UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Funzioni e protocolli

Luca Veltri

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

Corso di Reti di Telecomunicazione, a.a. 2016/2017

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

Protocolli e funzioni

- Con il passare del tempo sono stati definiti, e verranno definiti nel futuro, sempre nuovi protocolli con lo scopo di adattarsi a nuove esigenze
  - **nuove applicazioni o nuovi contesti di rete**
  - **nuovi strati che raggruppano funzioni comuni (ad esempio come sotto-strato del livello applicativo)**
  - **nuovi formati (ascii, binari, ASN.1, XML, etc.)**
- I principi su cui si basano le funzioni base che possono essere realizzate rimangono quasi sempre le stesse
- In questa sezione verranno definite e analizzate le principali funzioni che vengono realizzate dai protocolli di comunicazione

Principali funzioni

- Trasmissione/Ricezione
- Delimitazione delle UI
- Sequenzializzazione
- Multiplazione
- 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..

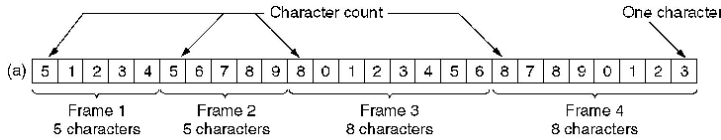
Delimitazione

Delimitazione

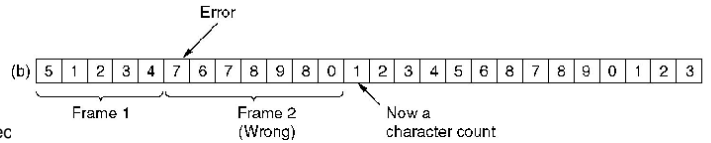
- Delimitazione delle UI all'interno di un flusso o blocco di bit/byte
  - **per riconoscere/distinguere l'inizio e fine di ciascuna UI**
- Funzione eseguita in genere quando uno strato si deve interfacciare con uno strato sottostante di tipo Stream oriented
  - **in molti casi implementata sopra i protocolli 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**

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



Metodo del conteggio di caratteri (cont.)

- Vantaggi
  - **semplice da implementare**
  - **basso contenuto di overhead**
- Svantagi
  - **difficile gestione della sincronizzazione in caso di errori**

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

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

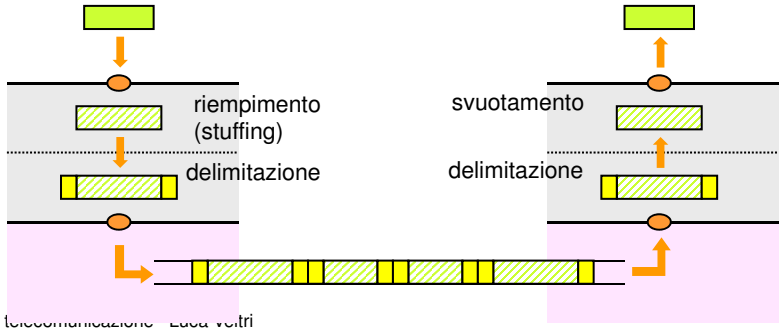
SF					EF	SF			EF			SF		

- 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**

Bit-stuffing & Char-stuffing

- Obbiettivo 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
  - **cioè processano sequenzialmente i simboli elementari (bit o byte) componenti la UI**



Bit stuffing: esempio (HDLC/LAPB/X.25)

- 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/LAPD, X.25

Sequenza originaria di cifre binarie

1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 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 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 1 1 **0** . . . .

Sequenza dopo l'operazione di svuotamento

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

Char stuffing: esempio (SLIP)

- 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

Inserimento di simboli di inizio e fine UI (cont.)

- Vantaggi
  - **più robusto in presenza di errori**
  - **ri-sincronizzazione automatica**
- Svantaggi
  - **se utilizza simboli speciali**
    - necessità di ridondanza di simboli (codifica)
  - **se utilizza bit o byte stuffing**
    - più complicato rispetto conteggio di caratteri
    - maggior overhead presente nel flusso trasmesso

Esempi di delimitazione

- 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)

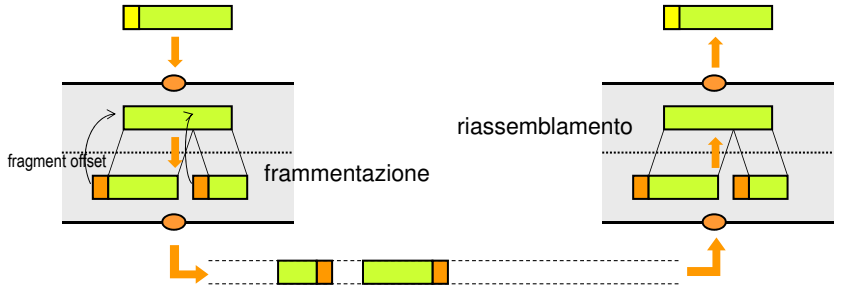
Esempi di delimitazione (cont.)

- HDLC, LAPB/LAPD, X.25
  - **delimitazione tramite Flag (01111110)**
    - utilizzo di bit stuffing
- HTTP
  - **delimitazione tramite flag di fine message header (linea vuota = CRLF), più conteggio dei carattteri 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

Frammentazione/aggregazione

Frammentazione

- Funzione che permette di incapsulare e spedire una unica SDU tramite di due o più PDU
  - **In ricezione viene eseguita la funzione opposta (riassemblamento)**



- Se sono previsti nodi intermedi, le funzioni di frammentazione e riassemblamento possono essere svolte
  - **solo nei nodi terminali**
  - **sia nei terminali che nei nodi intermedi**

Frammentazione (cont.)

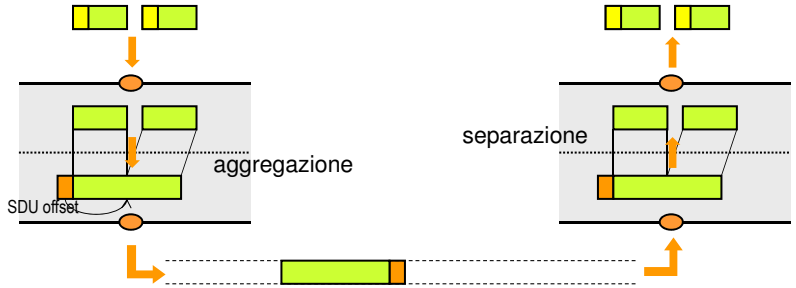
- Tale funzione viene normalmente svolta da un (N)-protocollo (che la supporta) quando:
  - **il (N-1)-protocollo sottostante ha una dimensione massima di (N-1)-SDU, minore della dimensione della (N)-SDU**
  - **si vuole limitare la dimensione complessiva delle (N)-PDU**
- Un limite alla dimensione delle SDU (o PDU) può essere utile per:
  - **limitare il tempo di trasmissione delle UI**
  - **limitare il tempo di attraversamento di un nodo di commutazione**
  - **limitare la probabilità di errore delle UI (frame/packet error rate)**
    - a parità di BER (bit error rate), il packet error rate cresce al crescere della dimensione delle UI
- In un protocollo, la dimensione massima di SDU (se presente) viene spesso indicata come MTU
  - **Maximum Transfer Unit (o anche Maximum Transmission Unit)**

Frammentazione (cont.)

- Al fine di riassemblare la SDU in ricezione, nel caso di possibilità di perdita o di ricezione con diverso ordine è necessario mantenere informazione della posizione delle frammenti rispetto alla SDU originaria
  - **aggiunta del numero di sequenza del frammento, oppure**
  - **aggiunta del numero di sequenza dei bit/byte contenuti**
    - in genere il numero di sequenza si riferisce al primo bit/byte trasportato
    - questa tecnica rende più semplice effettuare frammentazione multipla
- È necessario un meccanismo per riconoscere l'ultimo frammento
  - **un campo lunghezza totale (in termini di bit/byte o di frammenti)**
  - **un flag che caratterizza l'ultimo segmento**

Aggregazione

- Funzione che permette di incapsulare e spedire due o più SDU attraverso una sola PDU
  - **In ricezione viene eseguita la funzione opposta (separazione)**



- Se vengono aggregate SDU di utenti (del servizio) diversi, allora tale funzione viene svolta insieme a quella di multiplazione

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)
- Le UI fuori sequenza possono essere memorizzazione in ricezione e opportunamente riordinate, oppure scartate
- In caso di perdita, alcuni protocolli legano la funzione di controllo di sequenza con quella di recupero di errore

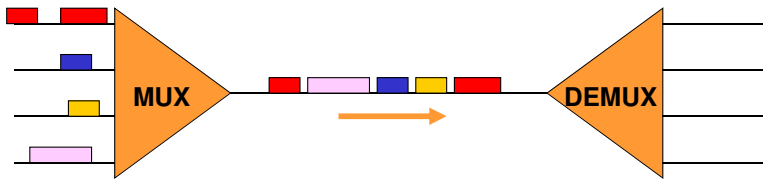
Multiplazione

Multiplazione

- Per una risorsa trasmissiva condivisa, 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 pacchetti o 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, e/o**
  - **uno strato deve gestire più pacchetti o flussi di strato superiore (relativi a protocolli differenti o ad un unico protocollo) multiplandoli nello stesso flusso di UI passato allo strato inferiore**
- La maggior parte dei protocolli offrono questa funzionalità
  - **e.g. PCM/PDH, SDH, Ethernet, IP, TCP, UDP**

