati e di ack, che viaggiano nella stessa direzione



■ Per inviare un ack da B ad A, si aspetta che sia pronto un frame dati che B deve inviare ad A, e si sfrutta tale frame dati per inserire anche l'ack di ritorno da B ad A (**piggybacking, letteralmente portare a spalle**)

Livello data-link

S.Balsamo 2010

R4.41

## Livello Data Link protocollo a finestra scorrevole

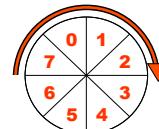
2/6

■ Ogni frame inviato ha un numero di sequenza, in  $[0, 2^n-1]$ , il campo **Seq** è di **n** bit

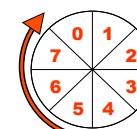
■ Finestra del **mittente** e finestra del **destinatario**

■ Es.  $n=3$

- numeri di sequenza  $[0,7]$
- finestra del mittente: da spedire o spediti ancora senza ack
- finestra del destinatario: da accettare



Finestra del mittente



Finestra del destinatario

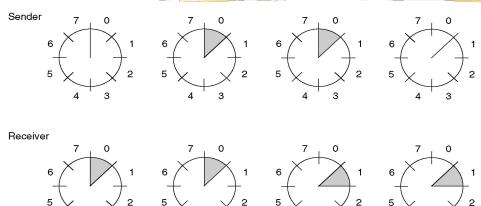
Livello data-link

S.Balsamo 2010

R4.42

## Livello Data Link protocollo a finestra scorrevole

3/6



Esempio: finestra

- (a) inizio
- (b) dopo la spedizione del primo frame
- (c) dopo che il primo frame è stato ricevuto
- (d) dopo che il primo ack è stato ricevuto

Livello data-link

S.Balsamo 2010

R4.43

## Livello Data Link protocollo a finestra scorrevole

4/6

■ Ad ogni istante il **mittente** mantiene una finestra scorrevole sugli **indici** dei frame:

- Solo quelli entro la finestra possono essere trasmessi
- I numeri di sequenza nella finestra indicano frame **da spedire o spediti**, ma non ancora confermati
- Quando **arriva** dal livello network **un pacchetto**, si aggiunge un nuovo indice nella finestra
- Quando **arriva** un **ack**, l'indice corrispondente esce dalla **finestra**
- I frame dentro la finestra devono essere **mantenuti** in memoria per la possibile **ritrasmissione**
- Se il **buffer** è **pieno**, il livello data link deve costringere il livello network a **sospendere** la consegna di pacchetti

Livello data-link

S.Balsamo 2010

R4.44

## Livello Data Link protocollo a finestra scorrevole

5/6

- il destinatario mantiene una finestra con gli indici dei frame che possono essere accettati
- se arriva un frame il cui indice è **fuori dalla finestra**, il frame viene **scartato** (*non si invia al livello superiore*)
- se arriva un frame il cui indice è nella finestra:
  - il frame viene accettato
  - viene spedito il relativo ack
  - la finestra del destinatario viene spostata in avanti
- La finestra del destinatario rimane di dimensione **costante**
- Solo se la dimensione è 1 è garantito l'ordine, altrimenti no

Livello data-link

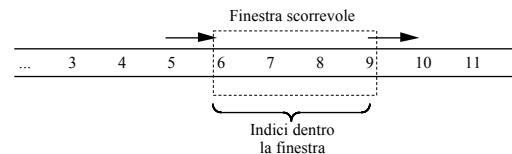
S.Balsamo 2010

R4.45

## Livello Data Link protocollo a finestra scorrevole

6/6

- I pacchetti devono essere riconsegnati ordinati al livello network
- Il canale fisico è **wire-like**, cioè consegna i frame nell'ordine di partenza
- Le finestre di mittente e destinatario non devono necessariamente avere uguali dimensioni, né uguali limiti inferiori o superiori



