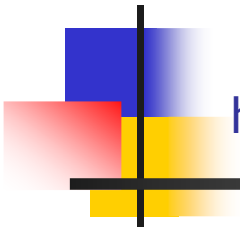
Reti di Calcolatori
AA 2009/2010

UNIVERSITÀ DEGLI STUDI DI TRENTO



http://disi.unitn.it/locigno/index.php/teaching-duties/computer-networks

Il livello Trasporto: UDP e TCP

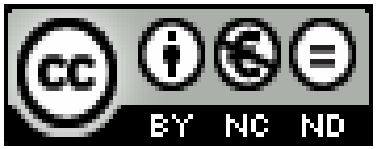
Renato Lo Cigno

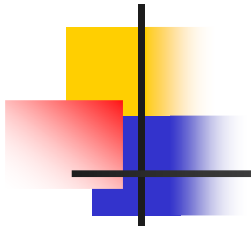
Copyright

Quest'opera è protetta dalla licenza:

Creative Commons
Attribuzione-Non commerciale-Non opere derivate
2.5 Italia License

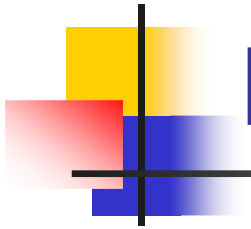
Per i dettagli, consultare
http://creativecommons.org/licenses/by-nc-nd/2.5/it/





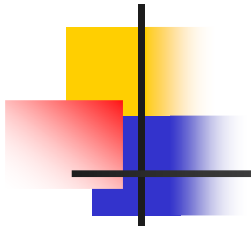
Livello trasporto in Internet

- Due protocolli di trasporto alternativi: TCP e UDP
- Modelli di servizio diversi
  - TCP orientato alla connessione, affidabile, controllo di flusso e congestione, mantiene lo stato delal connessione
  - UDP non connesso, non-affidabile, senza stato
- Caratteristiche comuni:
  - multiplazione e demultiplazione mediante le porte
  - rilevazione (non correzione) errori su header



Posizionamento e visione

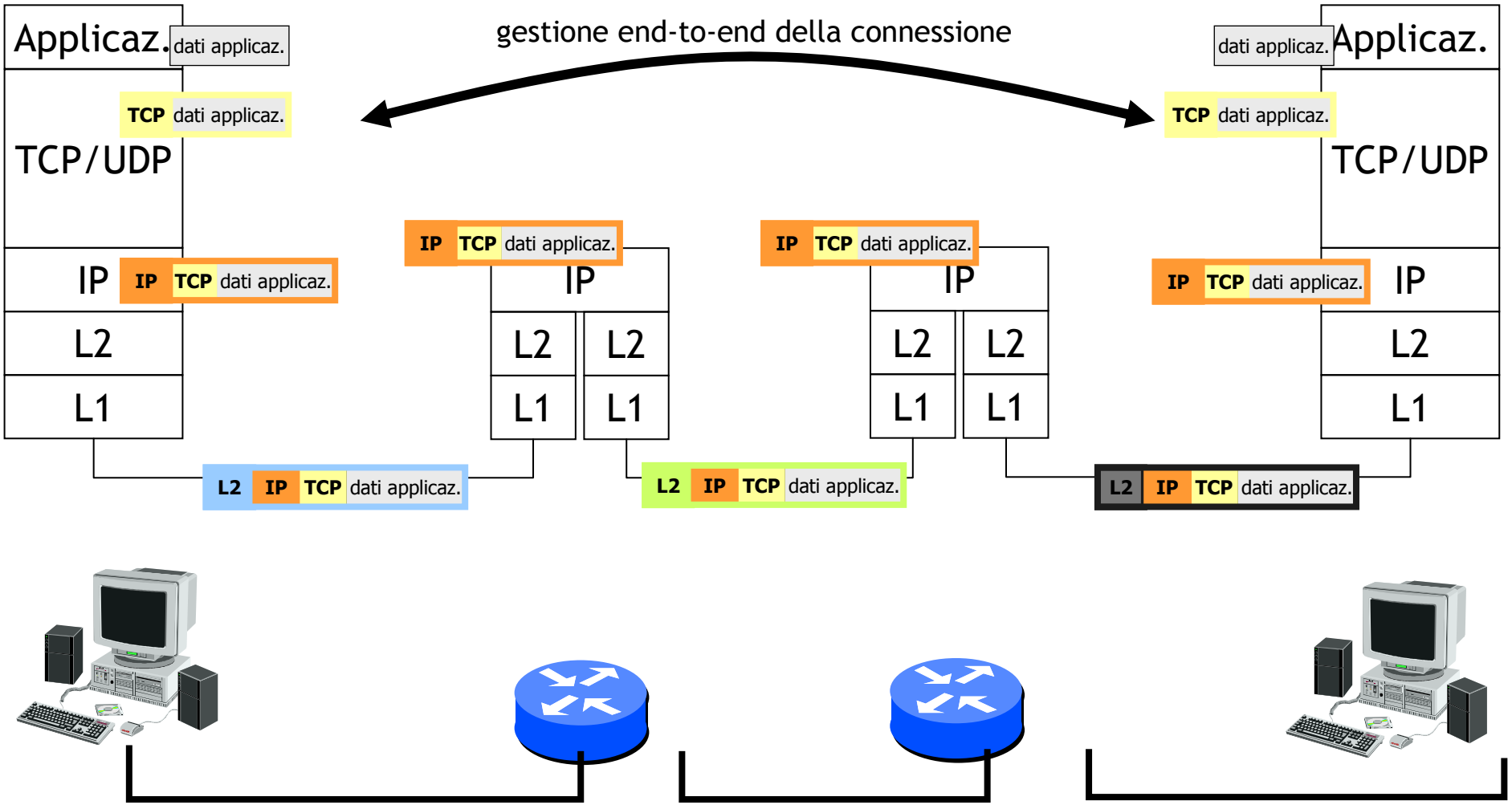
- Il livello trasporto, come quello di applicazione, non ha visibilità dei dettagli della rete
- La semantica del protocollo è end-to-end e la rete viene trattata come una scatola nera
- Lo scopo "generale" del protocollo è fornire una astrazione di comunicazione ai protocolli di livello applicativo
  - attraverso servizi astratti dedicati (SAP – Service Access Point)
  - chiamati socket in gergo Internet



Indirizzamento: mux/demux

- Il destinatario finale dei dati non è un host ma un processo in esecuzione sull'host
- L'interfaccia tra processi applicativi e strato trasporto è rappresentata da una "porta"
  - numero intero su 16 bit
  - associazione tra porte e processi
  - processi server standard e "pubblici" sono associati a porta "ben nota", inferiore a 1024 (es: 80 per WWW, 25 per email)
  - processi client e server non standard o "nascosti" usano una porta superiore a 1024, nel caso dei client assegnata dinamicamente dal sistema operativo

Visione d'insieme



Livello di trasporto

- Fornisce un canale di trasporto end-to-end ideale e privo di errori tra due utenti, indipendentemente dalla rete
- Per compiere questo obiettivo, come tutti i livelli OSI, il livello di trasporto offre, attraverso delle primitive, dei servizi al livello superiore e svolge una serie di funzioni
- Servizi offerti al livello applicativo:
  - connection-oriented affidabile
    - per il trasferimento dei dati viene attivata una connessione
    - ogni pacchetto (→ segmento) inviato viene "riscontrato" in modo individuale

    TCP
  - connectionless non affidabile
    - non viene attivata nessuna connessione
    - invio delle trame senza attendere alcun feedback dalla destinazione sulla corretta ricezione
      - se una trama viene persa non ci sono tentativi per recuperarla

    UDP

Funzioni svolte dal livello di trasporto

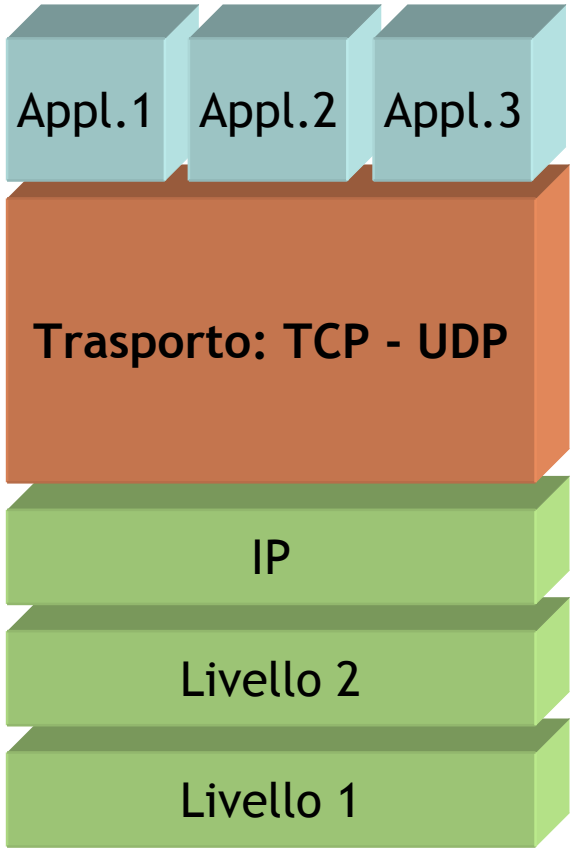
- Indirizzamento a livello applicativo / multiplazione / demultiplazione
  - multiplazione e indirizamento a livello applicativo
  - scarto PDU malformate o con errori nell'header
- Instaurazione, gestione e rilascio delle connessioni
  - si preoccupa di gestire la connessione, ovvero lo scambio di informazioni necessarie per concordare l'attivazione di un canale di comunicazione
- Recupero degli errori (sui dati)
  - politiche implementate e azioni da svolgere in caso di errore o perdita di segmenti
- Consegna ordinata dei segmenti
- Controllo di flusso
  - azione preventiva finalizzata a limitare l'immissione di dati in rete a seconda della capacità end-to-end di questa
- Controllo della congestione
  - azioni da intraprendere come reazione alla congestione di rete

UDP

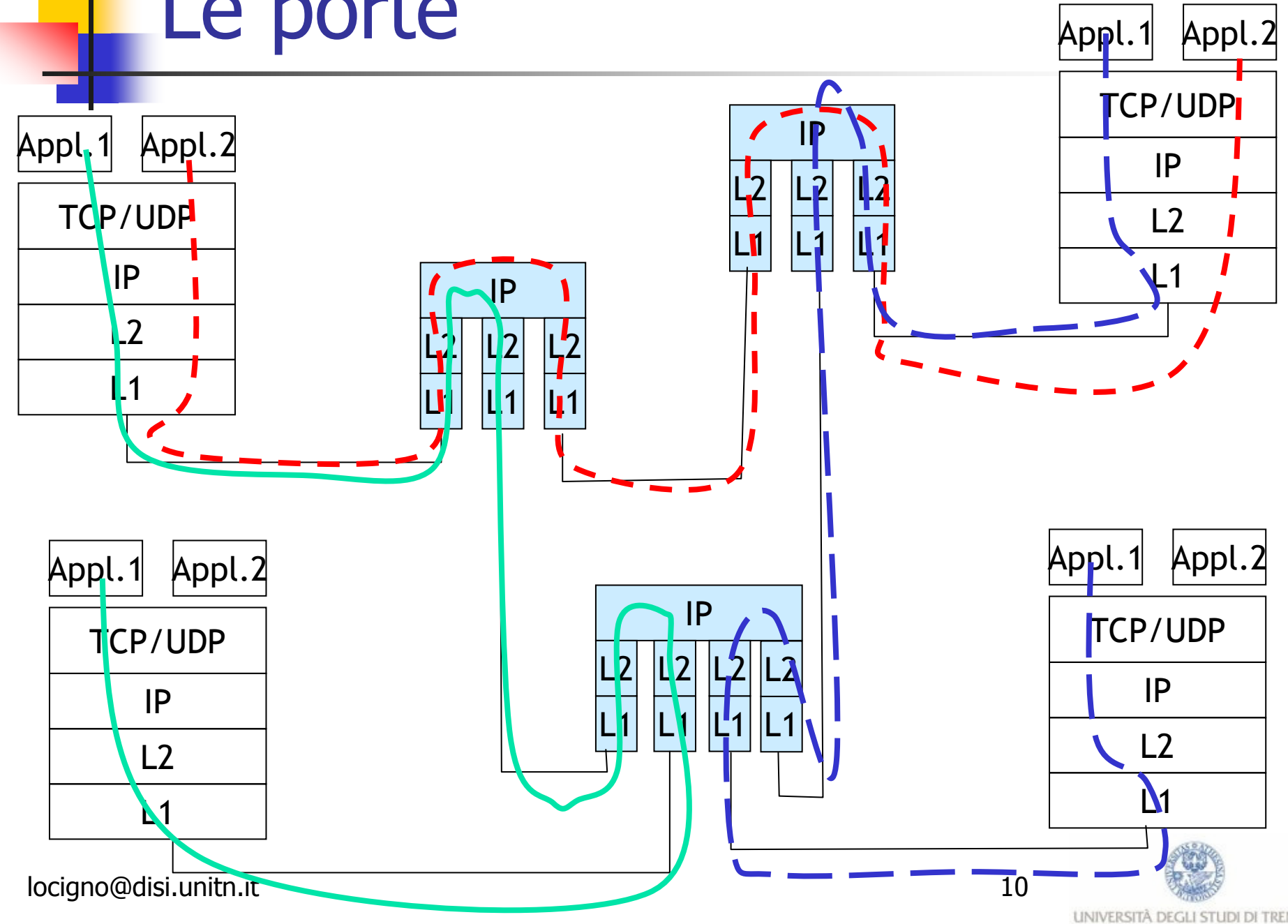
TCP

Generalità

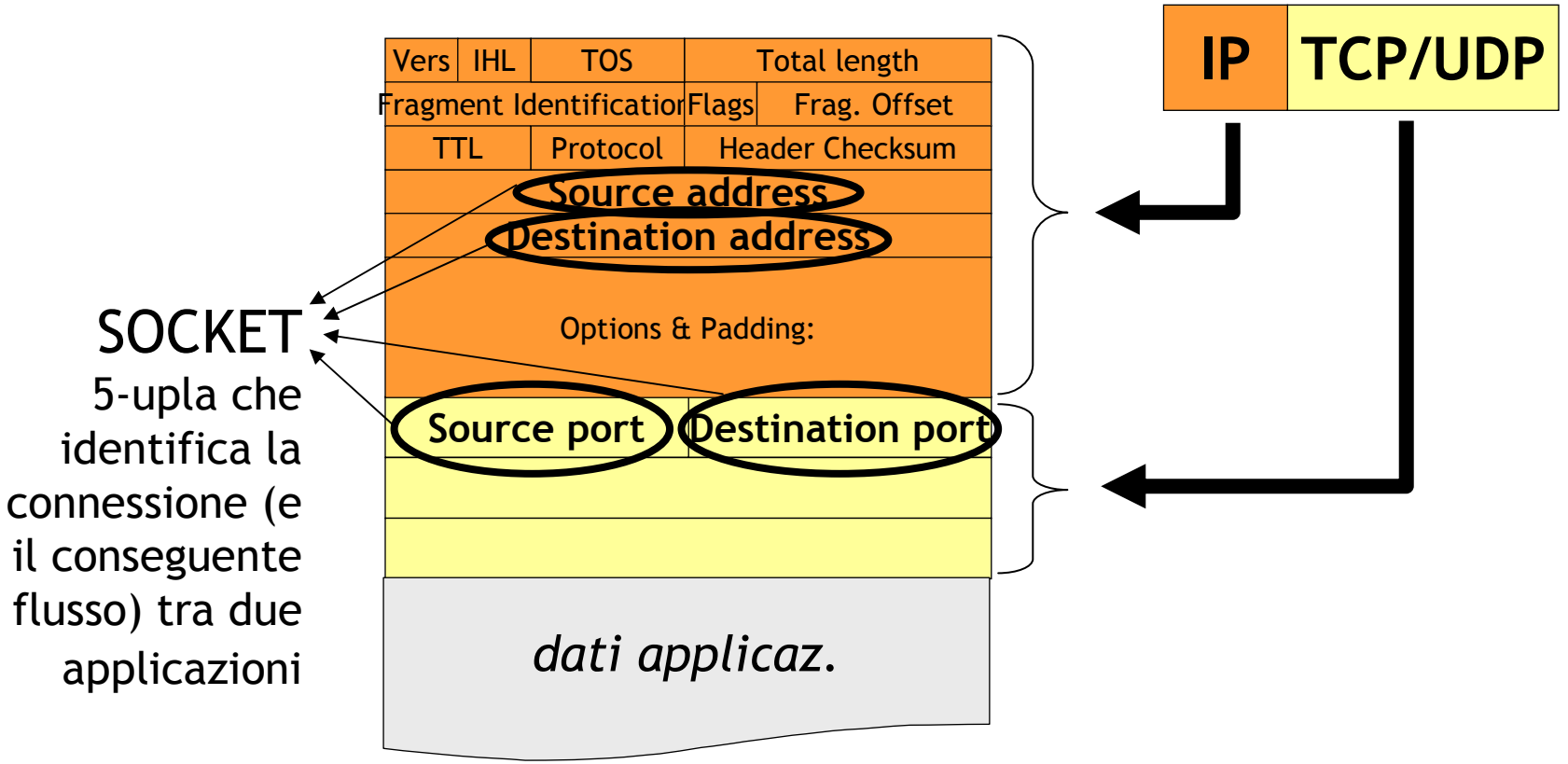
- Gli indirizzi di livello IP vengono utilizzati per l'instradamento dei pacchetti all'interno della rete e identificano univocamente il sistema sorgente e destinazione
- Quando il pacchetto giunge alla destinazione e viene passato al livello di trasporto nasce un ulteriore problema:
  - poiché sul livello di trasporto si appoggiano più applicazioni, com'è possibile distinguere l'una dall'altra?
- Si introduce il concetto di porta, che non e' altro che un codice che identifica l'applicazione



Le porte



Il concetto di Socket

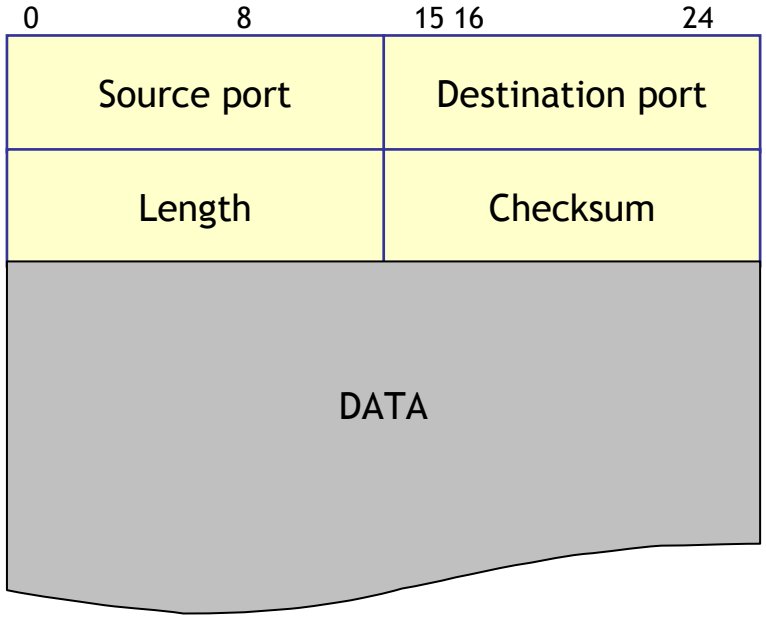


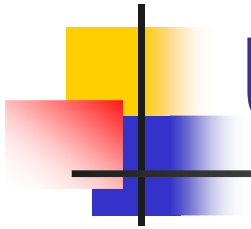
Indirizzamento a livello Trasporto

- Il numero di porta può essere
  - statico (well known port)
    - sono identificativi associati ad applicazioni largamente utilizzate (posta elettronica, web, ftp, dns, ...)
    - esempio: la porta 80 identifica l'applicazione web
  - dinamico (emphemeral)
    - sono identificativi assegnati direttamente dal sistema operativo al momento dell'apertura della connessione
- La porta sorgente e la porta destinazione non sono uguali
- L'utilizzo delle porte, insieme agli indirizzi IP sorgente e destinazione (→socket) servono per il mux/demx dei dati
- I socket identificano univocamente un flusso o connessione
  - passano dati di una sola applicazione
  - una singola appicazione può usare più flussi

Formato dei pacchetti

- Source Port e Destination Port [16 bit]: identificano i processi sorgente e destinazione dei dati
- Length [16 bit]: lunghezza totale (espressa in byte) del datagramma, compreso l'header UDP
- Checksum [16 bit]: campo di controllo che serve per sapere se il datagramma corrente contiene errori