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 sovrapposizioni/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)**



Multiplazione

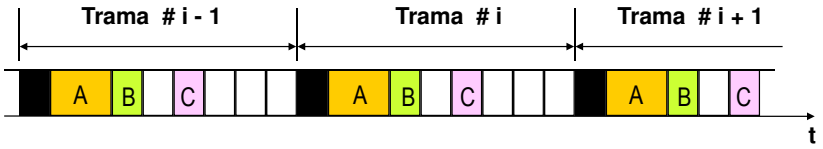
- Nel caso di protocolli di strato PH la risorsa di comunicazione condivisa (canale multiplato) è il mezzo trasmissivo, e si possono distinguere i seguenti schemi di multiplazione:
  - **Time Division Multiplexing (TDM)**
  - **Space Division Multiplexing (SDM)**
  - **Frequency Division Multiplexing (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**
  - **In alcuni casi è possibile operare anche in modalità SDM, sfruttando più connessioni o interfacce**

Multiplazione

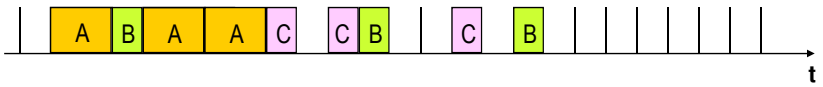
- Si possono evidenziare 2 differenti approcci TDM
  - **Multiplazione statica (sincrona)**
    - i flussi vengono multiplati in modo sincrono (con una cadenza o successione temporale fissata)
      - in genere tale cadenza temporale è di tipo periodica
    - esempi, PCM/PDH, SDH
  - **Multiplazione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

Multiplazione sincrona e asincrona

Esempio di multiplazione sincrona

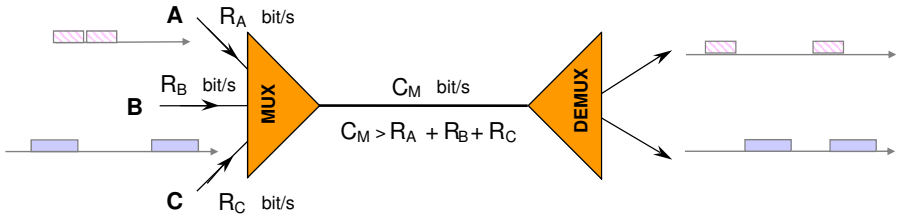


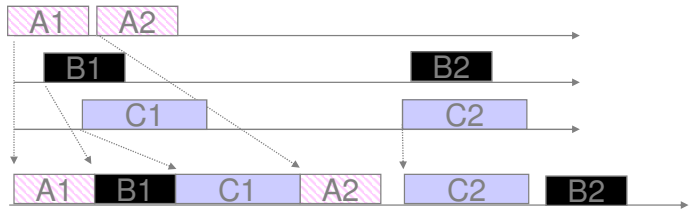
Esempio di multiplazione asincrona



(è necessario indirizzamento esplicito delle UI)

Canale multiplato





Controllo di Accesso Multiplo

Accesso multiplo

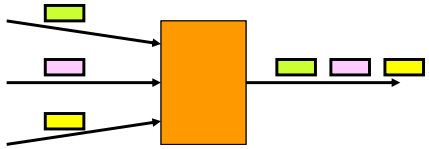
- Quando:
  - **accesso diretto ad un mezzo trasmissivo condiviso (a divisione di tempo)**
  - **la trasmissione è di tipo broadcast (un sistema trasmette e tutti gli altri possono ricevere)**
    - ricezione contemporanea da parte di più sistemi terminali
- Quando un sistema trasmette diventa proprietario 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

Accesso multiplo (cont.)

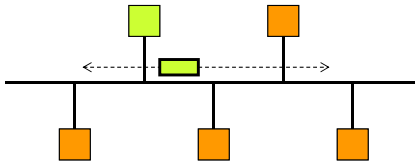
- Simile alla funzione di multiplazione, ma in questo caso il flusso multiplato è gestito da entità distribuite (e non da un'unica entità concentrata)
  - **multiplazione → soluzione concentrata**
  - **accesso multiplo → soluzione distribuita**
- Tipico dei protocolli di accesso ad un mezzo fisico condiviso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

Multiplazione e accesso multiplo

- Multiplazione



- Accesso multiplo



Tecniche di accesso multiplo

- Modalità
  - **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
  - **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*
- Esempi di protocolli MAC:
  - **Aloha, CSMA/CD (Ethernet), CSMA/CA (WiFi), FDDI**

Accesso controllato

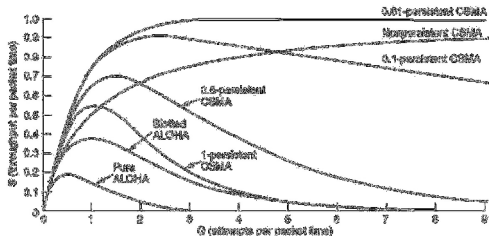
- Si distinguono i casi di:

  - 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)**

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**



Indirizzamento e risoluzione degli indirizzi

Identificazione/indirizzamento

- Identificazione delle UI (PDU/SDU) 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) nel caso di comunicazione con connessione
- I protocolli CO in aggiunta hanno bisogno di identificare la sorgente e la destinazione in fase di instaurazione

Identificazione/indirizzamento (cont.)

- Modalità di indirizzamento delle UI
  - **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
  - **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, UDP, TCP, HTTP
    - esempi di VCI:
      - X.25, ATM, MPLS

Identificazione/indirizzamento (cont.)

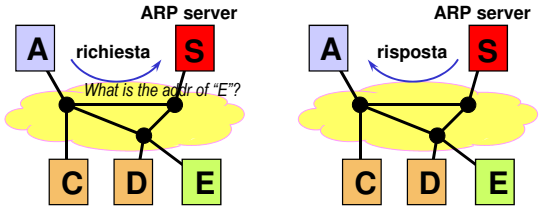
- Protocolli diversi possono usare piani di indirizzamento differenti
- Questi possono essere di tipo
  - **non gerarchico (opaco, piatto)**
  - **gerarchico (indirizzo diviso in parti o livelli)**
    - gerarchico su base geografica (e.g. PSTN)
    - gerarchico su base non geografica
- Possono essere di tipo
  - **numerico: utilizzo di stringhe numeriche**
    - spesso sono accompagnati da una rappresentazione canonica, in notazione esadecimale o decimale
    - e.g. numeri di telefono, indirizzi IP
  - **mnemonico: utilizzo di stringhe di caratteri, nomi**
    - e.g. URI/URL (esempio: http://www.unipr.it/veltri)

Risoluzione degli indirizzi

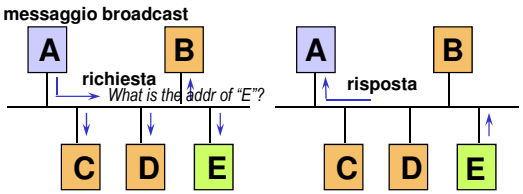
- Ogni qual volta si deve rilanciare un pacchetto di un protocollo N da un nodo ad un altro bisogna conoscere l'indirizzo del protocollo (N-1) sottostante del nodo successivo a livello N
- E' quindi necessaria una funzione di mappaggio da indirizzi di livello N ad indirizzi di livello (N-1), se utilizzati
  - **questa funzione può essere svolta sulla base di una tabella di mapping**
  - **non è necessario nei collegamenti punto punto in cui l'indirizzamento a livello (N-1) diventa inutile o non è presente**
- Mapping statico
  - **la tabella di associazione viene predisposta staticamente**
- Mapping dinamico
  - **la tabella viene costruita dinamicamente attraverso un opportuno protocollo ARP (Address Resolution Protocol)**
  - **due approcci possibili per il mapping dinamico:**
    - centralizzato: utilizzo di server di risoluzione (ARP server)
      - tipicamente su reti non broadcast
    - distribuito: utilizzo di comunicazioni broadcast (e.g. nelle LAN)
      - solo per reti con supporto di comunicazioni broadcast

Risoluzione dinamica degli indirizzi

- 1) Centralizzato: utilizzo di server di risoluzione (ARP or Name server)



- 2) Distribuito: utilizzo di comunicazioni broadcast



Risoluzione dinamica degli indirizzi (cont.)

