



UNIVERSITA' DEGLI STUDI DI PARMA  
Dipartimento di Ingegneria dell'Informazione

## Funzioni e protocolli

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Corso di Reti di Telecomunicazioni A, a.a. 2006/2007

<http://www.tlc.unipr.it/veltri>



Università degli Studi di Parma  
Dipartimento di Ingegneria dell'Informazione

Funzioni e protocolli

## Protocolli e funzioni

- Con il passare del tempo vengono definiti sempre nuovi protocolli con lo scopo di adattarsi a nuove esigenze
  - nuovi formati (ascii, binari, ASN.1, XML, etc.)
  - nuovi strati (soprattutto come sotto-strato del livello applicativo)
  - nuove applicazioni
- I principi su cui si basano e le funzioni base che possono essere realizzate rimangono quasi sempre le stesse

2



Università degli Studi di Parma  
Dipartimento di Ingegneria dell'Informazione

Funzioni e protocolli

## Principali funzioni

- Trasmissione/Ricezione
- Delimitazione delle UI
- Sequenzializzazione
- Multiplexing
- Accesso multiplo
- Indirizzamento
- Commutazione
- Rivelazione di errore
- Recupero di errore
- Controllo di flusso
- Controllo di congestione
- Compressione e/o criptaggio dei dati
- Autenticazione
- Gestione della mobilità
- altre..

## Delimitazione

3

## Delimitazione

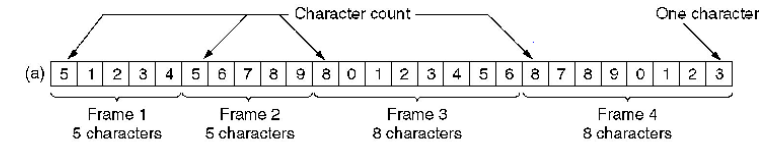
- Delimitazione delle UI all'interno di un flusso o blocco di bit/byte
  - per riconoscere/distinguere l'inizio e fine di ciascuna UI
- Funzione eseguita in genere quando uno strato si deve interfacciare con uno strato sottostante di tipo Stream oriented
  - in molti casi implementata sopra i protocolli di strato PH, ma anche sopra protocolli di trasporto come TCP, TLS o altri di tipo Stream
- Modalità di implementazione:
  - Conteggio di caratteri (byte) o delle cifre binarie
    - tramite campi indicatori della lunghezza della UI; oppure
    - tramite UI di lunghezza fissa
  - Inserimento di delimitatori (di inizio e fine)
    - inserimento di simboli speciali/riservati (e.g. manchester, 4/5, etc)
    - inserimento di simboli (sequenze di bit o byte) non riservati, + funzione bit- o char- stuffing
- Sono ovviamente possibili combinazioni delle stesse
  - esempio, delimitatori di inizio + conteggio dei caratteri per la fine

5

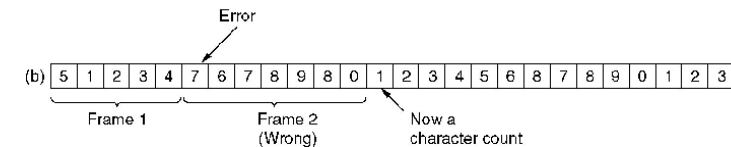
## Metodo del conteggio di caratteri

- Esempio
  - inserimento della lunghezza della UI come primo byte

- Esempio di comunicazione senza errori



- Esempio di comunicazione con 1 errore nel campo lunghezza



6

## Inserimento di simboli di inizio e fine UI

- L'inizio e la fine della UI vengono identificati con speciali campi delimitatori (detti anche flag) composti da particolari sequenze di bit o da caratteri di controllo
- Esempi:

Campo di delimitazione (01111110)

0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 0 0 0 1 0 0

Caratteri/flag di inizio (SF) e fine (EF)

SF					EF	SF				EF				SF			
----	--	--	--	--	----	----	--	--	--	----	--	--	--	----	--	--	--

- Problema:
  - bisogna evitare che i delimitatori appaiano all'interno delle trame

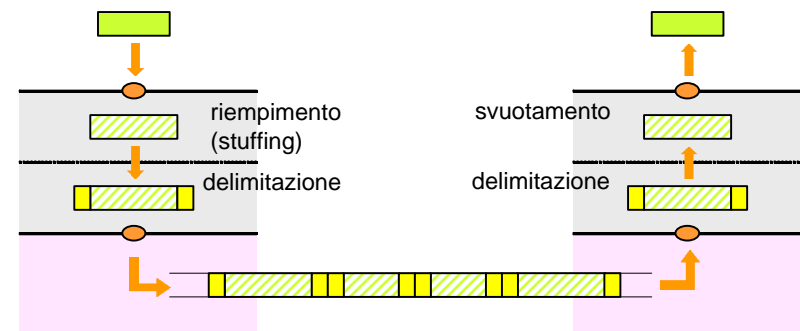
Soluzioni:

- 1) i delimitatori sono caratteri non ammessi (simboli riservati)
- 2) bit-stuffing/char-stuffing

7

## bit-stuffing & char-stuffing

- Obiettivo del bit/char stuffing: eliminare (mascherare) eventuali presenze dei simboli delimitatori all'interno della UI
- Normalmente gli algoritmi utilizzati operano sul blocco dati da inviare in modo sequenziale
  - operando in modo sequenziale sui simboli elementari (bit o byte) componenti la UI



8

## bit stuffing: esempio

- Flag di delimitazione di inizio e di fine: 01111110
- Algoritmo di bit-stuffing:
  - stuffing: dopo ogni cinque "1" consecutivi, aggiungere uno "0"
  - de-stuffing: dopo cinque "1" togliere lo "0" seguente
- Utilizzato da HDLC, LAPB, X.25

Sequenza originaria di cifre binarie

1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1

Sequenza dopo l'operazione di riempimento bit di stuffing

1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1

Sequenza inviata (e ricevuta) flag delimitatori

0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 . . . .

Sequenza dopo l'operazione di svuotamento

1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1

9

## char stuffing: esempio

- Char di delimitazione di inizio e di fine: END (decimal 192)
- Stuffing:
  - If a data byte is the same code as END char, a two byte sequence of ESC (decimal 219) + 220 is sent instead (i.e. 192 → 219+220)
  - If a data byte is the same code as ESC char, a two byte sequence of ESC+221 is sent instead (i.e. 219 → 219+221)

Sequenza originaria di cifre binarie

11	12	9	219	7	192	220	14
----	----	---	-----	---	-----	-----	----

Sequenza dopo l'operazione di riempimento char di stuffing

11	12	9	219	221	7	219	220	220	14
----	----	---	-----	-----	---	-----	-----	-----	----

Sequenza inviata (e ricevuta) flag delimitatori

192	11	12	9	219	221	7	219	220	220	14	192	. . . .
-----	----	----	---	-----	-----	---	-----	-----	-----	----	-----	---------

Sequenza dopo l'operazione di svuotamento

11	12	9	219	7	192	220	14
----	----	---	-----	---	-----	-----	----

10

## Esempi di delimitazione (1/2)

- RS-232
  - trasmissione seriale asincrona
  - delimitazione attraverso approccio misto
    - bit di start (1 transizione alto-basso) e conteggio dei bit (8 bits) per la fine
- PCM
  - trasmissione sincrona (plesiocrona)
  - delimitazione attraverso conteggio dei bit e dei bytes (allineamento di byte e allineamento di trama)
- Ethernet
  - delimitazione attraverso "simboli" speciali
    - codifica di manchester (Ethernet 10Mb/s): bit di start dopo preambolo e assenza di carrier per la fine della trama
    - codifica 4B/5B (Ethernet 100Mb/s): in altri casi, utilizzo di simboli riservati
- SLIP
  - delimitazione attraverso carattere di END alla fine (e opz. anche all'inizio)
    - utilizzo di char stuffing ESC (gli stessi dell'esempio precedente)