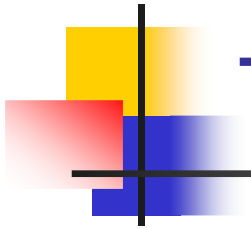


User Datagram Protocol

- E' un protocollo di trasporto connectionless non affidabile
- Svolge solo funzione di indirizzamento delle applicazioni (porte)
- NON gestisce:
  - connessioni
  - controllo di flusso
  - recupero degli errori (solo rilevamento)
  - controllo della congestione
  - riordino dei pacchetti
- E' utilizzato per il supporto di transazioni semplici tra applicativi e per le applicazioni multimediali
- Un'applicazione che usa UDP deve risolvere problemi di affidabilità, perdita di pacchetti, duplicazione, controllo di sequenza, controllo di flusso, controllo di congestione
- Standardizzato in RFC 768

UDP: applicabilità

- **Utile quando:**
  - **Si opera su rete locale (affidabilità)**
  - **Applicazione mette tutti i dati in un singolo pacchetto (non apro connessione)**
  - **Non è importante che tutti i pacchetti arrivino a destinazione**
  - **Necessità di protocollo veloce**
    - **Evita overhead apertura connessione**
    - **Meccanismi di ritrasmissione per affidabilità non utilizzabili per vincoli temporali**
  - **Applicazione gestisce meccanismi di ritrasmissione**

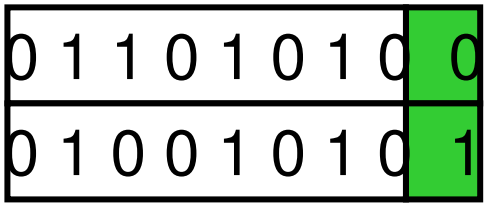


Tecniche ARQ e protocolli a finestra

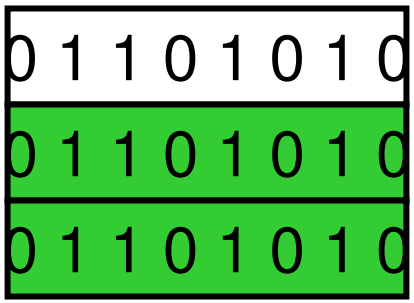
- Per ottenere affidabilità si usano tecniche di ritrasmissione automatica delle informazioni corrotte o perse
  - ARQ – Automatic Retransmission reQuest
- La decisione (automatica) se ritrasmettere l'informazione viene presa in base a protocolli che sfruttano finestre di trasmissione e ricezione
  - stop and wait (finestra pari a una unità dati – PDU)
  - finestre di dimensione fissa
  - finestre con dimensione variabile
- Servono tecniche per
  - rilevare le PDU errate
  - rilevare le PDU mancanti

Esempi di protezione dagli errori

- bit di parità (riconosce errori in numero dispari)

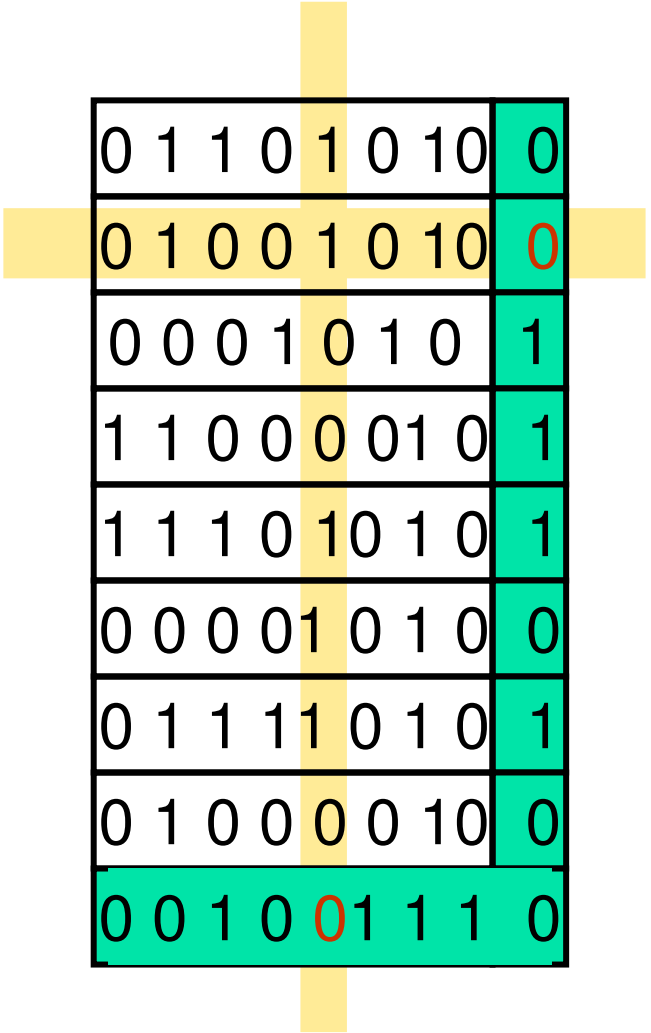


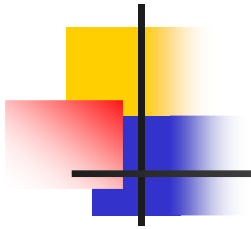
- codice a ripetizione (decisione a maggioranza: permette di correggere errori)



Esempi di protezione dagli errori

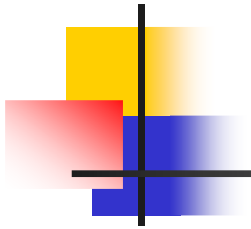
- parità di riga e colonna (consente la correzione di errori singoli)





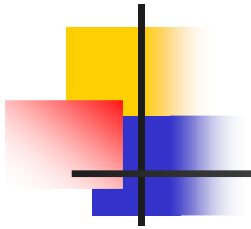
Stop and wait

- Il trasmettitore:
  - invia una PDU dopo avere fatta una copia
  - attiva un orologio (tempo di timeout)
  - si pone in attesa della conferma di ricezione (acknowledgment - ACK)
  - se scade il timeout prima dell'arrivo della conferma, ripete la trasmissione
- Il trasmettitore, quando riceve un ACK:
  - controlla la correttezza dell'ACK
  - controlla il numero di sequenza
  - se l'ACK è relativo all'ultima PDU trasmessa, si abilita la trasmissione della prossima PDU



Stop and wait

- Il ricevitore, quando riceve una PDU:
  - controlla la correttezza della PDU
  - controlla il numero di sequenza
  - se la PDU è corretta invia la conferma di ricezione
  - se la PDU è quella attesa, essa viene consegnata ai livelli superiori



Go Back N

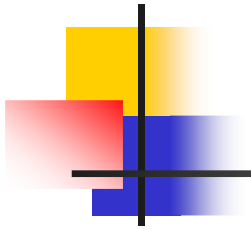
- Il protocollo Stop and wait può essere poco efficiente a causa di elevati ritardi di attesa delle conferme
- Permettere la trasmissione di più di una PDU prima di fermarsi in attesa delle conferme migliora le prestazioni

Finestra di trasmissione

- La finestra di trasmissione W_T rappresenta

  la quantità massima di PDU in sequenza che il trasmettitore è autorizzato ad inviare in rete senza averne ricevuto riscontro (ACK)

- La dimensione della finestra è limitata dalla quantità di memoria allocata in trasmissione
- W_T rappresenta anche il massimo numero di PDU contemporaneamente presenti sul canale o in rete



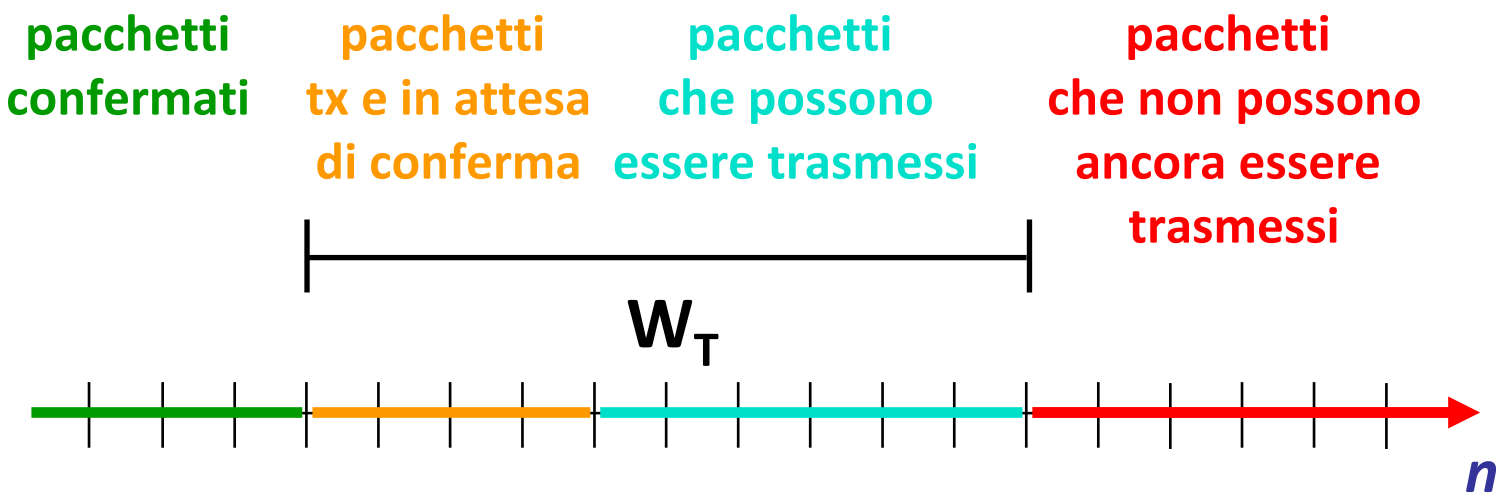
Finestra di ricezione

- La finestra di ricezione W_R rappresenta

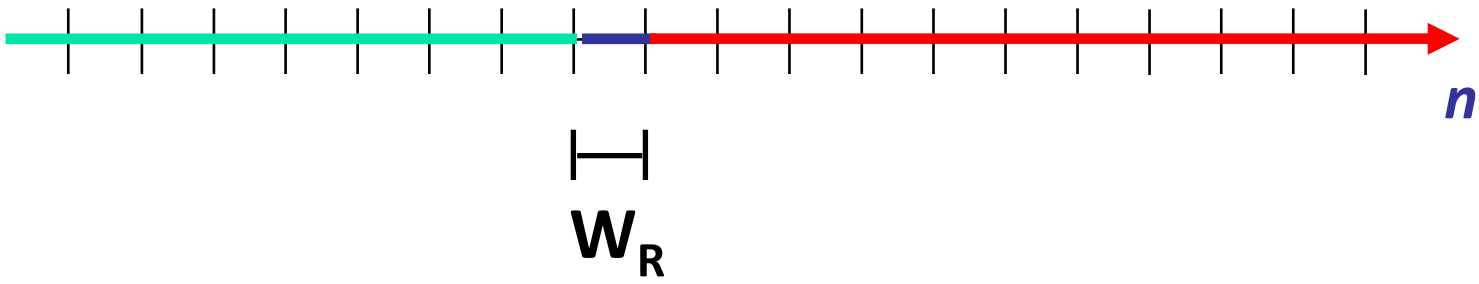
  la sequenza di PDU che il ricevitore è disposto ad accettare e memorizzare

- La dimensione della finestra è limitata dalla quantità di memoria allocata in ricezione

Finestra di trasmissione



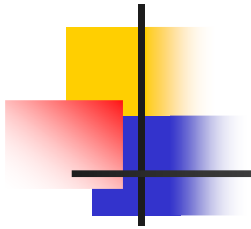
Finestra di ricezione unitaria



pacchetti confermati

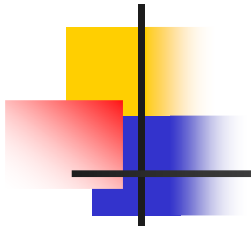
pacchetto atteso

pacchetti fuori sequenza che non possono essere accettati



Go Back N

- Il trasmettitore con finestra N:
  - invia fino ad N = W_T PDU, facendo di ognuna una copia
  - attiva un solo orologio per le N PDU (che viene resettato ad ogni trasmissione di PDU)
  - si pone in attesa delle conferme di ricezione (ACK)
  - se scade il timeout prima della conferma di ricezione relativa alla PDU che ha settato il timeout, ripete la trasmissione di tutte le PDU non ancora confermate

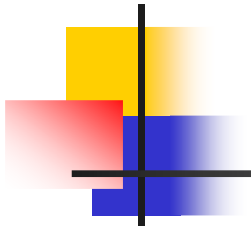


Go Back N

- Il ricevitore, quando riceve una PDU:
  - controlla la correttezza della PDU
  - controlla il numero di sequenza
  - se la PDU è corretta invia la conferma di ricezione
  - se la PDU contiene il primo numero di sequenza non ancora ricevuto, viene consegnata ai livelli superiori

Semantica dei pacchetti di riscontro

- La semantica associata al pacchetto di riscontro può essere:
  - **ACK individuale (o selettivo)**: si notifica la corretta ricezione di un pacchetto particolare.
    ACK(*n*) significa "ho ricevuto il pacchetto *n*"
  - **ACK cumulativo**: si notifica la corretta ricezione di tutti i pacchetti con numero di sequenza inferiore a quello specificato nell'ACK.
    ACK*(n*) significa 'ho ricevuto tutto fino ad *n escluso*"
  - **ACK negativo (NAK)**: si notifica la richiesta di ritrasmissione di un singolo pacchetto.
    NAK(*n*) significa "ritrasmetti il pacchetto *n*"
- Trasmettitore e Ricevitore si devono accordare preventivamente sulla semantica degli ACK

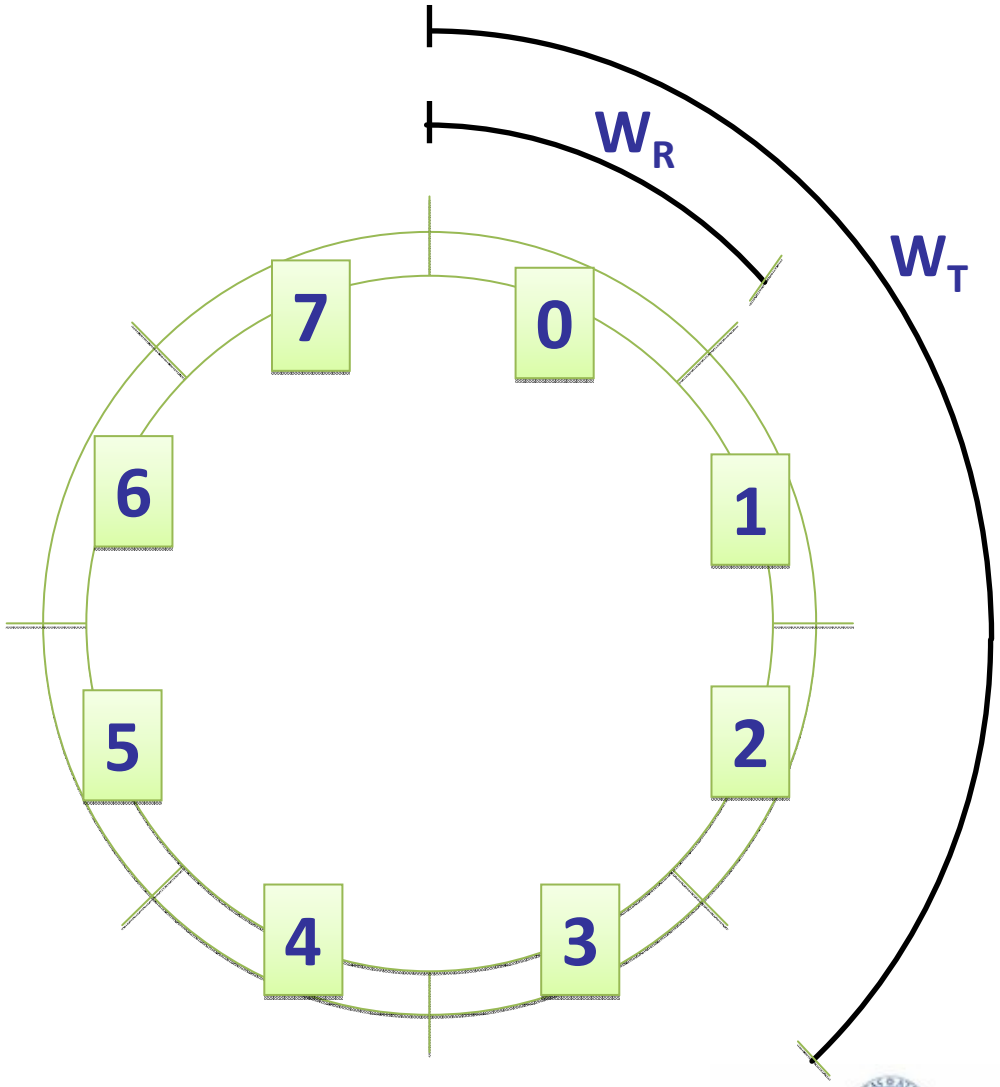


Piggybacking

- Nel caso di flussi di informazione bidirezionali, è sovente possibile scrivere l'informazione di riscontro (ACK) nella intestazione di PDU di informazione che viaggiano nella direzione opposta.

Numerazione PDU

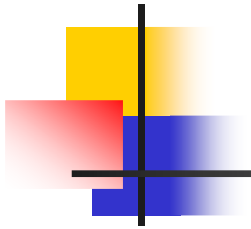
- La numerazione delle PDU è ciclica:
  - k bit di numerazione
  - numerazione modulo 2^k



3 bit di numerazione

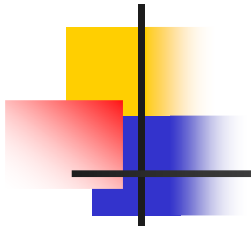
$W_R = 1$

$W_T = 3$



Go Back N

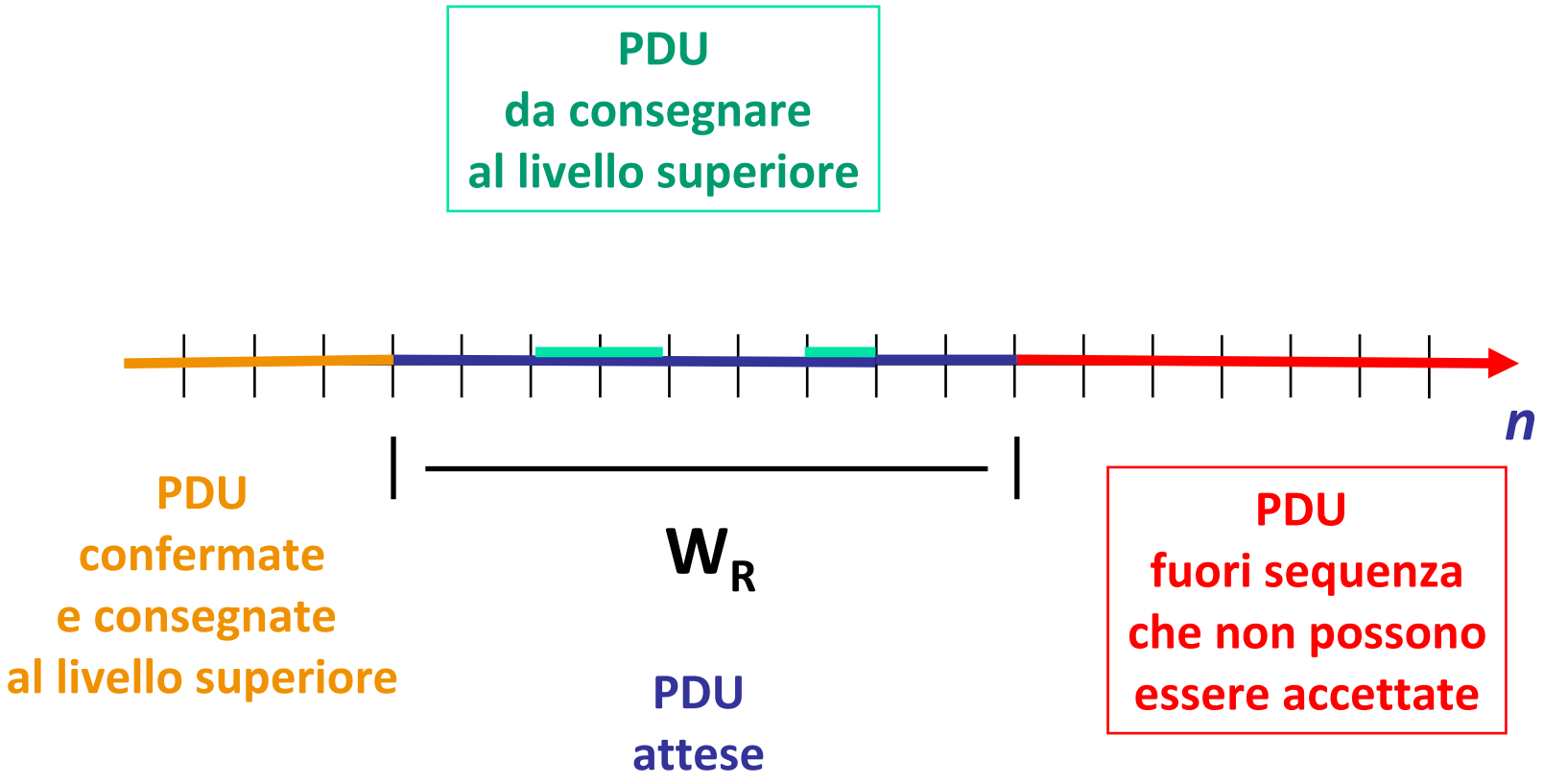
- Il Trasmettitore è significativamente più complesso rispetto al caso dello Stop and wait
  - quantità di memoria
  - gestione dell'orologio
  - algoritmi
- Ricevitore inalterato (finestra 1)
- Si possono usare conferme cumulative (su gruppi di PDU)
  - orologio al ricevitore
- La finestra di trasmissione non può avere dimensioni arbitrarie: $W_T < 2^k$