- 3) Distribuito: utilizzo di una struttura di server
  - **simile al metodo con di server di risoluzione, in cui però il DB del server è sostituito da un DB distribuito gerarchicamente su più server che vengono acceduti in modo iterativo o ricorsivo**
    - e.g. DNS
- 4) Distribuito: utilizzo di un sistema distribuito P2P
  - **i nodi collaborano al fine di mantenere un DB distribuito**
  - **I nodi possono entrare e uscire dinamicamente dal sistema**
  - **spesso realizzato tramite Distributed Hash Table (DHT)**
    - e.g. Kademlia (BitTorrent, Kad/eMule), Chord

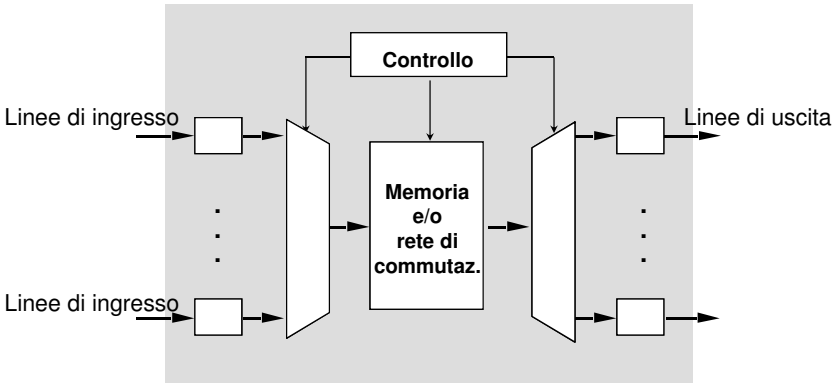
Commutazione

Commutazione

- E' la funzione che permette alle UI di attraversare i nodi di rete (nodi di commutazione) e raggiungere il nodo (o i nodi) di destinazione, 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

Nodi di commutazione

- Schema generale di un commutatore a divisione di tempo



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 istradamento (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 istradamento**
    - 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

Commutazione CO e CL

- La funzione di instradamento può essere effettuata:
  - **durante la fase di instaurazione della connessione (per servizi di trasferimento orientati alla connessione)**
  - **indipendentemente per ciascuna UI (per servizi ti trasferimento senza connessione)**
- Ciò corrisponde a due differenti modalità di commutazione
  - **commutazione CO (utilizzata solo da protocolli CO)**
  - **commutazione CL**

Commutazione CL (a datagramma)

- Nel caso di commutazione CL (o a datagramma), le UI vengono rilanciate dai nodi di commutazione in modo indipendente l'una dall'altra
  - **ogni UI viene elaborata singolarmente**
  - **sulla UI viene eseguita la funzione di istradamento**
    - individuazione del ramo attraverso cui inoltrare la UI, oppure
    - o equivalentemente, scelta del nodo successivo verso cui rilanciare la UI
  - **tale funzione viene di solito svolta tramite consultazione di opportuna tabella di istradamento (routing table)**
  - **le tabelle di routing riportano per varie possibili destinazioni la direzione verso cui rilanciare le UI con tale destinazione**
    - l'identificativo del ramo di uscita, oppure
    - l'identificativo del nodo successivo

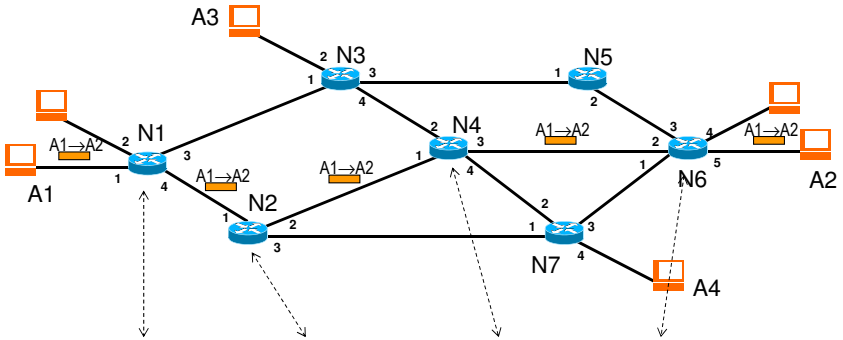
RoutingTable

Dest	Direction
D1	N3
D2	N1
D3	N2
D4	N2

Commutazione CL (a datagramma) (cont.)

- Nelle tabelle di routing, come possibili destinazioni possono essere presenti gli identificativi dei singoli nodi o delle (sotto)reti che li contengono
  - **nel secondo caso deve essere possibile riconoscere dall'indirizzo di destinazione delle UI la loro eventuale appartenenza a specifiche (sotto)reti**
    - questo dipende dal sistema di indirizzamento utilizzato
    - in tal caso si parla di istradamento gerarchico
    - questo ha come obiettivo quello di semplificare la funzione di istradamento soprattutto in presenza di reti di grandi dimensioni
- Nelle tabelle di routing la direzione può essere specificata attraverso vari campi
  - nodo successivo (next-hop), interfaccia, metrica, etc.
- Se non è possibile individuare alcuna direzione, la UI viene scartata

Esempio di commutazione CL (a datagramma)



Routing Table N1

Dest	Next-hop
A1	A1
A2	N2
A3	N3
A4	N2

Routing Table N2

Dest	Next-hop
A1	N1
A2	N4
A3	N1
A4	N7

Routing Table N4

Dest	Next-hop
A1	N2
A2	N6
A3	N3
A4	N7

Routing Table N6

Dest	Next-hop
A1	N4
A2	A2
A3	N5
A4	N7

Commutazione CL (a datagramma) (cont.)

- Il percorso attraverso la rete è il risultato della funzione di instradamento eseguito in successione dai vari nodi di commutazione intermedi incontrati
- La funzione di instradamento può avere risultati diversi in momenti diversi
  - **UI dirette alla stessa destinazione possono in generale seguire percorsi differenti**
  - **per la stessa ragione, UI diretti a stessa destinazione possono arrivare in ordine differente rispetto a quello di partenza**

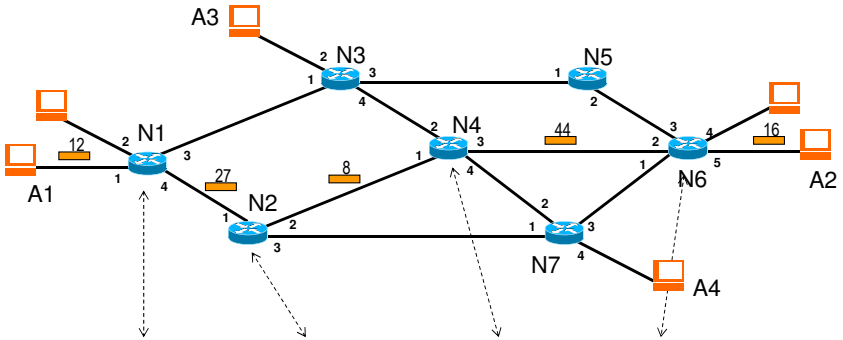
Commutazione CO

- Nel caso di commutazione CO, la funzione di istradamento viene eseguita durante la fase di instaurazione della connessione
  - **in questa fase viene scelto il percorso che seguiranno le UI all'interno della rete dal nodo sorgente al nodo di destinazione**
  - **spesso questo percorso viene instaurato in modo progressivo nodo per nodo tramite tabelle di istradamento**
    - ogni nodo decide in base a delle tabelle di istradamento (routing) verso quale nodo rilanciare il flusso di UI
  - **le tabelle di istradamento sono del tutto simili a quelle utilizzate nella commutazione CL**
    - la differenza sta nel fatto che in questo caso queste sono consultate solo durante la fase di instaurazione
  - **il risultato della funzione di istradamento viene memorizzato in una ulteriore tabella**
    - tabella di commutazione (switching table)
    - questa viene consultata durante la fase di trasferimento delle UI, per la funzione di attraversamento

Commutazione CO (cont.)

- Nel caso di multiplazione statistica, durante la fase di instaurazione vengono anche scelti gli identificatori di connessione o circuito virtuale (VCI, label, tag, etichette) utilizzati successivamente per il trasferimento e commutazione delle UI
  - **tali VCI vengono scelti e associati ramo per ramo**
- Durante la fase di trasferimento dati
  - **le UI vengono identificate su ogni ramo mediante i VCI scelti durante la fase di instaurazione**
  - **su ogni ramo le UI avranno in generale un VCI diverso**
  - **quindi i nodi di commutazione, durante la funzione di attraversamento, modificano il VCI delle UI ricevute in accordo al ramo di uscita su cui rilanciano le UI stesse**
- Per tale motivo nelle Switching Table in genere sono presenti:
  - **ramo e VCI di ingresso**
  - **ramo e VCI di uscita**

Esempio di Commutazione CO



Forwarding Table N1

In		Out	
link	label	link	label
1	12	4	27

Forwarding Table N2

In		Out	
link	label	link	label
1	27	2	8

Forwarding Table N4

In		Out	
link	label	link	label
1	8	3	44

Forwarding Table N6

In		Out	
link	label	link	label
2	44	5	16

Funzione di istradamento

- La sequenza dei nodi attraversati dipende dalla successione della funzione di istradamento eseguita nei vari nodi
- Principali proprietà
  - **semplicità**
    - facile da implementare e leggero da eseguire
  - **efficienza**
    - capacità nel guidare le UI tramite un percorso (quasi)ottimo, minimizzando il numero di rilanci inutili
  - **robustezza**
    - capacità nel tenere conto di guasti, malfunzionamenti o disconfigurazioni
  - **stabilità**
    - capacità di funzionare correttamente (garantendo un percoso ottimo) anche in presenza di variazioni della topologia e dei nodi
      - Il tempo di eventuali fenomeni transitori è minimizzato

Funzione di istradamento (cont.)