11

## Esempi di delimitazione (2/2)

- HDLC, LAPB, X.25
  - delimitazione tramite Flag (01111110)
    - utilizzo di bit stuffing
- HTTP
  - delimitazione tramite flag di fine message header (linea vuota = CRLF), più conteggio dei caratteri del payload
  - ovvero [RFC 2616]:
    - Any response message which does not include a message-body (such as the 1xx, 204, and 304 responses) is always terminated by the first empty line (CRLF) after the header fields
    - If a Content-Length header field is present, its decimal value in OCTETs represents the body (payload) length

12

## Controllo di sequenza

### Controllo di sequenza/Risequenzializzazione

- Recupero in ricezione della corretta sequenza delle UI inviate, in modo da consegnare le UI allo strato superiore nel corretto ordine
- Utilizzo di numeri di sequenza delle UI
  - **alle UI emesse viene aggiunto un numero di sequenza (in ordine crescente e modulo N)**
    - vengono contate direttamente le UI, oppure
    - vengono contati i byte emessi; in questo caso nelle UI viene inserito il numero di sequenza del primo byte trasportato (e.g. TCP)
- Sono possibili tecniche che fanno ricorso a memorizzazione in ricezione delle UI e successivo riordinamento, oppure tramite tecniche generali di recupero di errore

14

## Multiplazione

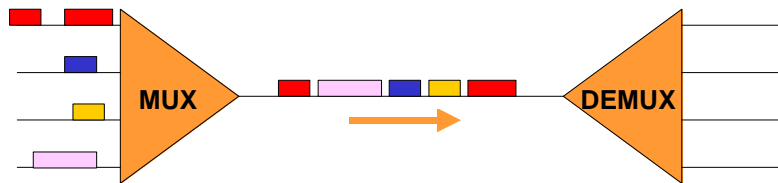
### Multiplazione

- Per una risorsa trasmissiva, definisce il modo secondo cui le UI condividono logicamente la capacità di trasferimento
  - **la risorsa trasmissiva può essere logica (coincidente con il servizio di trasferimento offerto dallo strato sottostante) o fisica (il canale trasmissivo)**
  - **viene svolta dalle entità alla pari estreme alla risorsa stessa**
- In generale tale funzione può essere vista come la multiplazione di più flussi all'interno di un unico flusso (flusso multiplato)
- Si applica ogni qualvolta
  - **un strato deve inviare su una uscita UI ricevute da ingressi differenti**
  - **uno strato deve gestire più flussi di strato superiore (relativi a protocolli differenti o ad un unico protocollo) multiplandoli nello stesso flusso di UI verso lo strato inferiore**
- La maggior parte dei protocolli offrono questa funzionalità
  - **e.g. PCM/PDH, SDH, Ethernet, IP, TCP, UDP**

16

## Multiplazione

- La funzione di multiplazione è in generale svolta, in modo cooperativo, da due entità operanti alle estremità del canale multiplato che svolgono rispettivamente il ruolo di multiplatore e di demultiplatore
  - il multiplatore deve provvedere affinché le UI emesse accedano al canale multiplato in modo ordinato (senza collisioni) e in accordo ad opportune strategie di assegnazione della risorsa ai singoli flussi (flussi tributari)
  - il demultiplatore deve essere in grado di identificare le UI che gli pervengono (funzione di identificazione e indirizzamento)



17

## Multiplazione

- Nel caso di protocolli di strato PH si possono distinguere i seguenti schemi di multiplazione:
  - Time Division Multiplexing (TDM)
  - Space Division Multiplexing (SDM)
  - Frequency Division Multiplexing (FDM)
  - Code Division Multiplexing (CDM)
- Nel caso di protocolli di livello superiore al PH, si considerano in genere solo schemi di multiplazione TDM
  - l'occupazione della risorsa da parte di una UI avviene in un intervallo di tempo che non si sovrappone con quelli relativi alle altre UI
  - Eventuali contese di utilizzazione possono essere risolte tramite code di attesa (buffer + algoritmi di scheduling) o perdita

18

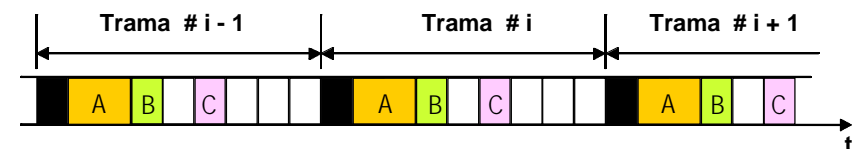
## Multiplazione

- Si possono evidenziare 2 differenti approcci TDM
  - Multiplazione statica (sincrona)**
    - i flussi vengono multiplati in modo sincrono (con una cadenza temporale regolare)
    - esempi, PCM/PDH, SDH
  - Multiplazione dinamica (asincrona/statistica)**
    - i flussi vengono multiplati tra loro senza una precisa cadenza temporale
      - strategia FIFO
      - strategie WFQ/CBQ
- Alla strategia di multiplazione può essere legata la funzione di identificazione delle UI (indirizzamento)
  - multiplazione sincrona → identificazione/indirizzamento implicito
  - multiplazione asincrona → identificazione esplicita (indirizzi)

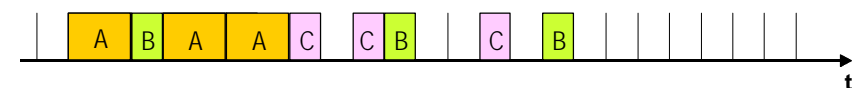
19

## Multiplazione sincrona e asincrona

Esempio di multiplazione sincrona



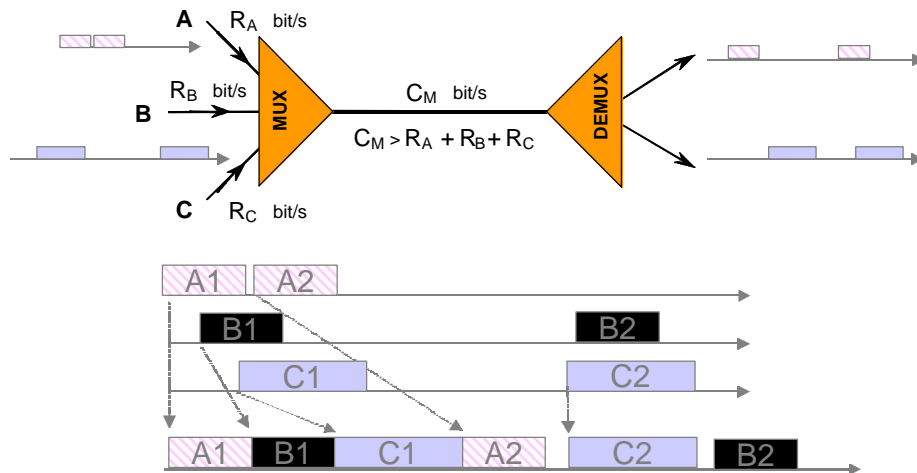
Esempio di multiplazione asincrona



(è necessario indirizzamento esplicito delle UI)

20

## Canale multiplato



21

## Controllo di Accesso Multiplo

## Accesso Multiplo

- Quando:
  - **accesso diretto ad un mezzo trasmissivo condiviso (a divisione di tempo)**
  - **la trasmissione è di tipo broadcast (un sistema trasmette e tutti gli altri possono ricevere)**
- Quando un sistema trasmette diventa proprietario temporaneamente dell'intera capacità trasmissiva
- Occorre un meccanismo per arbitrare l'accesso al mezzo trasmissivo: Controllo di accesso al mezzo
  - **Medium Access Control (MAC)**
- E' necessaria la presenza di indirizzi per stabilire chi sono il reale destinatario e il mittente della trasmissione