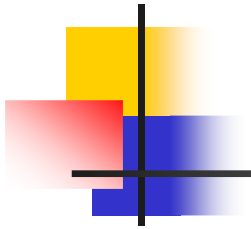


Selective Repeat

- Nel protocollo Go back N il ricevitore può accettare solo PDU in sequenza
- Accettare PDU corrette, ma fuori sequenza, migliora le prestazioni: Selective repeat
- Il protocollo Selective Repeat usa finestra di trasmissione e finestra di ricezione di dimensioni maggiori di 1 (di solito di pari dimensione)
- Esistono diverse possibili implementazioni che si differenziano per:
  - uso di ACK selettivi o cumulativi;
  - uso di timer associati alle singole PDU o alla finestra
  - comportamenti del trasmettitore e del ricevitore
- Descriviamo caso con ACK cumulativi e timer associati alla finestra

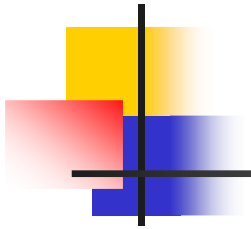
Finestra di ricezione maggiore di 1





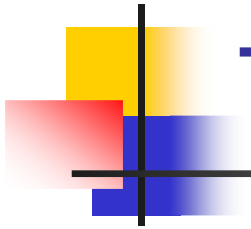
Selective Repeat

- Il trasmettitore:
  - invia fino ad N = W_T PDU, facendo di ognuna una copia
  - attiva un solo orologio per le N PDU (che viene resettato ad ogni trasmissione di PDU)
  - si pone in attesa delle conferme di ricezione (ACK)
  - se scade il timeout prima della conferma di ricezione relativa alla PDU che ha settato il timeout, ripete la trasmissione di tutte le PDU non ancora confermate



Selective Repeat

- Il ricevitore:
  - riceve una PDU
  - controlla la correttezza della PDU
  - controlla il numero di sequenza
  - se la PDU è corretta ed in sequenza la consegna al livello superiore (eventualmente insieme ad altre PDU ricevute in sequenza)
  - se la PDU è corretta ma non in sequenza:
    - se è entro la finestra di ricezione la memorizza
    - se è fuori dalla finestra di ricezione la scarta
  - invia un ACK relativo all'ultima PDU ricevuta in sequenza (ack cumulativi)



TCP ... finalmente

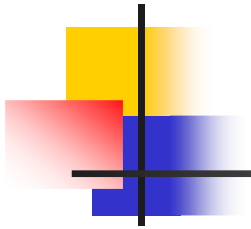
- Il "Transmission Control Protocol" è un protocollo a finestra con ACK cumulativi, go-back-N (versione base)
  - La finestra di ricezione e trasmissione possono essere variate dinamicamente durante la comunicazione
  - versioni e patch successive lo hanno reso uno dei protocolli più complessi della suite di Internet
- TCP è orientato alla connessione
- Affidaibile
- Implementa controllo di flusso
- Cerca di controllare il traffico iniettato in funzione della congestione nella rete

TCP: riferimenti bibliografici

- Richard Stevens: TCP Illustrated, vol.1
- RFC 793 (1981)
  - Transmission Control Protocol
- RFC 1122/1123: (1989)
  - Requirements for Internet Hosts
- RFC 1323: (1992)
  - TCP Extensions for High Performance (PRP STD)

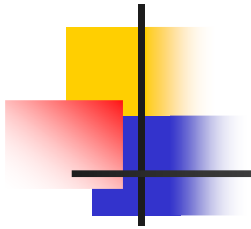
TCP: riferimenti bibliografici

- RFC 2018: (1996)
  - TCP Selective Acknowledgment Options (PRP STD)
- RFC 2581:
  - TCP Congestion Control (PRP STD)
- RFC 2582:
  - The NewReno Modification to TCP's Fast Recovery Algorithm
- RFC 2883:
  - An Extension to the Selective Acknowledgement (SACK) Option for TCP
- RFC 2988:
  - Computing TCP's Retransmission Timer (PRP STD)



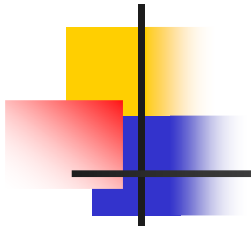
TCP: compiti

- Fornisce porte per (de)multiplazione
- Una entità TCP di un host, quando deve comunicare con un'entità TCP di un altro host, crea una connessione fornendo un servizio simile ad un circuito virtuale
  - bidirezionale (full duplex)
  - con controllo di errore e di sequenza
- Richiede maggiore capacità di elaborazione rispetto a UDP e di mantere informazioni di stato negli host per ogni connessione
- TCP segmenta e riassembla i dati secondo le sue necessità:
  - tratta stream di dati (byte) *non strutturati* dai livelli superiori
  - non garantisce nessuna relazione tra il numero di read e quello di write (buffer tra TCP e livello applicazione)



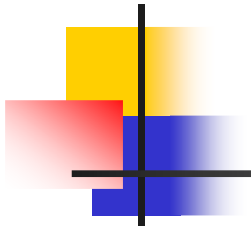
Identificazione di connessioni

- Una connessione TCP tra due processi è definita dai suoi endpoints (punti terminali), univocamente identificati da un socket:
  - Indirizzi IP host sorgente e host destinazione
  - Numeri di porta TCP host sorgente e host destinazione
- Nota: TCP o UDP usano porte indipendenti
- Esempio: connessione TCP tra porta 15320 host 130.192.24.5 e porta 80 host 193.45.3.10



Trasmettitore TCP

- Suddivide i dati dell'applicazione in segmenti
- Usa protocollo a finestra con $W_T \geq 1$
- Attiva timer quando invia i segmenti:
  - segmenti non confermati allo scadere del timer (timeout - RTO) provocano ritrasmissioni
- Stima RTT per impostare timeout
- Calcola e trasmette checksum obbligatorio su header e dati
- Regola velocità con dimensione finestra
  - controllo di flusso e congestione



Ricevitore TCP

- Riordina segmenti fuori sequenza e scarta segmenti errati
  - consegna stream ordinato e corretto a processo applicativo
- Invia ACK cumulativi
- Annuncia negli ACK lo spazio libero nel buffer di ricezione per controllare velocità trasmettitore (controllo di flusso)
- Segmento corretto ed in sequenza
  - Memorizza (ed eventualmente consegna al livello superiore) ed invia ACK cumulativo
- Segmento duplicato
  - Scarta ed invia ACK relativo all'ultimo segmento ricevuto in sequenza
- Segmento con checksum errato
  - Scarta senza inviare ACK
- Segmento fuori sequenza
  - Memorizza (non obbligatorio, ma standard de facto) ed invia ACK relativo all'ultimo segmento ricevuto in sequenza (ACK duplicato)

TCP: generazione ACK

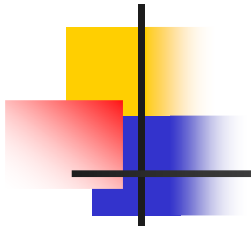
Eventi	**Azioni ricevitore TCP**
arrivo segmento in ordine, inviato ACK correttamente per tutti segmenti precedenti	**invia ACK**
arrivo segmento fuori sequenza con numero maggiore di quello atteso: vuoto rilevato	**invia ACK duplicato, indicando come numero di sequenza il prossimo byte che si attendedi ricevere**
arrivo di segmento che riempie vuoti parzialmente o completamente	**ACK immediato se il segmento copre parte iniziale della finestra**

Controlli di flusso e congestione

- Generico protocollo a finestra: la velocità di trasmissione in assenza di errori è

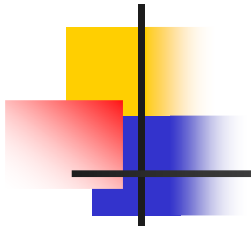
$$\frac{\text{Finestra di trasmissione}}{\text{Round trip time}}$$

  - Connessioni "corte" ottengono banda maggiore
- Per regolare velocità di trasmissione posso agire su
  - round trip time (ritardando invio di ack)
    - genero ritrasmissioni
  - dimensione finestra



Controlli di flusso e congestione

- Se cresce finestra oltre valore per cui bit rate in trasmissione supera capacità del collo di bottiglia
  - si memorizzano dati nei buffer lungo il percorso, e cresce round trip time
  - si perdono dati nei buffer dei nodi intermedi
- TCP: Velocità di trasmissione di un trasmettitore è regolata da:
  - controllo di flusso: evita che un host veloce saturi un ricevitore lento
  - controllo di congestione: evita che un host 'aggressivo' saturi la rete trasmettendo sempre alla velocità massima consentita dal ricevitore

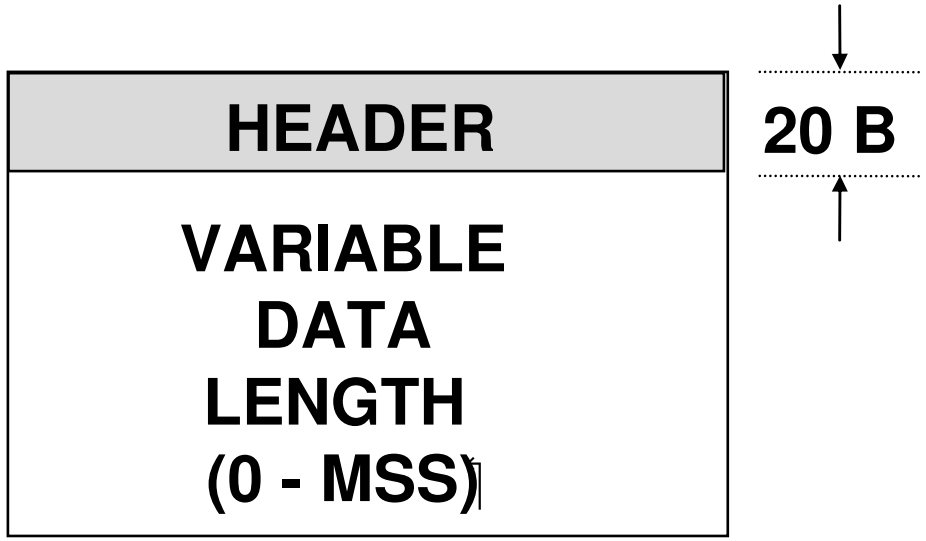


Controlli di flusso e congestione

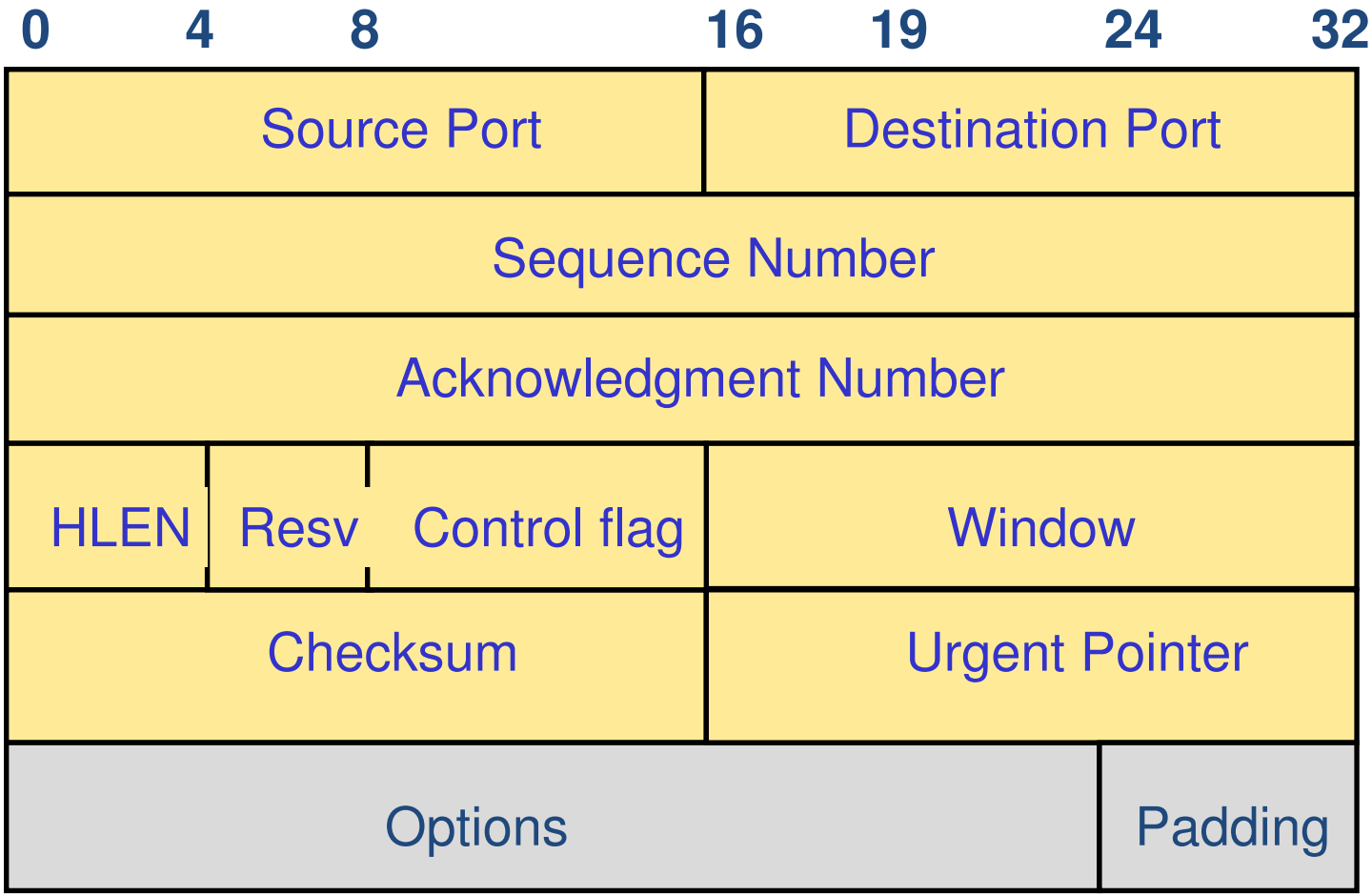
- TCP impiega un controllo end-to-end basato su controllo della dimensione della finestra del trasmettitore:
  - **controllo di flusso**: il ricevitore impone la dimensione massima della finestra del trasmettitore, indicando negli ACK la finestra di ricezione disponibile
  - **controllo di congestione**: il trasmettitore si autoimpone una dimensione massima della finestra in funzione delle perdite riscontrate per mancato arrivo di ACK

La PDU TCP

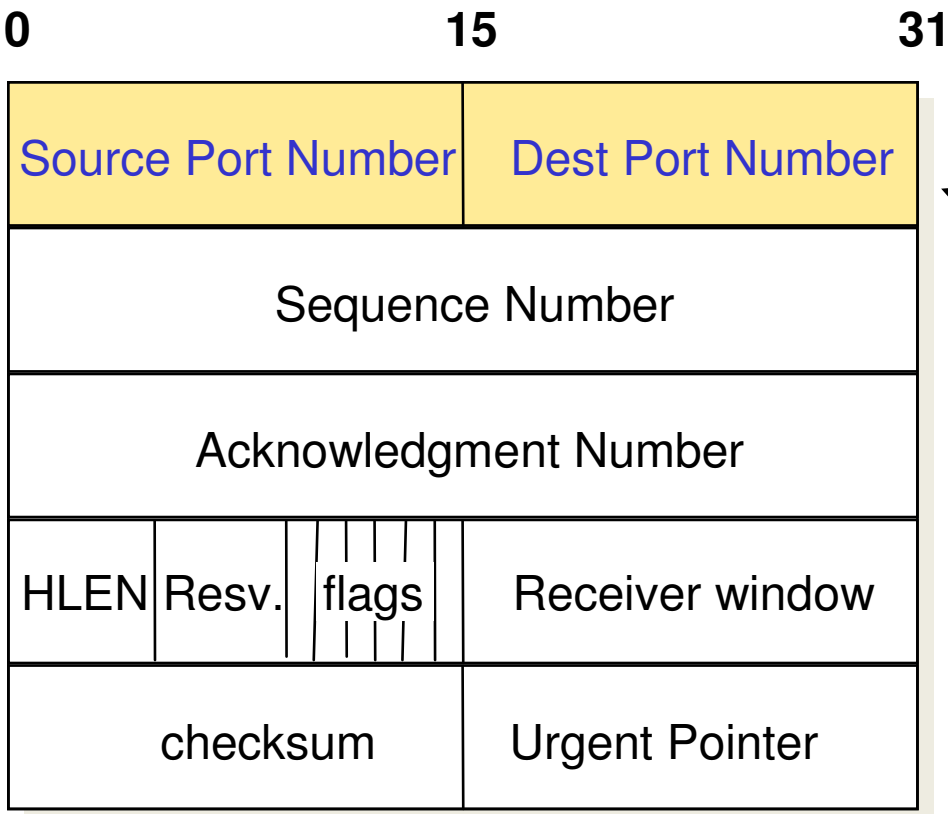
- La PDU di TCP è detta *segmento*
- La dimensione dei segmenti può variare dal solo header (ACK, 20 byte) fino ad un valore massimo MSS concordato con il ricevitore e dipendente dalla MTU IP
- La dimensione del singolo segmento dipende dallo stream dei livelli superiori



TCP header

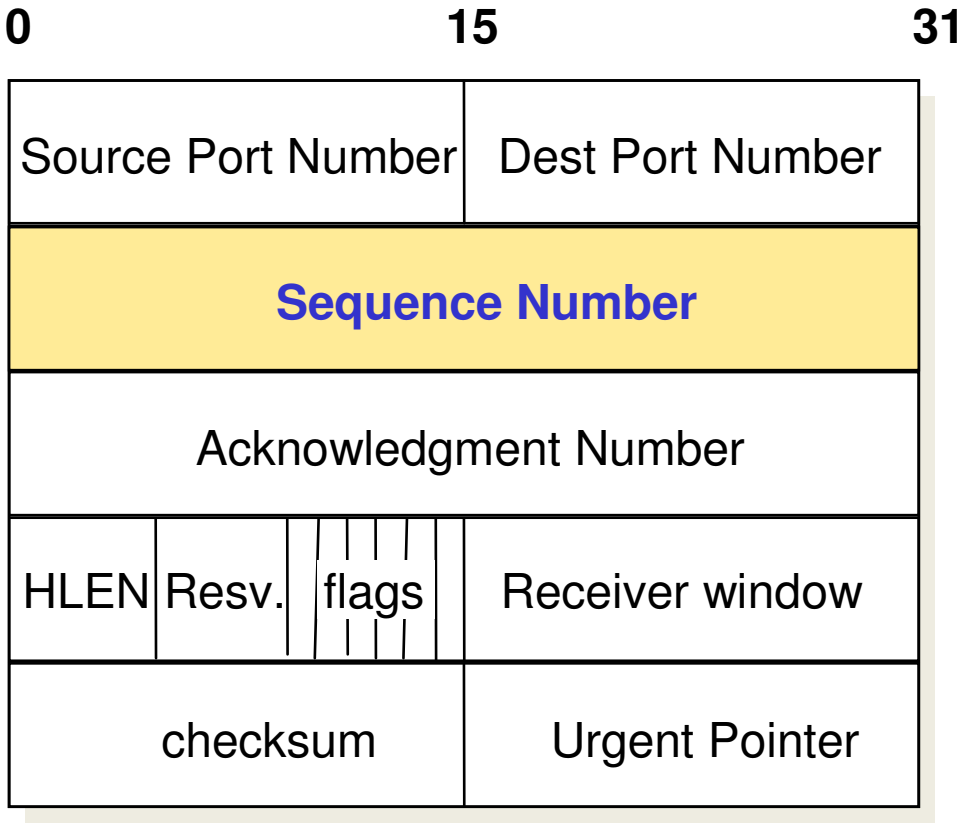

0
4
8
16
19
24
32
Source Port
Destination Port
Sequence Number
Acknowledgment Number
HLEN
Resv
Control flag
Window
Checksum
Urgent Pointer
Options
Padding