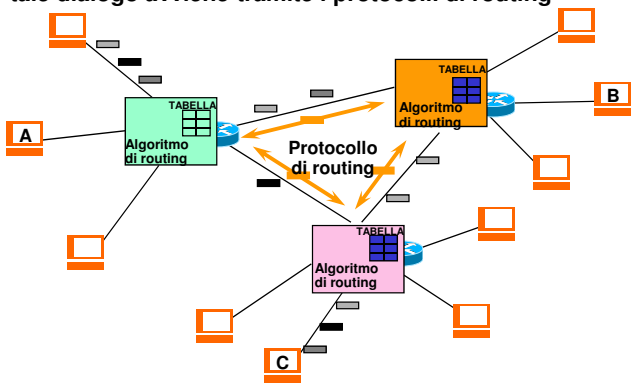
- Tecniche di istradamento
  - **con tabelle di istradamento**
    - istradamento statico
      - configurazione manuale o gerarchico
    - istradamento dinamico
      - protocolli di istradamento
  - **senza tabelle di istradamento**
    - istradamento casuale (random)
    - istradamento ad inondazione (flooding)
    - istradamento guidato dalla sorgente (source routing)
- Altra distinzione, in base al nodo che effettua la scelta
  - **istradamento centralizzato**
  - **istradamento distribuito**

Instradamento statico e dinamico

- Le tabelle di istradamento possono essere configurate
  - **staticamente → routing statico**
    - le tabelle vengono create staticamente e aggiornate manualmente
      - le tabelle vengono modificate dal gestore
      - il gestore ha un totale controllo dei flussi di traffico
      - deve intervenire manualmente per riconfigurare la rete
    - utilizzato in genere nelle reti di piccole dimensione, nelle reti non magliate, o nelle zone di accesso della rete (inclusi i terminali)
  - **dinamicamente → routing dinamico**
    - le tabelle sono calcolate automaticamente dai nodi e aggiornate periodicamente al variare dello stato della rete
    - sulla base di informazioni scambiate periodicamente dai nodi stessi e di appositi algoritmi di istradamento
      - l'algoritmo di istradamento è utilizzato per calcolare le rotte presenti nelle tabelle di routing
    - I nodi utilizzano un protocollo di routing che definisce sia l'algoritmo di routing utilizzato che i messaggi scambiati

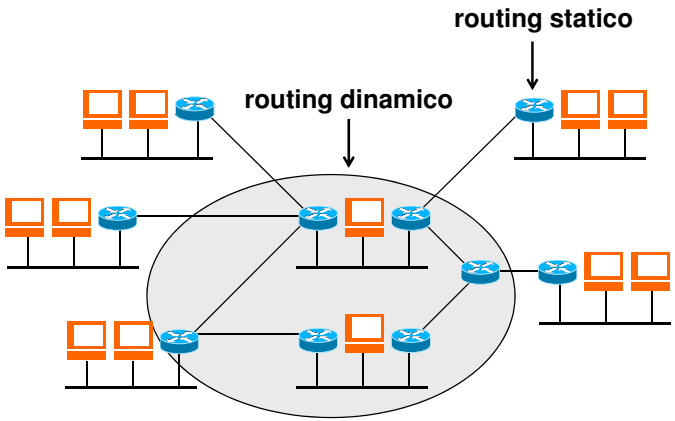
Instradamento dinamico

- Ogni nodo calcola le sue tabelle dialogando con gli altri nodi
  - **per determinare le rotte verso le reti non direttamente connesse**
  - **tale dialogo avviene tramite i protocolli di routing**



- Esempi di protoocolli di routing: RIP, IS-IS, OSPF, Spanning Tree

Instradamento statico vs. dinamico



Algoritmi di istradamento dinamico

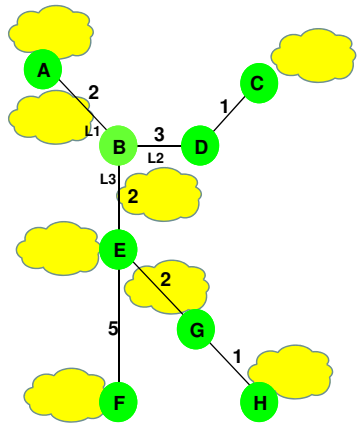
- Gli algoritmi di istradamento dinamico vengono spesso classificati in due tipologie:
  - **Algoritmi di tipo Distance Vector**
    - più semplici
    - impegnano meno risorse nei nodi
    - meno efficienti
    - adatti a reti piccole
  - **Algoritmi di tipo Link State**
    - molto più complessi
    - molto più efficienti
    - impegnano più risorse nei nodi
    - adatti a reti grandi

Link State

- Ogni nodo impara il suo ambito locale: link e nodi adiacenti
- Trasmette queste informazioni a tutti gli altri nodi della rete tramite un messaggio di Link State Advertisement (LSA)
  - **chiamato anche Link State Packet**
- Tutti i nodi, memorizzando i LSA trasmessi dagli altri nodi, si costruiscono una mappa della rete
- Ogni nodo calcola indipendentemente le sue tabelle di instradamento applicando alla mappa della rete un algoritmo di calcolo del cammino minimo
  - **algoritmo di Dijkstra (Shortest Path First)**
    - La complessità è O(*R log N*)
      - *R* è il numero di rami, *N* è il numero di nodi

Link State (cont.)

- Ogni nodo calcola indipendentemente le sue tabelle di istradamento applicando alla mappa della rete l'algoritmo di Dijkstra



Node	LS Advertisement
A	B(2)
B	A(2) D(3) E(2)
C	D(1)
D	B(3) C(1)
E	B(2) F(5) G(2)
F	E(5)
G	E(2) H(1)
H	G(1)

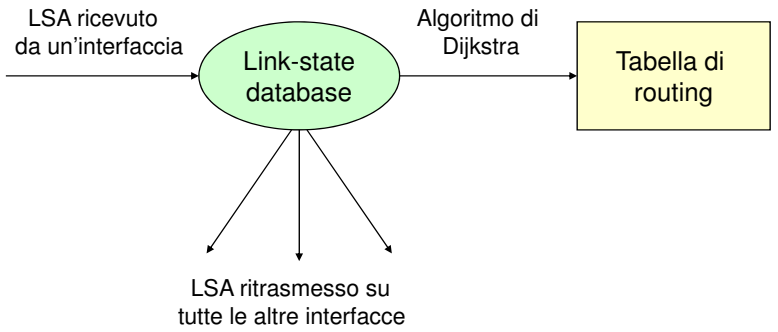
RT-B

Node	NH	Link
A	-	L1
C	D	L2
D	-	L2
E	-	L3
F	E	L3
G	E	L3
H	E	L3

Link State: Flooding

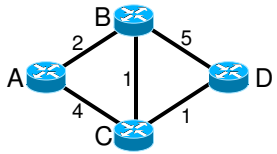
- I messaggi di LSA vengono trasmessi in flooding su tutti i link del nodo che li ha originati
- Un nodo che riceve un LSA lo ritrasmette in flooding solo se esso ha modificato il LSA database del nodo stesso (selective flooding)
  - **All'atto del ricevimento di un LSA un nodo compie le seguenti azioni:**
    - se non ha mai ricevuto LSA da quel mittente o se il num di sequenza del LSA è maggiore di quello del LSA memorizzato nel database, allora memorizza il pacchetto nel database e lo ritrasmette in flooding su tutte le linee eccetto quella da cui l'ha ricevuto
    - se il LSA ricevuto ha lo stesso numero di sequenza di quello posseduto, allora non viene fatto nulla
    - se il LSA è più vecchio di quello posseduto, cioè è obsoleto, allora il nodo ricevente trasmette il LSA aggiornato al nodo mittente
- Questo meccanismo serve a fare in modo che i LSA database di tutti i nodo si mantengano perfettamente allineati e coerenti, condizione indispensabile per un corretto instradamento

Link State (cont.)



Link State - esempio

- Topologia di rete (e costi):



- Messaggi di Link State Advertisement:

LSA_A	
Link	Costo
a-b	2
a-c	4

LSA_B	
Link	Costo
b-a	2
b-c	1
b-d	5

LSA_C	
Link	Costo
c-a	4
c-b	1
c-d	1

LSA_D	
Link	Costo
d-b	5
d-c	1

- Routing Table:

RT-A

A	0	A
B	2	B
C	3	B
D	4	B

RT-B

A	2	A
B	0	B
C	1	C
D	2	C

RT-C

A	3	B
B	1	B
C	0	C
D	1	D

RT-D

A	4	C
B	2	C
C	1	C
D	0	D

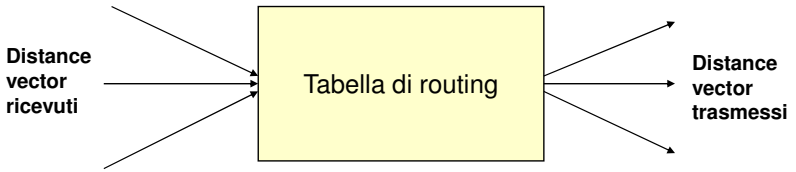
Link State: caratteristiche

- Vantaggi:
  - **Può gestire reti di grandi dimensioni**
  - **Ha una convergenza rapida**
  - **Difficilmente genera loop, e comunque è in grado di identificarli e interromperli facilmente**
  - **Facile da capire: ogni nodo ha la mappa della rete**
- Svantaggi:
  - **Più complesso da realizzare**
- Esempi di protocolli LS:
  - **IS-IS, OSPF**

Distance Vector

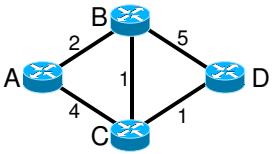
- Noto anche come algoritmo di Bellman-Ford (distribuito)
- Ogni nodo mantiene un database con le distanze minime tra sé stesso e tutte le possibili destinazioni
- Ogni nodo, quando modifica le proprie tabelle di instradamento, invia ai nodi adiacenti un *Distance Dector* (DV)
- Il *Distance Vector* è un insieme di coppie
  - **{indirizzo destinazione, distanza}**
- Quando un nodo *X* riceve un DV da un nodo *Y* adiacente, ricalcola la tabella delle distanze minime; se ci sono modifiche invia il suo nuovo DV (aggiornato) ai nodi adiacenti
  - **$D'_x(z) = \min\{D_x(z), D_y(z)+C_x(y)\}$, $\forall$ z**
  - **dove:**
    - $D_x(z)$ = distanza da X a Z (calcolata da X)
    - $C_x(y)$ = costo del ramo tra il nodo X e il nodo adiacente Y
- La distanza è espressa tramite metriche classiche quali numero di salti e costo

Distance Vector (cont.)