23

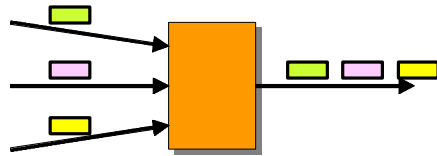
## Controllo di Accesso Multiplo (MAC)

- Simile alla funzione di moltiplicazione, ma in questo caso il flusso multiplato è gestito da entità distribuite (e non da un'unica entità concentrata)
  - **moltiplicazione → soluzione concentrata**
  - **accesso multiplo → soluzione distribuita**
- Tipico dei protocolli di accesso ad un mezzo fisico condiviso da diverse stazioni
  - **spesso nelle LAN**
  - **e.g. controllo di accesso ad bus, ad una risorsa radio, etc.**
- Modalità
  - **discipline ad accesso controllato (senza collisione)**
  - **discipline ad accesso casuale (con collisione)**

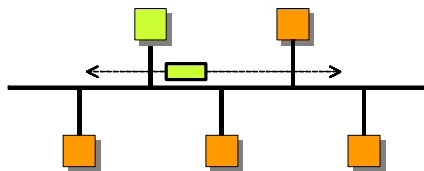
24

## Multiplicazione e accesso multiplo

- Multiplicazione



- Accesso multiplo



25

## Controllo di Accesso Multiplo (MAC)

- Modalità

- **Accesso controllato (senza collisione)**

- un nodo prima di trasmettere acquisisce esplicitamente il controllo della rete (ovvero il diritto a trasmettere)
- non possono verificarsi 'collisioni'
- spesso gestito tramite la presenza di un *token*

- **Accesso casuale (con collisione)**

- non c'è un'acquisizione esplicita del diritto a trasmettere
- possono verificarsi collisioni
- le collisioni possono essere gestite in modo diverso a seconda del tipo di protocollo

- Esempi: Aloha, CSMA/CD (Ethernet), CSMA/CA (WiFi), FDDI

26

## Accesso controllato

- Si distinguono i casi di:

- controllo centralizzato

- Una delle stazioni (**primaria**) provvede ad abilitare ognuna delle altre (**secondarie**) ad emettere

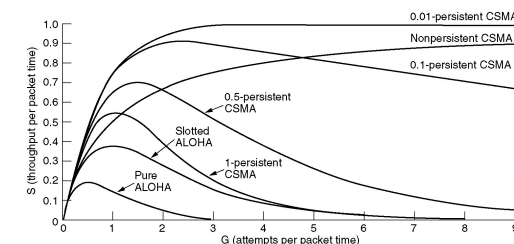
- controllo distribuito

- Il controllo passa ordinatamente da stazione a stazione
- Le stazioni non attive non assorbono risorse
- Si consegue un'efficiente ripartizione della capacità di trasferimento del mezzo di comunicazione fra le sole stazioni attive
- tipico dei protocolli a token (e.g. Token Ring, FDDI)

27

## Protocolli ad accesso casuale

- Ogni stazione emette quando ha una UI pronta
- Se il mezzo è libero la emissione ha successo, altrimenti (collisione) occorre riprovare successivamente
- Se il traffico generato dalle stazioni aumenta, cresce anche il numero delle collisioni
  - **ciò può limitare fortemente il traffico globale smaltito dal sistema**



28

## Indirizzamento

### Identificazione/indirizzamento

- Identificazione delle UI in modo da determinare
  - **entità sorgente**
    - indispensabile per determinare il mittente
  - **entità di destinazione**
    - indispensabile per la funzione di commutazione e demultiplazione
- Deve essere possibile determinare i SAP sorgente e destinazione (indirizzo del S-SAP e del D-SAP) o gli eventuali CEP (Connection End Point)
- Modalità
  - **indirizzamento implicito**
    - in base alla posizione spaziale e/o temporale della UI all'interno di una struttura di multiplazione (trama multiplata)
    - eventualmente con l'ausilio di opportuni puntatori e unità di riempimento/stuffing (per ridurre le esigenze di sincronismo)
    - e.g. PCM/PDH, SDH

30

### Identificazione/indirizzamento

- Modalità (cont.)
  - **indirizzamento esplicito**
    - presenza nella UI dell'indirizzo completo del SAP/CEP
    - oppure, tramite la presenza nella UI di un identificatore di circuito virtuale (VCI, label, etichetta), a cui è associata la coppia di SAP/CEP
    - esempi di indirizzamento completo: Ethernet, IP
    - esempi di VCI: X.25, ATM, MPLS
- I protocolli CO, hanno inoltre la necessità di identificare la sorgente e destinazione in fase di instaurazione
  - esempio: numeri di telefono nella rete telefonica

31

### Piani di indirizzamento

- Esistono differenti piani di indirizzamento
- Questi possono essere di tipo
  - **gerarchico su base geografica (e.g. PSTN)**
  - **gerarchico su base non geografica (e.g. Ethernet)**
- Possono essere di tipo
  - **numerico**
    - e.g. numeri di telefono, indirizzi IP
  - **mnemonico**
    - e.g. URI/URL (esempio: <http://www.unipr.it/veltri>)

32



## Commutazione

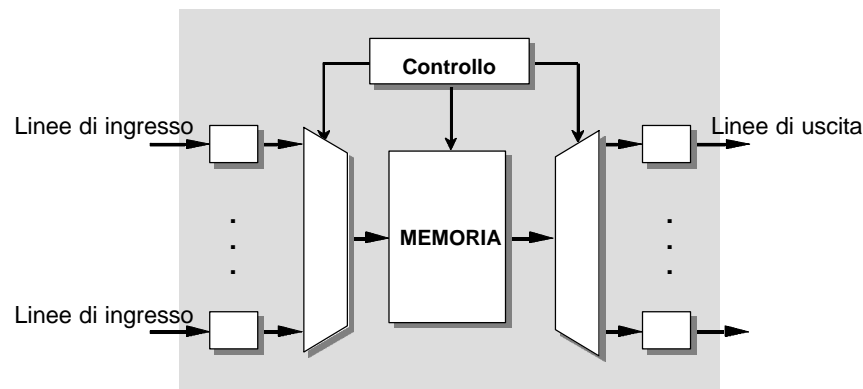
- E' la funzione che permette alle UI di attraversare i nodi di rete (nodi di commutazione), seguendo un percorso di rete dal terminale sorgente al terminale/i di destinazione
- Consiste nell'associazione logica tra una terminazione d'ingresso e una particolare terminazione d'uscita del nodo per la durata necessaria al trasferimento della UI stessa



34

## Nodi di commutazione

- Schema generale di un commutatore a divisione di tempo



35

## Commutazione

- Si compone di due funzioni:
  - **INSTRADAMENTO (Routing)**, funzione decisionale (intelligente) che ha lo scopo di stabilire il ramo di uscita verso cui deve essere inoltrata la UI pervenuta da un ramo d'ingresso
    - avviene attraverso la consultazione di opportune tabelle di instradamento (tabelle di routing)
    - tali tabelle possono essere configurate staticamente o dinamicamente
    - l'istradamento dipende dalla UI (in genere dipende dal destinatario/i della UI)
    - tale funzione può introdurre un ritardo di elaborazione
  - **ATTRAVERSAMENTO (Forwarding)**, funzione attuativa (più "meccanica", spesso implementata in HW), che ha lo scopo di trasferire una UI da un ramo d'ingresso ad uno di uscita in accordo a quanto deciso dalla funzione di instradamento
    - può avvenire direttamente (tramite un percorso interno), o attraverso un immagazzinamento e rilancio (ogni UI viene memorizzata prima di essere rilanciata verso l'uscita)
    - è caratterizzato da un ritardo di attraversamento che può essere costante o variabile