TCP header



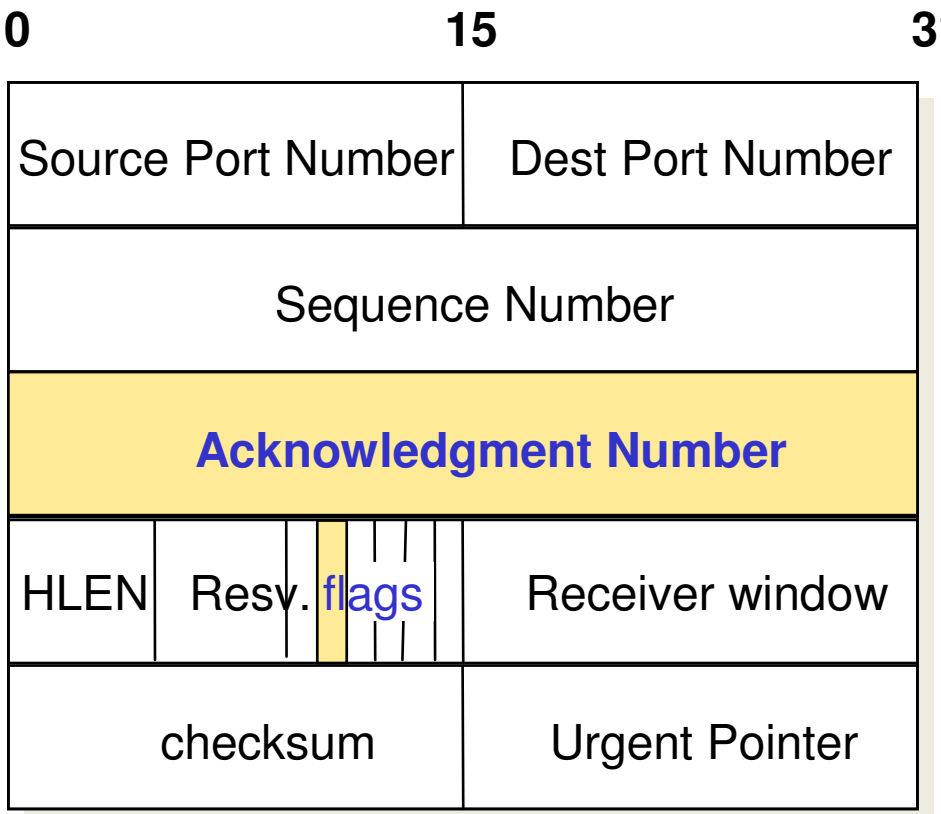
Identificano l'applicazione che sta inviando e ricevendo dati.
Combinati con i rispettivi indirizzi IP, identificano in modo univoco una connessione

TCP header



- Identifica, nello stream di dati, la posizione del primo byte del payload del segmento
- Numerazione ciclica su 32 bit
- Ogni direzione della connessione procede con numeri di sequenza diversi e indipendenti

TCP header



Numero di sequenza più 1 dell'ultimo byte di dati ricevuto correttamente

Valido solo con ACK flag settato

Numeri di sequenza e di ack

Host A

Host B

Utente preme tasto 'C'

Seq=42, ACK=79, data = 'C'

host conferma ricezione di 'C', ed invia echo

Seq=79, ACK=43, data = 'C'

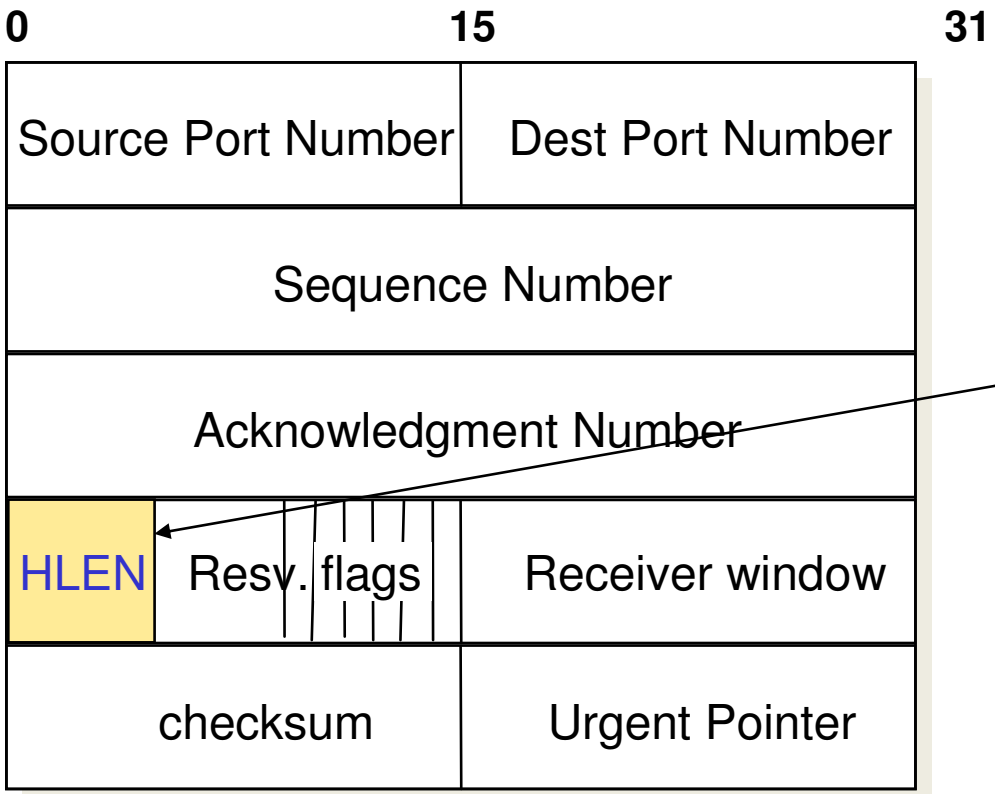
host conferma ricezione dell'eco 'C'

Seq=42, ACK=80

tempo

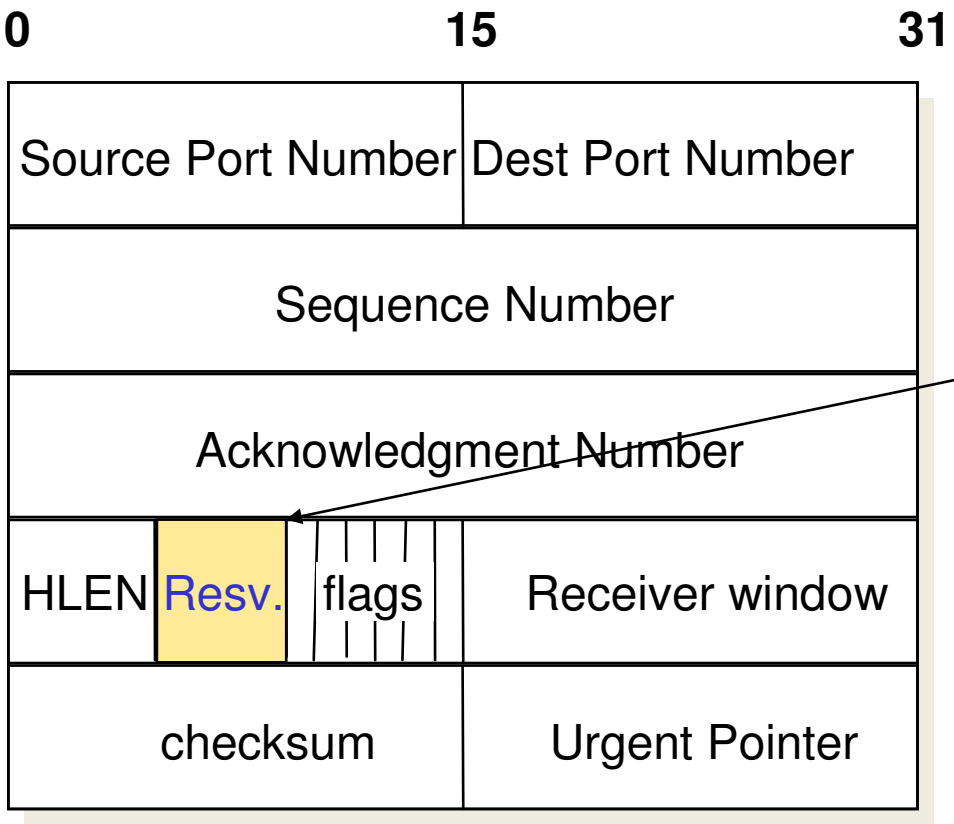
Esempio di sessione telnet

TCP header



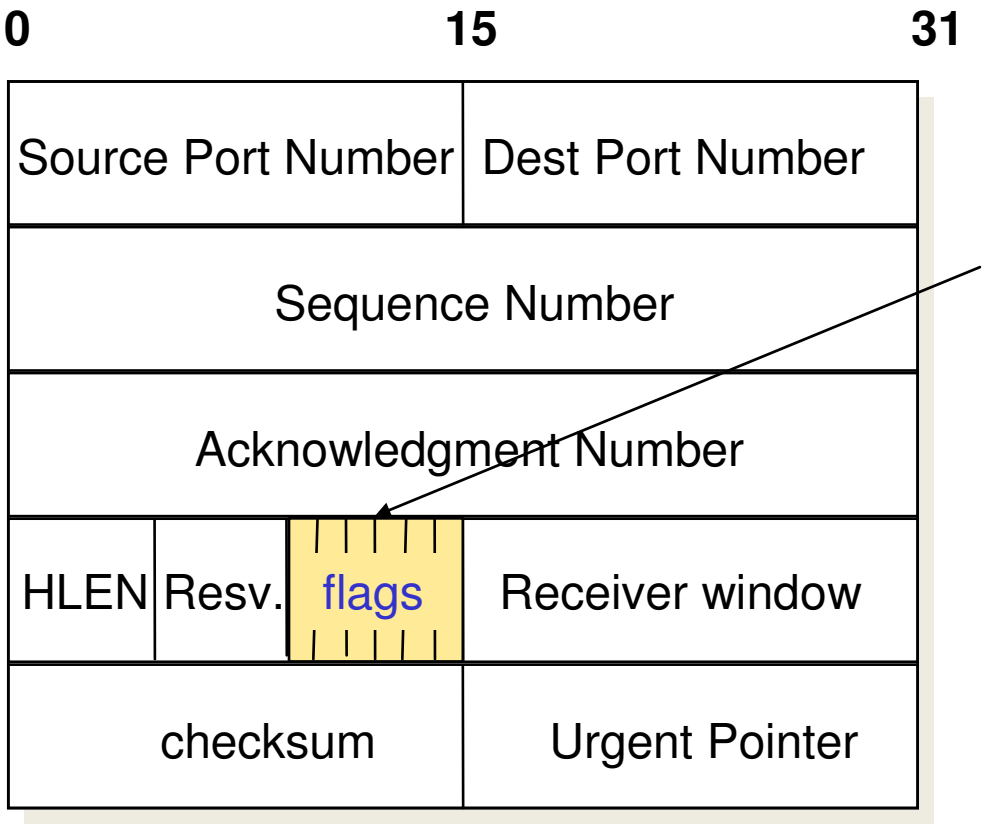
Lunghezza dell'header in parole di 32 bit

TCP header


0
15
31
Source Port Number
Dest Port Number
Sequence Number
Acknowledgment Number
HLEN
Resv.
flags
Receiver window
checksum
Urgent Pointer
RiservatI per usi futuri
(ECN)

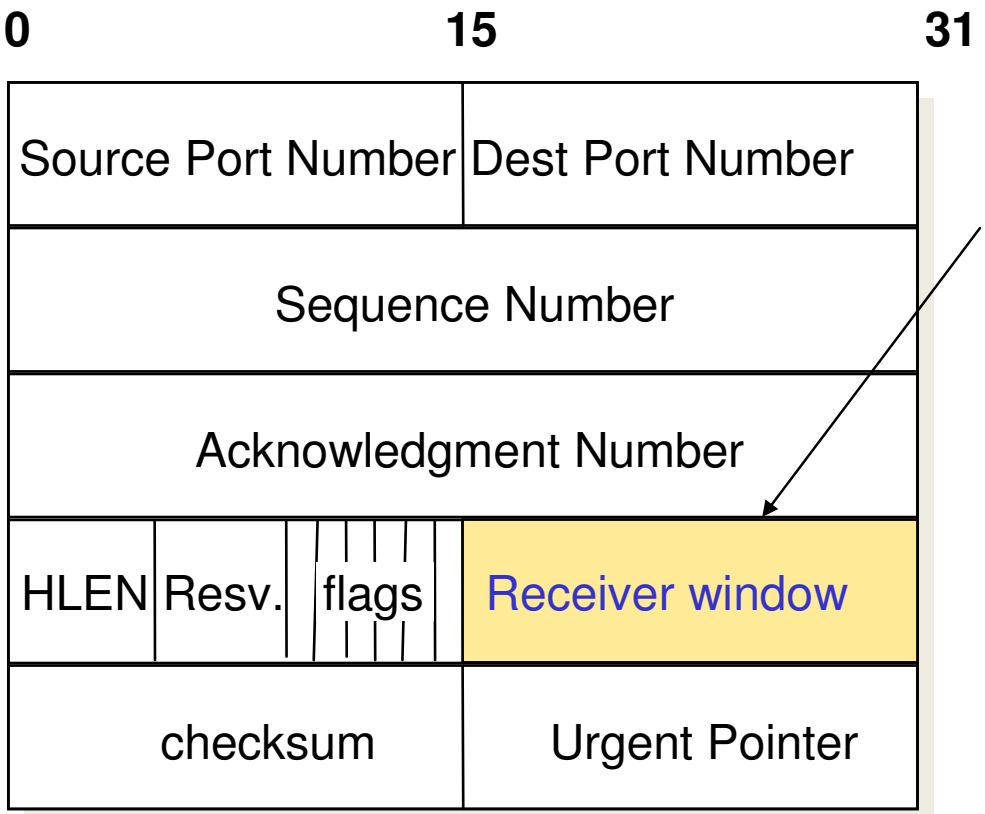
TCP header

Gestione connessione



- Sei bit di flag, uno o più possono essere settati insieme:
  - URG: urgent pointer valido
  - ACK: numero di ack valido
  - PSH: forza passaggio dati applicazione
  - RST: reset connessione
  - SYN: synchronize seq. No. Apertura connessione
  - FIN: chiusura connessione

TCP header



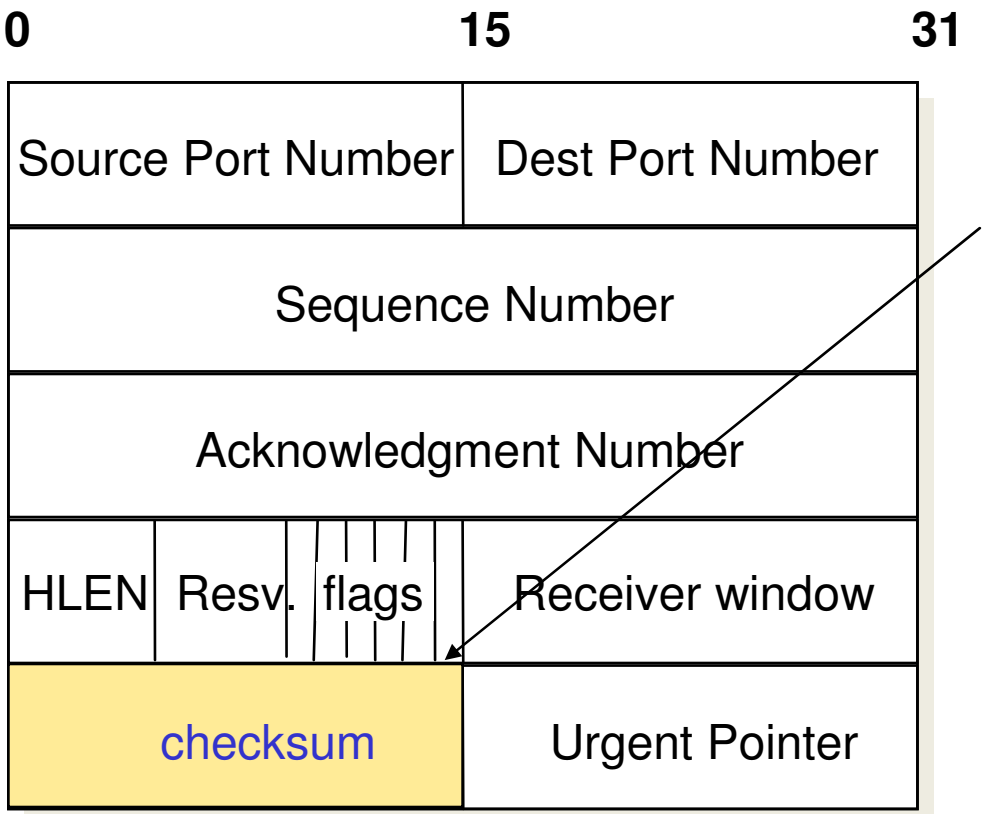
Numero di byte, a partire da quello nel campo di ACK, che il ricevitore è disposto ad accettare per controllo di flusso
Valore massimo rwnd 65535 byte

Finestra necessaria per ottenere velocità massima

- Massima quantità di dati in transito per RTT:
  - 16-bit rwnd = 64KB max
- Prodotto banda x ritardo dato RTT=100ms

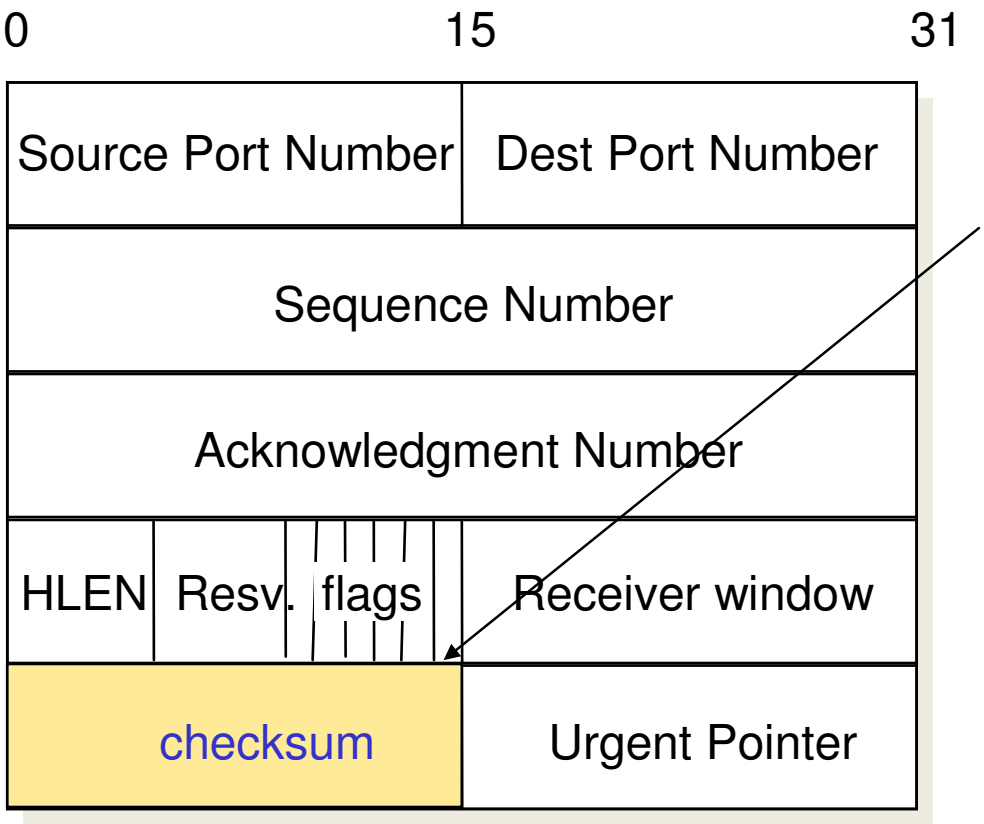
Banda		banda x ritardo
T1	(1.5Mbps)	18KB
Ethernet	(10Mbps)	122KB
T3	(45Mbps)	549KB
FastEthernet	(100Mbps)	1.2MB
STS-3	(155Mbps)	1.8MB
STS-12	(622Mbps)	7.4MB
STS-48	(2.5Gbps)	29.6MB

