Livello 2: Data-Link Control Layer

- ✓ Il livello 2 compie il seguente passaggio logico:

trasmissione di segnali attraverso un canale fisico { **trasmissione di dati attraverso un canale dati**

- ✓ In pratica la procedura di *data-link control* trasforma il mezzo trasmissivo in un *data-link.*
- ✓ Per avere una trasmissione dati efficiente tra due stazioni trasmittenti/riceventi occorre soddisfare una serie di requisiti e obiettivi non garantiti dal livello 1:

Livello 2: Data-Link Control Layer

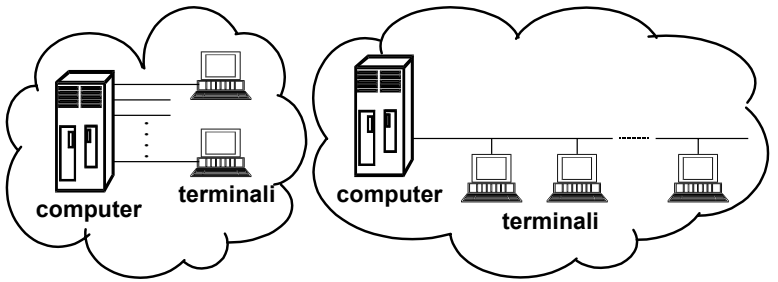
- ➢ **sincronizzazione dei frame**: identificazione di inizio e fine dei blocchi di dati trasmessi (*frames*)
- ➢ **configurazioni di linea**: possono essere di vario tipo
- ➢ **controllo di flusso**: per evitare che la stazione tx invii più dati di quanti la stazione rx sia in grado di ricevere
- ➢ **controllo d'errore**: per recuperare o identificare i bit errati introdotti dal sistema di trasmissione
- ➢ **indirizzamento**: identificazione delle stazioni tx/rx in un sistema multi-punto
- ➢ **gestione di dati e controlli sulla stessa linea**: utile in molte applicazioni (circuiti più semplici, cfr. ISDN)
- ➢ **gestione del link**: coordinamento tra le stazioni rx/tx

DLC: Configurazione della linea

- ✓ Vedremo in dettaglio le caratteristiche chiave gestite da un protocollo DLC, ovvero: configurazione di linea, controllo di flusso, controllo d'errore.
- ✓ Caratteristiche che distinguono le configurazioni di linea:
  - ➢ **TOPOLOGIA** (*topology*): organizzazione delle stazioni sulla linea di trasmissione
  - ➢ **FUNZIONAMENTO IN DUPLICE** (*duplexity*): direzione e temporizzazione del flusso di segnale su una linea
  - ➢ **REGOLAMENTAZIONE DI LINEA** (*line discipline*): insieme di regole e convenzioni da rispettare nell'uso della linea

Configurazione della linea: Topology e Duplex

- ✓ Topologie possibili:
  - ➢ point-to-point: connessione tra due stazioni
  - ➢ multi-point: connessione tra più di due stazioni



Configurazione della linea: Topology e Duplex

- ✓ Vantaggi della struttura *multi-point*:
  - ➢ il computer non necessita di una porta I/O per ogni terminale
  - ➢ non occorre avere una linea separata per ogni terminale
- ✓ Tipi di funzionamento in duplice:
  - ➢ ***simplex***: il flusso del segnale è monodirezionale
    - Esempi: sensori, lettori di schede (solo rx), stampanti (solo tx)
  - ➢ ***half-duplex***: trasmette e riceve ma non nello stesso tempo (*bidirezionale alternato*)
  - ➢ ***full-duplex***: trasmette e riceve anche in contemporanea (*bidirezionale simultaneo*)

NB. Nella tx digitale dipende dal numero di fili, in quella analogica dipende dalla frequenza

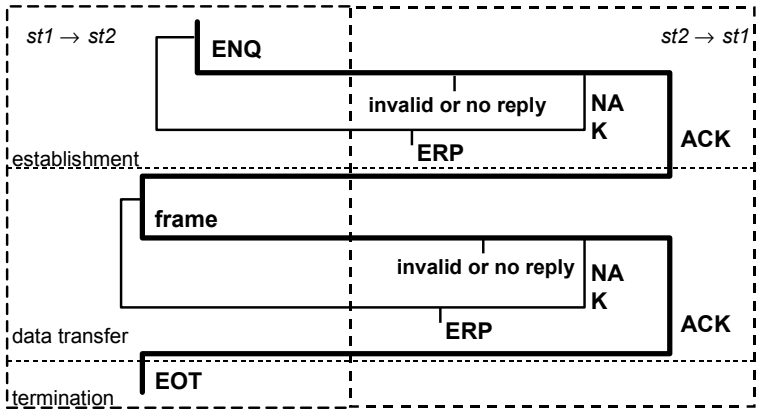
Configurazione della linea: Line discipline

- ✓ Si differenzia a seconda della topologia:

  POINT-TO-POINT
  - ➢ tre fasi: connessione, trasferimento dati, terminazione
  - ➢ caso half-duplex: una delle due linee inizia lo scambio:
    - Manda richiesta (**enq**) } **connessione**
    - Aspetta conferma (**ack**) }
    - Invia **dati** } **trasferimento dati**
    - Aspetta **ack** }
    - Invia un terminatore (**eot**) } **terminazione**
  - ➢ Si può avere conferma negativa (**nak**) (sistema non pronto o errore in tx) ⇒ procedure di recupero (**erp**).

Configurazione della linea: Line discipline

- ✓ Esempio (*point-to-point link control / half duplex*)



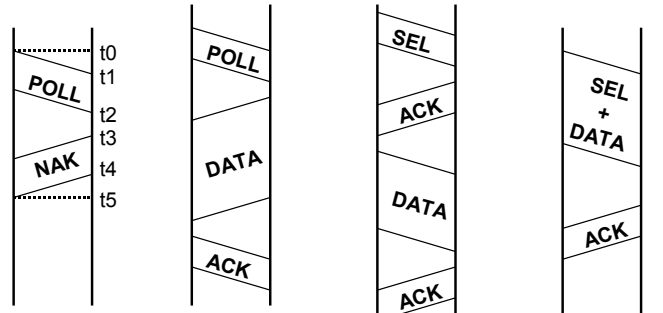
Configurazione della linea: Line discipline

MULTI-POINT

- ➢ dipende dalla configurazione:
  - primario - secondari: scambi diretti non possibili tra secondari
  - tutti alla pari: qualsiasi scambio possibile
- ➢ caso 1: le strategie note sono varianti dello schema poll&select
  - poll: il primario richiede dati da un secondario
  - select: il primario informa il secondario che ha dati da inviare
- ➢ caso 2: si utilizzano varie procedure, spesso la contention:
  - una qualsiasi delle stazioni può inviare dati verificando prima che la linea sia libera, altrimenti deve aspettare.