Livello data-link

S.Balsamo 2010

R4.46

## Livello Data Link protocollo a finestra scorrevole di 1 bit 1/4

- E' un protocollo **stop-and-wait**
- Il **mittente**, quando invia un frame:
  - fa partire un **timer**:
    - se prima che scada il timer arriva un ack con lo stesso numero di sequenza del frame che si sta cercando di trasmettere, si avanza la finestra e si passa a trasmettere il frame successivo
    - se arriva un ack diverso o scade il timer, si ritrasmette il frame
- Il destinatario quando arriva un frame corretto, senza errori:
  - Invia un **ack** col corrispondente **numero** di sequenza
  - Se il frame non è un duplciato lo passa al livello network e avanza la finestra
  - (*l'ack è etichettato col numero di sequenza del frame a cui si riferisce con valori che possono solo essere 0 e 1, come nel PROTOCOLLO 3*)
- Variabili che memorizzano il numero di frame
  - **n\_frame\_da\_inviare** come mittente
  - **n\_frame\_atteso** come destinatario

Livello data-link

S.Balsamo 2010

R4.47

## Livello Data Link protocollo a finestra scorrevole di 1 bit 2/4

- **PROTOCOLLO 4 - finestra scorrevole di un bit**
- Mittente e destinatario (loop infinito; [seq] e [ack] rappresentano risp. i campi seq e ack di un frame)
  - 0) **n\_frame\_da\_inviare = 0;**
  - 1) **n\_frame\_atteso = 0;**
  - 2) **prendi un pacchetto dal livello network;**
  - 4) **costruisci frame dati con**

```
[seq]= n_frame_da_inviare   e
[ack] = 1 - n_frame_atteso;
```
  - 5) **passa il frame dati al livello fisico;**
  - 6) **resetta il timer(seq);**
  - 7) **attendi un evento:**
    - \* **timer scaduto: torna a 4)**

Livello data-link

S.Balsamo 2010

R4.48

## Livello Data Link protocollo a finestra scorrevole di 1 bit 3/4

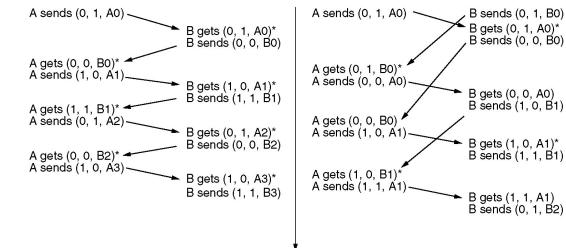
- 7) attendi un evento:
  - \* timer scaduto: torna a 4)
  - \* arrivo frame dal livello fisico
- 8) se (`[seq] == n_frame_atteso`)
  - 8.1) estra pacchetto
  - 8.2) lo consegna al livello network
  - 8.3) `n_frame_atteso = 1 - n_frame_atteso`
- 9) se (`[ack] == n_frame_da_inviare`)
  - 9.1) ferma timer(ack)
  - 9.2) prendi un pacchetto dal livello network;
  - 9.3) `n_frame_da_inviare = 1 - n_frame_da_inviare`
- 10) torna a 4)

Livello data-link

S.Balsamo 2010

R4.49

## Livello Data Link protocollo a finestra scorrevole di 1 bit 4/4



Esempi (a) Caso normale

Notazione (seq, ack, numero di pacchetto)

L'asterisco indica quando il livello rete accetta il pacchetto

Livello data-link

S.Balsamo 2010

R4.50

## Livello Data Link - protocollo go-back-n 1/7

- Il destinatario quando riceve un frame danneggiato o con un numero di sequenza non progressivo, lo ignora assieme a tutti i successivi, non inviando i relativi ack
- Il mittente quando il **scatta il time-out** sul frame inviato con errore, e poi su tutti quelli successivi (scartati dal destinatario), **rtrasmette** la sequenza di frame che inizia con quello per il quale si è verificato il time-out
- Il mittente mantiene in un apposito **buffer** tutti i frame non confermati per poterli rtrasmettere
- Se il buffer si riempie, il mittente deve bloccare il livello network fino a che non si ricrea dello spazio
- Vi è spreco di banda se il tasso d'errore è alto e/o il time-out è lungo