TCP header



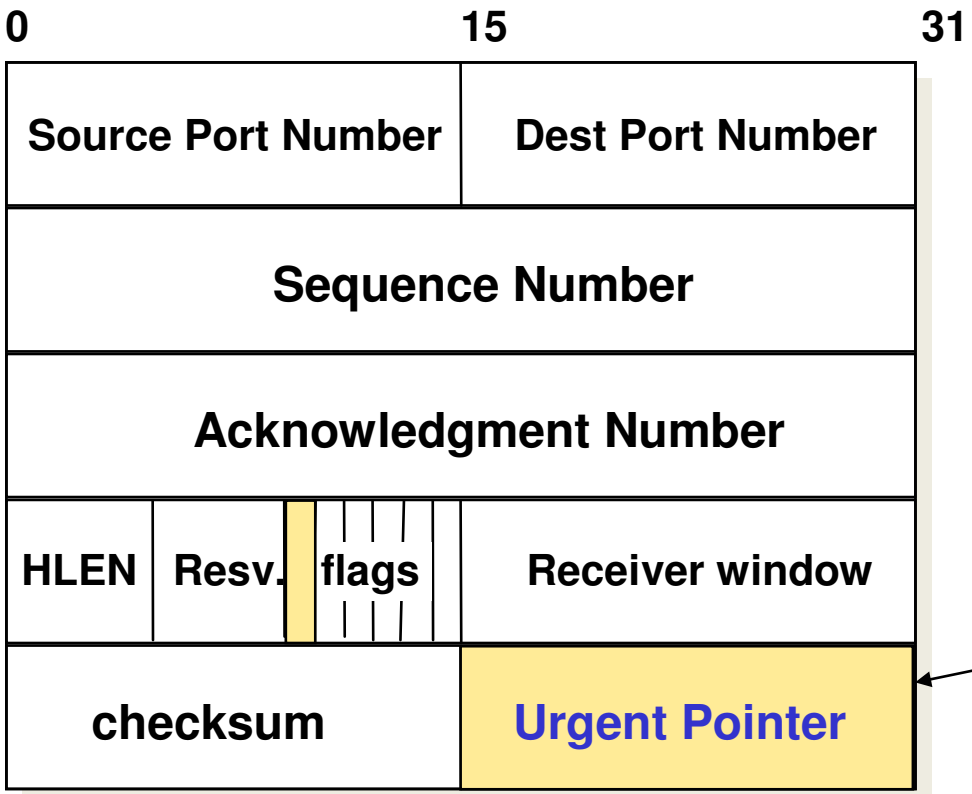
Checksum obbligatorio su header e dati, più pseudo-header che include indirizzi IP e tipo di protocollo (violazione del principio di stratificazione OSI)

TCP header



- Algoritmo di checksum
  - allineamento di header, dati e pseudo-header su 16 bit
  - somma in complemento a 1 di ogni riga
  - si ottiene numero a 32 bit, che si divide in due parti di 16 bit
  - somma in complemento a 1 delle due parti, incluso il riporto
  - inserisco nell'header i 16 bit risultanti

TCP header

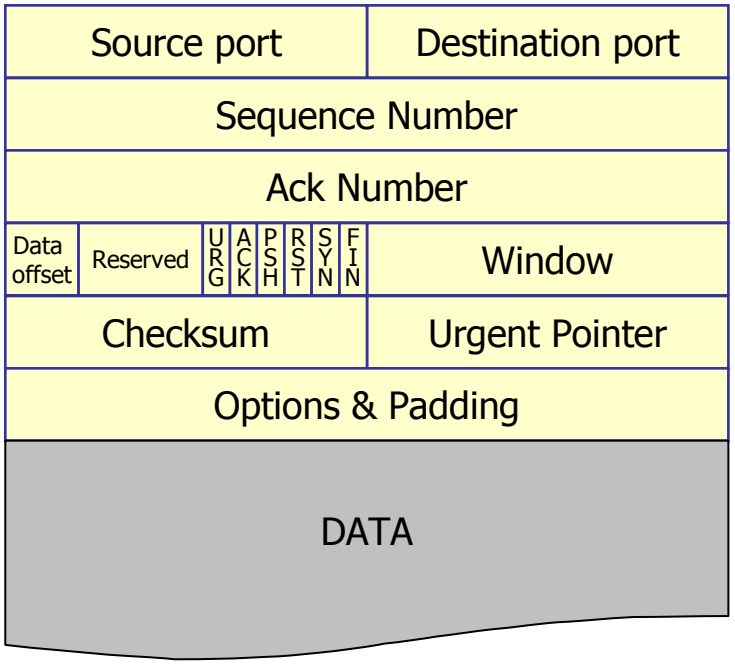


Puntatore a "dati urgenti" nel campo dati (es. ctrl-C in una sessione telnet).
Offset rispetto al num. di seq.

Valido solo se flag URG è settato

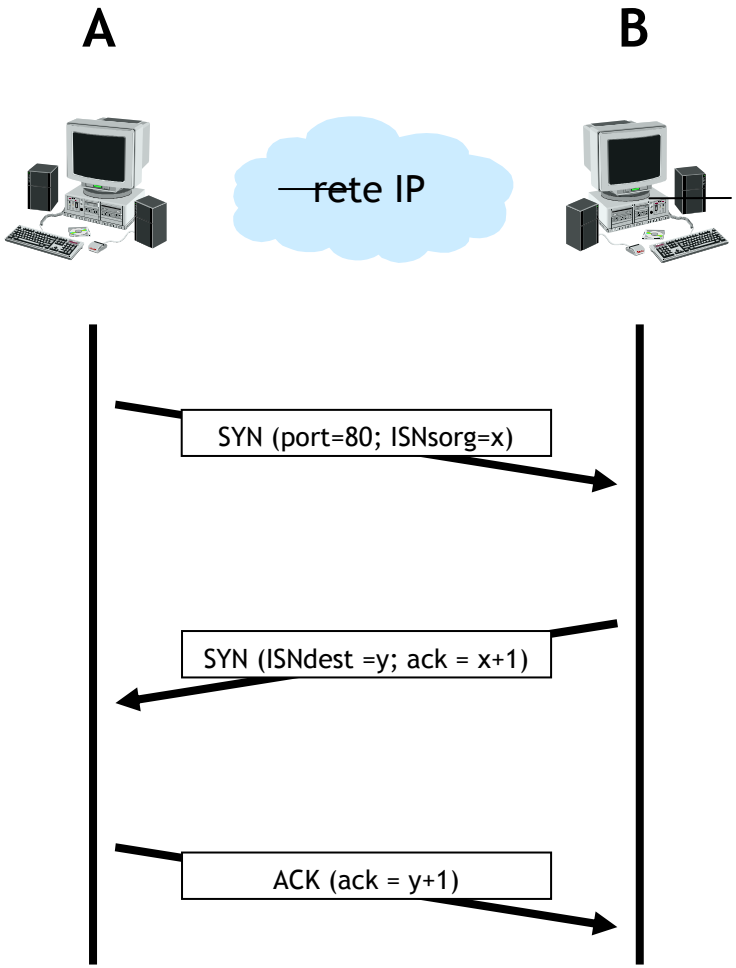
Instaurazione della connessione

- I segmenti SYN, ACK, dati, etc, sono segmenti TCP in cui i campi dell'header assumono valori particolari
  - esempi:
    - un segmento SYN è un segmento vuoto (è presente solo l'header) in cui il bit SYN è posto a 1
    - un segmento ACK è un segmento vuoto (è presente solo l'header) in cui il bit ACK è posto a 1
    - un segmento SYN ACK è un segmento vuoto (è presente solo l'header) in cui i bit SYN e ACK sono posti a 1
    - un segmento dati è un segmento contenente i dati dell'applicazione in cui i bit SYN e ACK sono posti a 0



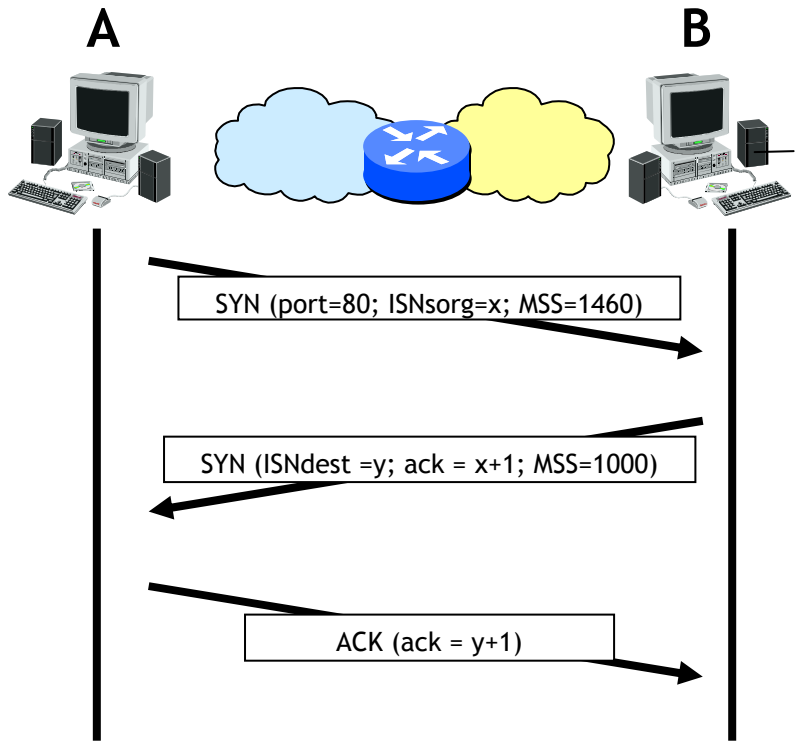
Instaurazione della connessione

- Il TCP è un protocollo ***connection oriented***
  - prima di iniziare a trasferire i dati ci deve essere una connessione tra i due end-system
- L'instaurazione della connessione avviene secondo la procedura detta di "three-way handshake"
  - la stazione che richiede la connessione (A) invia un segmento di SYN
    - parametri specificati: numero di porta dell'applicazione cui si intende accedere e Initial Sequence Number (ISNA)
  - la stazione che riceve la richiesta (B) risponde con un segmento SYN
    - parametri specificati: ISNB e riscontro (ACK) ISNA
  - la stazione A riscontra il segmento SYN della stazione B (invia un ACK alla stazione B)



Maximum Segment Size (MSS)

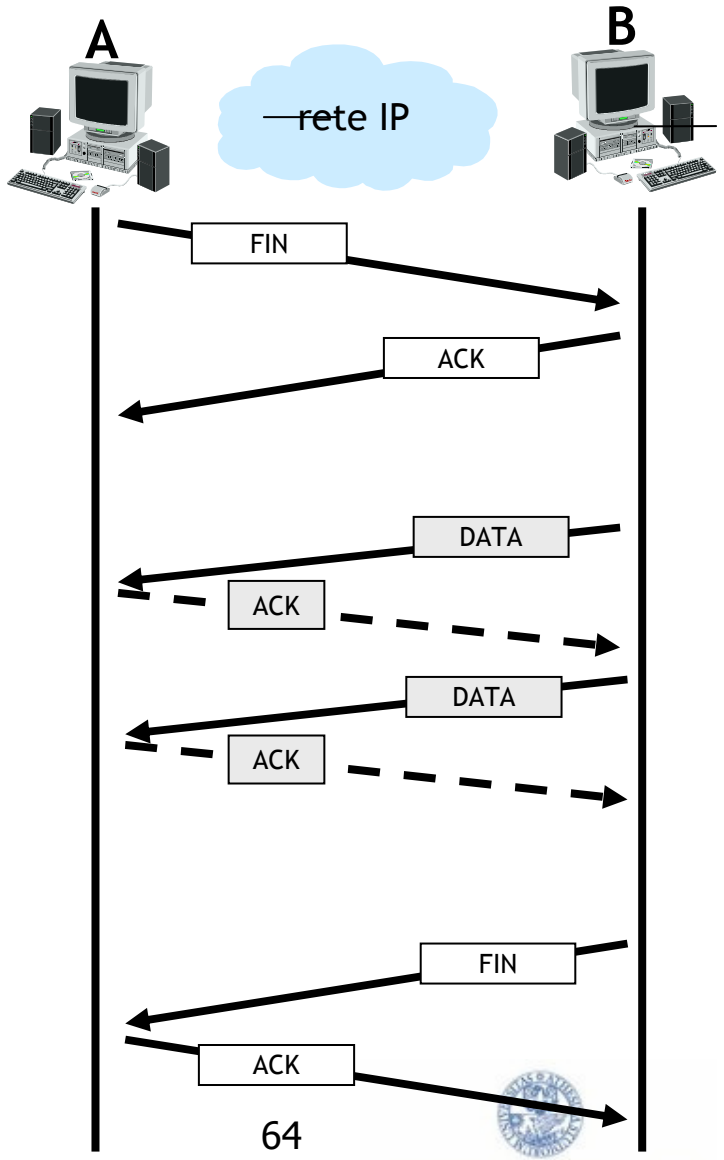
- Tra i vari parametri scambiati all'inizio di una connessione vi può essere anche la MSS
- La MSS consente ad una stazione di specificare la dimensione massima (espressa in numero di byte) del campo dati dei segmenti che è disponibile a ricevere
  - esistono meccanismi per far scoprire alla stazione la MSS in base alle caratteristiche della rete adiacente
  - se non viene specificato nessun valore, si considera il valore di default pari a 536 byte

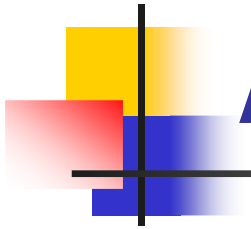


A annuncia a B di poter accettare solo segmenti la cui dimensione massima del campo dati è pari a 1460 byte;
B annuncia ad A che la sua MSS è invece di 1000 byte; le due stazioni dunque trasmetteranno segmenti con un campo dati lungo al massimo 1000 byte

Terminazione di una connessione

- Poiché la connessione è bidirezionale, la terminazione deve avvenire in entrambe le direzioni
- Procedura di terminazione
  - la stazione che non ha più dati da trasmettere e decide di chiudere la connessione invia un segmento FIN (segmento con il campo FIN posto a 1 e il campo dati vuoto)
  - la stazione che riceve il segmento FIN invia un ACK e indica all'applicazione che la comunicazione è stata chiusa nella direzione entrante
- Se questa procedura avviene solo in una direzione (half close), nell'altra il trasferimento dati può continuare (gli ACK non sono considerati come traffico originato, ma come risposta al traffico)
  - per chiudere completamente la connessione, la procedura di half close deve avvenire anche nell'altra direzione

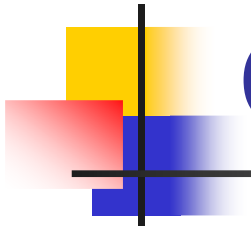




Affidabilità

- TCP è un protocollo connection oriented affidabile, ovvero garantisce la corretta e ordinata consegna dei segmenti
- TCP deve essere in grado di gestire
  - le situazioni di errore
    - individuate con l'uso del campo "checksum"
  - la perdita dei segmenti (ed eventualmente dei riscontri)
    - individuate grazie al riscontro di ciascun segmento; in caso di perdita del segmento non vi sarà il riscontro
- Se si verificano le suddette situazioni, TCP ritrasmette i segmenti interessati
- Tale tecnica viene detta "positive acknowledgement with retransmission"
- Nella versione base della tecnica (→stop and wait), la sorgente non trasmette un nuovo segmento fino a quando l'ultimo non viene riscontrato

- **Problema: quanto tempo devo aspettare prima di poter considerare il segmento perso e ritrasmetterlo?**



Gestione del timeout

- Il timeout (RTO → Retransimission Time Out) indica il tempo entro il quale la sorgente si aspetta di ricevere il riscontro (ack)
  - nel caso in cui il riscontro non arrivi, la sorgente procede alla ritrasmissione
- RTO non può essere un valore statico predefinito
  - il tempo di percorrenza sperimentato dai segmenti è variabile e dipende
    - dalla distanza tra sorgente e destinazione
    - dalle condizioni della rete
    - dalla disponibilità della destinazione
  - esempio: fattorino che deve consegnare un pacco in città
- RTO deve dunque essere calcolato dinamicamente di volta in volta
  - durante la fase di instaurazione della connessione
  - durante la trasmissione dei dati
- Il calcolo di RTO si basa sulla misura del RTT (Round Trip Time)
  - RTT: intervallo di tempo tra l'invio di un segmento e la ricezione del riscontro di quel segmento

Stima del Round Trip Time

SRTT = (1- α) SRTT + α RTT

RTT = valore del campione attuale di RTT
SRTT = stima smoothed di RTT

- Poiché RTT può variare anche molto in base alle condizioni della rete, il valore di RTT (SRTT, Smoothed RTT) utilizzato per il calcolo di RTO risulta una stima del valor medio di RTT sperimentato dai diversi segmenti
- Il parametro α è regolabile e, a seconda dei valori assunti, rende il peso della misura di RTT istantaneo più o meno incisivo
  - se α → 1 il SRTT stimato risulta abbastanza stabile e non viene influenzato da singoli segmenti che sperimentano RTT molto diversi
  - se α → 0 il SRTT stimato dipende fortemente dalla misura puntuale dei singoli RTT istantanei
  - tipicamente α = 0.125 = (1/8)

Stima del Round Trip Time

RTTVAR = (1- b) RTTVAR + b |SRTT – RTT|

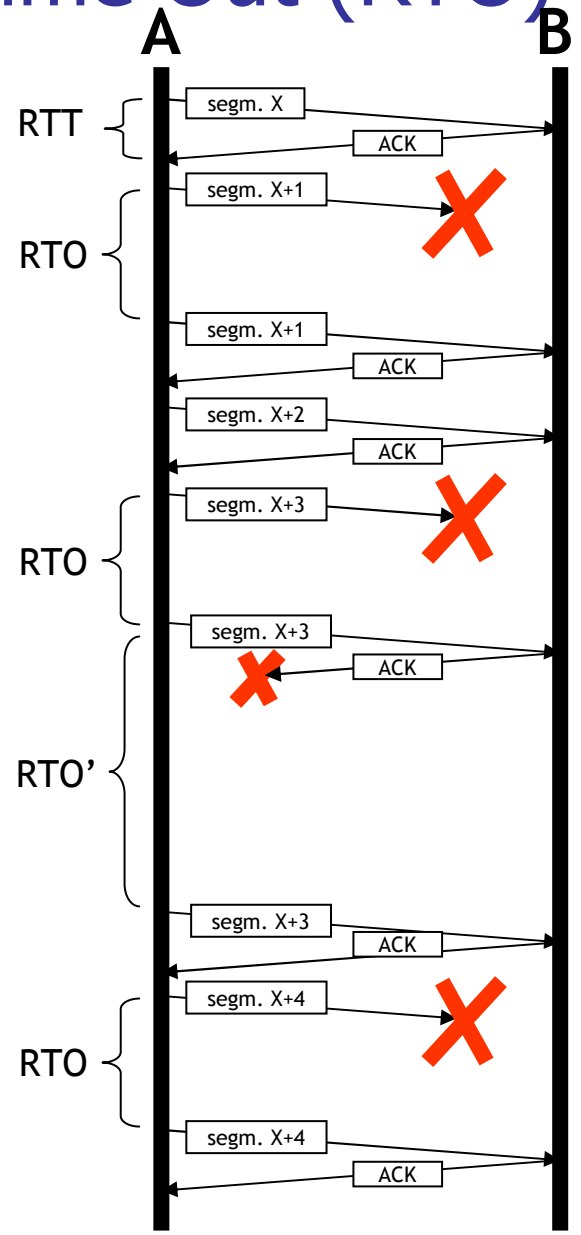
RTTVAR = stima smoothed della varianza di RTT

- Tipicamente b = 0.25 = (1/4)
- La RTTVAR può anche essere molto maggiore di SRTT
- La sua stima usa un filtro meno stretto, per cui la stima è più "reattiva"
- Conoscere media e varianza di RTT serve a impostare correttamente il timeout di ritrasmissione

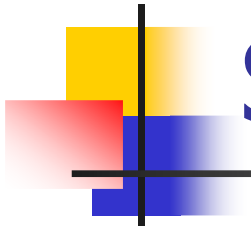
Stima del Retransmission Time Out (RTO)

- **RTO = SRTT + 4 RTTVAR**

- La sorgente attende il RTT medio (SRTT) più quattro volte la sua varianza (RTTVAR) prima di considerare il segmento perso e ritrasmetterlo
- In caso di ritrasmissione, il RTO per quel segmento viene ricalcolato in base ad un processo di exponential backoff
  - se è scaduto il RTO, probabilmente c'è congestione, quindi meglio aumentare il RTO per quel segmento
  - RTO-retransmission = 2*RTO
- RTO viene riportato al suo valore "calcolato" senza backoff dopo la trasmissione corretta di un segmento nuovo



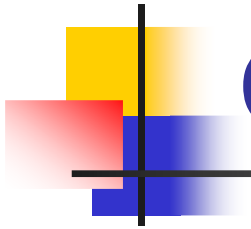
locigno@disi.unitn.it



Stima di RTT e RTO

- RTO ha un valore minimo (normalmente intorno a 200ms) e uno massimo (normalmente 60s e oltre)
- Alcune implementazioni di TCP utilizzano algoritmi più evoluti per il calcolo di RTT e RTO, ma i principi base di adattamento alle condizioni della rete rimane lo stesso:
  - viene stimato un valore medio e una deviazione media di RTT
  - RTO viene calcolato in base a questi valori medi
- Nelle implementazioni, inoltre, si tiene conto di altri fattori che possono influenzare il calcolo di RTT e di RTO:
  - esempio: il RTT dei segmenti ritrasmessi dovrebbe influenzare il SRTT e quindi il RTO?