Line discipline: Poll-Select

- ✓ Schema poll-select: alcuni esempi di scambi



Line discipline: Poll-Select

- ✓ Schema 1: - la stazione primaria fa un *poll* sulla secondaria
  - la secondaria non ha nulla da inviare (risponde *nak*)
  - ➢ Calcoliamo il tempo totale richiesto dall'operazione:

$$T_N = t_{prop} + t_{poll} + t_{proc} + t_{ack} + t_{prop}$$

- t_{prop} = tempo di propagazione $(t_1 - t_0) = (t_5 - t_4)$
- t_{poll} = tempo per trasmettere un poll $(t_2 - t_1)$
- t_{proc} = tempo per elaborare il poll prima dell'ack $(t_3 - t_2)$
- t_{ack} = tempo per trasmettere un *ack* o un *nak* $(t_4 - t_3)$
- t_{data} = tempo per trasmettere i dati (*non presente in questo es.*)

Line discipline: Poll-Select

- ✓ Schema 2: - la stazione primaria fa un *poll* sulla secondaria
  - la secondaria invia dati (confermati con un *ack*)

$$T_P = 3\,t_{prop} + t_{poll} + t_{ack} + t_{data} + 2\,t_{proc} = T_N + t_{proc} + t_{data} + t_{prop}$$

- ✓ È comune il *roll-call polling* (interrogazione secondari in ordine predefinito (es. *round-robin*: $S_1, S_2, ..., S_n, S_1, ...$).

$$T_C = nT_N + kT_D$$ (tempo di ciclo)

- T_N = tempo medio per interrogare una secondaria
- T_C = tempo per un ciclo completo di polling
- T_D = tempo per trasmettere i dati $(t_{prop} + t_{data} + t_{proc})$
- n = numero totale di secondarie di cui k hanno dati da inviare

Line discipline: Poll-Select

- ✓ Schema 3: - la stazione primaria invia un select alla secondaria
  - dopo conferma, invia anche i dati e aspetta *ack*
  - ➢ NB: occorrono quattro tx separate per ogni trasferimento
- ✓ Schema 4: - la stazione primaria invia i select+dati e aspetta ack
  - ➢ conviene con messaggi brevi (in media risparmio tempo)

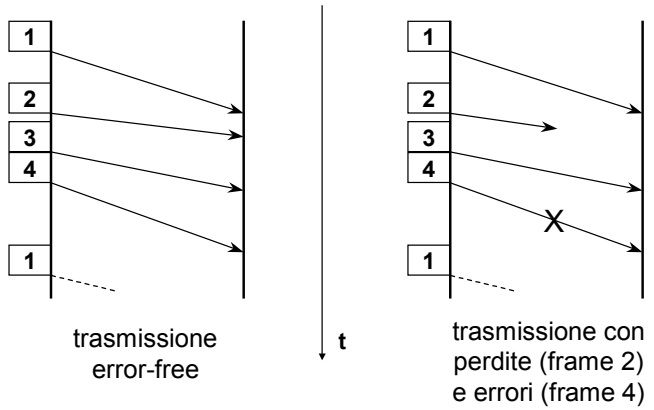
- ✓ Indirizzamento: tre possibilità
  - *point-to-point*: non servono indirizzi
  - *multi-point primario-secondari*: serve indirizzo secondario
  - *multi-point peer*: servono due indirizzi (entrambe le stazioni)
  - ➢ NB: normalmente la formula usata nei DLC è la seconda (anche per il point-to-point)

DLC: Controllo di flusso

- Tecnica utilizzata per evitare che la stazione trasmittente riempia il buffer di ingresso di quella ricevente (*overflow*).
- IPOTESI
  - per semplicità consideriamo il caso di *assenza di errori*
    - tutti i frame spediti arrivano a destinazione (no perdita)
    - nessun frame arriva con errori
    - i frame arrivano nello stesso ordine con cui sono stati spediti
    - l'unica incertezza è nel ritardo tra tx e rx.
  - usiamo i diagrammi a "sequenza temporale verticale"
    - la direzione temporale scorre dall'alto verso il basso
    - ogni freccia rappresenta un singolo frame (dati + informazioni di controllo) che attraversa un data link tra due stazioni

Modello a sequenza temporale verticale

- Esempi in assenza e presenza di errore:



Controllo di flusso: Strategia Stop & Wait

- È la forma più semplice di controllo di flusso
  - l'entità sorgente trasmette un frame
  - la destinazione indica la sua disponibilità a ricevere il frame successivo mandando una conferma del frame ricevuto
  - la sorgente aspetta l'*ack* prima di mandare un nuovo frame
- In pratica il destinatario esercita il controllo bloccando nuovi invii mediante mancata conferma.

Lo *stop&wait* funziona bene quando il messaggio è suddiviso in pochi pacchetti sufficientemente lunghi

Controllo di flusso: Strategia Stop & Wait

- In realtà di solito la sorgente spezza messaggi lunghi in insiemi di *frame* brevi, poiché:
  - messaggi lunghi ⇒ maggiore probabilità d'errore (perdita di efficienza, il pacchetto va ritrasmesso per intero)
  - su linea multi-point non posso permettere che una sorgente occupi il canale troppo a lungo senza interruzioni
  - il *buffer* in ricezione è in genere limitato
- In questi casi lo stop&wait si dimostra inadeguato.
- Per capire il perché è inadeguato dobbiamo studiare l'effetto del *ritardo di propagazione* sul *rate di trasmissione*.