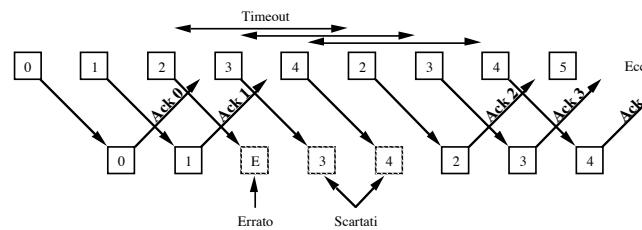
Livello data-link

S.Balsamo 2010

R4.51

## Livello Data Link - protocollo go-back-n 2/7

### esempio



Livello data-link

S.Balsamo 2010

R4.52

## Livello Data Link - protocollo *go-back-n* 3/7

- Il mittente può inviare più frame, fino a  $\text{MAX\_SEQ} = 2^n - 1$ , senza aspettare l'ack
- Assumiamo che
  - il livello rete **non** sia sempre pronto a trasmettere un pacchetto
  - se il livello rete ha un pacchetto da inviare lo segnala al livello data-link
  - di confrontare i numeri di sequenza secondo un ordinamento circolare in  $[0, 2^n - 1]$

Livello data-link

S.Balsamo 2010

R4.53

## Livello Data Link - protocollo *go-back-n* 4/7

- $[\text{ack\_atteso}, \text{frame\_da\_inviare} - 1]$  finestra del mittente (cardinalità  $< 2^n$ )
- $[\text{frame\_atteso}, \text{frame\_atteso} + 1]$  finestra del destinatario (cardinalità=1)

### Mittente e destinatario: loop infinito

```
Procedura per costruire ed inviare un frame
Static void send_data(seq_nr n_frame, seq_nr frame_atteso, packet
    buffer[])
{
    frame s;
    s.info = buffer[n_frame];
    s.seq = n_frame;
    s.ack = (frame_atteso + MAX_SEQ) mod (MAX_SEQ + 1);
    passa il frame dati al livello fisico;
    resetta il timer(n_frame);
}
```

Livello data-link

S.Balsamo 2010

R4.54

## Livello Data Link - protocollo *go-back-n* 5/7

- PROTOCOLLO 5 - finestra scorrevole go-back-n**
  - abilita il livello rete a segnalare eventi;
  - $\text{ack\_atteso} = 0$ ; futuro ack in ingresso
  - $\text{frame\_da\_inviare} = 0$ ;
  - $\text{frame\_atteso} = 0$ ;
  - $\text{in\_coda} = 0$ ;
  - attendi un evento
  - case of {
 **\* livello rete pronto**
    - 6.1) prendi un pacchetto dal livello network e ponilo in buffer;
    - 6.2)  $\text{in\_coda} = \text{in\_coda} + 1$ ; aumenta la finestra di invio
    - 6.3) `send_data(frame_da_inviare, frame_atteso, buffer)`
    - 6.4)  $\text{frame\_da\_inviare} = \text{frame\_da\_inviare} + 1$ ;

Livello data-link

S.Balsamo 2010

R4.55

## Livello Data Link - protocollo *go-back-n* 6/7

- \* arrivo frame dal livello fisico**
  - 7) se ( $\text{seq} == \text{frame\_atteso}$ )
    - 7.1) estrae pacchetto
    - 7.2) lo consegna al livello network
    - 7.3)  $\text{frame\_atteso} = 1 + \text{frame\_atteso}$
  - 8) while  $\text{ack\_atteso} \leq \text{ack} < \text{frame\_da\_inviare}$  do
    - 8.1)  $\text{in\_coda} = \text{in\_coda} - 1$ ;
    - 8.2) ferma timer( $\text{ack\_atteso}$ );
    - 8.3)  $\text{ack\_atteso} = \text{ack\_atteso} + 1$ ;
  - \* errori
  - ;

aumenta la finestra  
di ricezione

Controlla ack precedenti:  
elimina i frame in coda  
ferma i timer

Livello data-link

S.Balsamo 2010

R4.56

## Livello Data Link - protocollo *go-back-n* 7/7

```

* timer scaduto
  10) frame_da_inviare = ack_atteso;
  11) for i=1,..., in_coda
      11.1) send_data(frame_da_inviare,frame_atteso,buffer);
      11.2) frame_da_inviare = frame_da_inviare + 1;
  end of case)

  12) If in_coda < MAX-SEQ
      then abilita il livello rete
      else disabilita il livello rete
  13) Torna a 5).