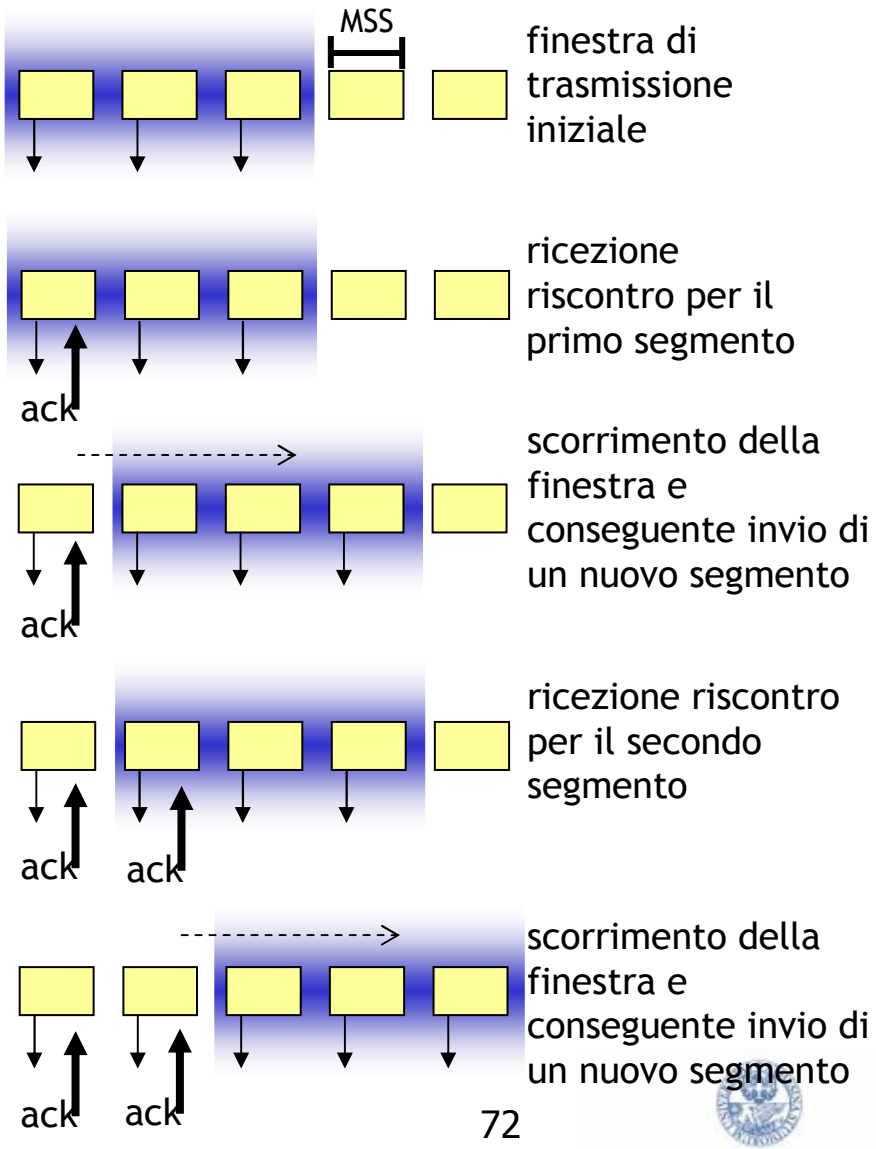
Controllo di flusso: generalità



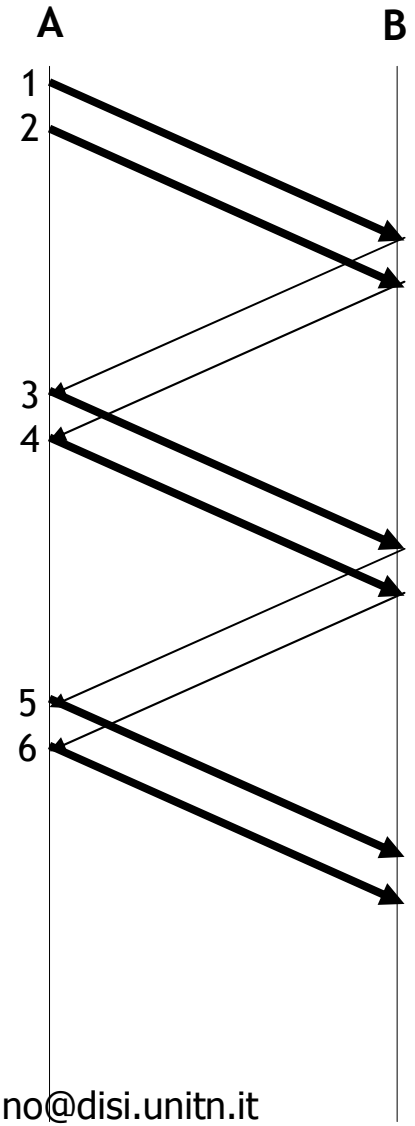
- Richiamiamo qui la differenza tra *controllo di flusso* e *controllo di congestione*
  - Controllo di flusso: ***azione preventiva*** finalizzata a limitare l'immissione di dati in rete in caso il ricevitore non li possa ricevere
  - Controllo della congestione: azioni da intraprendere come ***reazione*** alla congestione di rete (e non del ricevitore)
- Problemi associati al controllo di flusso:
  - com'è possibile da parte di una stazione determinare la capacità e lo stato del ricevitore?
- Con la tecnica di *positive acknoledgement with retransmission* abbiamo visto che una stazione invia un nuovo segmento solo se l'ultimo è stato riscontrato (stop and wait)
- Questa tecnica di per sé è già un controllo di flusso
  - il tasso di immissione di nuovi segmenti è determinato dalle condizioni della rete, poiché avviene in base alla ricezione dei riscontri
- Tuttavia tale tecnica risulta poco efficiente, in quanto viene sprecata banda nell'attesa dei singoli riscontri

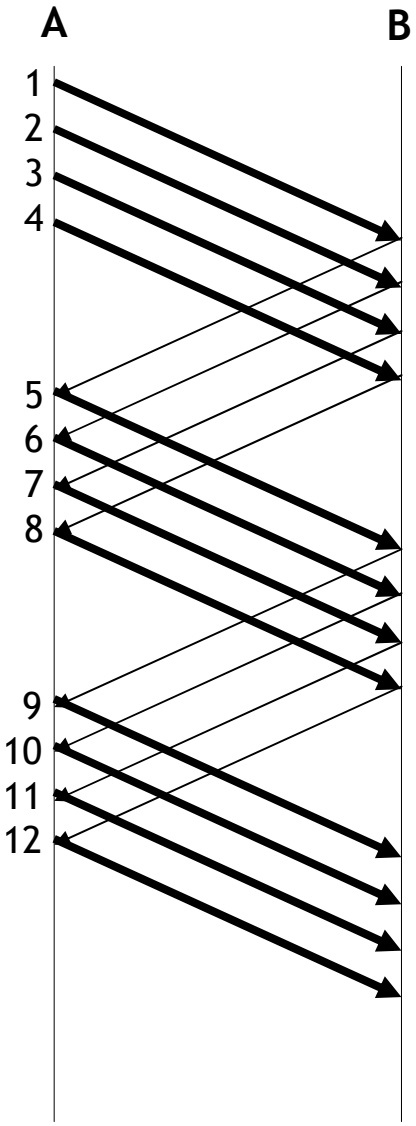
Controllo di flusso a finestra

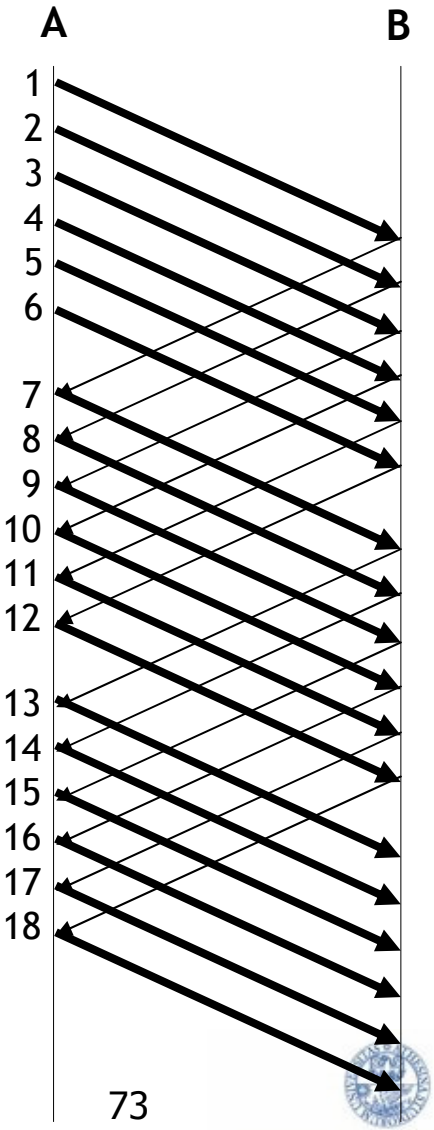
- Per incrementare l'efficienza è possibile trasmettere più segmenti consecutivamente senza attendere ogni singolo riscontro
  - considerando i segmenti in sequenza, l'insieme dei segmenti trasmessi rientrano in una "finestra di trasmissione"
    - la finestra di trasmissione è l'intervallo di segmenti trasmessi
- Alla ricezione dei riscontri dei segmenti iniziali della sequenza, la finestra scorre a destra permettendo al trasmissione di nuovi segmenti
- Tale tecnica è denominata "finestra scorrevole" (sliding window)

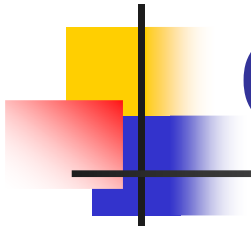


Influenza della finestra





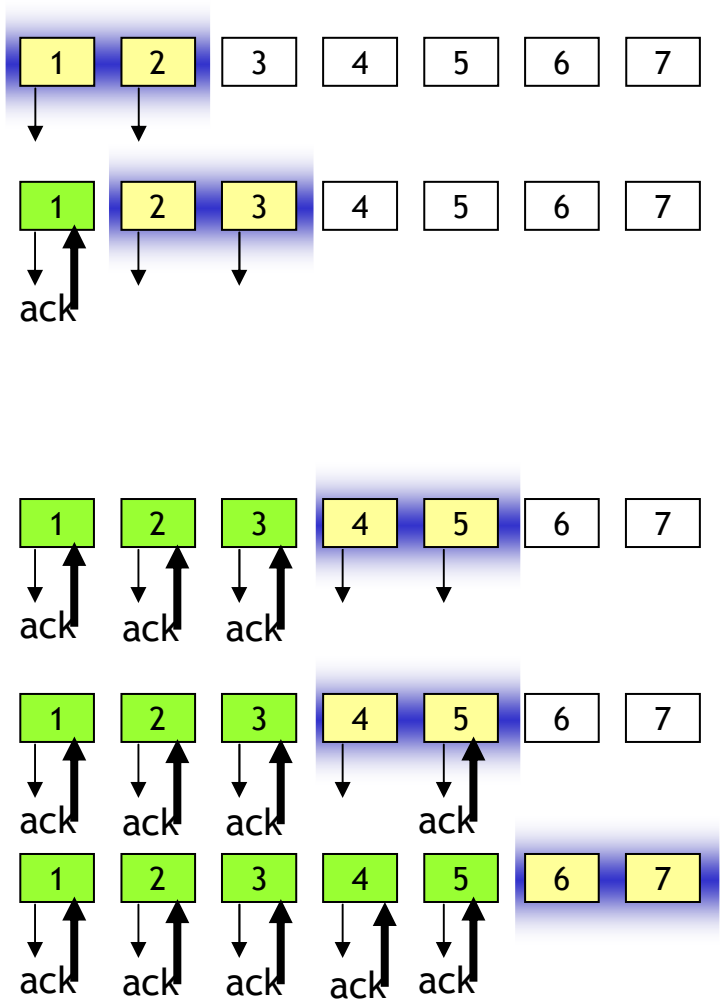


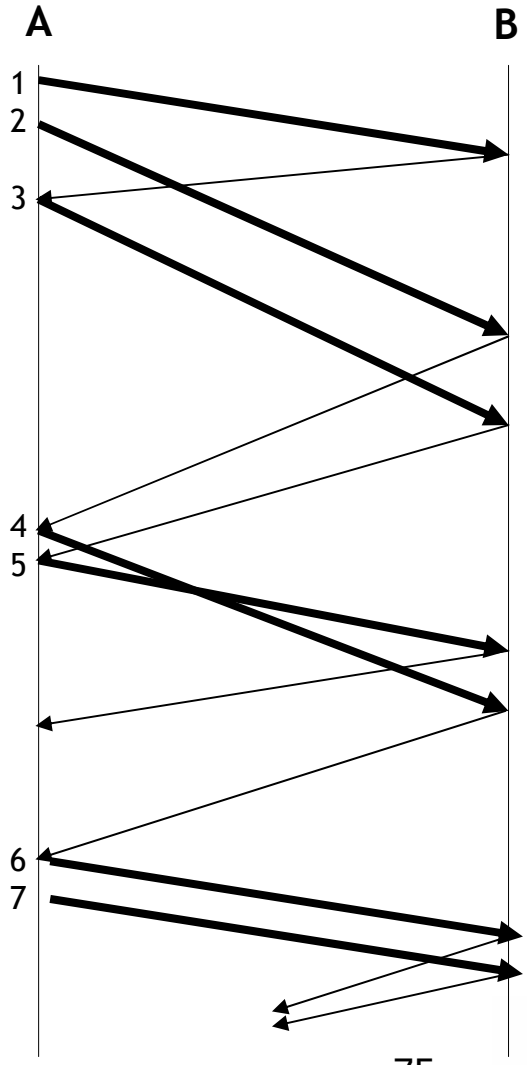


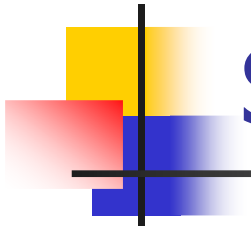
Considerazioni

- Se la dimensione della finestra è troppo piccola → sottoutilizzo la banda
- TCP usa il campo Receiver Window (RCWN) negli ACK per "dettare" al trasmettitore il numero di bytes che è possibile trasmettere con la certezza che il ricevitore li accetti
- RCWN viene impostata dinamicamente in funzione dei dati già consegnati all'applicazione
- Consente di implementare in modo semplice il buffer di riassemblaggio in caso di perdite o segment fuori ordine

Esempio di funzionamento

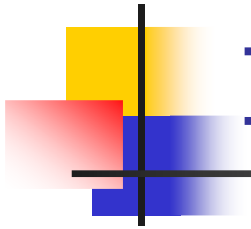






Sliding Window

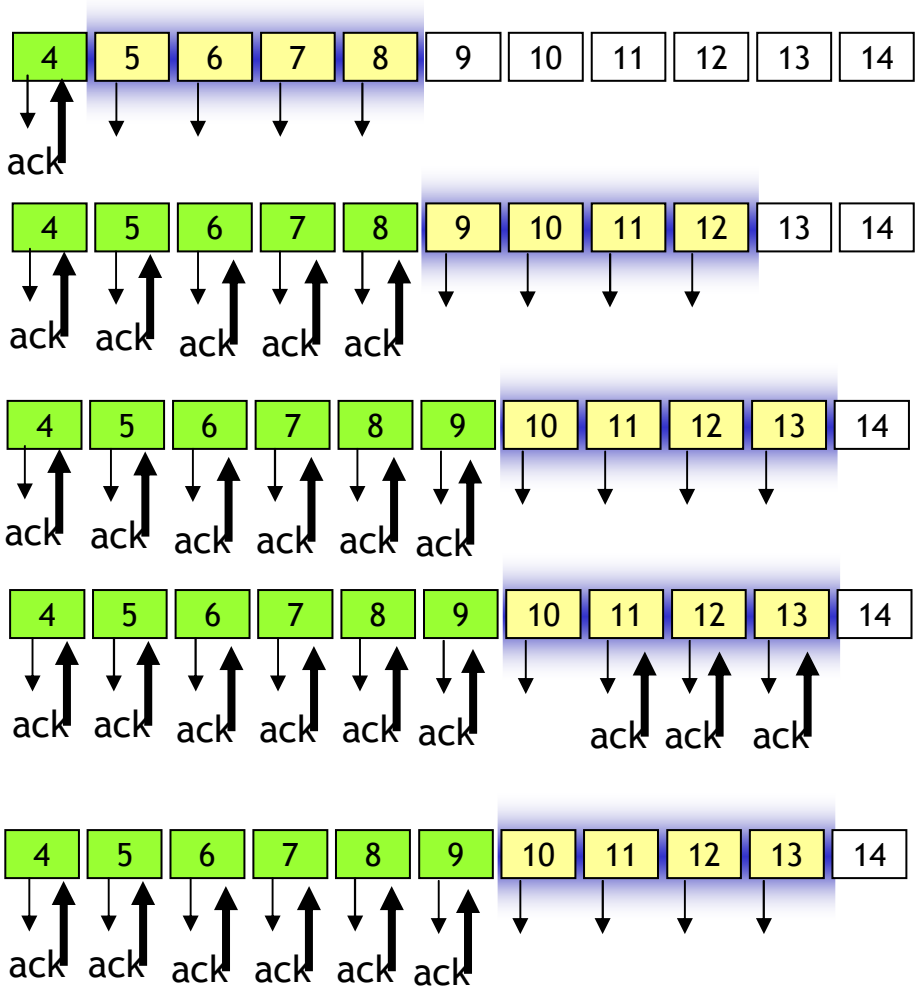
- I dati provenienti dall'applicazione vengono ordinati in una sequenza di byte numerati progressivamente (a partire dall'ISN, Initial Sequence Number)
- In base alla dimensione della MSS, tali byte sono suddivisi in segmenti
- Il meccanismo a finestra scorrevole è applicato alla sequenza di segmenti così determinata
- Problema: quanto deve essere grande la finestra?
  - la dimensione della finestra è fissata in base al valore "Window" (contenuto nell'header TCP) espresso dalle due stazioni durante la fase di connessione e durante lo scambio di dati
  - scegliere la finestra basandosi su tale valore significa fare controllo di flusso considerando solo le condizioni della destinazione, ma non le condizioni di rete...

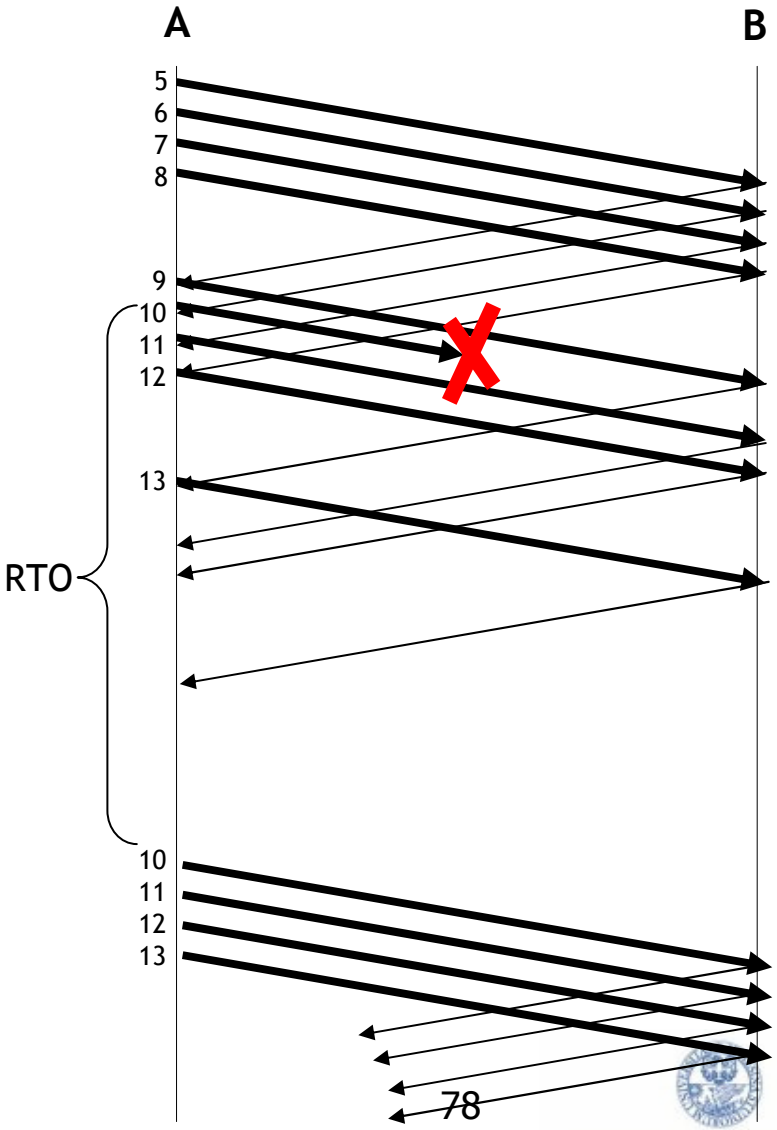


In caso di perdita dei segmenti...