Stop&Wait: Ritardo di Propagazione

- Massima efficienza possibile per una linea half-duplex con lo schema stop&wait:
  - messaggio trasmesso tramite sequenza di frame f_1 , f_2 ,..., f_n
  - la sequenza di eventi per la procedura di polling è:
    - S_1 manda un poll a S_2
    - S_2 risponde con f_1
    - S_1 manda conferma
    - S_2 manda f_2
    - S_1 manda conferma

      . . . . . .
    - S_2 manda f_n
    - S_1 manda conferma

Stop&Wait: Ritardo di Propagazione

- Il tempo totale richiesto per l'invio del messaggio sarà:

$$T_D = T_I + nT_F$$

  - T_I = tempo per iniziare la sequenza (t_{prop}+ t_{poll} + t_{proc})
  - T_F = tempo per mandare un frame ($2t_{prop}$+ t_{frame}+ $2t_{proc}$+ t_{ack})
- L'espressione precedente si può semplificare:
  - ignoriamo T_I (sequenza lunga)
  - t_{proc} tra trasmissione e ricezione trascurabile
  - tempo di conferma (t_{ack}) molto breve

$$T_D = n\,(2t_{prop} + t_{frame})$$

Stop&Wait: Efficienza

- Nella formula precedente, solo $n \times t_{frame}$ è tempo speso per trasmettere dati, il resto è ***overhead***
  - L'efficienza si misurerà come:

$$U = \frac{n \cdot t_{frame}}{n\left(2t_{prop} + t_{frame}\right)} = \frac{t_{frame}}{2t_{prop} + t_{frame}}$$

e definendo il parametro a = t_{prop} / t_{frame} :

$$U = \frac{1}{1+2a}$$

massima utilizzazione del link (supponendo frame di soli dati)

Stop&Wait: Efficienza

- "a" è costante se lo sono t_{prop} e t_{frame}, ovvero se:
  - frame di lunghezza fissa (spesso vero, a parte l'ultimo)
  - ritardo di propagazione costante (vero nel *p-to-p*)
  - nel multi-point si può approssimare considerando il max ritardo
- "a" può essere riscritto come:

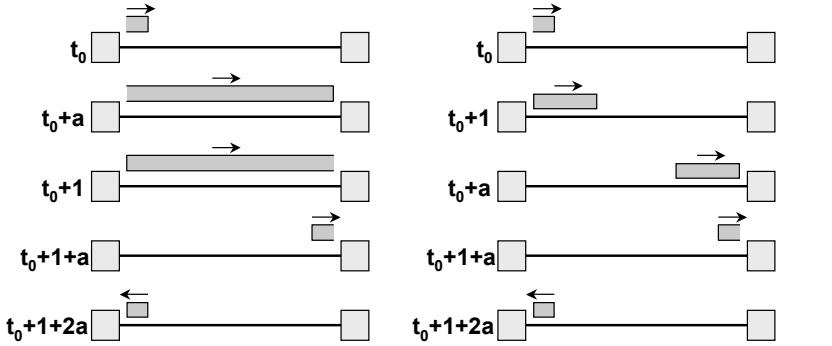
$$a = \frac{d/V}{L/R} = \frac{Rd}{VL}$$

- dove: d/V = tempo di propagazione (d=distanza, V=velocità) L/R = tempo di tx (L=lunghezza frame, R=data rate)

In altre parole "a" rappresenta la lunghezza in bit del mezzo (Rd/V) in confronto alla lunghezza del frame (L)

Stop&Wait: Efficienza (t_{frame}=1)

- ✓ Vediamo graficamente l'effetto di "a" sull'efficienza:



tempo di trasmissione = 1; a < 1 ***tempo di trasmissione = 1; a > 1***

Stop&Wait: Efficienza

- ✓ ESEMPIO 1: trasmissione satellitare
  - t_{prop} = 270 ms
  - data rate = 56 kbps
  - frame length = 4000 bit

  ⟹ tempo di tx frame = 71 ms
  ⟹ a = 270/71 = 3.8
  ⟹ efficienza = 0.12 **BASSA**

- ✓ ESEMPIO 2: reti locali
  - distanza = 0.1÷10 km
  - velocità = 2×10^8 m/s
  - data rate = 0.1÷10 Mbps
  - frame length = 500 bit

  ⟹ t_{prop} = d/V = 0.5 ÷ 50 µs
  ⟹ a = 10^{-4} ÷ 1
  ⟹ efficienza = 0.83 ÷ 0.98 **ALTA**
  (*situazione tipica*)

Controllo di flusso: Strategia Sliding Window

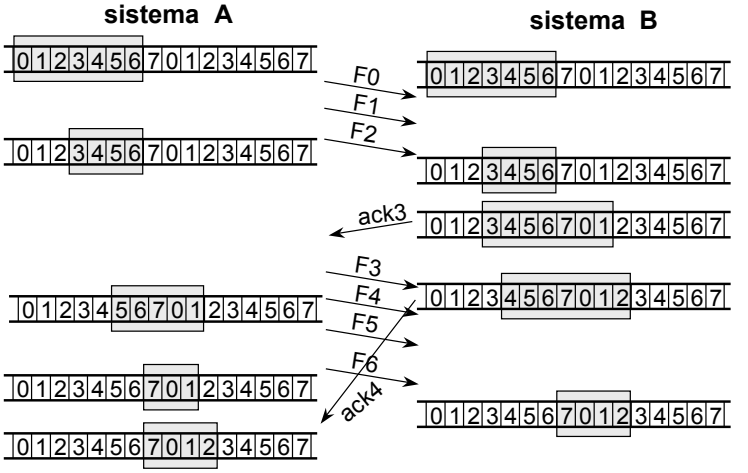
- ✓ Il problema di efficienza nel S&W è dovuto al fatto che può transitare sul link un solo frame alla volta.
- ✓ Per superare questo problema occorre:
  - allargare il buffer (n frame anziché 1)
  - consentire la trasmissione di *n* frame senza conferma
  - introdurre la numerazione sequenziale dei frame
- ✓ Il meccanismo usato è il seguente:
  - la stazione rx invia conferma specificando il numero del prossimo frame atteso (non necessariamente ogni frame)
  - l'*ack* indica anche che la stazione rx è pronta a ricevere n nuovi frame a partire da quello specificato