```

Ritrasmette tutti i frame senza ack

Prepara il prossimo indice per il frame

Livello data-link

S.Balsamo 2010

R4.57

## Livello Data Link protocollo *selective repeat* 1/10

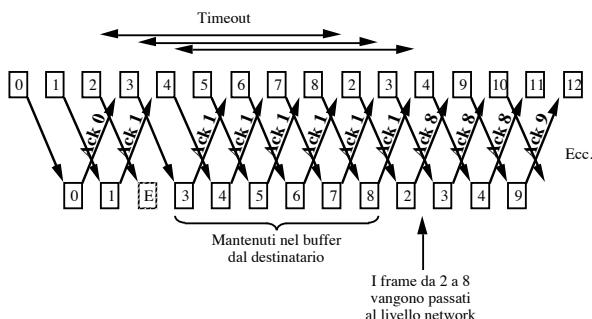
- Il **destinatario** mantiene nel suo **buffer** tutti i frame ricevuti successivamente ad un eventuale frame rovinato; non appena tale frame arriva nuovamente (senza errori), viene consegnato al livello rete insieme a tutti i frame successivi contigui mantenuti nel buffer
- Per ogni frame arrivato correttamente, il **destinatario** invia un **ack** col **numero più alto** della sequenza completa arrivata fino a quel momento
- Quando si verifica un **timeout**, il mittente rispedisce il frame corrispondente

Livello data-link

S.Balsamo 2010

R4.58

## Livello Data Link protocollo *selective repeat* 2/10



Livello data-link

S.Balsamo 2010

R4.59

## Livello Data Link protocollo *selective repeat* 3/10

- Nota:
    - mittente e destinatario devono entrambi gestire un **buffer** per mantenervi i frame:
  - non confermati (mittente)      successivi ad un errore (destinatario)
- 
- Per entrambi i precedenti protocolli:
    - è necessaria la gestione di **timer multipli** (uno per ogni frame inviato e non confermato)
    - Il ricevente, per inviare gli ack, usa il piggybacking se possibile, altrimenti invia un apposito frame

Livello data-link

S.Balsamo 2010

R4.60

## Livello Data Link protocollo *selective repeat* 4/10

- Con questo protocollo non necessariamente i frame sono ricevuti in ordine dal destinatario a livello data link
- Vi è un basso spreco di banda, che si può ulteriormente diminuire mandando un **NACK** (Negative ACKnowledgement) quando:
  - Arriva un frame danneggiato
  - Arriva un frame diverso da quello atteso (ciò può indicare l'avvenuta perdita del frame precedente)

Livello data-link

S.Balsamo 2010

R4.61

## Livello Data Link - protocollo *selective repeat* 5/10

Mittente e destinatario: loop infinito.  
 MAX-SEQ come per *go-back-n*. NBUF=(MAX\_SEQ + 1)/2  
`no_nak=true` non è stato invitato un nack  
*Procedure per costruire ed inviare un frame, un ack o nack*

```
Static void send_frame(tipo_frame tipo, seq_nr n_frame,
                      seq_nr frame_atteso, packet buffer[])
{
    frame s;
    s.kind = tipo;
    if (tipo == data) s.info = buffer[n_frame mod NBUF];
    s.seq = n_frame;
    s.ack = (frame_atteso + MAX_SEQ) mod (MAX_SEQ + 1);
    if (tipo == nak) no_nack = false; ← 1 solo nack per frame
    passa il frame dati al livello fisico;
    if (tipo == data) resetta il timer(n_frame mod NBUF);
    ferma timer_ack(); ← Non serve un frame di solo ack
    (si sta già inviando un frame)
}
```

Livello data-link S.Balsamo 2010 R4.62

## Livello Data Link - protocollo *selective repeat* 6/10

- `in_buf[NBUF], out_buf[NBUF]` pacchetti in arrivo e in partenza
- `arrivati[NBUF]` booleani, mappa di bit dei pacchetti arrivati
- $[ack\_atteso, frame\_da\_inviare - 1]$  finestra del **mittente**  
(cardinalità  $<(2^n/2)$ )
- $[frame\_atteso, sup - 1]$  finestra del **destinatario**  
(cardinalità  $<(2^n/2)$ )
- `timer_ack` se occorre inviare un frame di solo ack, timer associato
- `no_nak = true` se è stato inviato un nack
- `vecchio_frame=MAX_SEQ+1` traccia il frame più vecchio senza ack