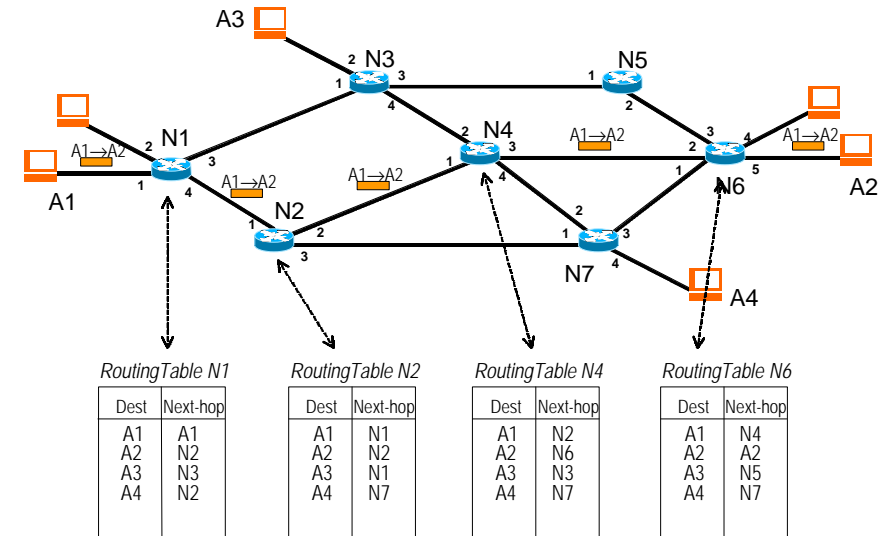
36

## Commutazione

- La funzione di instradamento può essere effettuata:
  - durante la fase di instaurazione della chiamata (per servizi di trasferimento orientati alla connessione)
  - indipendentemente per ciascuna UI (per servizi di trasferimento senza connessione)
- Ciò corrisponde a due differenti modalità di commutazione
  - Commutazione CO (utilizzata solo da protocolli CO)
  - Commutazione CL
- Nel caso di commutazione CO
  - a seguito della funzione di instradamento il risultato viene memorizzato in una tabella di switching (commutazione), consultata durante la fase di attraversamento delle UI
  - nel caso di moltiplicazione statistica le UI vengono identificate mediante identificatori di circuito virtuale (VCI, Label, etichette), associati alle UI link per link (label switching)

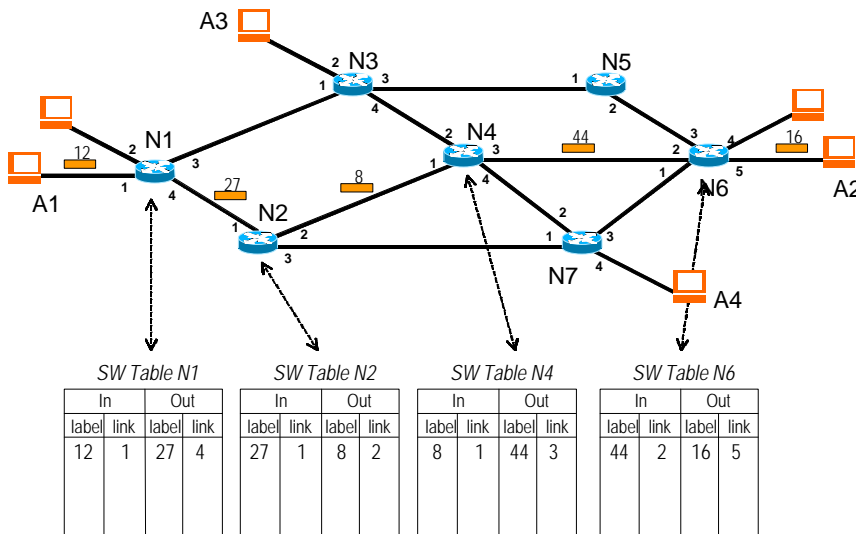
37

## Esempio di Commutazione CL (a datagramma)



38

## Esempio di Commutazione CO



39

## Commutazione

- La funzione di instradamento (routing) viene in genere effettuata tramite consultazione di opportune tabelle (di instradamento)
  - Routing Table
- Un aspetto importante legato all'istradamento è la modalità con cui le tabelle vengono calcolate e aggiornate
  - routing statico
    - le tabelle sono create e mantenute staticamente su base gestione della rete (spesso attraverso una configurazione manuale delle stesse)
  - routing dinamico
    - le tabelle sono aggiornate automaticamente dai nodi sulla base di informazione di routing scambiata periodicamente dai nodi stessi (protocolli di routing)
- Esempi di protocolli di routing:
  - Spanning Tree (Ethernet),
  - RIP, OSPF (IP)

40

## Controllo di errore (rivelazione, correzione, e recupero)

- La funzione di rivelazione di errore permette in ricezione di rilevare eventuali errori nelle UI ricevute (in genere errori trasmissivi)
  - controllo di parità
  - controllo di parità a blocchi
  - codici rivelatori d'errore polinomiali
  - e.g. campo CRC di Ethernet, IP, UDP, etc
- La funzione di correzione di errore ha il compito di correggere all'interno delle UI eventuali bit o byte errati
  - Controllo di errore in "avanti" o FEC (Forward Error Correction)

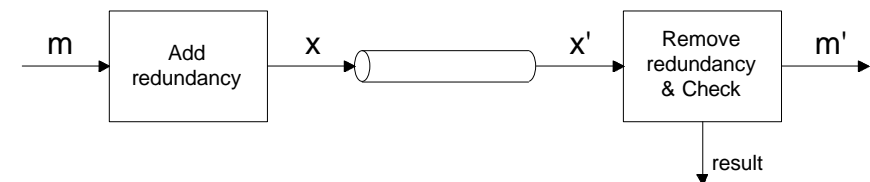
42

## Rivelazione di errore

- Aggiunta di ridondanza in trasmissione per ogni UI (codice a rivelazione di errore) → tra i campi del PCI
- Uso della ridondanza in ricezione per rivelare la presenza di errori e non per correggerli
- Stesso principio dell'uso di MIC (Message Integrity Check), o MAC (Message Authentication Code), usato come protezione da attacchi che tentano di modificare i dati scambiati
- Eventualmente uso di messaggi di servizio per la conferma di corretta ricezione o per la richiesta di ritrasmissioni → vedi recupero di errore
- La ridondanza richiesta per rivelare gli errori è molto più contenuta di quella eventualmente richiesta per la correzione
  - un'aggiunta di 16-32 bit è in genere sufficiente

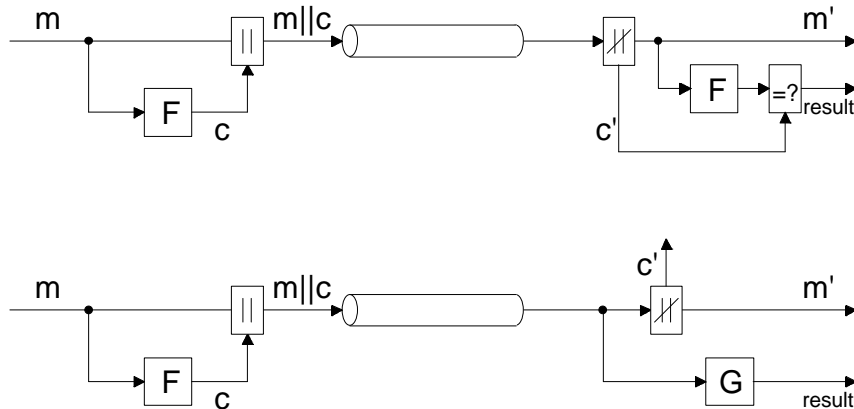
43

## Rivelazione di errore: schema generale



44

## Rivelazione di errore: schema con aggiunta di "checksum"



45

## Rivelazione di errore: Controllo di parità

- Il più semplice codice a rivelazione di errore è il controllo di parità
  - per ogni blocco di  $N$  bit viene aggiunto un bit pari a 1 se il numero di 1 nel blocco è dispari, mentre viene aggiunto uno 0 se pari
  - se sono presenti  $k$  blocchi di  $N$  bit, verranno generati  $k$  bit di parità
  - tali bit possono essere singolarmente aggiunti di seguito a ciascun blocco o tutti insieme in punto preciso della UI (e.g. alla fine)