Controllo di flusso: Strategia Sliding Window

- ✓ Lo spazio all'interno del frame per il numero sequenziale è fisso ⟹ occorre ciclare:
  - Esempio: k bit ⟹ indice da 0 a (2^k-1) [modulo 2^k]
- ✓ Il nome 'sliding window' (finestra scorrevole) deriva proprio da questo: indica la finestra di frame che possono essere trasmessi:
  - quando trasmetto la coda avanza (la finestra si stringe)
  - quando ricevo conferma la testa avanza (la finestra si allarga, fino ad un massimo di 2^k)
  - analogamente dalla parte del ricevitore

Controllo di flusso: Strategia Sliding Window

Strategia Sliding Window: Esempio



Strategia Sliding Window: Efficienza

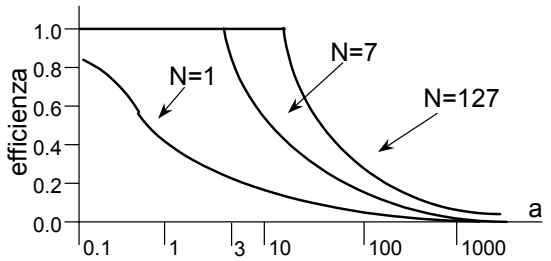
- ✓ L'efficienza in questo caso è proporzionale ad "a" e N (ampiezza della finestra)
- ✓ Supponendo il tempo di trasmissione di un frame unitario:
  - la stazione A inizia a emettere frame a t_0
  - il primo frame arriva in (t_0+a) e viene assorbito in (t_0+a+1)
  - dopo un tempo trascurabile la stazione B conferma
  - la conferma arriva ad A in (t_0+2a+1)
- ✓ Due casi possibili:
  - ***N > 2a+1***: *ack1* arriva prima di fine finestra [tx senza pause]
  - ***N < 2a+1***: A finisce la finestra in t_0+N e non può inviare altri frame fino a (t_0+2a+1) [tx viene bloccata]

Strategia Sliding Window: Efficienza

- ✓ Dalle osservazioni precedenti deriva che:

$$U = \begin{cases} 1 & N > 2a+1 \\ N/(1+2a) & N < 2a+1 \end{cases}$$

- ✓ Il grafico seguente mostra l'efficienza max in funzione di *a*



DLC: Controllo di Errore

- Meccanismo utilizzato per identificare e correggere errori eventuali nella trasmissione di frame.
- **Alcune IPOTESI**:
  - i dati sono inviati sequenzialmente e arrivano ordinati
  - il ritardo è arbitrario e variabile
  - questa volta però sono possibili due tipi di errori:
    - Perdita di frame (il frame trasmesso non arriva a destinazione, ovvero è talmente danneggiato da non essere riconoscibile)
    - Danneggiamento del frame (alcuni dei bit del frame sono alterati durante la trasmissione)

DLC: Controllo di Errore

- I metodi più comuni si basano su combinazioni di questi quattro meccanismi:
  - Detezione d'errore (tipicamente uso di codici, es. CRC)
  - Conferma positiva (la stazione ricevente invia conferma per i frame ricevuti senza errori)
  - Ritrasmissione dopo il *timeout* (se la sorgente non riceve conferma dopo un tempo prefissato, invia di nuovo il frame)
  - Conferma negativa e ritrasmissione (la destinazione invia una conferma negativa in caso di errore su un frame; la sorgente ritrasmette tale frame)
- Tali meccanismi vanno sotto il nome generale di **ARQ** (*automatic repeat request*)

DLC: Controllo di Errore

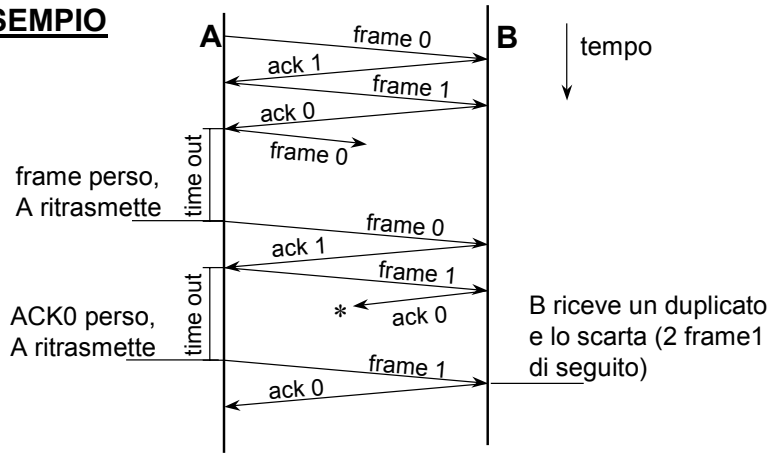
- I meccanismi di ARQ più noti sono i seguenti:
  - Stop & Wait ARQ
  - Go-back-N ARQ
  - Selective-Reject (o Selective-Repeat) ARQ
- Tali meccanismi fanno uso delle tecniche di controllo di flusso precedentemente definite, in congiunzione con tecniche di correzione d'errore.
  - Stop & Wait ARQ ⇔ Stop & Wait flow control
  - Go-back-N ARQ ⇔ Sliding Window flow control
  - Selective Reject ARQ ⇔ Sliding Window flow control

Controllo di Errore: Stop&Wait ARQ

- Il meccanismo è semplice: non è possibile trasmettere altro frame, finchè non c'è *ack* positivo dal ricevente:
  - se il frame viene perso, (dati o ack) viene di nuovo trasmesso dopo un timeout
  - se arriva errato, il ricevente risponde con un *nak* e il trasmettitore lo rimanda
- Un possibile problema è il seguente:
  - il frame è ricevuto corretto, ma l'ack non arriva. Il tx rimanda il frame, che viene ricevuto doppio
- Per evitare il problema, i frame sono numerati 0 e 1 in alternanza, così come gli ack (ack0 conferma un frame 1 e viceversa)

Controllo di Errore: Stop&Wait ARQ

ESEMPIO



Controllo di Errore: Go-back-N ARQ

- ✓ Trasmissione continua (*continuous ARQ*) in una finestra scorrevole. Se non ci sono errori il ricevente conferma.
- ✓ Esaminiamo tutti i possibili casi di errore:
  - ➢ Frame danneggiato → **A** trasmette il frame **i** e...