Livello data-link

S.Balsamo 2010

R4.63

## Livello Data Link - protocollo *selective repeat* 7/10

### PROTOCOLLO 6 - finestra scorrevole selective repeat

- abilita il livello rete a segnalare eventi;
  - `ack_atteso = 0;` ← futuro ack in ingresso
  - `frame_da_inviare = 0;`
  - `frame_atteso = 0;`
  - `sup = NBUF;`
  - `in_coda = 0;`
  - per  $i=0, \dots, NBUF - 1 : arrivati[i] = false$
  - attendi un evento
- case of**
- \* **livello rete pronto**
    1. `in_coda = in_coda + 1;` ← aumenta la finestra di invio
    2. prendi un pacchetto dal livello network e ponilo in `out_buf[frame_da_inviare mod NBUF];`
    3. `send_frame(data, frame_da_inviare, frame_atteso, out_buf)`
    4. `frame_da_inviare = frame_da_inviare + 1;`
- Livello data-link S.Balsamo 2010 =====>>> R4.64

### Livello Data Link - protocollo selective repeat 8/10

```
* arrivo frame dal livello fisico
8) if ([kind] == data) then {
    9) if (([seq] != n_frame_atteso) and no_nack)
        then send_frame(nak, 0, frame_atteso,out_buf)
        else resetta ack_timer();
10) if ((ack_atteso <= seq < sup) and not arrivati[seq mod NBUF])
then {
    arrivo corretto nella finestra di ricezione
    estrai il pacchetto e ponilo in in_buf[seq mod NBUF];
    while arrivati[frame_atteso mod NBUF] do {
        consegna al livello network in buf[frame_atteso mod NBUF];
        no_nak = true;
        arrivati[frame_atteso mod NBUF]= false;
        frame_atteso = 1 + frame_atteso;
        sup = 1 + sup;
        resetta ack_timer();
    }
    serve un frame ack separato?
}
}
Livello data-link S.Balsamo 2010 R4.65 =====>>>
```

### Livello Data Link - protocollo selective repeat 9/10

```
* arrivo frame dal livello fisico (continua)
8) if ([kind] == data) then {
    ...
11) if (([kind] == nak) and
    (ack_atteso <=(ack+1 mod MAX_SEQ)<frame_da_inviare))
    then
send_frame(data, (ack + 1 mod MAX_SEQ),frame_atteso,out_buf)
    gestione ack in piggyback, nella finestra di invio
12) while ack_atteso <= ack < frame_da_inviare do
    12.1) in_coda = in_coda - 1;
    12.2) ferma timer(ack_atteso mod NBUF);
    12.3) ack_atteso = ack_atteso + 1;
    aumenta la finestra di invio
}
Livello data-link S.Balsamo 2010 R4.66
```

### Livello Data Link - protocollo selective repeat 10/10

```
* errori
13) if (no_nack) then
    send_frame(nak, 0, frame_atteso, out_buf)
* timer scaduto
14) send_frame(data, vecchio_frame, frame_atteso, out_buf)
* ack_timer scaduto
15) send_frame(ack, 0, frame_atteso, out_buf)
end of case
    invio frame di solo ack

16) If in_coda < N_BUF
    then abilita il livello rete
    else disabilita il livello rete
17) Torna a 7).

Livello data-link S.Balsamo 2010 R4.67
```

### Livello Data Link protocolli go-back-n e selective repeat

Dimensione della finestra  
grande → per elevato valore di banda x ritardo di trasmissione  
piccola per piccolo

Protocollo go-back-n

Protocollo selective repeat

Uso di tecnica di pipeline per inviare frame successivi

F	lunghezza frame	bit
B	banda	bps (bit per sec)
F/B	tempo di frame	sec
T	tempo di propagazione	

U	utilizzazione della linea (percentuale di utilizzo) nel caso più semplice
---	--

$$U = F / (F + T B) \quad \text{se } F < TB \quad \text{allora } U < 50\%$$

Livello data-link S.Balsamo 2010 R4.68

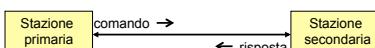
## Livello Data Link - alcuni protocolli protocollo **hdlc** 1/3