- Esempio ( $N=8$ )

10010010 1 10100011 0

oppure

10010010 10100011 10

- Il bit di parità permette di riconoscere errori in numero dispari

46

## Controllo di parità a blocchi

- Per correggere errori a burst con i bit di parità, si può suddividere la sequenza di bit in  $k$  parole di  $N$  bit e organizzarla come una matrice  $k \times N$ , con una parola per riga
- I bit di parità vengono calcolati sulle colonne
  - $N$  bit di parità
- Un errore a burst di lunghezza  $k$  influenzerà al più 1 bit per parola
  - così si riesce a correggere l'intera sequenza

47

## Controllo di parità a blocchi (righe e colonne)

a) blocco di carattere originario e bit di parità

b) errore trasmissivo non rivelato dal controllo di parità

0	1	0	1	0	0	1	1
0	0	0	0	1	1	0	0
0	0	1	1	1	0	1	0
1	1	0	0	0	1	0	1
0	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0
1	0	0	0	1	1	0	0
0	1	0	1	0	0	0	0

Bit di parità per righe

a)

0	1	0	1	0	0	1	1
0	1	0	0	1	0	0	0
0	0	1	1	1	0	1	0
1	1	0	0	0	1	0	1
0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	0
1	0	0	0	1	1	0	0
0	1	0	1	0	0	0	0

b)

48

## Codici a somma complemento a 1 (Checksum)

- La sequenza di bit da proteggere viene suddivisa in k parole di N bit e organizzarla come una matrice kxN, con una parola per riga
- le righe vengono sommate tra loro con aritmetica complemento a 1
- la somma, eventualmente complementata, è il codice rivelazione di errore (checksum)

**Esempio**

0100	0101	0000	0000
0000	0000	0011	0000
1100	0001	0000	0000
0100	0000	0000	0000
1000	0000	0000	0110
0000	0000	0000	0000
0000	0001	1001	1000
0001	0101	1110	1011
1010	0000	0100	1110
0001	1101	0011	1110
<hr/>			
1001	1011	0100	0111 = sum
0110	0100	1011	1000 = checksum

49

## Esempio di calcolo del Checksum

	1	1	1		
	10111	0110	1111	100	
<hr/>					
0100	0101	0000	0000	+	45 00 +
<hr/>					
0000	0000	0011	0000		00 30
<hr/>					
1100	0001	0000	0000		C1 00
<hr/>					
0100	0000	0000	0000		40 00
<hr/>					
1000	0000	0000	0110		80 06
<hr/>					
0000	0000	0000	0000		00 00
<hr/>					
0000	0001	1001	1000		01 98
<hr/>					
0001	0101	1110	1011		15 EB
<hr/>					
1010	0000	0100	1110		A0 4E
<hr/>					
0001	1101	0011	1110		1D 3E
<hr/>					
				=	----- =
somma parziale:				1001	1011 0100 0101
<hr/>					
riporto:				10	2
<hr/>					
somma (complemento a 1):				1001	1011 0100 0111
<hr/>					
				=	----- =
				1001	1011 0100 0111 ⊕
<hr/>					
				1111	1111 1111 1111
<hr/>					
checksum:				0110	0100 1011 1000
<hr/>					
				64 B8	= 0110 0100 1011 1000

50

## Codici a somma complemento a 1 (Checksum)

FCS (Frame Check Sequence), o checksum, è il codice a rivelazione di errore usato in Internet (es. IP, UDP, TCP)

### Trasmettitore

- tratta il contenuto dell'UI come una sequenza di interi a 16-bit mettendo a zero i bit relativi al checksum
- sum: somma (complemento a 1) del contenuto del segmento
- checksum: complemento della somma (0→1, 1→0)
- inserisce il valore della checksum nel campo checksum dell'header

### Ricevitore

- calcola la checksum del segmento ricevuto
- se la checksum calcolata è composta da sedici 1 (FF FF) allora la UI è rivelata corretta
- Nota: è possibile che venga rivelata corretta una UI errata

51

## Codici rivelatori di errore polinomiali (CRC)

- Codici polinomiali o anche Cyclic Redundancy Code (CRC)
- Le singole cifre binarie di una stringa da emettere sono trattate come coefficienti (di valore "0" o "1") di un polinomio P(x)
  - Le cifre binarie di una UI di lunghezza uguale a k cifre binarie sono quindi considerate come i coefficienti di un polinomio completo di grado k-1. In particolare, l'i-esimo bit della trama è il coefficiente  $X^{i-1}$  di P(x)
- Le entità emittente e ricevente utilizzano un polinomio comune G(x), detto polinomio generatore, come divisore del polinomio P(x)

$$\frac{P(x)}{G(x)} = N(x) + \frac{R(x)}{G(x)}$$

dove: N(x) è il polinomio quoziente

R(x) è il polinomio resto

- La divisione viene effettuata in algebra modulo 2

52

## CRC

- La entità emittente inserisce i coefficienti del resto  $R(x)$  in un apposito campo del PCI della UI
- La entità ricevente esegue, con il polinomio generatore, la stessa operazione di divisione effettuata in emissione e confronta il resto ottenuto localmente con quello contenuto nella UI
- Se i due resti sono uguali, la UI è considerata corretta; altrimenti uno o più errori si sono verificati nel corso del trasferimento e la UI viene considerata errata (in genere viene scartata)

53

## CRC

- Si può dimostrare che scegliendo opportunamente  $G(x)$ , il CRC può rivelare sempre
  - Errori su un bit singolo
  - Errori su 2 bit se  $G(x)$  ha almeno 3 coefficienti non nulli
  - Errori su un numero dispari di bit se  $G(x)$  è divisibile per  $(x+1)$
  - Errori a "burst" (cioè consecutivi) con lunghezza del burst minore di  $r+1$  bit

54

## CRC

- I polinomi divisori  $G(x)$  più usati in Europa e Nord America sono rispettivamente:

$$G(x) = x^{16} + x^{15} + x^{12} + 1$$

$$G(x) = x^{16} + x^{15} + x^2 + 1$$

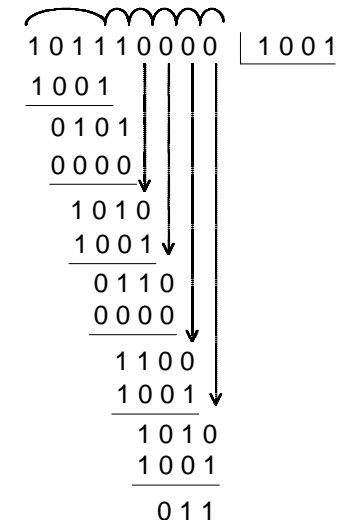
- Tecnica usata per esempio in Ethernet o in altri protocolli di Livello 2 (Data Link)