Distance Vector - esempio

- Topologia di rete (e costi):



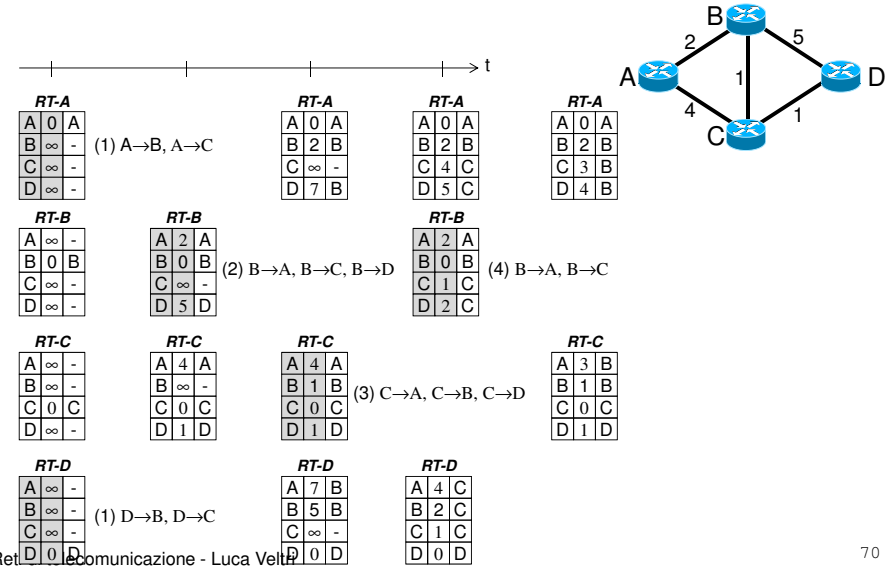
- Routing Table *RT-X*:

Dest.	Dist.	Via
A	$D_x(A)$	$R_x(A)$
B	$D_x(B)$	$R_x(B)$
C	$D_x(C)$	$R_x(C)$
D	$D_x(D)$	$R_x(D)$

- Distance Vector DV_x:

Dest.	Dist.
A	$D_x(A)$
B	$D_x(B)$
C	$D_x(C)$
D	$D_x(D)$

Distance Vector – esempio (cont.)



Distance Vector: caratteristiche

- Vantaggi:
  - **Molto semplice da implementare**
- Svantaggi
  - **Possono innescarsi dei loop a causa di particolari variazioni della topologia**
  - **Converge alla velocità del link più lento e del router più lento**
  - **Difficile capirne e prevederne il comportamento su reti grandi: nessun nodo ha una mappa della rete**
  - **L'implementazione di meccanismi migliorativi appesantisce notevolmente il protocollo**
- Esempi di protocolli DV:
  - **RIP, IGRP (Cisco)**

Distance Vector vs. Link State

- Nel LS i router cooperano per mantenere aggiornata la mappa della rete, poi ogni router calcola il proprio spanning tree autonomamente; nel DV i router cooperano per calcolare direttamente le tabelle di instradamento
- L'algoritmo LS può gestire reti di grandi dimensioni (10000 nodi), il DV generalmente non supera i 1000
- LS ha convergenza rapida, difficilmente genera loop, e comunque è in grado di identificarli e interromperli facilmente; ed è facile da capire e prevedere poiché ogni nodo contiene l'intera mappa della rete

Controllo di errore: rivelazione, correzione, e recupero

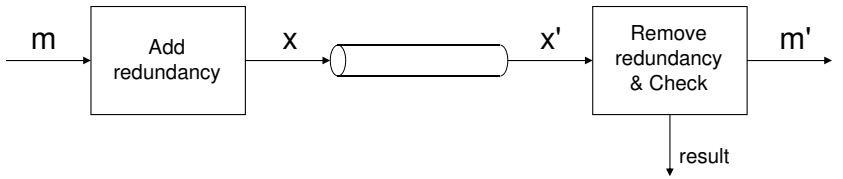
Controllo di errore

- La funzione di controllo di errore si occupa genericamente di rilevare gli errori subiti dalle UI durante il loro trasferimento ed eventualmente ripristinare il corretto flusso informativo
- Può aver a che fare con errori di vario tipo
  - **errori trasmissivi o procedurali, duplicazione, alterazione dell'ordine o perdita di UI**
- Tale funzione in realtà si può comporre di una o più delle seguenti funzioni:
  - **rivelazione di errore**
    - rilevare in ricezione eventuali errori nelle UI ricevute (in genere dovuti a errori trasmissivi)
  - **correzione di errore**
    - correggere eventuali errori di bit o byte all'interno delle UI
  - **recupero di errore**
    - in presenza di errori riportare alla normalità il flusso di UI trasferite tra due entità tramite ritrasmissione delle UI stesse

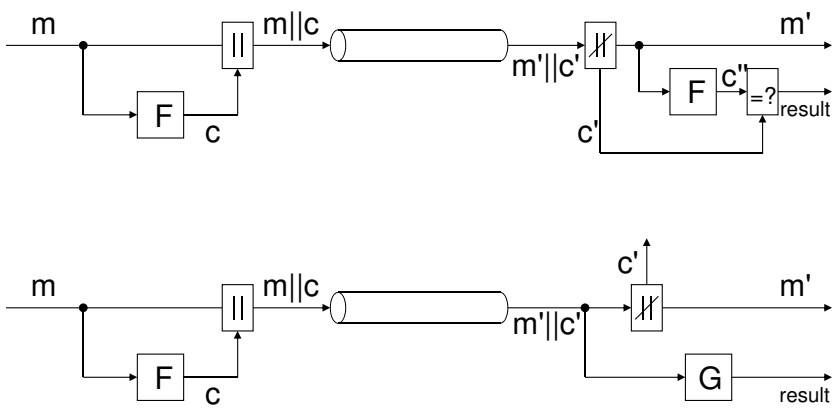
Rivelazione di errore

- Ha come obiettivo quello di rilevare in ricezione eventuali errori nelle UI ricevute (in genere errori trasmissivi)
- Normalmente si basa sull'aggiunta nelle UI in trasmissione di ridondanza (codice di rivelazione di errore all'interno del PCI)
  - **utilizzata in ricezione per rivelare la presenza di errori (non per correggerli)**
- Può essere la base di una eventuale funzione di recupero errore
  - **permettendo il ricevitore di verificare la corretta ricezione o, nel caso contrario, chiedere la ritrasmissione (vedi recupero di errore)**
- 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
  - **la differenza tra un codice di rivelazione di errore e un MIC è che il primo deve rilevare solo modifiche casuali**

Rivelazione di errore: schema generale



Rivelazione di errore: schema con aggiunta di un codice di rivelazione



Rivelazione di errore (cont.)