### **High-level Data Link Control**

- Standard ISO. Modificato LAP (Link Access Procedure) standard CCITT per X.25
- Protocollo **bit oriented**, che usa la tecnica del **bit stuffing**
- Con flag di 8 bit 01111110

### Comunicazione

- Punto a punto
- Per link multipunto



### Configurazioni

- Normale: **normal response mode (NRM)**

- Stazione primaria e secondaria
- Comunicazione punto a punto e multipunto



- Asincrona: **asynchronous balanced mode (ABM)**

- Configurazione bilanciata
- Comunicazione punto a punto



Livello data-link

S.Balsamo 2010

R4.69

## Livello Data Link protocollo **hdlc** 2/3

### Tre tipi di frame per diversi tipi di contenuti

- informazione** - per trasmissione dati e ack,
- supervisione** - per controllo
- non numerati** - per la gestione del sistema stesso

I-frame

S-frame

U-frame (*unnumbered*)

- Il tipo di frame ne definisce alcuni campi

### Campi del frame HDLC

Bit:	8	8	8	$\geq 0$	16	8
	01111110	Address	Control	Dati	Checksum	01111110

flag

flag

### I campi del frame hanno le seguenti funzioni:

- Address**: Identifica i diversi terminali (il protocollo offre funzioni per il dialogo fra un concentratore e diversi terminali)
- Control**: Contiene numeri di sequenza, ack, ecc.
- Dati**: Contiene i dati da trasportare
- Checksum**: per il rilevamento di errori, calcolato con **CRC-CCITT**

Livello data-link

S.Balsamo 2010

R4.70

## Livello Data Link protocollo **hdlc** 3/3

- Usa una finestra scorrevole con numeri di sequenza a  $n = 3$  bit, contenuti dentro un campo **Seq** del campo **Control**
- utilizza il campo **Next** (in Control), per il *piggybacking* degli ack
- ha tre tipi di frame (identificati dai primi due bit di Control)

Bits	1	3	1	3
(a)	0	Seq	P/F	Next
(b)	1	0	Type	P/F
(c)	1	1	Type	P/F

Information - trasmissione dati

Supervisory - per comandare diverse modalità di ritrasmissione

Unnumbered: per controllo o trasporto di traffico per servizi senza connessione e non affidabili

P/F bit di controllo, se posto a 1 indica

*Poll* se il frame va da una stazione primaria a secondaria  
*Final* se il frame va da una stazione secondaria a primaria

Livello data-link

S.Balsamo 2010

R4.71

## Livello Data Link protocollo **slip**

- Nato per collegare via modem macchine Sun ad Internet

- Spedisce sulla linea pacchetti IP terminati col byte **0xC0**. Usa **character stuffing**

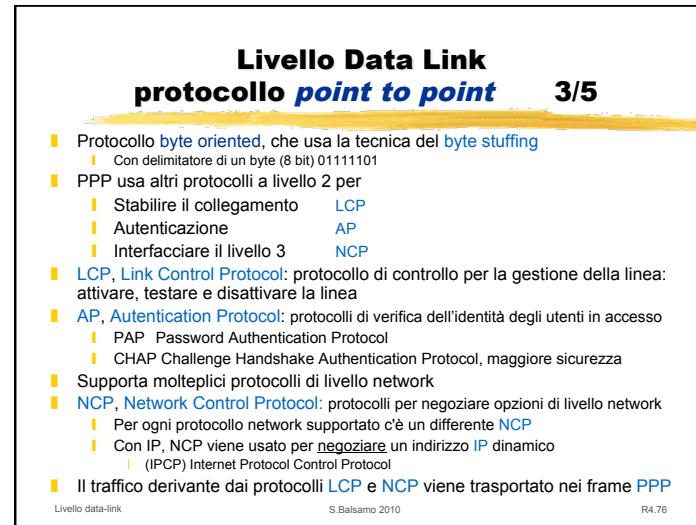
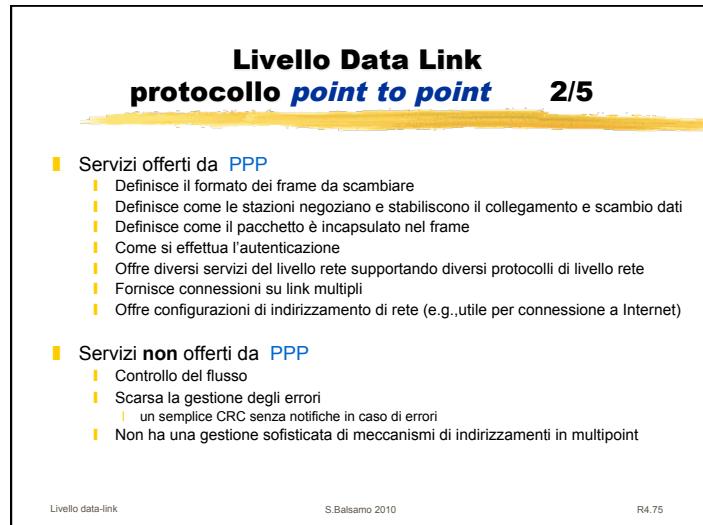
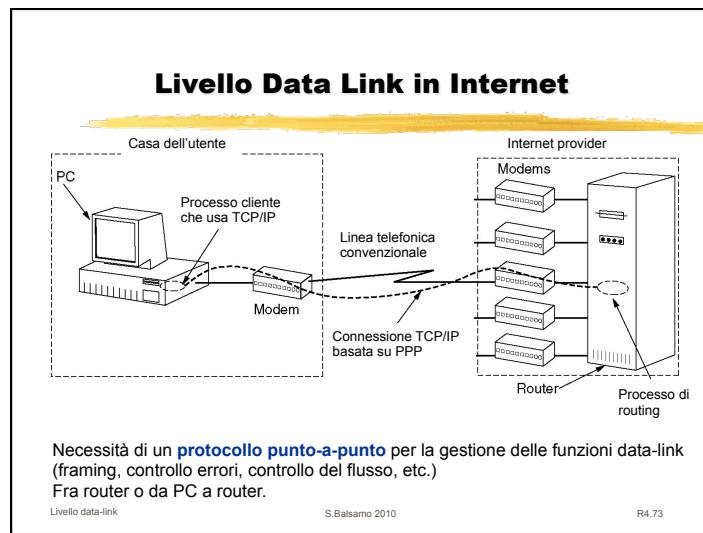
- Ha diverse limitazioni:

- Non c'è controllo degli errori**
- Supporta solo IP**, e per di più **solo indirizzi statici**
- Non è uno standard ufficiale di Internet**

Livello data-link

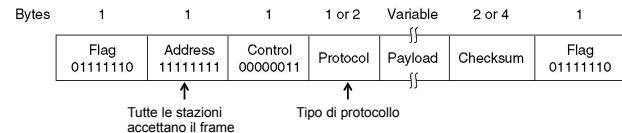
S.Balsamo 2010

R4.72



## Livello Data Link protocollo *point to point* 4/5

- Formato del frame PPP simile a quello per HDLC
- Byte stuffing sulle linee telefoniche
  - Esempio: formato del frame per il caso *Unnumbered*



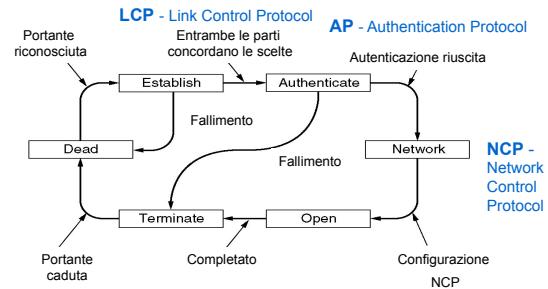
- PPP: meccanismo multiprotocollo per la trasmissione via *modem*, *linee seriali HDLC*, *SONET (Synchronous Optical Network)*, etc.
- Supporta: *negoziazione delle scelte*, *rilevazione errori*, *compressione dell'intestazione*, *possibile trasmissione affidabile*.

Livello data-link

S.Balsamo 2010

R4.77

## Livello Data Link protocollo *point to point* 5/5



Esempio di diagramma delle fasi di attivazione e di scollegamento di una linea

Livello data-link

S.Balsamo 2010

R4.78