55

## Esempio CRC

- $M$  = msg originale = 101110
- $G$  = polinomio generatore = 1001
- $r$  = 3 bit di ridondanza (grado del polinomio  $G$ )
- $Q$  = polinomio quoziente
- $R$  = resto = CRC = 011
- $P$  = msg inviato =  $M \parallel R$  = 101110011

$$\frac{x^r D(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$



56

## Correzione di errore

- FEC (Forward Error Correction)
  - aggiunta di ridondanza in trasmissione
  - in ricezione si usa la ridondanza per rimediare ai bit errati (se in numero contenuto)
  - nessun riscontro di corretta ricezione → comunicazione unidirezionale
  - non c'è necessità di buffer per le UI inviate
  - teoria dei codici (codici a blocco, codici convoluzionali)
  - la ridondanza aggiunta è in genere maggiore di quella usata per la sola rivelazione
  - la ridondanza introdotta per correggere l'errore è costante (fissa)
    - al contrario dei meccanismi di recupero di errore che prevedono meccanismi di ack e ritrasmissione (vedi seguente)
- e.g. codice a ripetizione
  - ripetendo N volte lo stesso bit si correggono sino a  $N/2 - 1$  errori

57

## Recupero di errore

- Ha lo scopo di riportare alla normalità il flusso di UI trasferite tra due entità nel caso di errori di duplicazioni, di perdite di UI o di alterazioni nel loro ordine originale
- Controllo di errore che opera tramite ritrasmissione automatica
  - ARQ (Automatic Repeat Request)
- Meccanismi impiegati per la ritrasmissione:
  - 1) codici di rivelazione di errore
  - 2) uso di riscontri positivi (ACK) e/o negativi (NACK)
  - 3) uso di temporizzatori
  - 4) introduzione dei numeri di sequenza (SQN) delle UI
  - 5) uso di richieste di ri-emissione selettive (SEL REJ)
- Alcune tecniche: stop&wait, sliding window go-back-N or selective repeat
- Esempi: X.25/HDLC, TCP, SIP

58

## Procedure di recupero ARQ

### Automatic Repeat Request (ARQ)

- 1) modalità a riscontro positivo con riemissione (stop and wait)
  - finestra  $Tx=1$
- 2) modalità a finestra variabile (sliding window) con riemissione non selettiva (go-back-N)
  - finestra  $Tx=N (>1)$ , finestra  $Rx=1$
- 3) modalità a finestra variabile (sliding window) con riemissione selettiva (selective repeat)
  - finestra  $Tx=N (>1)$ , finestra  $Rx=K (>1)$

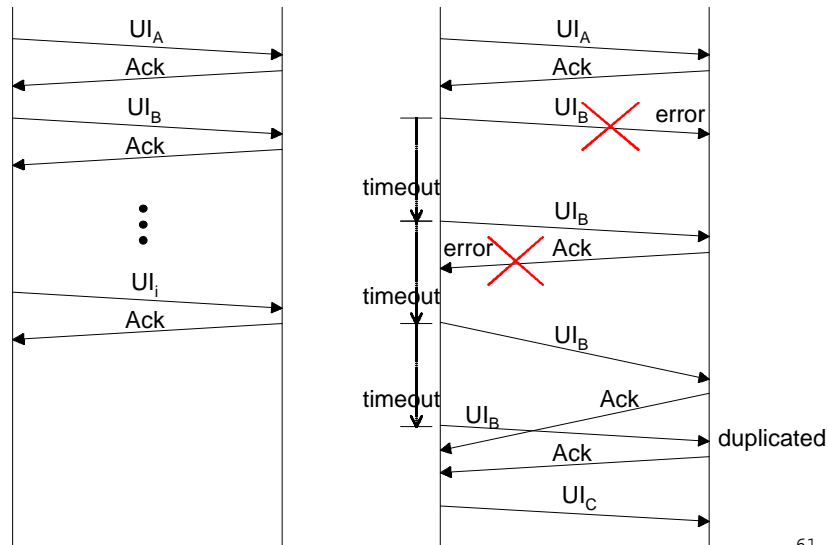
59

## Stop&Wait

- Per ogni verso della comunicazione le UI dati vengono inviate singolarmente (una per volta)
- Prima di inviare la successiva UI si attende un messaggio di riscontro (ACK)
- Il messaggio di riscontro è di una UI contenente dati diretti nel verso opposto o vuota in cui è presente una informazione di controllo di ACK (acknowledgment)
- Per poter far fronte ad eventuali casi di errore (mancanza di consegna di UI dati o ACK, o ricezione errata) è necessario attivare in trasmissione per ogni UI inviata un timeout di ritrasmissione
- Alla scadenza del timeout, se non è stato ricevuto alcun ACK, la UI viene ritrasmessa

60

## Stop&Wait con ACK e Timeout



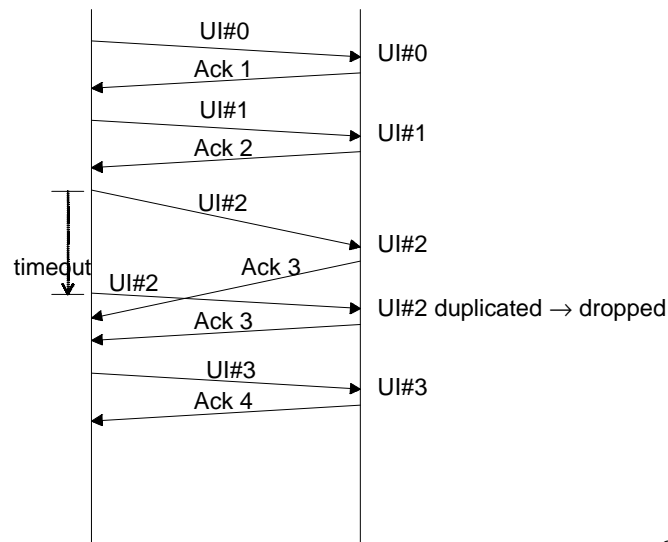
61

## Stop&Wait: SQN

- Per far fronte ad eventuali duplicazioni di UI dati e/o di ACK è importante utilizzare numeri di sequenza (SQN)
- Per i numeri di sequenza è normalmente riservato nel PCI (header) un numero di bit fissato (e limitato)
  - le UI sono quindi numerate modulo  $N$
  - se  $k$  bit per SQN allora si ha  $N=2^k$
- Nel caso di Stop&Wait, il valore minimo di  $N$  è 2
  - un solo bit per il conteggio del numero di sequenza

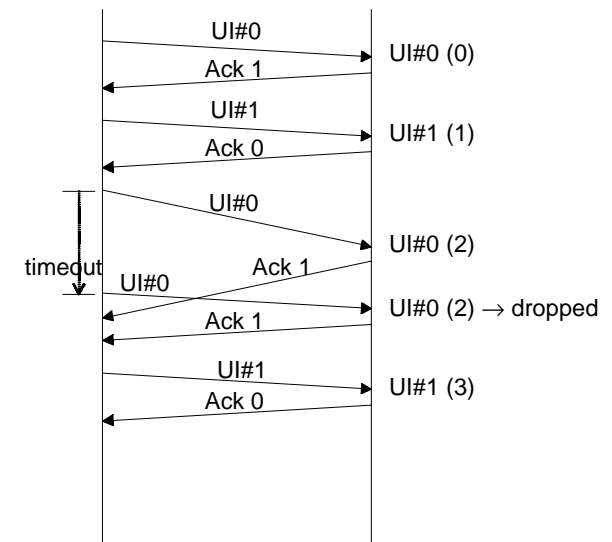
62

## Stop&Wait con ACK, Timeout, e SQN



63

## Stop&Wait con ACK, Timeout, e SQN



64



## Stop&Wait

- Il trasmettitore
  - invia una UI
  - attiva un orologio con un timeout fissato
  - si pone in attesa della conferma di ricezione (ACK)
  - se scade il timeout prima dell'arrivo del ACK, ripete la trasmissione
  - se riceve un ACK prima della scadenza del timeout, controlla il numero di sequenza e se corretto abilita la trasmissione della UI successiva
- Il ricevitore
  - riceve una UI
  - controlla la correttezza della UI
  - controlla il numero di sequenza
  - se la UI è corretta invia la conferma di ricezione (ACK)