- La ridondanza richiesta per rivelare gli errori è molto più contenuta rispetto a quella che sarebbe richiesta per la funzione di correzione di errore
  - **un'aggiunta di 16-32 bit è in genere sufficiente**
- Esistono differenti meccanismi di generazione del codice di rivelazione di errore
  - **controllo di parità**
  - **controllo di parità a blocchi**
  - **codici rivelatori d'errore polinomiali**
  - **altri (e.g. checksum)**
- Esempi
  - **campo CRC o FCS in Ethernet e in PPP, checksum in IP, in UDP e in TCP, etc.**

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

Controllo di parità per colonne

- Con bit di parità, non vengono rivelati errori consecutivi (a burst) in numero pari all'interno dello stesso blocco (falsi negativi)
- Per rivelare errori a burst si può suddividere la sequenza di bit in *k* parole di *N* bit e organizzarla come una matrice *kxN*, con una parola per riga
- I bit di parità vengono calcolati sulle colonne
  - ***N* bit di parità**
- Se è presente un errore a burst di lunghezza $h \leq N$, questo influenzerà al più 1 bit per colonna
  - **si riesce a rivelare l'errore**

Controllo di parità a blocchi (righe e colonne)

a) blocco di carattere originario e bit di parità

b) errore trasmissivo non rilevato 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	1
0	1	0	1	0	0	0	0

Bit di parità per righe

Bit di parità per colonne

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	1
0	1	0	1	0	0	0	0

b)

Codici a somma complemento a 1 (Checksum)

- La sequenza di bit da proteggere viene suddivisa in k parole di N bit e organizzata 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)
- corrisponde al checksum usato per la rivelazione di errore dai protocolli IP, UDP, TCP, ed altri protocolli di Internet
- 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
-------------------
1001 1011 0100 0111 = sum
0110 0100 1011 1000 = checksum
```


Esempio di calcolo del Checksum

```
                1         1    1
               10111 0110 1111 100

                0100 0101 0000 0000 +    45 00 +
                0000 0000 0011 0000      00 30
                1100 0001 0000 0000      C1 00
                0100 0000 0000 0000      40 00
                1000 0000 0000 0110      80 06
                0000 0000 0000 0000      00 00
                0000 0001 1001 1000      01 98
                0001 0101 1110 1011      15 EB
                1010 0000 0100 1110      A0 4E
                0001 1101 0011 1110      1D 3E
                -------------------  =   -----  =
somma parziale: 1001 1011 0100 0101      9B 45
riporto:                      10           2
                -------------------  =   -----  =
somma (complemento a 1): 1001 1011 0100 0111   9B 47 = 1001 1011 0100 0111

                1001 1011 0100 0111 ⊕            1001 1011 0100 0111 ⊕
                1111 1111 1111 1111              1111 1111 1111 1111
                -------------------  =           -------------------  =
checksum:       0110 0100 1011 1000      64 B8 = 0110 0100 1011 1000
```


Codici a somma complemeto a 1 (Checksum)

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 rilevata corretta
- Nota: è possibile che venga rilevata corretta una UI errata

Codici rivelatori di errore polinomiali (CRC)

- Codici polinomiali o anche Cyclic Redundancy Check (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 considerate come i coefficienti di un polinomio completo di grado**
  - **l'i-esimo bit è il coefficiente di 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)

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

- dove:
  - **Q(x) è il polinomio quoziente**
  - **R(x) è il polinomio resto**
- La divisione viene effettuata in algebra modulo 2

CRC (cont.)

- I coefficienti del resto R(x) vengono utilizzati come codice di rivelazione di errore e inserito dalla entità emittente 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)

Esempio CRC

- M(X) = msg originale = 101110
- P(X) = x^rM(x) = 101110 000
  - **in questo esempio *P* include anche *r* bit destinati al campo di rivelazione di errore, e posti a 0**
- G(X) = polinomio generatore= 1001
- r = mum. bit di ridondanza = grado del polinomio G = 3
- Q(X) = polinomio quoziente
- R(X) = resto = CRC = 011
- S = msg inviato = M || R = 101110011

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

```
1 0 1 1 1 0 0 0 0 | 1 0 0 1
1 0 0 1
  0 1 0 1
  0 0 0 0
    1 0 1 0
    1 0 0 1
      0 1 1 0
      0 0 0 0
        1 1 0 0
        1 0 0 1
          1 0 1 0
          1 0 0 1
            0 1 1
```


CRC (cont.)

- Si può dimostrare che scegliendo opportunamente G(x), il CRC può rilevare 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 (con r il grado di G(x))**

CRC (cont.)

- I polinomi divisori G(x) di grado 16 più usati in Europa e Nord America sono rispettivamente:

$$G(x)=x^{16}+x^{12}+x^5+1$$

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

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

- Nota: il controllo di errore con bit di parità, può essere visto come semplice CRC di 1, bit con polinomio generatore x+1

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)**
  - **non sono necessari messaggi di 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 rilevazione**
  - **la ridondanza introdotta per correggere l'errore è costante (fissa)**
    - al contrario dei meccanismi di recupero di errorre 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**

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), e/o uso di richieste di ri-emissione selettive (SEL REJ)**
  - **3) uso di temporizzatori**
  - **4) introduzione dei numeri di sequenza (SQN) delle UI**
- Alcune tecniche ARQ: stop&wait, sliding window go-back-N or selective repeat
- Esempi: X.25/HDLC, TCP, SIP

Procedure di recupero ARQ

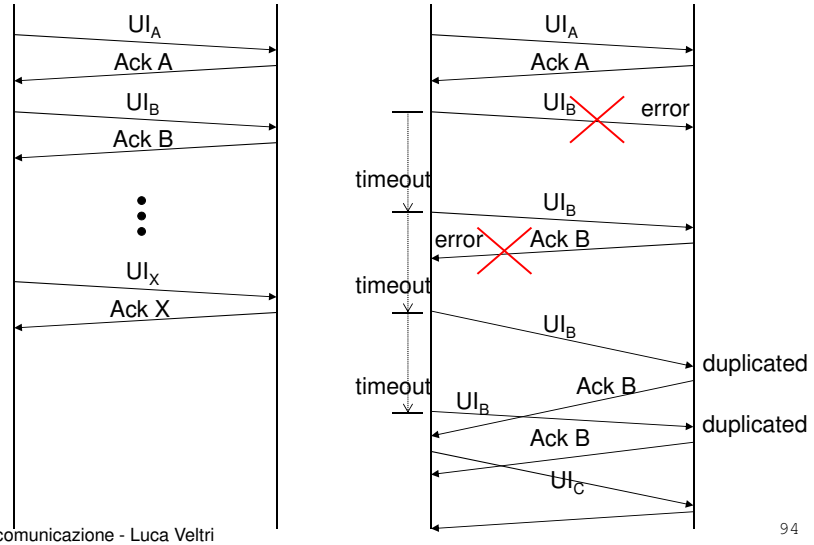
Automatic Repeat Request (ARQ)

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

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 una UI, che può contenere dati diretti nel verso opposto oppure vuota, in cui è presente 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

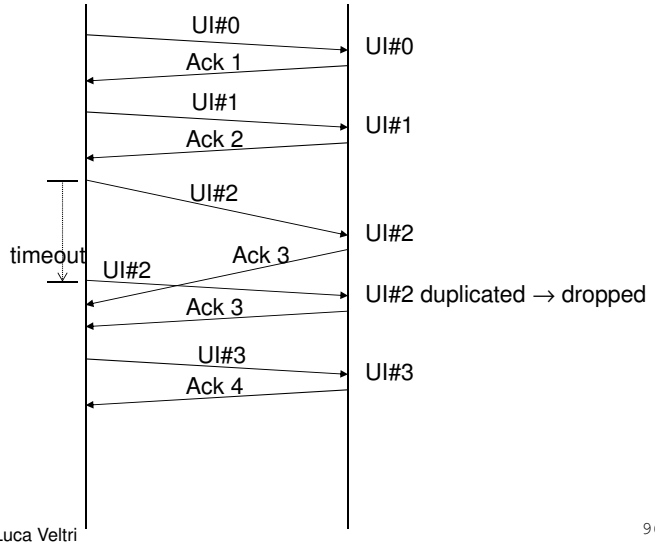
Stop&Wait con ACK e Timeout



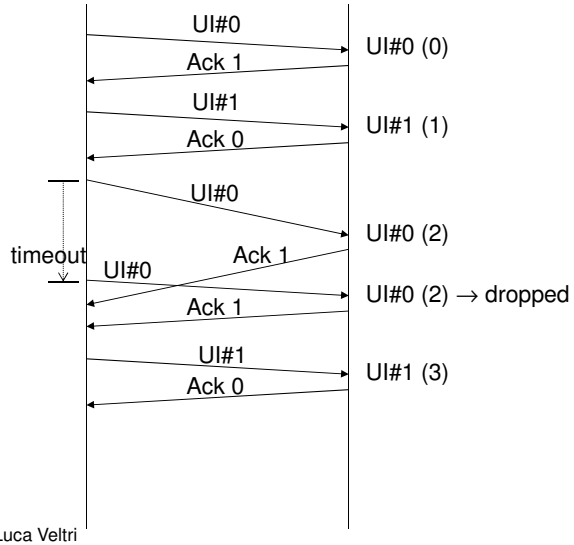
Stop&Wait: SQN

- Per far fronte ad eventuali duplicazioni di UI dati e/o di ACK è importante utilizzare dei numeri sequenza (SQN)
  - **impiegati come identificativi di UI**
- Normalmente negli ACK si preferisce riportare il numero di sequenza successivo a quello riscontrato
  - **ACK *n* ⇒ ricezione corretta di tutte le UI sino al quella con SQN=*n*-1 (ACK cumulativi)**
- Per i numeri di sequenza (SQN) è normalmente riservato nel PCI (header) un numero di bit fissato (e limitato)
  - **se *k* è il numero di bit per SQN, si ha al più *N*=2^k numeri diversi**
  - **le UI sono quindi numerate modulo *N***
- Se non sono possibili fuori sequenza, nello Stop&Wait il valore minimo di *N* per evitare duplicazioni è 2 (1 bit)
  - **se durante il trasferimento è mantenuta la sequenzialità delle UI è sufficiente un solo bit per il conteggio del SQN**

Stop&Wait senza fuori sequenza



Stop&Wait senza fuori sequenza (cont.)



Stop&Wait: Timeout

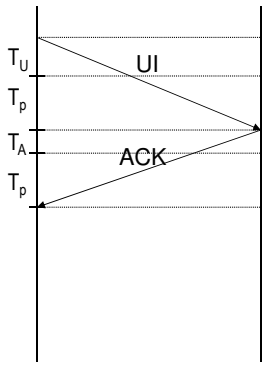
- Il timeout deve essere sufficientemente grande da consentire la trasmissione della UI e della relativa conferma (ACK)
- Un timeout troppo grande rallenta eventuali ritrasmissioni
- Il valore del timeout deve quindi approssimare per eccesso il ritardo A/R ovvero il RTT (Round Trip Time)
- La stima di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

Stop&Wait

- Il trasmettitore
  - **se non impegnato da (ri-)trasmissione di altra UI, riceve nuova UI dallo strato superiore e ne fa una copia e la invia**
    - attiva un orologio (timer)
    - si pone in attesa della conferma di ricezione (ACK)
  - **se scade un timer prima dell'arrivo del ACK corrispondente, ripete la trasmissione e riattiva il timer**
  - **altrimenti se riceve l'ACK prima della scadenza del timer, controlla il numero di sequenza (SQN) e se corretto cancella la copia della UI e diventa disponibile per trasmissione di una UI successiva**
- Il ricevitore
  - **quando riceve una UI controlla la sua integrità; in caso di errore scarta la UI e non effettua altra operazione; altrimenti:**
    - controlla il SQN; se corrisponde al primo SQN non ancora ricevuto, *la UI viene consegnata ai livelli superiori, altrimenti viene scartata*
    - invia un ACK con ACKN uguale al SQN della UI successiva
- Nota: Stop&Wait equivale a Go-back-N con finestra di trasmissione = 1

Stop&Wait - prestazioni in assenza di errori

- Recupero di errore con modalità stop&wait
- Nell'ipotesi che non si verificano errori trasmissivi (UI dati e ACK sempre ricevuti correttamente)
- Indicando con:
  - **T_U = tempo di trasmissione di una UI dati**
  - **T_A = tempo di trasmissione di un ACK**
  - **T_p = tempo di trasferimento end-to-end nella rete, escluso il tempo di trasmissione introdotto dal nodo sorgente**
    - nel caso di un unico ramo rappresenterebbe il tempo di propagazione
- Tempo complessivo necessario per inviare una UI e ricevere il riscontro:
  - **$T_1 = T_U + T_p + T_A + T_p = T_U + T_A + 2\,T_p$**



- Grado di utilizzazione massimo del canale di comunicazione
  - **valore max in assenza di errori (e di ritrasmissioni)**
  - **$\rho = T_U/T_1 = T_U / (T_U + T_A + 2T_p)$**
  - **Se $T_U = T_A$**
    - $\rho = 1/ (2+2T_p/T_U)$
    - se $T_U >> T_p$, $\rho \rightarrow 1/2$
    - se $T_U << T_p$, $\rho \rightarrow 0$
  - **Se $T_U >> T_A$**
    - $\rho = 1/ (1+2T_p/T_U)$
    - se $T_U >> T_p$, $\rho \rightarrow 1$
    - se $T_U << T_p$, $\rho \rightarrow 0$
  - **Se $Tp >> T_U$**
    - $\rho \approx 0$

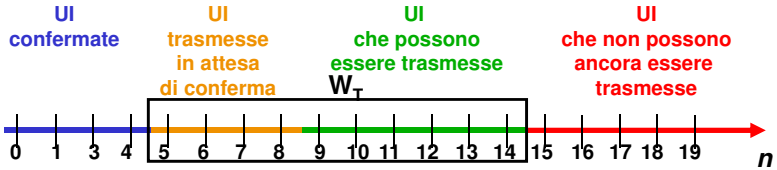
Stop&Wait - prestazioni in presenza di errori

- Recupero di errore con modalità stop&wait
- La sorgente non riceve degli ACK (quando UI o ACK errati/persi)
- Ipotesi:
  - **Indipendenza statistica degli eventi di UI e ACK persi**
  - **Pr {non si riceva un ACK relativo ad una UI dati inviata} = p**
- Tempo medio per inviare una UI e ricevere il riscontro:
  - **a = numero di tentativi per inviare una UI e ricevere il riscontro**
  - **$P_a(1) = Pr\{a=1\} = 1-p$**
  - **$P_a(2) = p(1-p)$**
  - **$P_a(k) = p^{k-1}(1-p)$**
  - **valore medio di a = $E\{a\} = \sum_k p^{k-1}(1-p) = (1-p) \sum_k p^{k-1} =$**
    $= (1-p) \sum d(p^k)/dp = (1-p)\, d (\sum p^k)/dp = (1-p)\, d\, (1/(1-p))/dp =$
    $= (1-p) / (1-p)^2 = 1/(1-p)$
  - **tempo medio $E\{T\} = (E\{a\}-1)TO + T1 = TO\ p/(1-p) + T1$**

- Se timeout di ritrasmissione TO circa uguale al tempo complessivo T_1 necessario per inviare una UI e ricevere il riscontro ($TO \approx T_1$):
  - **$E\{T\} = (E\{a\}-1)TO + T_1 = E\{a\}\ T_1 = (T_U+T_A+2T_p)/(1-p)$**
- Rendimento o grado di utilizzazione del collegamento
  - **$\rho = T_U / E\{T\} = (1-p)\ T_U/T_1 = (1-p)\ \rho_0$**
  - **dove si è indicato con il ρ_0 rendimento nel caso di assenza di errori**

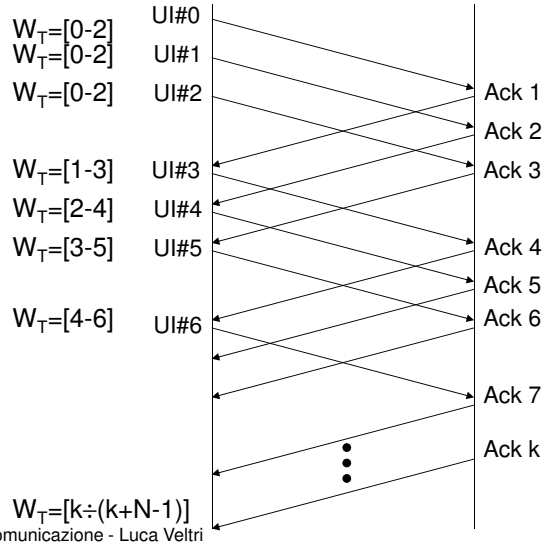
Sliding window

- Il throughput (portata) può essere aumentato consentendo al trasmettitore l'invio di UI successive senza aspettare i relativi riscontri
- Per poterle ritrasmettere, per ogni UI deve essere mantenuta copia siano alla ricezione di un riscontro positivo
- Al fine di limitare la memoria richiesta in trasmissione si può limitare il numero di UI consecutive che possono essere emesse senza ricevere riscontro = N_T
- 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

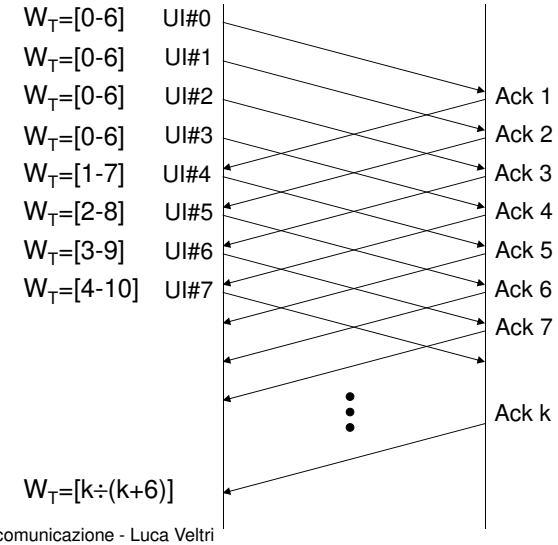


W_T: finestra di trasmissione (dimensione N_T)

Sliding window (N_T=3)



Sliding window (N_T=7)



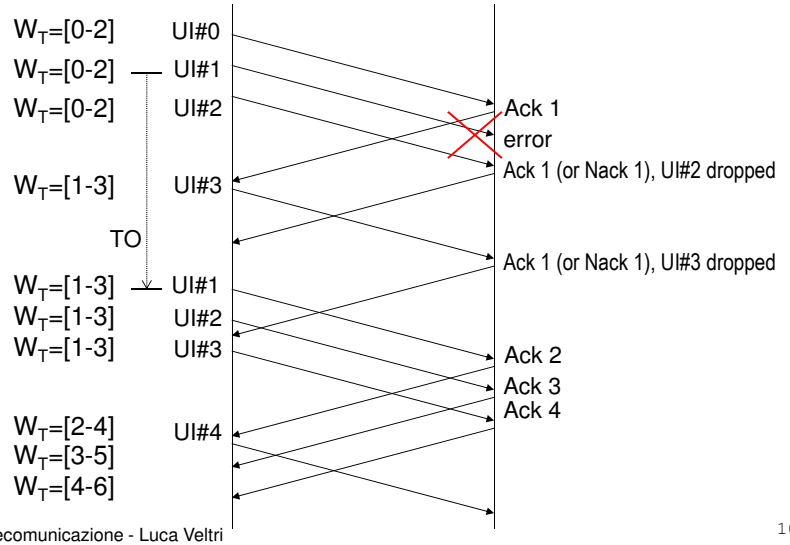
Sliding window (cont.)

- In genere si considerano ACK cumulativi
  - **con ACK-*i* si notifica la corretta ricezione di TUTTE le UI con SQN 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**
- I protocollo *sliding window* si possono differenziare in base alla 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**
- Due tipi di protocolli sliding window
  - **Go-back-N (N_R=1)**
  - **Selective repeat (N_R>1)**

Go-back-N

- Il trasmettitore
  - **invia le UI passate dal livello superiore sino ad un massimo di *N* non confermate, facendo di ognuna una copia in buffer di trasmissione**
    - per ogni UI attiva un orologio (timer)
    - si pone in attesa delle conferme di ricezione (ACK)
  - **se scade un timer prima dell'arrivo di una conferma, *ripete la trasmissione delle UI nel buffer di trasmissione (non ancora confermate),* senza aspettare la scadenza dei timeout successivi**
  - **se arriva un ACK, si considerano confermate tutte le UI trasmesse con numero di sequenza (SQN) minore al numero di ACK (ACKN)**
    - la finestra di trasmissione si sposta in avanti con inizio il ACKN; si terminano i timer relativi a queste UI e si eliminano le UI dal buffer di trasmissione
- Il ricevitore
  - **quando riceve una UI controlla la sua integrità; in caso di errore scarta la UI e non effettua altra operazione; altrimenti:**
    - controlla il SQN; se corrisponde al primo SQN non ancora ricevuto, *la UI viene consegnata ai livelli superiori, altrimenti viene scartata*
    - invia un ACK con ACKN uguale al SQN della prima UI non ricevuta in sequenza
- Nota: Go-back-N è molto simile a Selective repeating con W_R = 1

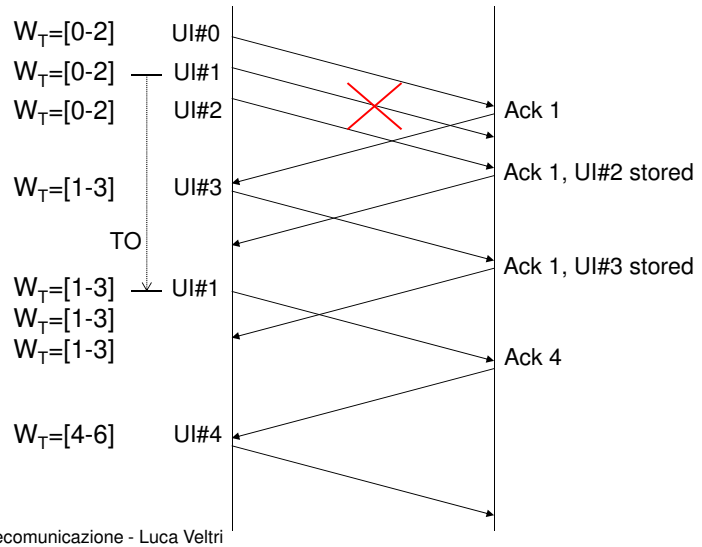
Sliding window Go-Back-N (N_T=3)



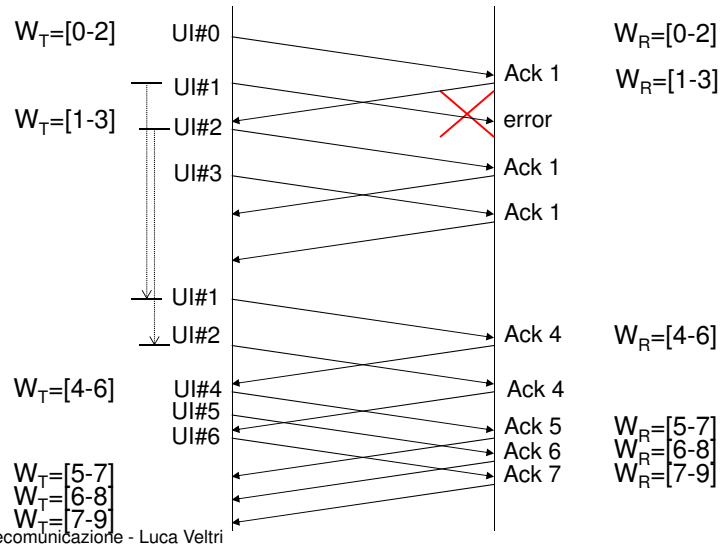
Selective repeating

- Il trasmettitore
  - **invia le UI passate dal livello superiore sino ad un massimo di *N* non confermate, facendo di ognuna una copia in buffer di trasmissione**
    - per ogni UI attiva un orologio (timer)
    - si pone in attesa delle conferme di ricezione (ACK)
  - **se scade un timer prima dell'arrivo di una conferma, *ripete la trasmissione della singola UI associata al timer scaduto***
  - **se arriva un ACK, si considerano confermate tutte le UI trasmesse con numero di sequenza (SQN) minore al numero di ACK (ACKN)**
    - la finestra di trasmissione si sposta in avanti con inizio il ACKN; si terminano i timer relativi a queste UI e si eliminano le UI dal buffer di trasmissione
- il ricevitore
  - **quando riceve una UI controlla la sua integrità; in caso di errore scarta la UI e non effettua altra operazione; altrimenti:**
    - controlla il SQN; se uguale al primo SQN non ancora ricevuto, *la UI viene consegnata ai livelli superiori insieme a quelle immediatamente successive già presenti nel buffer di ricezione; altrimenti se il SQN è all'interno della finestra di ricezione la UI viene salvata nel buffer di ricezione; altrimenti la UI è scartata*
    - invia un ACK con ACKN uguale al SQN della prima UI non ricevuta in sequenza

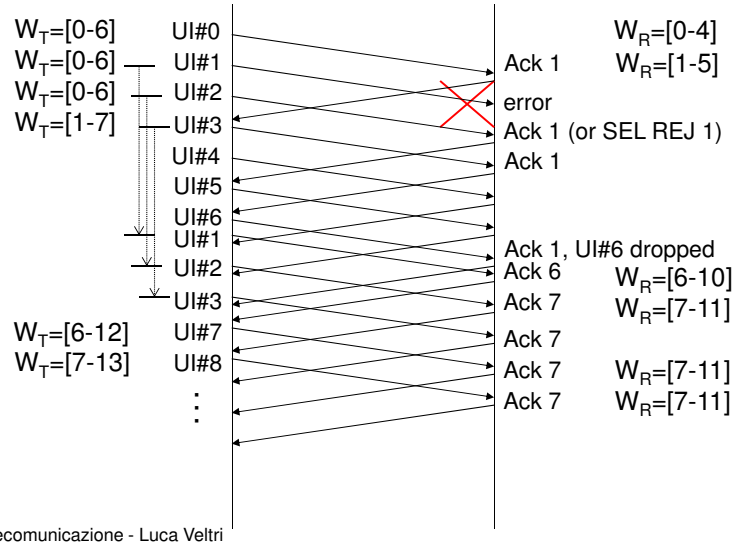
Selective repeating



Sliding window Selective repeating (N_T=3,N_R=3)



Sliding window Selective repeating (N_T=7,N_R=5)



Piggybacking

- Se la comunicazione è di tipo bidirezionale, le stesse UI dati possono essere usate per inviare le informazioni 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 "piggybacking" si intende quella tecnica che ritarda leggermente l'invio di ACK nell'attesa di eventuali UI dati da inviare nello stesso verso del ACK

Sliding window - prestazioni in assenza di errori

- Recupero di errore con modalità sliding window
- Nell'ipotesi che non si verifichino errori trasmissivi (UI dati e ACK sempre ricevuti correttamente)
- Indicando con:
  - **W_T = dimensione della finestra di trasmissione**
- Il tempo complessivo T_n necessario per inviare n UI consecutive e correttamente riscontrate, nel caso che $n < W_T$, è:
  - **$T_n = n\,T_U + 2\,T_p + T_A = (n-1)\,T_U + T_1$**

- In generale, per qualsiasi valore di W_T e n, è possibile ricavare che:
  - **se $T_1 \le W_T T_U$:**
    - $T = (n-1)T_U + T_1$
  - **se $T_1 \ge W_T T_U$:**
    - $T = ((n-1) \text{ div } W_T)T_1 + ((n-1) \bmod W_T))T_U + T_1 =$

      $= T_1(1+((n-1) \text{ div } W_T)) + T_U((n-1) \bmod W_T))$

      con:
      x div y = $\lfloor x/y \rfloor$ = parte intera di x/y
      x mod y = resto della divisione x/y

Controllo di flusso

Controllo di flusso

- Ha il compito di assicurare che il 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 di indicazione da parte del ricevitore della flow window, ovvero dimensione della finestra di numeri di sequenza ammessi dal ricevitore (e.g. TCP)**

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 trasmettitore, 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

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**

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

Altre funzioni

Altre funzioni

- Interfacciamento Fisico
  - **connettori, cavi e mezzi fisici**
- Trasmissione/Ricezione
  - **modulazione/demodulazione, codifica di canale, codifica di linea, etc.**
- Adattamento/Compressione
  - **identificazione, riempimento, compressione, etc.**
  - **e.g. AAL, LLC, IP, L2TP, PPTP, IPSec, etc.**
- 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 connessione o instaurazione di chiamata (Call Setup)
  - **e.g. PPP, TCP, SIP**

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 si 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**