- In caso di perdita di un segmento, dopo lo scadere di un timeout (RTO), il TCP provvede a ritrasmetterlo
  - il comportamento risulta intuitivo se la finestra è pari a 1 segmento
- Problema: come si comporta il TCP quando il segmento appartiene ad una finestra di trasmissione (più segmenti trasmessi contemporaneamente)?
- Possibili soluzioni
  - far tornare la finestra indietro e ritrasmettere tutti i segmenti (Go-back-N)
  - ritrasmettere solo il segmento perso (Selective Repeat)

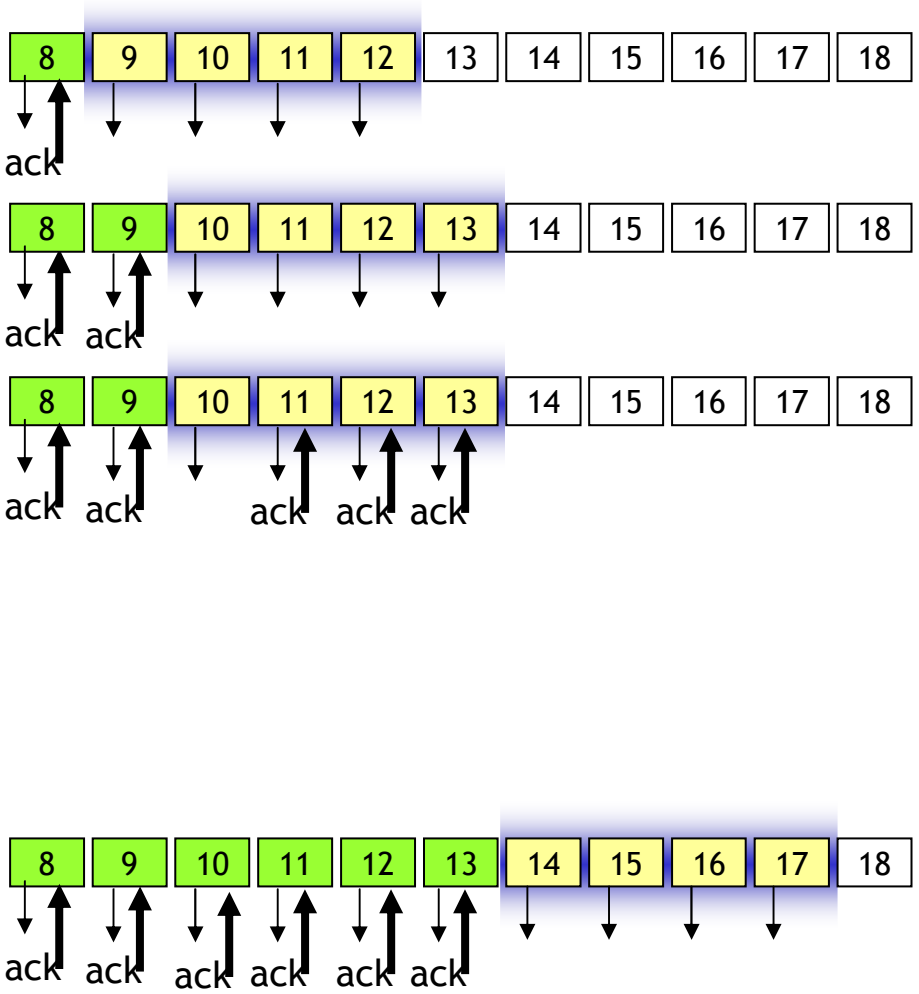
Go-back-N

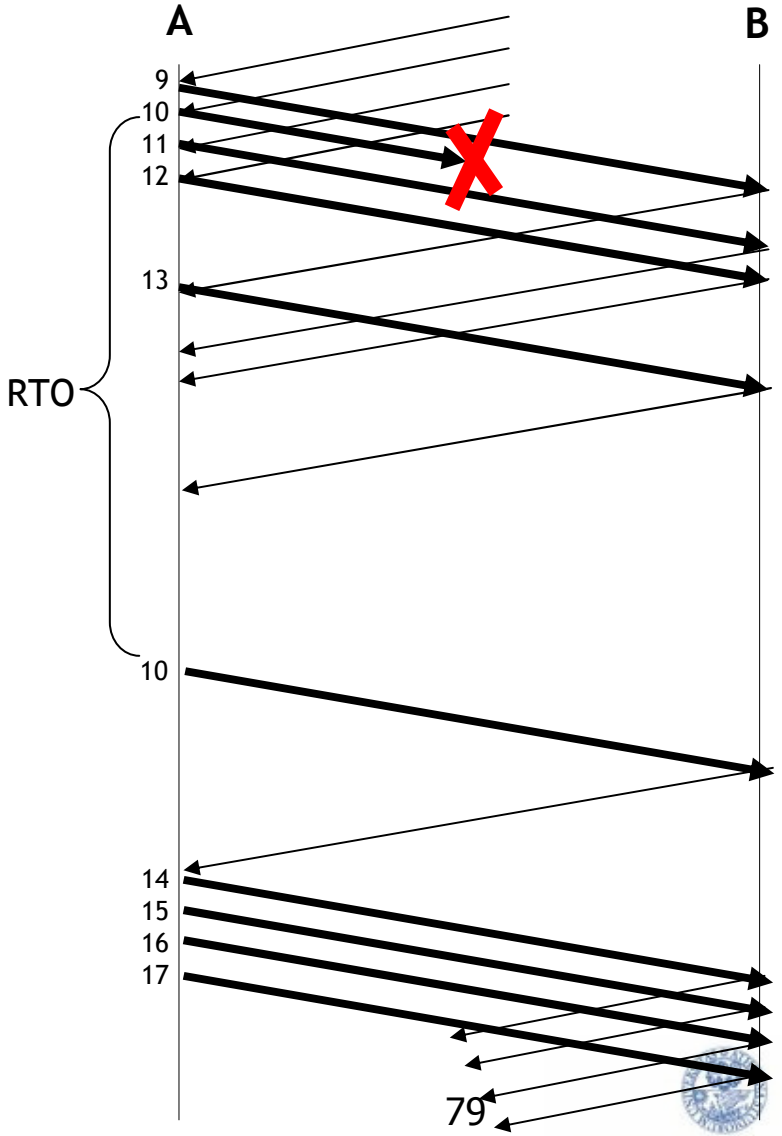




locigno@disi.unitn.it

Selective Repeat





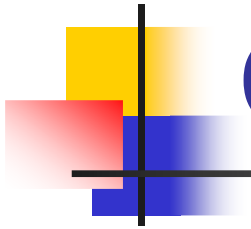
locigno@disi.unitn.it

Considerazioni

- Go-back-N
  - semplice da implementare
  - è stato pensato per un ambiente in cui ci sono perdite multiple contemporanee (burst di segmenti persi)
  - inefficiente in caso di perdite singole
- Selective Repeat
  - esistono diverse modalità di ritrasmissione selettiva
    - sia sorgente e destinazione devono implementare la stessa modalità
  - maggior complessità implementativa
  - buona efficienza in caso di perdite singole
  - non sovraccarica la rete con ritrasmissioni già riscontrate

- Le prime implementazioni di TCP utilizzavano la tecnica per il recupero degli errori Go-ack-N;
- Le attuali implementazioni di TCP utilizzano la tecnica Selective Repeat (con ACK cumulativi)

- Problema: la perdita dei segmenti potrebbe essere causata dalla congestione della rete? Se fosse così, non sarebbe opportuno ridurre il tasso di immissione dei nuovi segmenti?

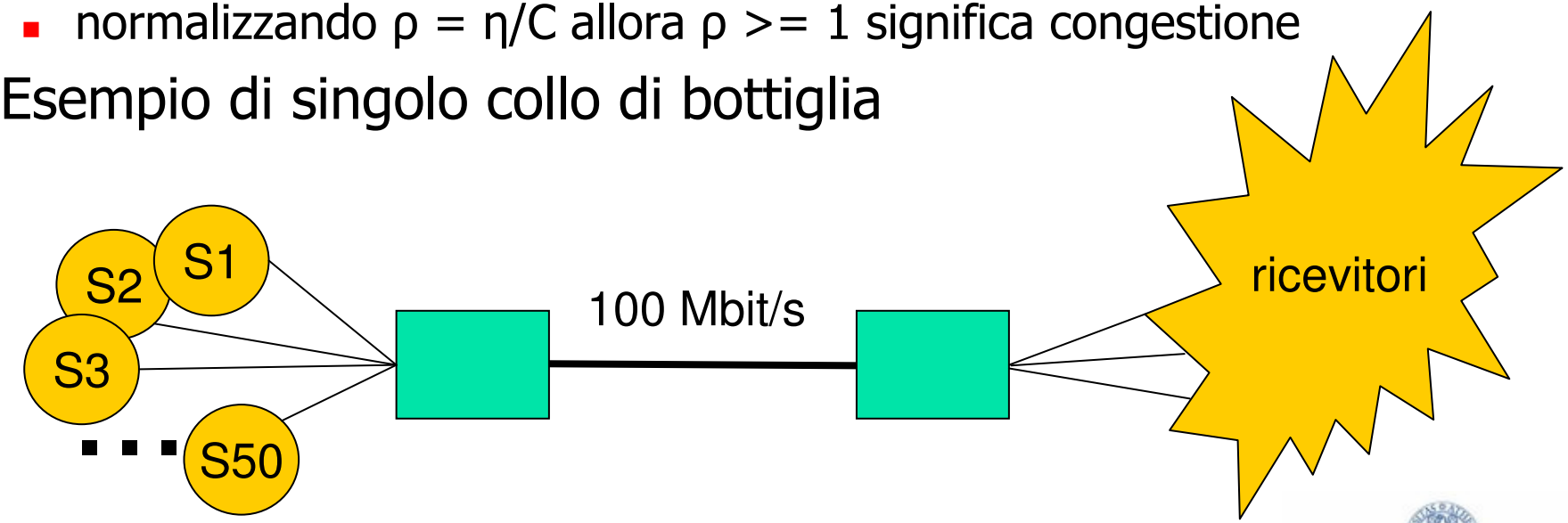


Congestione: generalità

- In caso di congestione della rete, a causa dei buffer limitati degli apparati di rete, alcuni segmenti potrebbero venire persi
- La perdita dei segmenti e il relativo scadere del timeout di ritrasmissione è considerato un sintomo di congestione
  - il TCP non ha altri mezzi per conoscere lo stato della rete
- La sorgente dovrebbe essere in grado di reagire diminuendo il tasso di immissione dei nuovi segmenti
- Questa reazione viene detta "controllo della congestione"
  - si differenzia dal controllo di flusso che invece definisce tecniche per evitare il sovraccarico del ricevitore

Congestione: cause

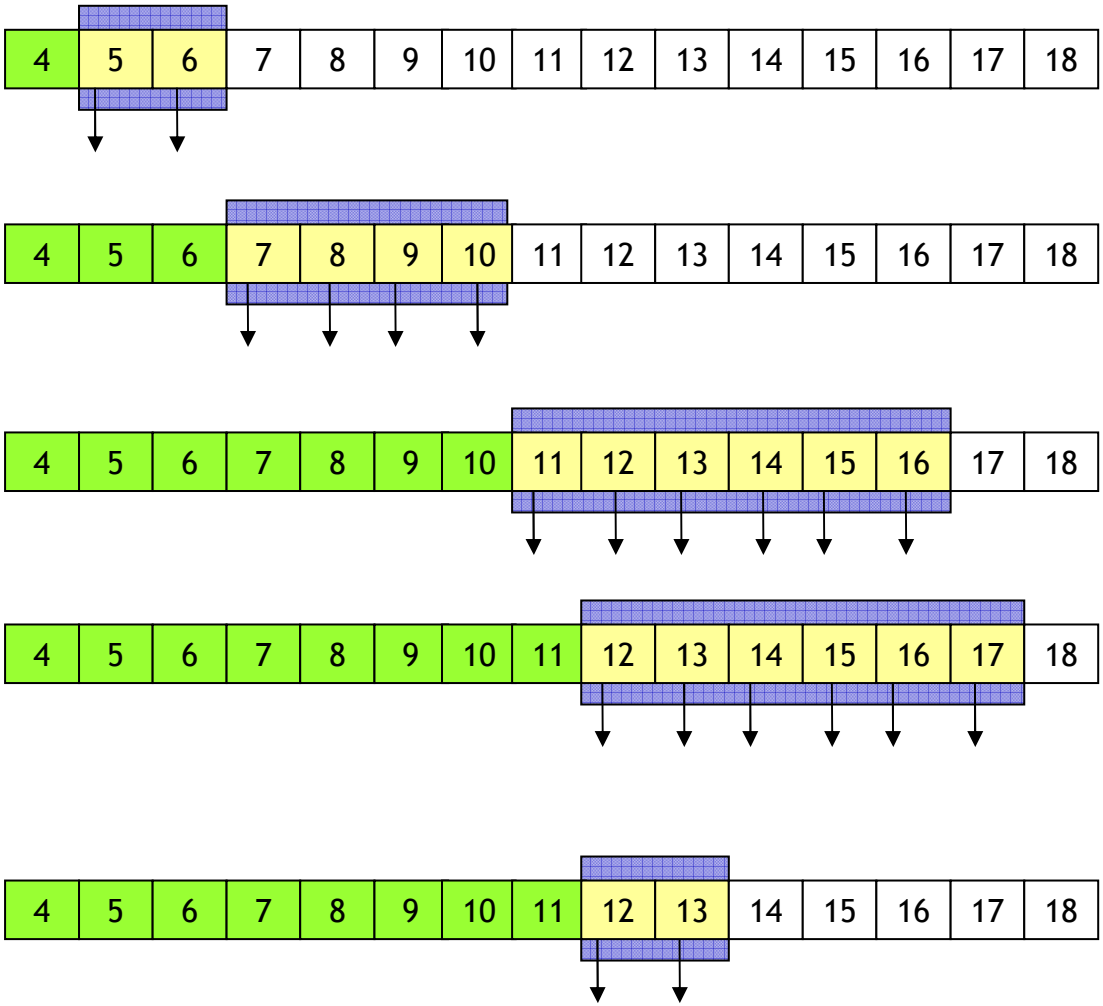
- Le ritrasmissioni rischiano di essere inutili nel caso in cui i pacchetti vengono buttati via per mancanza di risorse
- Cos'è la congestione e come si genera?
- **Congestione**: lo stato di una rete in cui il traffico offerto η è maggiore della capacità delal rete C
  - normalizzando ρ = η/C allora ρ >= 1 significa congestione
- Esempio di singolo collo di bottiglia

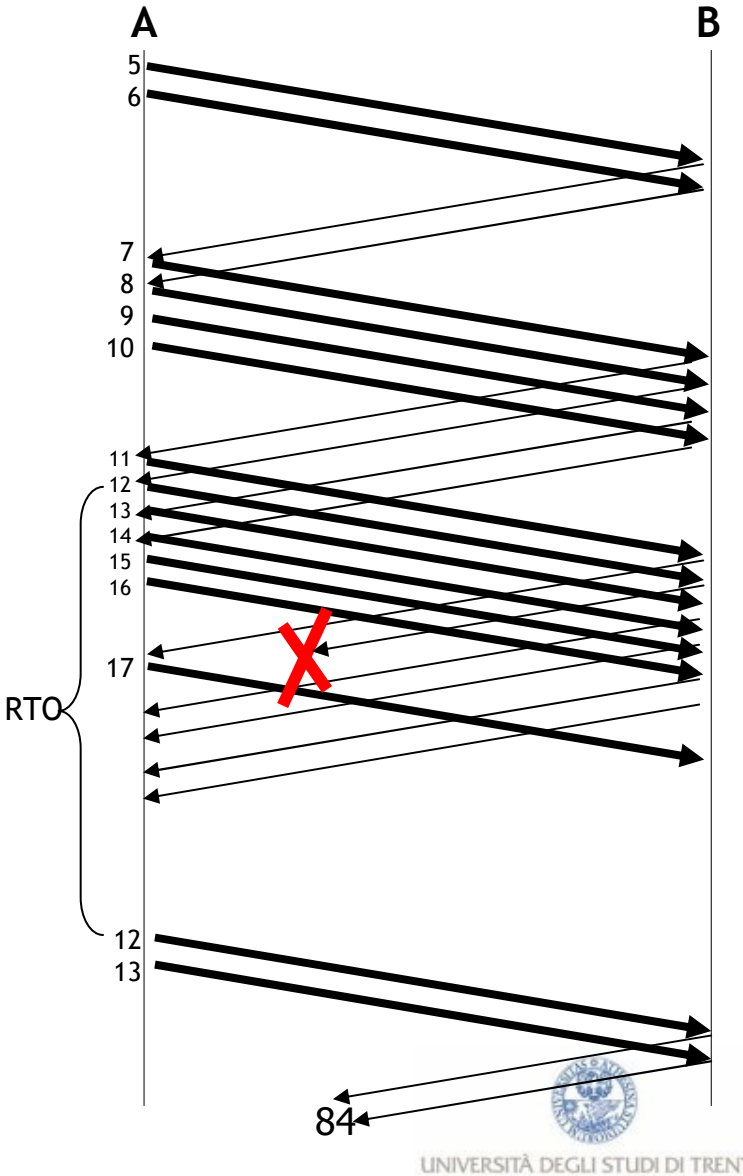


Congestion Window

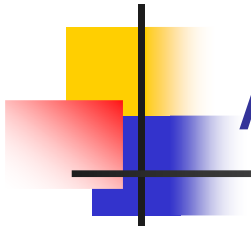
- In caso di congestione, il controllo di flusso a finestra, protegge implicitamente anche la rete:
  - se la rete è congestionata, arriveranno meno riscontri e quindi il tasso di immissione diminuisce automaticamente
  - l'attesa dello scadere del timeout lascia un intervallo di tempo in cui non vengono immessi nuovi segmenti
- Tuttavia queste misure non potrebbero essere sufficienti
  - rimane da determinare la dimensione della finestra ottimale
- Soluzione per il controllo della congestione → variare dinamicamente la dimensione della finestra di trasmissione
  - la dimensione della finestra non è dunque decisa a priori e statica, ma si adatta alle situazioni di sovraccarico e reagisce alle congestione
  - tale finestra scorrevole e variabile viene denominata "congestion window" (CWND)

Esempio



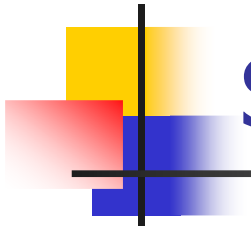


locigno@disi.unitn.it



Algoritmi per il controllo della congestione

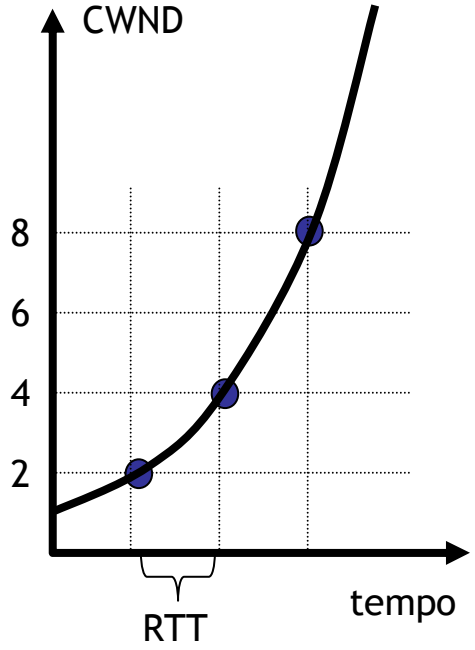
- Esistono diversi algoritmi che regolano la dimensione della finestra (CWND) al variare delle condizioni di rete
- I due algoritmi base sono:
  - Slow Start
  - Congestion Avoidance
- Altri algoritmi sono stati definiti per aumentare l'efficienza del TCP
  - Fast Retransmit
  - Fast Recovery
  - SACK