B rileva un errore in **i**. **i-1** era stato ricevuto corretto	⟹	**B** manda un *NAK-i* ad **A**. **A** rimanda i frame da **i** in poi
il frame **i** viene perso, **A** invia il frame **i+1** che arriva a **B**	⟹	**B** si accorge che **i+1** è fuori ordine e manda un *NAK-i*
il frame **i** viene perso e **A** non invia altri frame	⟹	**B** non invia alcuna conferma Al timeout **A** ritrasmette **i**

Controllo di Errore: Go-back-N ARQ

- ➢ Conferma positiva (*ack*) danneggiata → **B** riceve il frame **i** e manda un *ACK-i+1*, che è perso in trasmissione, quindi...

B invia un ack successivo prima del timeout di **A**	⟹	L'ack è cumulativo, accetta anche il precedente perduto
Il timer di **A** segnala il timeout	⟹	**A** ritrasmette frame **i** e tutti i seguenti (event. doppi)

- ➢ Conferma negativa (*nak*) danneggiata → **B** riceve il frame **i** errato e manda un *NAK-i*, perso in trasmissione, quindi...

Il timer di **A** segnala il timeout	⟹	**A** ritrasmette frame **i** e tutti i seguenti (event. doppi)

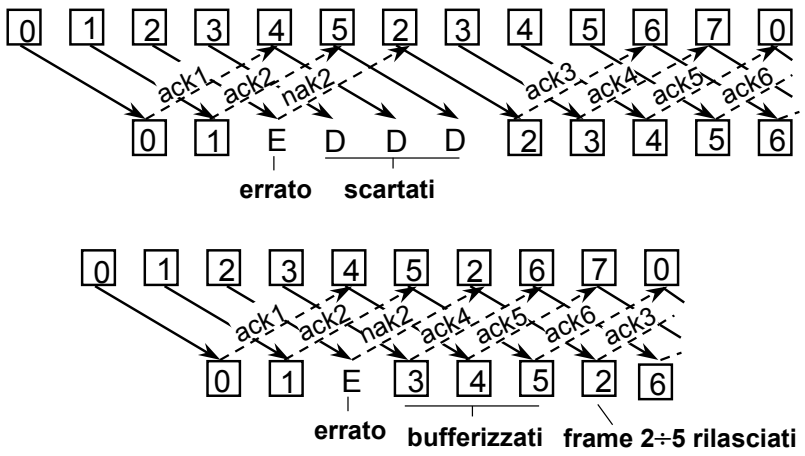
Controllo di Errore: Go-back-N ARQ

- ✓ Con il Go-back-N non è richiesto confermare ogni frame
  - ➢ Es. **A** tx ***fr0*** - **B** tx ***ack1*** - **A** tx ***fr1***, ***fr2*** , ***fr3*** - **B** tx ***ack4***. Con quest'ultimo, conferma anche *fr1* e *fr2*.
- ✓ Utilizzando *n* bit per il numero di sequenza, la dimensione della finestra è normalmente 2^n-1 anziché 2^n, infatti:
  - ➢ nei sistemi full-duplex, la conferma è spesso inviata all'interno di un messaggio per ridurre l'*handshake* (**piggybacking**)
  - ➢ spesso il campo è fisso, e quindi l'ack viene mandato anche quando non è necessario
  - ➢ in questi casi si usa una configurazione di bit particolare può servire per indicare che l'ack non è valido

Controllo di Errore: Selective-Reject ARQ

- ✓ Nel Go-back-N, la ritrasmissione di un frame implica la ritrasmissione anche di tutti i successivi già inviati.
- ✓ Questa inefficienza è eliminata nel Selective-Reject, che ritrasmette solo i frame che ricevono un *NAK* o un *timeout*
- ✓ In cambio di questi vantaggi, occorre che:
  - ➢ il trasmettitore contenga una logica più complessa per gestire la trasmissione di frame in ordine non corretto
  - ➢ il ricevitore disponga di un buffer per contenere i frame ricevuti dopo un *NAK* in attesa della ritrasmissione
  - ➢ il ricevitore contenga una logica per il riordinamento dei frame nella sequenza corretta

Confronto tra Go-back-N e Selective-Reject



Confronto tra Go-back-N e Selective-Reject

- ✓ Nonostante la maggiore efficienza, il selective-reject pone dei problemi molto 'sottili', che impongono delle limitazioni
  - ➢ Esempio:
    - **A** invia i frame 0-6 a stazione **B**
    - **B** riceve e conferma
    - le conferme vanno perse (es. a causa di un noise burst)
    - **A** va in timeout e ritrasmette il frame 0
    - **B** è già pronta ad accettare i frame 7,0,…,5, pensa che il frame 7 sia andato perso ed accetta e memorizza il 'nuovo' frame 0
  - ➢ Per evitare situazioni di questo tipo (sovrapposizione delle finestre in trasmissione e ricezione) occorre dimezzare la finestra ($2^n/2$ se n è il numero di bit per l'indice di sequenza)

Controllo di Errore: Efficienza tecniche ARQ

- ✓ <u>Efficienza massima nello Stop&Wait ARQ</u>
  - ➢ in assenza di errori $U = 1/(1+2a)$ (cfr. flow control stop&wait)
  - ➢ in presenza di errori le cose cambiano. Ridefinire U come:

$$U = T_f / T_t$$

T_f = tempo del tx per emettere un singolo frame
T_t = tempo totale per la tx di un singolo frame

data questa nuova definizione, in assenza di errori si ha:

$$U = T_f / (T_f + 2T_p)$$

T_p = tempo di propagazione

che in caso di errori si modifica in: $U = T_f / (N_r T_t)$

N_r = numero medio di trasmissioni del frame

Controllo di Errore: Efficienza tecniche ARQ

per uno Stop&Wait ARQ, si ha quindi:

$$U = 1/N_r(1+2a)$$

Introduciamo poi una semplice espressione per N_r basata sulla probabilità P che un frame contenga errori:

$$N_r = \sum_{i=1}^{\infty} i \cdot P^{i-1}(1-P) = 1/1-P$$

da cui ricaviamo infine, per lo S&W:

$$U = 1-P/1+2a \quad *$$

* vero posto che ACK e NAK siano sempre *error-free*

Controllo di Errore: Efficienza tecniche ARQ

- ✓ Efficienza massima nel Selective-Reject ARQ
  - si può ripetere il ragionamento fatto per lo S&W, si ricava l'espressione del metodo a finestra scorrevole, diviso per N_r:

$$U = \begin{cases} 1-P & N > 2a+1 \\ N(1-P)/1+2a & N < 2a+1 \end{cases}$$

- ✓ Efficienza massima nel Go-back-N ARQ
  - anche in questo caso il ragionamento è analogo, ma occorre calcolare con più precisione il valore di N_r, in quanto stavolta un errore implica la ritrasmissione di più frame:

$$N_r = \sum_{i=1}^{\infty} f(i) \cdot P^{i-1}(1-P)$$

numero di frame mediamente trasmessi per inviarne uno con esito positivo

Controllo di Errore: Efficienza tecniche ARQ

f(i) è il numero totale di frame trasmessi se il frame originale deve essere trasmesso *i* volte:

$$f(i) = 1 + (i-1)K = (1-K) + Ki$$

e sostituendo:

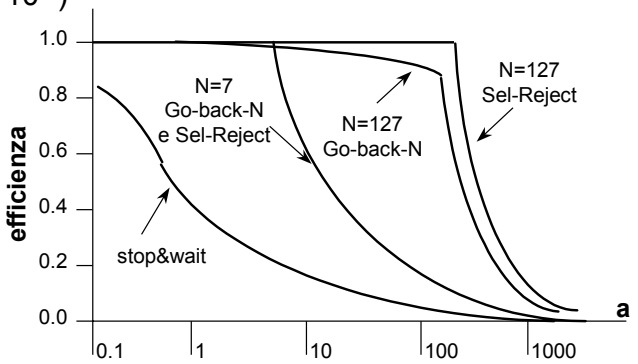
$$N_r = (1-K)\sum_{i=1}^{\infty} P^{i-1}(1-P) + K\sum_{i=1}^{\infty} iP^{i-1}(1-P) = \frac{1-P+KP}{1-P}$$

approssimando $K \cong (2a+1) \Leftrightarrow N > (2a+1)$ e $K \cong N \Leftrightarrow N < (2a+1)$:

$$U = \begin{cases} 1-P/1+2aP & N > 2a+1 \\ N(1-P)/(1+2a)(1-P+NP) & N < 2a+1 \end{cases}$$

Controllo di Errore: Efficienza tecniche ARQ

- ✓ Efficienza per varie tecniche di controllo di errore ($P=10^{-3}$)



DLC: Protocolli Standard

- ✓ I protocolli *bit-oriented* devono soddisfare vari requisiti:
  - gestione link *point-to-point* e *multi-point*
  - funzionamento in *full-duplex* e *half-duplex*
  - interazione primario-secondario o alla pari
  - link con valore di '*a*' grande o piccolo (trasmissione satellitare, link a breve distanza)
- ✓ Tra i protocolli più usati (*tutti molto simili*) possiamo citare:
  - **HDLC** (*High-level data link control*) [ISO3309-ISO4335]
  - **ADCCP** (*Advanced data comm. control proc.*) [ANSI X3.66]
  - **LAP-B** (*Link access procedure, balanced*) [CCITT X.25]
  - **SDLC** (*Synchronous data link control*) [IBM-proprietary]

Protocolli Standard di DLC: HDLC

- ✓ Ci limitiamo a descrivere in dettaglio l'HDLC in quanto:
  - ADCCP è concettualmente quasi identico all'HDLC
  - LAP-B è un sottinsieme dell'HDLC
  - SDLC è un sottinsieme dell'HDLC, con piccole aggiunte
- ✓ L'HDLC definisce:
  - 3 tipi di stazione (*primaria*, *secondaria* e *combinata*)
  - 2 configurazioni di link (*bilanciata* e *non bilanciata*)
  - 3 modi di trasferimento dati (*modo a risposta normale*, *modo asincrono bilanciato* e *modo a risposta asincrona*)
- ✓ Vediamo a che modalità di funzionamento corrispondono queste varie opzioni.

HDLC: Caratteristiche base

➢ tipi di stazione:

- **primaria**: ha il compito di controllare il funzionamento del link. I suoi frame sono detti comandi. Gestisce un link logico separato per ogni secondaria sulla linea.
- **secondaria**: opera sotto il controllo della stazione primaria. I suoi frame sono detti risposte.
- **combinata**: combina le caratteristiche di una stazione primaria e una secondaria. Può inviare sia comandi che risposte.

➢ configurazioni di link:

- **non bilanciato**: consiste in un primario e uno o più secondari. Supporta half- e full-duplex in point-to-point e multi-point.
- **bilanciato**: consiste in due stazioni combinate. Supporta half- e full-duplex ma solo in point-to-point.

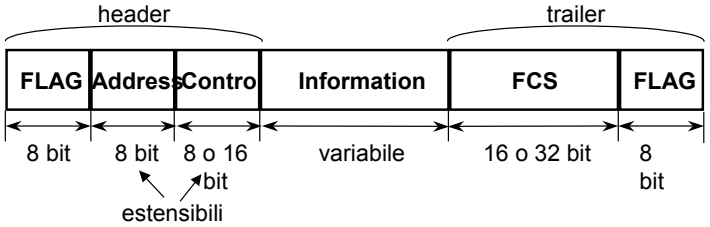
HDLC: Caratteristiche base

➢ modo di trasferimento dati :

- **NRM**: è una configurazione non bilanciata. Il primario può iniziare a trasferire dati a un secondario, mentre il secondario deve aspettare un *poll* dal primario e inviare i dati in risposta.
  - molto usato nelle linee *point-to-point*.
- **ABM**: è una configurazione bilanciata. Ognuna delle due stazioni combinate può iniziare la trasmissione senza bisogno di ricevere un permesso dall'altra.
  - efficiente nella gestione di link point-to-point full-duplex.
- **ARM**: è una configurazione non bilanciata, tuttavia il secondario può iniziare a trasmettere senza un esplicito permesso del primario. Il primario è comunque responsabile della gestione della linea (inizializzazione, controllo errore, ...)
  - poco usato in generale.