65

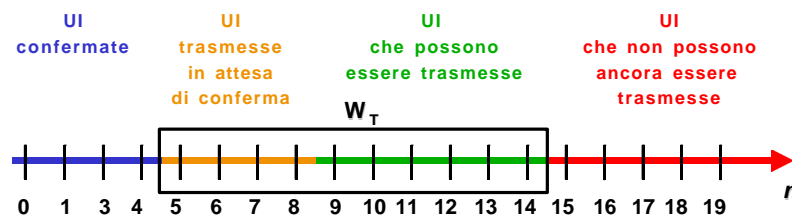
## Stop&Wait: Timeout

- Il timeout deve essere sufficientemente grande da consentire la trasmissione della UI e della relativa conferma (ACK)
- Un timeout troppo grande rallenta eventuali ritrasmissioni
- Il valore del timeout deve quindi approssimare per eccesso il ritardo A/R ovvero il RTT (Round Trip Time)
- La stima di tale valore è in genere un'operazione delicata

66

## Sliding window

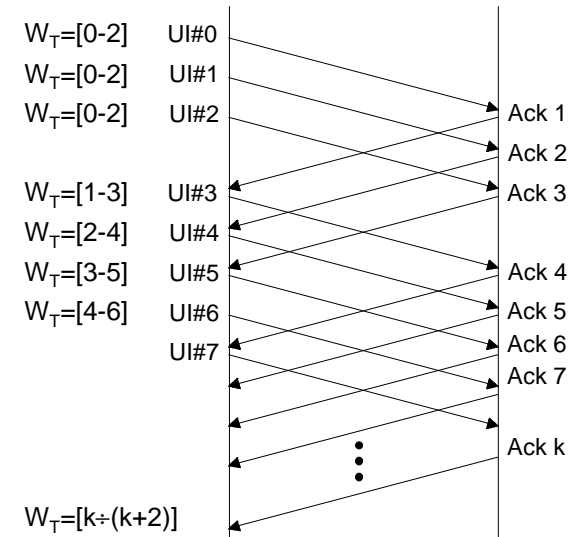
- Il throughput può essere aumentato consentendo al trasmettitore l'invio di  $N_T$  UI consecutive senza aspettare i relativi riscontri
- La gamma dei numeri di sequenza delle UI consecutive che possono essere emesse senza ricevere riscontro può essere considerata una "finestra" di dimensione  $N_T$  che si sposta in avanti di 1 per ogni ACK ricevuto



$W_T$ : finestra di trasmissione (dimensione  $N_T$ )

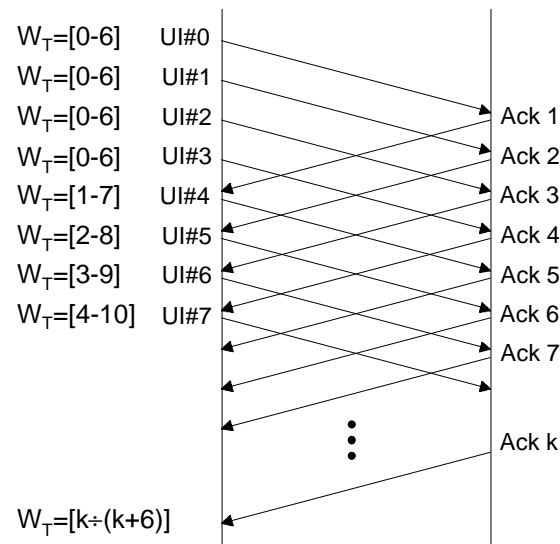
67

## Sliding window ( $N_T=3$ )



68

## Sliding window ( $N_T=7$ )



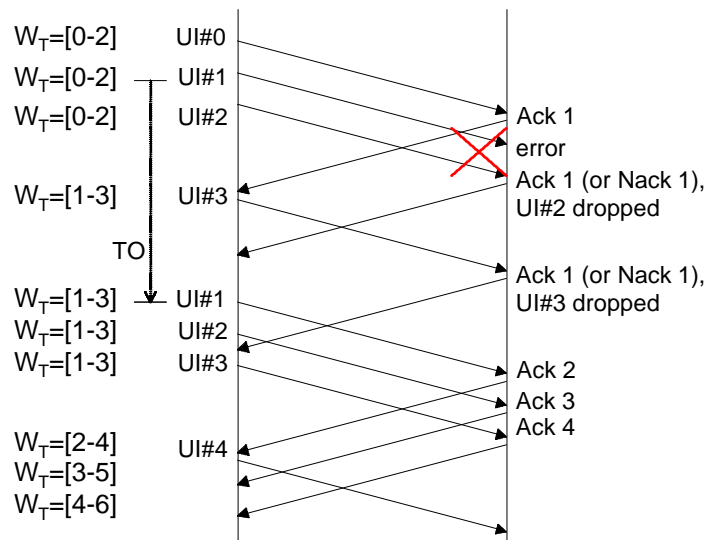
69

## Sliding window

- In genere si considerano ACK cumulativi
  - con **ACK- $i$**  si notifica la corretta ricezione di **TUTTE** le UI con **SNQ inferiore a  $i$**  (specificato nell'ACK)
  - se il trasmettitore riceve un **ACK- $k$**  con  $k$  maggiore della posizione più bassa della  $W_T$ , può avanzare la  $W_T$  fino alla posizione  $k$ , ignorando il fatto che alcuni ACK non siano ancora stati ricevuti
- Due tipi di protocolli sliding window
  - **go-Back-N**
  - **selective repeat**
- Si differenziano sulla capacità del ricevitore di mantenere le UI ricevute fuori sequenza (successive rispetto a UI non ancora ricevute)
  - **finestra di ricezione  $W_R$** , indica la gamma di UI che possono essere ricevute anche non in sequenza

70

## Sliding window Go-Back-N ( $N_T=3$ )



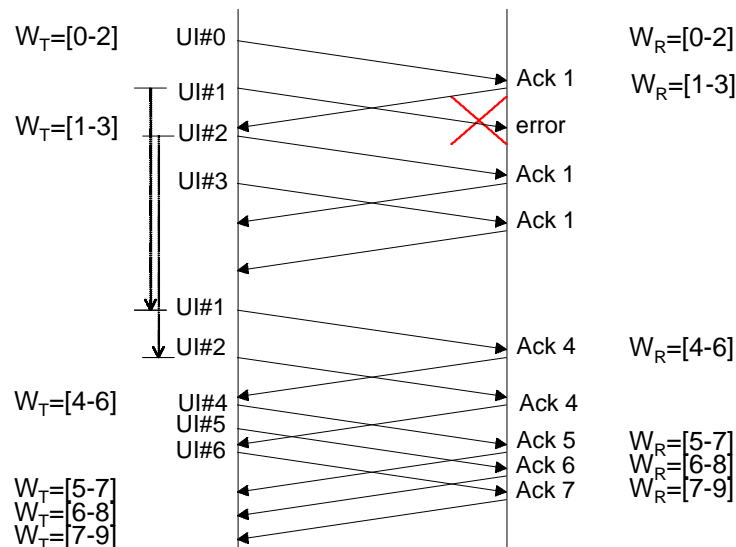
71

## Go-back-N

- Il trasmettitore
  - invia fino ad  $N$  UI facendo di ognuna una copia
  - attiva un orologio per ogni UI
  - si pone in attesa delle conferme di ricezione (ACK)
  - se scade un timeout prima dell'arrivo delle conferme, ripete la trasmissione
- Il ricevitore
  - quando riceve una UI controlla la sua correttezza e il numero di sequenza
  - se la UI contiene il primo numero di sequenza non ancora ricevuto, viene consegnata ai livelli superiori, altrimenti viene scartata
  - invia la conferma di ricezione (ACK)
- Nota: Equivale a Selective repeting con finestra di ricezione  $W_R=1$

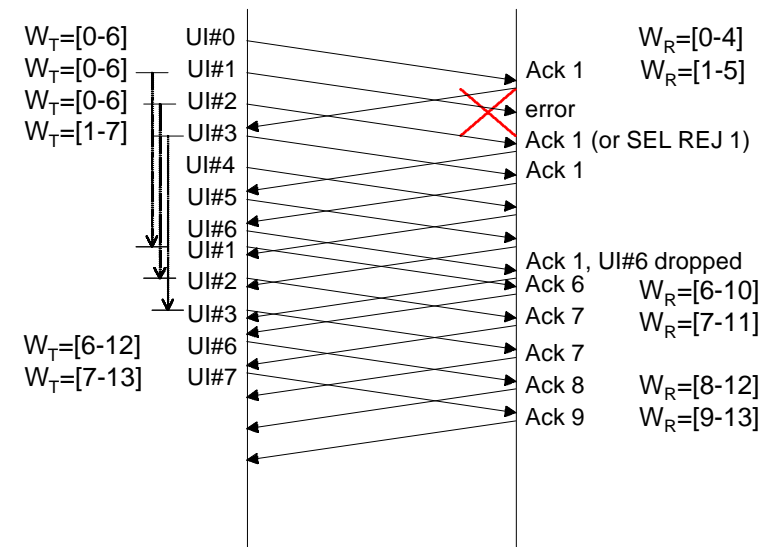
72

## Sliding window Selective repeting ( $N_T=3$ , $N_R=3$ )



73

## Sliding window Selective repeting ( $N_T=7$ , $N_R=5$ )



74

## Selective repeting

- Il trasmettitore
  - invia fino ad N UI
  - attiva un orologio per ogni UI
  - si pone in attesa delle conferme di ricezione (ACK)
  - se scade un timeout prima dell'arrivo della conferma, ripete la trasmissione della UI
- il ricevitore
  - riceve una UI e controlla la correttezza della UI e il numero di sequenza
  - se la UI è corretta e all'interno della finestra di ricezione invia un ACK
  - nell'ACK si richiede sempre la prima UI non ricevuta in sequenza (ovvero si conferma l'ultima UI ricevuta in sequenza)

75

## Piggybacking

- Se la comunicazione è di tipo bidirezionale, le stesse UI dati possono essere usate per inviare messaggi di ACK (controllo) relativi al flusso opposto
  - molti protocolli di tipo ARQ prevedono che una UI possa essere:
    - solo dati
    - solo controllo
    - contemporaneamente sia dati che controllo
- Per "piggibacking" si intende quella tecnica che ritarda leggermente l'invio di ACK relativi ad un flusso nell'attesa di eventuali UI dati nel verso opposto

76

## Controllo di flusso

- Ha il compito di assicurare che ritmo di arrivo delle UI non superi la capacità di memorizzazione e elaborazione della entità ricevente, in modo che non si verifichino perdite di informazione
  - **regolazione dell'emissione del flusso di dati di una entità emittente da parte della entità ricevente**
  - **dipende dalle capacità di memorizzazione del ricevente (buffer) e dalla sua velocità di elaborazione e smaltimento**
  - **spesso viene realizzata in interazione con la funzione di recupero di errore e/o di controllo di congestione**
- Tecniche:
  - **messaggi di riscontro tipo RR (Receiver Ready), RNR (Receiver Not Ready) (e.g. X.25), CTS (Clear to Send) (e.g. RS-232)**
  - **oppure tecniche indicazione della flow window di ricezione (e.g. TCP)**

78

## Controllo di flusso: esempi

- Esempio: seriale RS-232
  - **RTS/CTS (Controllo di flusso HW)**
    - E' presente una coppia di fili corrispondenti ai segnali RTS (*Request To Send - pin 4*) e CTS (*Clear To Send - pin 5*). Quando un dispositivo ricevente rileva l'attivazione del segnale RTS da parte del dispositivo trasmittente ed è pronto per ricevere, allora risponde attivando il CTS. Per interrompere l'invio dei dati da parte del trasmittente, il ricevitore può disattivare il segnale CTS, e riattivarlo quando sarà nuovamente in grado di ricevere i dati.
  - **XON/XOFF (Controllo di flusso SW)**
    - L'utilizzo dei caratteri XON e XOFF (codici 17 e 19 della tabella ASCII, talvolta identificati come DC1 e DC3 - *device control* numero 1 e 3 - e corrispondenti ai codici di controllo CTRL-Q e CTRL-S) permette di realizzare un controllo di flusso senza bisogno di segnali hardware dedicati, in quanto XON e XOFF viaggiano sugli stessi canali dei dati. Il ricevitore trasmette un XOFF quando non è più in grado di ricevere i dati e un XON quando è nuovamente in grado di riceverli

79

## Controllo di flusso: esempi

- Esempio: HDLC/X.25 livello 2
  - **controllo di flusso tramite trama (HDLC-PDU) di controllo RNR (Receiver Not Ready). È una trama utilizzata per indicare che la stazione è temporaneamente impossibilitata a ricevere nuovi I-frame (trame informative)**
  - **quando la stazione ricevente può ricominciare a ricevere UI invia una trama di controllo di tipo RR (Receive Ready)**
- Esempio: TCP
  - **controllo di flusso a finestra scorrevole di ampiezza variabile e operante a livello di ottetti (byte) numerati sequenzialmente**
  - **un segmento (TCP-PDU) di riscontro con ACK Number=n e Window=w significa che il trasmittente è autorizzato a trasmettere fino a ulteriori w ottetti a partire da n, ovvero fino all'ottetto numerato con n+w-1**

80

## Controllo di congestione

## Controllo di congestione

- Il controllo della congestione ha lo scopo di recuperare e/o evitare eventuali situazioni di sovraccarico nella rete
  - **regolazione dell'emissione dei dati in modo da ridurre il grado di congestione della rete**
- Tecniche:
  - **meccanismi con feedback dei nodi della rete**
  - **meccanismi end-to-end**
  - **routing adattativo**
  - **controllo di ammissione di chiamata (CAC), nel caso di comunicazioni CO**
- Esempi: ATM, TCP

82

## Altre funzioni

## Altre funzioni

- Interfacciamento Fisico
  - **connettori, cavi e mezzi fisici**
- Trasmissione/Ricezione
  - **modulazione/demodulazione, codifica di canale, codifica di linea, etc.**
- Incapsulamento/Adattamento/Compressione
  - **identificazione, riempimento, frammentazione o aggregazione, compressione, etc.**
  - **e.g. AAL, LLC, IP, L2TP, PPTP, IPSec, etc.**
- Interworking con traduzione di protocolli
  - **tanti problemi, e.g. indirizzamento, restrizioni sul formato, gestione errori, gestione QoS, ..**
  - **e.g. IEEE 802.1, NAT, NAT-PT (IPv4/IPv6), Media GWs, etc..**
- Instaurazione di chiamata (Call Setup)
  - **e.g. PPP, TCP, SIP**

84

## Altre funzioni (cont.)

- Gestione della mobilità
  - **Roaming, indirizzamento, paging, etc.**
  - **e.g. GSM/GPRS/UMTS, Mobile IP, SIP, etc.**
- Controllo della QoS
  - **gestione delle risorse per flusso o per classe di servizio**
  - **e.g. Intserv, Diffserv**
- Sincronizzazione/Equalizzazione
  - **Timestamps, playout buffer**
  - **e.g. RTP/RTCP, RTSP**
- Multicasting
  - **indirizzamento, routing**
  - **e.g. IP Multicast, SAP**
- Supporto della sicurezza
  - **Autenticazione, integrità, confidenzialità**
  - **e.g. IPSec, TLS/SSL**