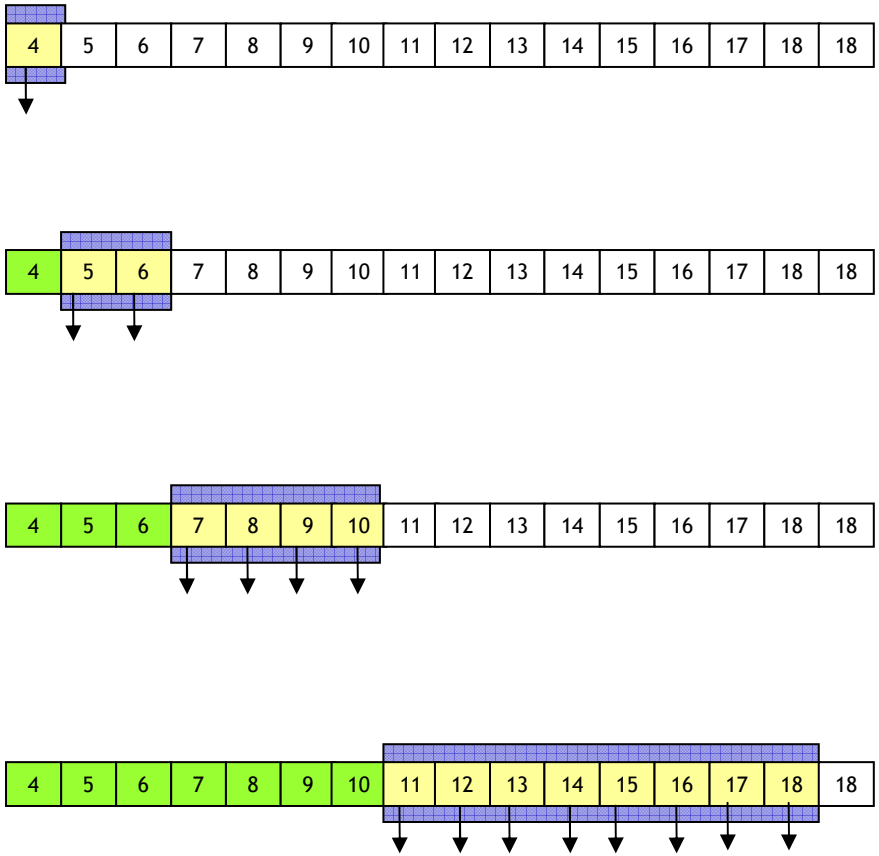


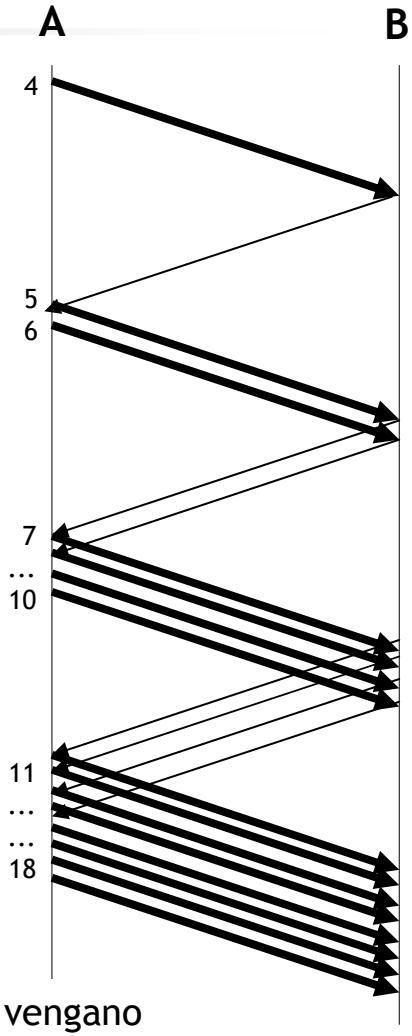
Slow Start e Congestion Avoidance

- Sono due diversi algoritmi che regolano la dimensione della finestra (l'utilizzo di uno esclude l'utilizzo dell'altro)
- Slow Start
  - per ciascun riscontro ricevuto la CWND può aumentare di un segmento
  - questo implica che, quando si riceve un riscontro, si trasmettono due nuovi segmenti (e non uno solo come nel caso in cui la CWND rimane fissa)
  - l'evoluzione della CWND ha un andamento esponenziale
    - al primo RTT la CWND=1, ricevuto il riscontro CWND = 2, ricevuti i due riscontri CWND = 4, ...
- Congestion avoidance
  - per ciascun riscontro ricevuto, la finestra aumento di 1/CWND (CWND è espressa in numero di segmenti)
  - questo implica che ad ogni RTT, in cui si ricevono un numero di riscontri pari alla CWND, la CWND aumenta di un segmento
  - l'evoluzione della CWND ha un andamento lineare

Esempio: Slow Start

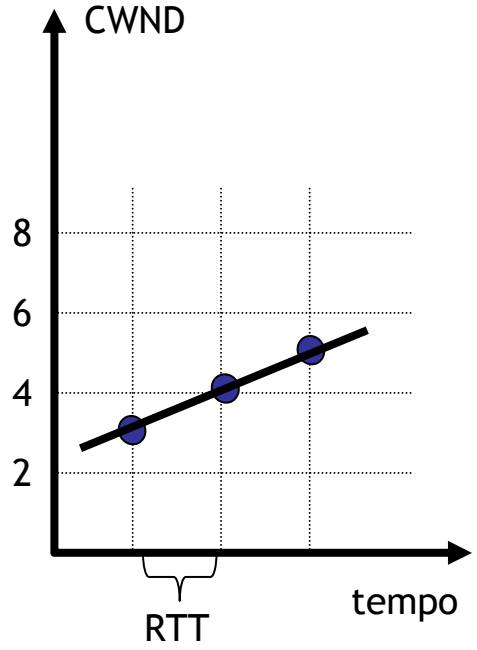


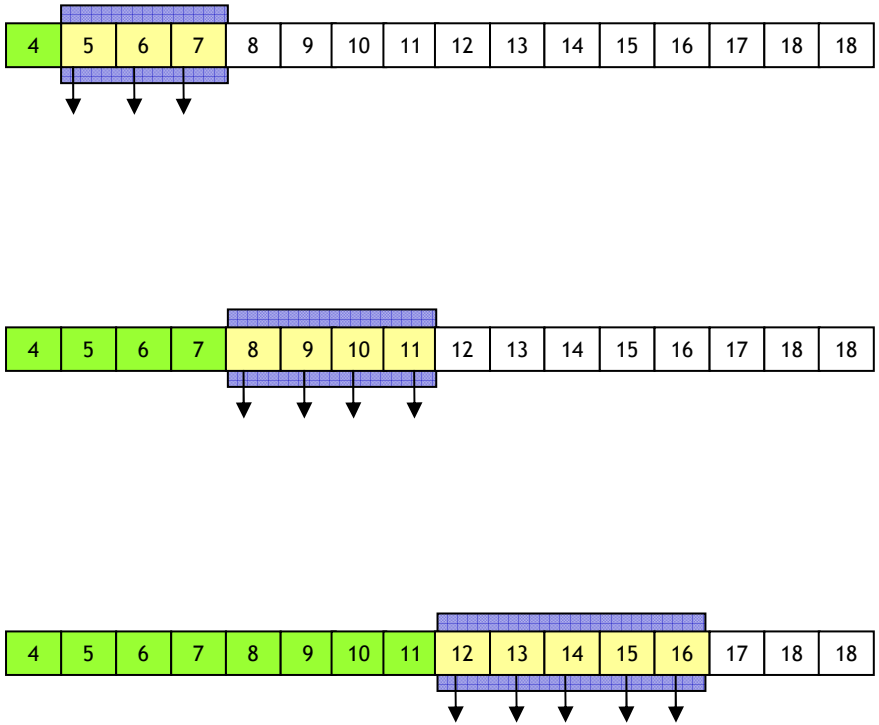


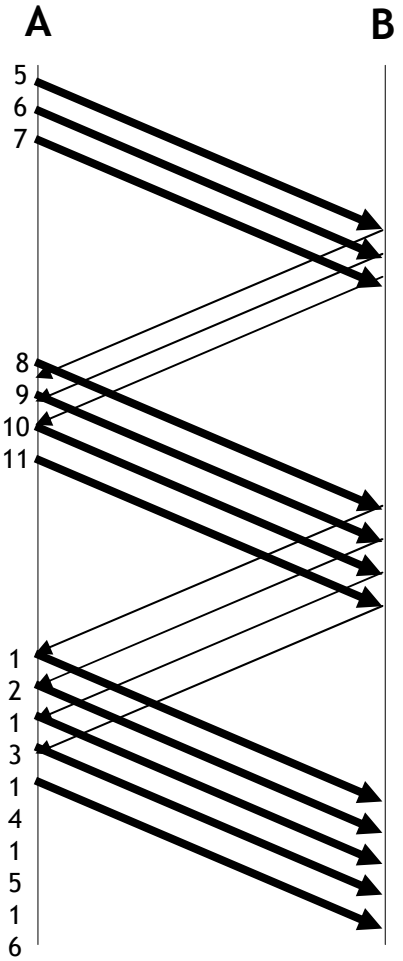


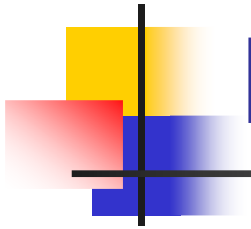
NOTA: per semplificare la rappresentazione grafica, si assume che i segmenti vengano generati e trasmessi tutti nello stesso istante e i corrispettivi riscontri vengano ricevuti di conseguenza tutti insieme dopo un tempo pari a RTT (supposto costante)

Esempio: Congestion Avoidance



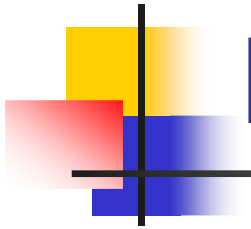






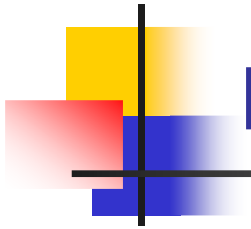
Evoluzione della CWND

- Variabili considerate:
  - Congestion Window (CWND)
    - è la dimensione della finestra (espressa in byte o in numero di segmenti) di trasmissione
  - Receive Window (RECWND)
    - è la dimensione della finestra di ricezione (espressa in byte o in numero di segmenti) annunciata dalla destinazione; è il limite massimo che la CWND può assumere
  - Slow Start Threshold (SSTHRESH)
    - è una dimensione della finestra (espressa in byte o in numero di segmenti) raggiunta la quale, invece di seguire l'algoritmo di Slow Start, si segue l'algoritmo di Congestion Avoidance
  - RTT
    - è il tempo trascorso tra l'invio di un segmento e la ricezione del riscontro; in condizioni di stabilità della rete e del carico, RTT rimane pressoché costante
  - RTO
    - è il tempo che la sorgente aspetta prima di ritrasmettere un segmento non riscontrato



Evoluzione della CWND

- L'algoritmo che regola la dimensione della CWND è il seguente:
  - all'inizio della trasmissione si pone
    - CWND = 1 segmento (ovvero un numero di byte pari a MSS)
    - SSTHRESH = RCVWND oppure SSTHRESH = RCVWND / 2 (dipende dalle implementazioni)
  - la CWND evolve secondo l'algoritmo di Slow Start fino al raggiungimento della SSTHRESH
  - raggiunta la soglia SSTHRESH, la dimensione di CWND è regolata dall'algoritmo di Congestion Avoidance
  - la finestra cresce fino al raggiungimento di RCVWND

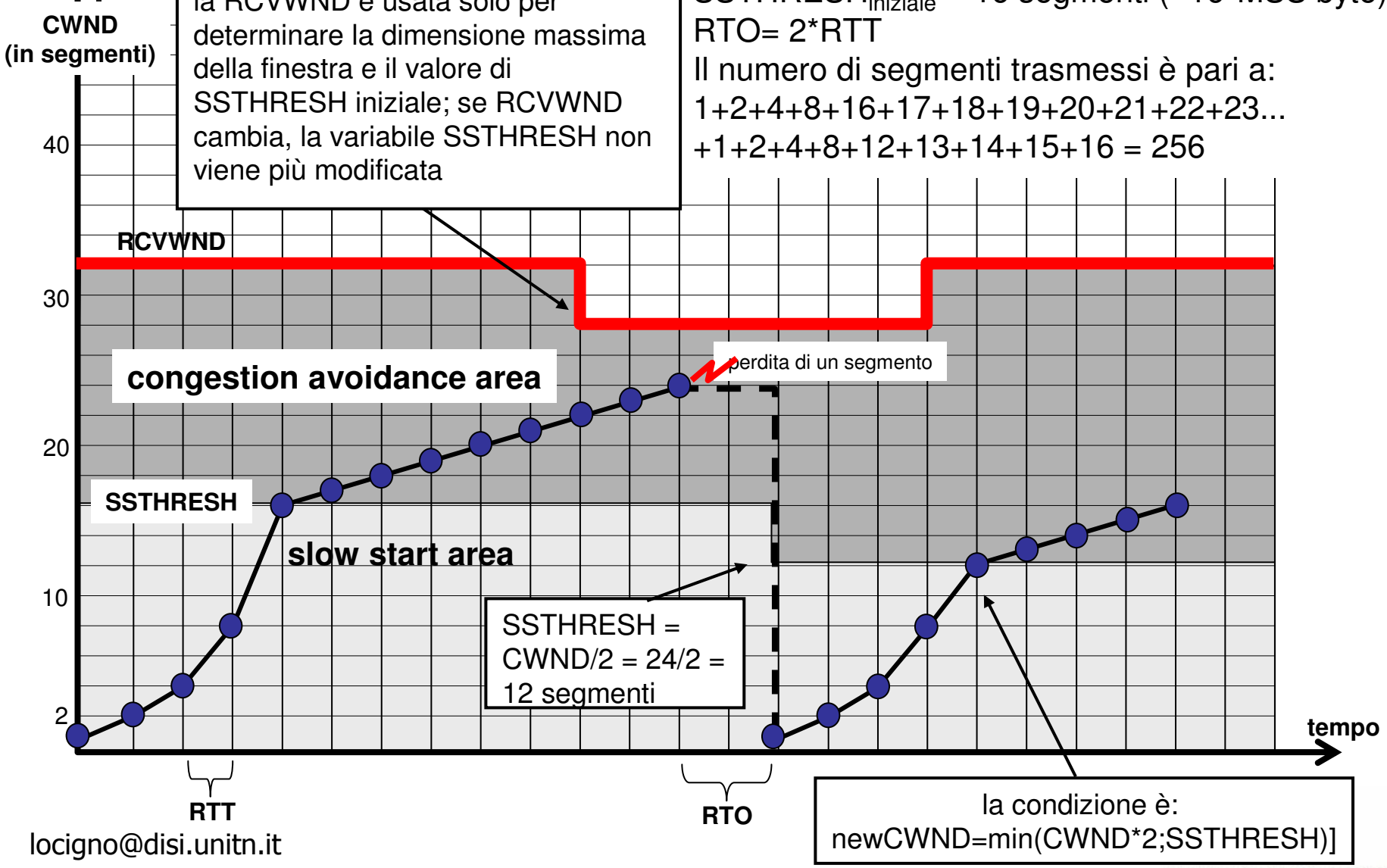


Evoluzione della CWND: errori o perdite

- In caso di errore o di perdita dei segmenti:
  - la trasmissione si interrompe (la finestra non si sposta non permettendo l'immissione di nuovi segmenti in rete)
  - si attende lo scadere del timeout RTO
  - allo scadere di RTO, si pone
    - SSTHRESH = CWND / 2
    - CWND = 1
  - si riprende a ritrasmettere con la tecnica di Go-back-N
  - l'evoluzione della finestra segue l'algoritmo di Slow Start fino al raggiungimento della SSTHRESH...
- In caso di errore o perdita consecutiva
  - al primo errore o perdita, quando il timeout scade e si ritrasmette il primo segmento non riscontrato, il timeout per quel segmento viene raddoppiato (exponential back off)
    - RTO_new=2*RTO_old
  - la CWND rimane = 1, mentre SSTHRESH si pone = 2 segmenti

Esempio

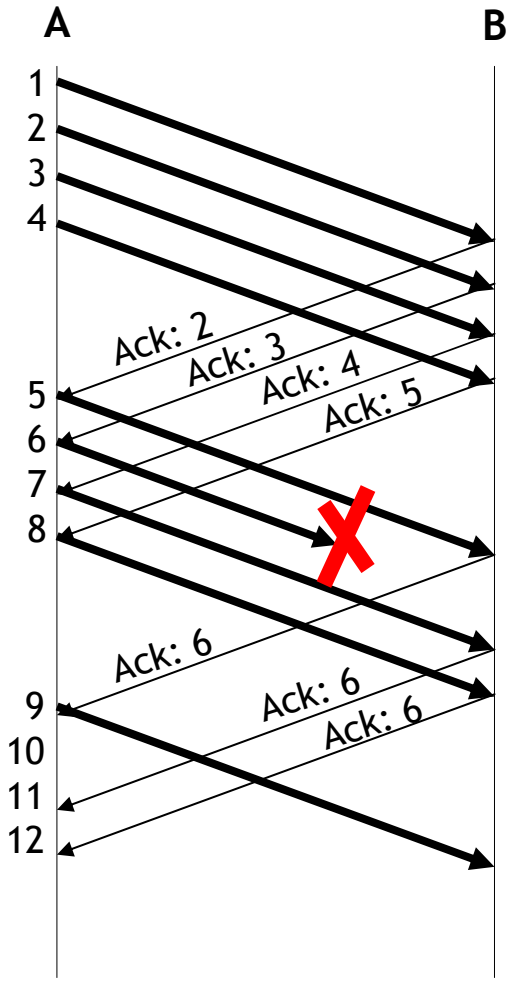
Condizioni iniziali:
RCVWND = 32 segmenti (ovvero 32*MSS byte)
$SSTHRESH_{iniziale}$ = 16 segmenti (=16*MSS byte)
RTO= 2*RTT
Il numero di segmenti trasmessi è pari a:
1+2+4+8+16+17+18+19+20+21+22+23...
+1+2+4+8+12+13+14+15+16 = 256

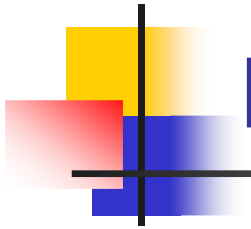


locigno@disi.unitn.it

ACK duplicati

- Negli ACK il campo ***Ack Number*** contiene il successivo numero di sequenza che ci si aspetta arrivi
  - questo dice implicitamente che tutti i segmenti precedenti sono stati ricevuti correttamente
- Se un segmento arriva fuori sequenza, la destinazione invia un ACK indicando il numero di sequenza del segmento che non è ancora arrivato
  - la ricezione di un numero sufficientemente alto di ACK duplicati può essere interpretata come forte di indicazione che è avvenuta una perdita





Fast Retransmit – Fast Recovery

- Esistono principalmente due motivi che causano la perdita di segmenti:
  - errori di trasmissione
    - influisce generalmente su un singolo segmento
  - congestione
    - provoca la perdita di più segmenti consecutivi
- Problema:
  - quando viene perso un solo segmento, si ha una reazione "eccessiva" da parte della sorgente
    - attesa dello scadere del timeout e chiusura della CWND
- Soluzione:
  - Fast Retransmit e Fast Recovery: due algoritmi (implementati sempre in coppia) specificatamente progettati per gestire le perdite singole
    - il segmento considerato perso viene subito ritrasmesso (fast retramsmit)
    - la CWND non viene chiusa eccessivamente (fast recovery)

Fast Retransmit – Fast Recovery: algoritmo

- Se alla sorgente arrivano 3 ACK duplicati →
  - si pone $SSTHRESH_{nuova}$ = CWND / 2
  - si ritrasmette il segmento senza attendere lo scadere del RTO
    - Fast Retransmit
  - si pone CWND = SSTHRESH + 3
    - Fast Recovery
  - per ogni successivo ACK duplicato la CWND aumenta di 1 segmento
    - permette la trasmissione di nuovi dati
- Quando la sorgente riceve l'ACK che conferma la ricezione del segmento ritrasmesso
  - si pone CWND = SSTHRESH
  - si procede la trasmissione con il Congestion Avoidance
- Casi paticolari:
  - se il segmento ritrasmesso viene perso, si attende lo scadere del RTO, la CWND torna a 1 e si riparte in Slow Start
  - se viene perso piu' di un segmento: l'algoritmo cerca di recuperare il primo, anche se ce la fa, arrivano gli ACK duplicati dei successivi segmenti persi che l'algoritmo non sa trattare, per cui scade il RTO

Fast Retransmit – Fast Recovery: esempio