HDLC: Struttura del frame

- ✓ Vediamo ora la struttura dei frame in HDLC
  - ➢ L'HDLC utilizza una trasmissione sincrona
  - ➢ Tutto ciò che si trasmette è convertito in frame
  - ➢ Un solo formato di frame (di seguito schematizzato) è usato per qualsiasi tipo di messaggio (dati, controlli, ...):



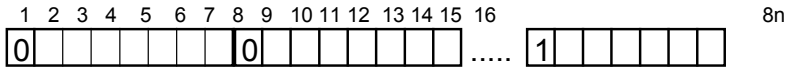
Struttura del frame HDLC: Campi Flag

- ➢ sono sequenze fisse di bit che delimitano il frame: **01111110**
- ➢ le stazioni rx cercano la sequenza flag per sincronizzarsi sull'inizio del frame o per determinarne la fine.
- ➢ un flag può indicare fine frame e inizio seguente: in caso di errore si ha uno *split* o un *merge* di due frame successivi.
- ➢ per evitare che nella parte dati ci sia una configurazione uguale al flag, si usa la tecnica di ***bit stuffing*** (inserimento di un bit '**0**' dopo cinque bit '**1**' consecutivi nella parte dati).

sequenza originale	11111111111011111101111110
sequenza bit-stuffed	111110111110110111110101111010

Struttura del frame HDLC: Campo Indirizzo

- ➢ È usato per identificare la stazione secondaria che ha trasmesso o sta per ricevere il frame.
- ➢ Non è richiesto nei link point-to-point.
- ➢ L'indirizzo è di solito 8 bit, ma può essere anche un multiplo di 7 bit (l'ottavo bit dice se il campo è finito o no)
- ➢ L'indirizzo **11111111** è usato per il broadcast



Esempio di indirizzo esteso

Struttura del frame HDLC: Campo Controllo

- ➢ L'HDLC definisce tre tipi di frame (identificati dai primi 1 o 2 bit), ognuno con un diverso formato di campo di controllo:
  - frame di informazione (I-frame): trasportano i dati utente ed eventualmente controlli di flusso ed errore (*piggybacking*)
  - frame di supervisione (S-frame): sono usati per l'ARQ nel caso non si impieghi il piggybacking.
  - frame non numerati (U-frame): usati per ulteriori funzioni di controllo del link.
- ➢ Anche il campo di controllo prevede un formato esteso (per *S-frame* e *I-frame*). I numeri di sequenza passano in questo caso da 3 a 7 bit

Struttura del frame HDLC: Campo Controllo

1	2	3	4	5	6	7	8	
0	N(S)			P/F	N(R)			I: Information
1	0	S		P/F	N(R)			S: Supervisory
1	1	M		P/F	M			U: Unnumbered

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	N(S)							**P/F**	N(R)							I: Information
1	0	S		0	0	0	0	**P/F**	N(R)							S: Supervisory

N(S) = send sequence number N(R) = receive sequence number
S = supervisory function bits M = unnumbered function bits
P/F = poll/final bit

Struttura del frame HDLC: Campi Dati e FCS

- ✓ Campo dati (informazione)
  - ➢ Il campo informazione è presente solo negli I-frame e talvolta nei non numerati.
  - ➢ Può contenere una sequenza di bit qualunque (lo standard non definisce una lunghezza massima che solitamente viene stabilita dalla specifica implementazione).
- ✓ Campo FCS (frame check sequence)
  - ➢ È un codice a correzione d'errore a 16-bit (CRC-16-CCITT)
  - ➢ In casi particolari (cattive prestazioni della linea, frame molto lunghi) si può usare un codice a 32 bit (CRC-32-CCITT)
  - ➢ È applicato a tutti i bit precedenti eccetto i FLAG.

HDLC: Funzionamento

- ✓ Il modo di operare dell'HDLC consiste nello scambio di frame di tipo I, S e U.
- ✓ Lo scambio può avvenire:
  - ➢ tra una stazione primaria e una secondaria
  - ➢ tra due stazioni primarie
- ✓ I frame possono contenere due tipi di istruzioni:
  - ➢ Comando (C)
  - ➢ Risposta (R)
- ✓ Vediamo in dettaglio quali tipi di comandi/risposte sono definiti in HDLC per ognuna delle tre tipologie di frame

Funzionamento HDLC: Frame di informazione

- ✓ Gli I-frame sono usati per trasferire i dati dell'utente.
- ✓ Sono di un solo tipo e possono essere usati sia per comando che per risposta (*piggybacking*).
- ✓ Oltre alle informazioni dell'utente contengono:
  - ➢ Il numero progressivo di frame inviato
  - ➢ L'acknowledge dell'ultimo frame ricevuto (ovvero numero del prossimo frame atteso); la dim. max finestra 7 o 127
  - ➢ Il bit poll/final:
    - in modo NRM è usato dal primario (C) per abilitare la trasmissione e dal secondario (R) per segnalare l'ultimo frame
    - in modo ARM o ABM è usato talvolta per coordinare lo scambio di frame S e U.

Funzionamento HDLC: Frame di supervisione

- ✓ Gli S-frame sono usati per controllo di flusso ed errore. Consentono ARQ go-back-N e selective-reject (più raro)
- ✓ Quattro tipi di istruzione, usati come comando o risposta:
  - Receive Ready (RR): *ack* e pronto a ricevere I-frame
  - Receive Not Ready (RNR): *ack* e blocca il flusso fino ad un RR
  - Reject (REJ): *nack*, applica go-back-N ARQ
  - Selective Reject (SREJ): *nack*, applica selective reject ARQ
- ✓ Oltre al comando, i frame contengono:
  - ➢ numero frame atteso (numero frame scartato per SREJ)
  - ➢ bit poll/final (se non disponibili I-frame per piggybacking):
    - in RR abilita tx (*prim.*, P=1) o segnala mancanza dati (*second.*, F=1)
    - in RNR sollecita un RR da parte di un secondario.

Funzionamento HDLC: Frame non numerati

- ✓ Gli U-frame sono usati per varie funzioni di controllo.
- ✓ Possono essere raggruppati in 4 categorie:
  - ➢ impostazione modo [*mode setting*] (C/R)
    - SNRM, SNRME, SARM, SARME, SABM, SABME, SIM, DISC, UA, RIM, RD
  - ➢ trasferimento di informazione [*information transfer*] (C/R)
    - UI, UP
  - ➢ recupero [*recovery*] (C/R)
    - FRMR, RSET
  - ➢ altri [*miscellaneous*] (C/R)
    - XID, TEST

Funzionamento HDLC: Frame non numerati

- ✓ Vediamo in dettaglio comandi/risposte per ogni categoria:
  - ➢ Impostazione modo:
    - SNRM/SNRME (C): imposta modo di risposta normale/esteso
    - SARM/SARME (C): imposta modo di risposta asincrono/esteso
    - SABM/SABME (C): imposta modo di risposta bilanciato/esteso
    - SIM (C): imposta modo di inizializzazione (la stazione che riceve il comando lancia una procedura di inizializzazione)
    - DISC (C): avvisa la staz. rx che la staz. tx sta per disconnetersi
    - UA (R): *ack* non-numerato, il bit F ha il valore del bit P ricevuto
    - DM (R): indica che la stazione è logicamente disconnessa
    - RIM (R): indica che la stazione non è pronta e richiede di far partire la procedura di inizializzazione
    - RD (R): richiede la disconnessione del link logico

Funzionamento HDLC: Frame non numerati

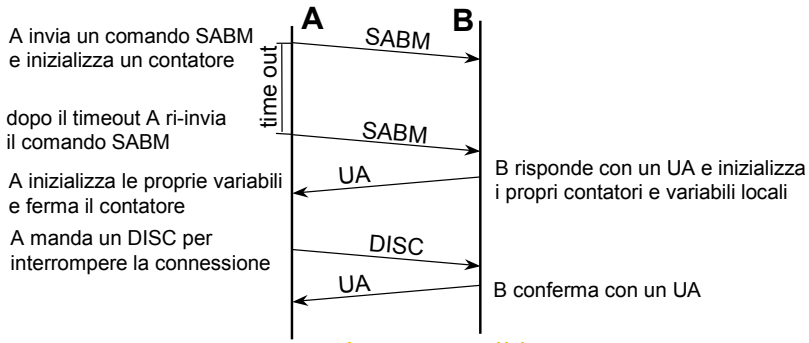
- ➢ Trasferimento informazione:
  - UI (C/R): scambio di informazioni tra stazioni (es. parametri di inizializzazione, interruzioni di funzionamento, sincronismo, ...)
  - UP (C): sollecita una risposta non numerata per verificare lo stato della stazione rx.
- ➢ Recupero
  - frame di comando e riposta usati quando il normale ARQ da problemi o non viene utilizzato
  - l'FRMR (R): segnala errori nel frame ricevuto:
    - campo di controllo non valido
    - campo dati troppo lungo
    - campo dati non consentito per il tipo di frame ricevuto
    - conteggio in ricezione non valido
  - RSET (C): ri-inizializza i contatori dopo un FRMR

Funzionamento HDLC: Frame non numerati

- Altri
  - XID (C/R): scambio di identificatori, usato per informazione reciproca dei numeri di identificazione e delle caratteristiche tra due stazioni (dipendente da implementazione).
  - TEST (C/R): usato per verificare lo stato della connessione e del funzionamento delle due stazioni collegate. La stazione che riceve un TEST deve farne eco appena possibile.

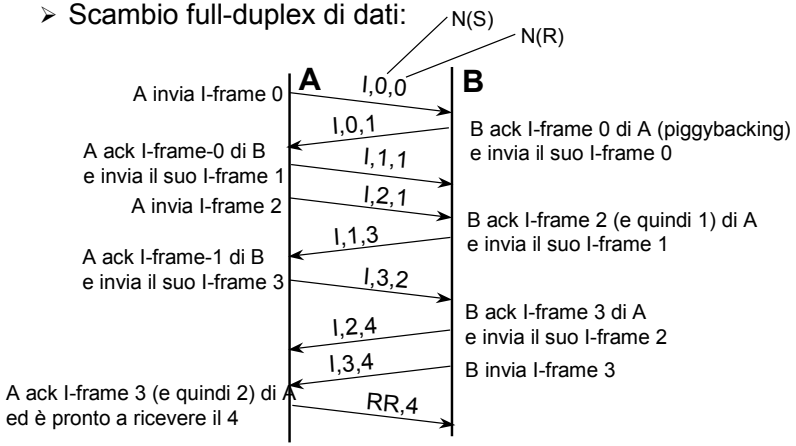
Funzionamento HDLC: ESEMPI

- ✓ Vediamo infine qualche esempio di operazione nel protocollo HDLC.
  - Creazione e disconnessione di un link:



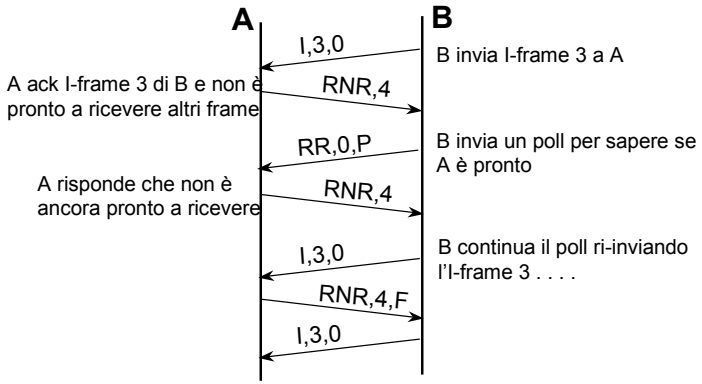
Funzionamento HDLC: ESEMPI

- Scambio full-duplex di dati:



Funzionamento HDLC: ESEMPI

- Gestione condizione 'occupato':



Funzionamento HDLC: ESEMPI

- Recupero di un errore tramite *reject*: