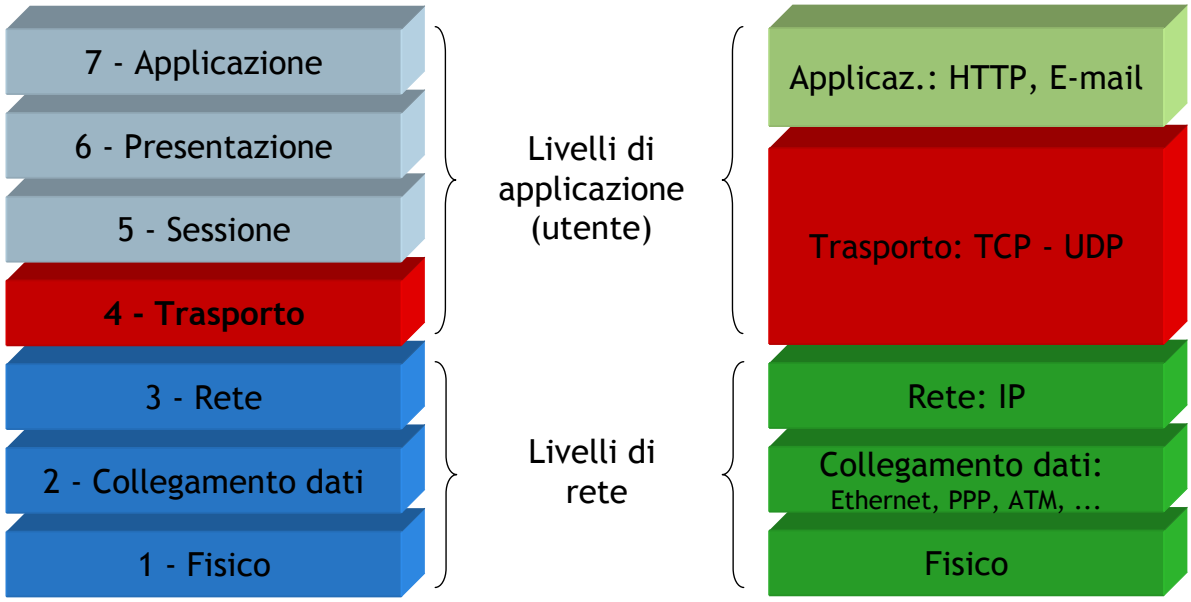
Reti di Calcolatori

Il livello di trasporto

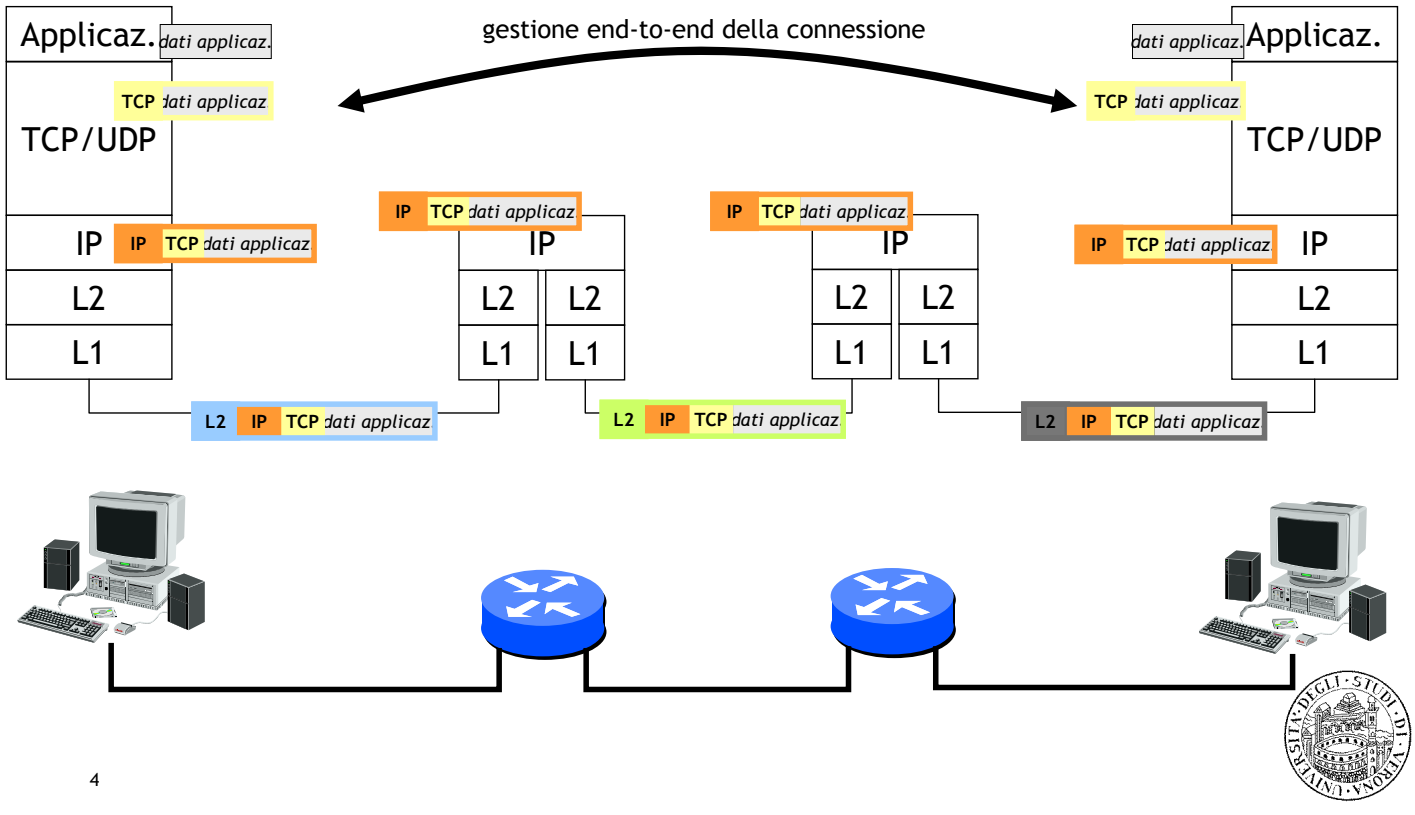
Universtità degli studi di Verona
Facoltà di Scienze MM.FF.NN.
A.A. 2009/2010
Laurea in Informatica
Docente: Damiano Carra

Livello Data Link



Generalità

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

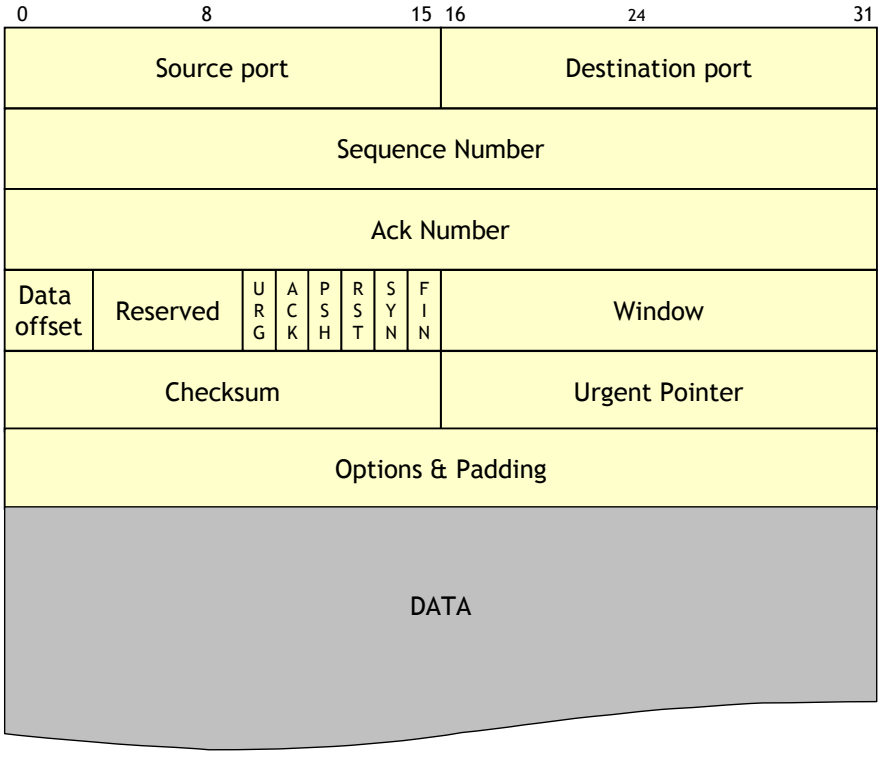
- Indirizzamento a livello applicativo / multiplazione / demultiplazione
  - modalità di differenziazione delle diverse applicazioni che utilizzano il protocollo di trasporto per trasferire i dati
  - multiplazione in caso di tratti di rete comuni

  UDP
- 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
  - 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

  TCP

Il protocollo TCP: formato dei segmenti

Header TCP



Significato dei campi

- *Source port - Destination port* [16 bit]: indirizzi della porta sorgente e della porta destinazione
- *Sequence Number* [32 bit]: numero di sequenza del primo byte del payload
- *Acknowledge Number* [32 bit]: numero di sequenza del prossimo byte che si intende ricevere (ha validità se il segmento è un ACK)
- *Offset* [4 bit]: lunghezza dell'header TCP, in multipli di 32 bit
- *Reserved* [6 bit]: riservato per usi futuri
- *Window* [16 bit]: ampiezza della finestra di ricezione (comunicato dalla destinazione alla sorgente)
- *Checksum* [16 bit]: risultato di un calcolo che serve per sapere se il segmento corrente contiene errori nel campo dati
- *Urgent pointer* [16 bit]: indica che il ricevente deve iniziare a leggere il campo dati a partire dal numero di byte specificato. Viene usato se si inviano comandi che danno inizio ad eventi asincroni "urgenti"

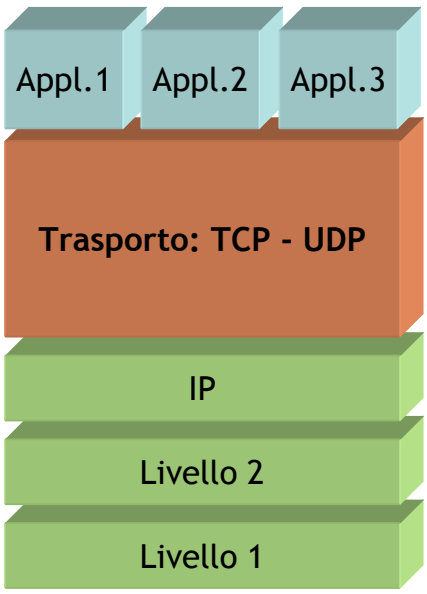
Significato dei campi

- *Flag* [ogni flag è lunga 1 bit]:
  - URG: vale uno se vi sono dati urgenti; in questo caso il campo urgent pointer ha senso
  - ACK: vale uno se il segmento è un ACK valido; in questo caso l'acknowledge number contiene un numero valido
  - PSH: vale uno quando il trasmettitore intende usare il comando di PUSH;
  - RST: reset; resetta la connessione senza un tear down esplicito
  - SYN: synchronize; usato durante il setup per comunicare i numeri di sequenza iniziale
  - FIN: usato per la chiusura esplicita di una connessione
- *Options and Padding* [lunghezza variabile]: riempimento (fino a multipli di 32 bit) e campi opzionali come ad esempio durante il setup per comunicare il MSS
- *Data*: i dati provenienti dall'applicazione

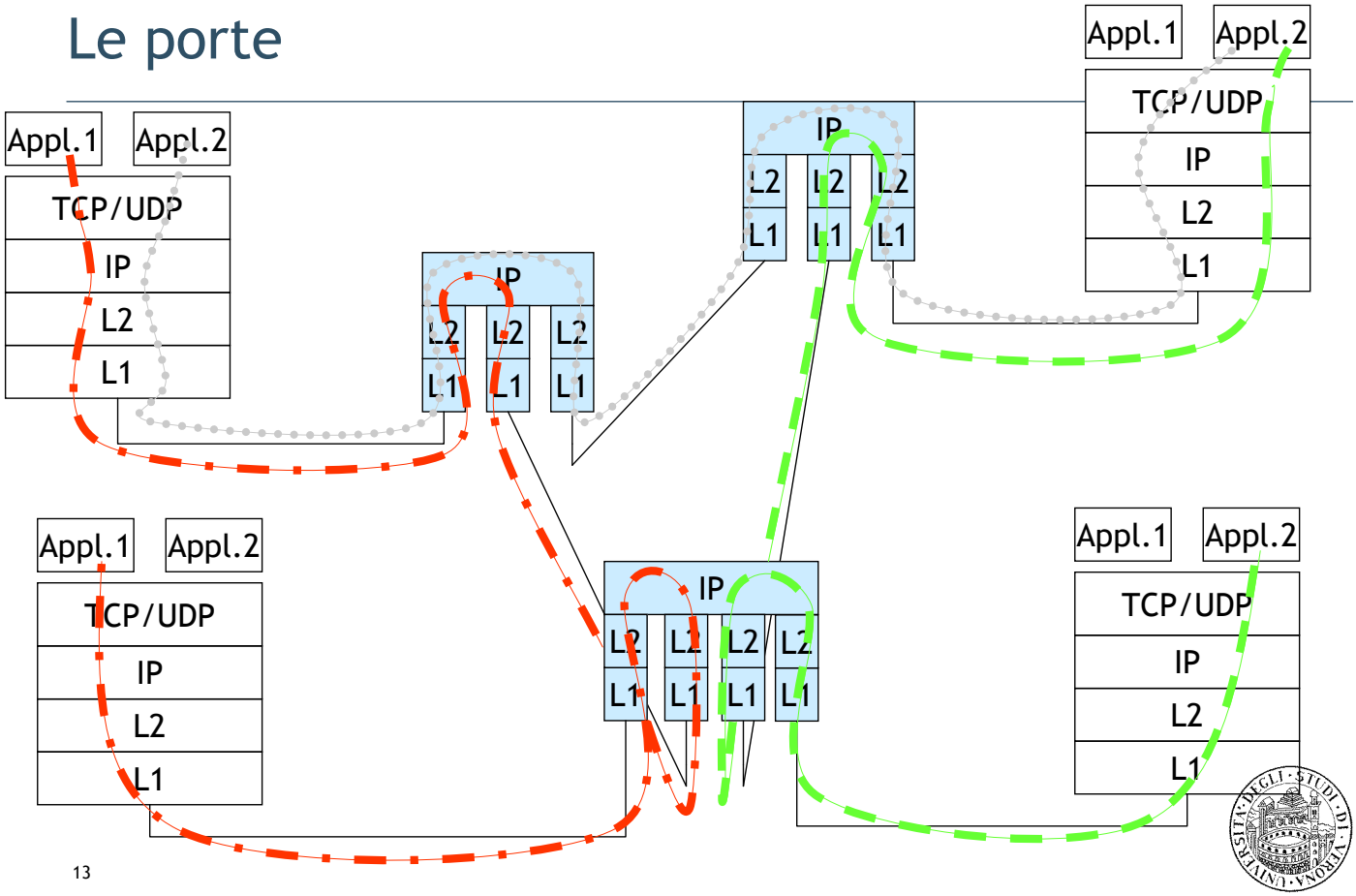
Indirizzamento

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

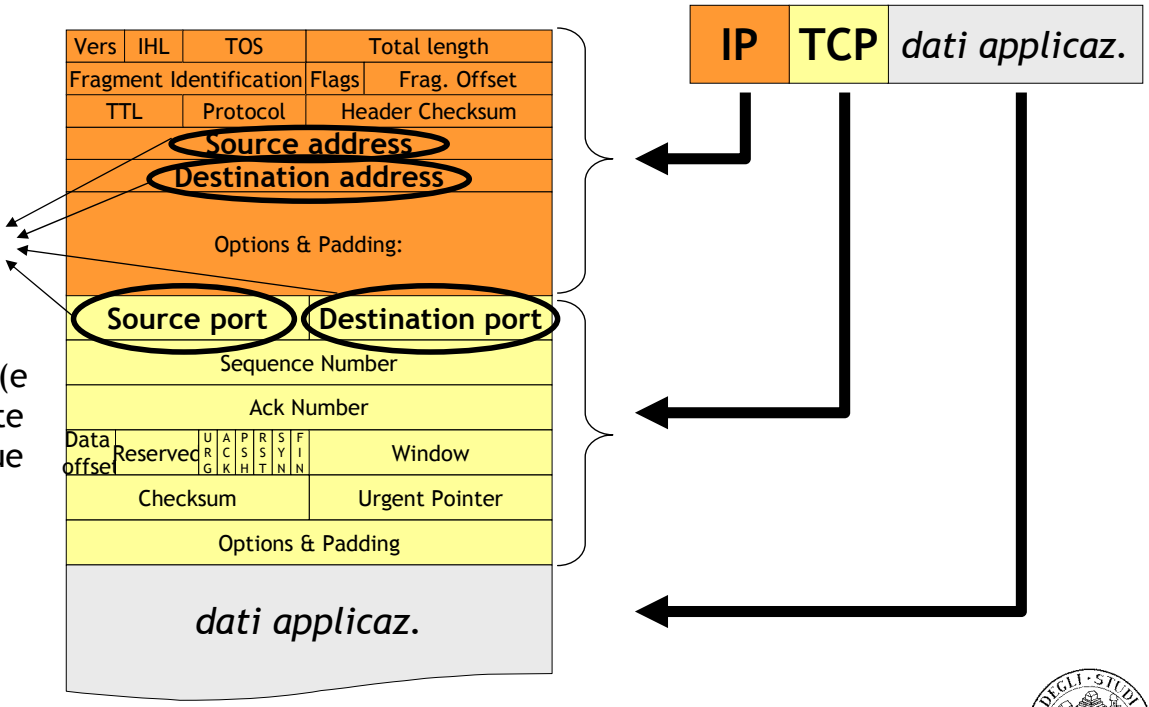


Le porte



Il concetto di Socket

SOCKET
tupla che identifica la connessione (e il conseguente flusso) tra due applicazioni



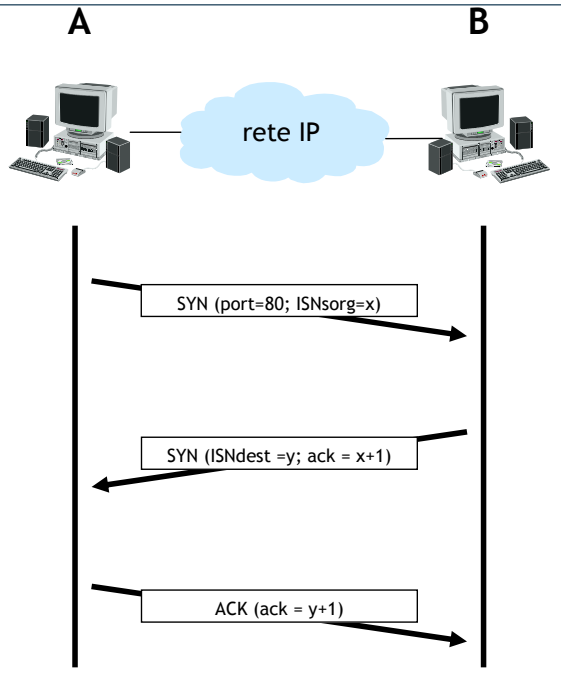
Indirizzamento TCP

- Il numero di porta può essere
  - statico (well known port)
    - sono identificativi associati ad applicazioni largamente utilizzate (posta elettronica, web, ftp, ...)
    - 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 necessariamente uguali
- L'utilizzo delle porte, insieme agli indirizzi IP sorgente e destinazione (→socket) servono per il multiplexing e demultiplexing dei dati da parte del TCP
- I socket identificano univocamente una connessione (su cui passa un unico flusso informativo)

Gestione delle connessioni

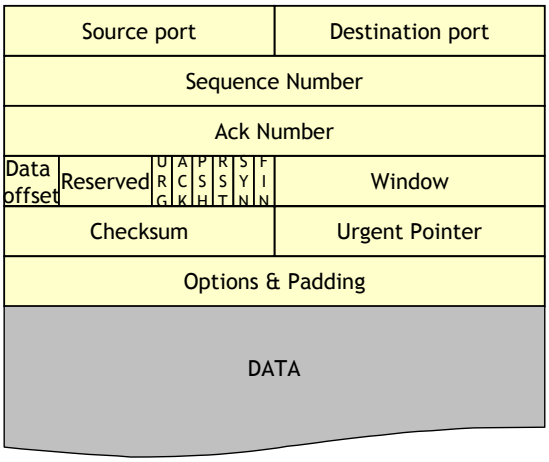
Instaurazione della connessione

- 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)



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

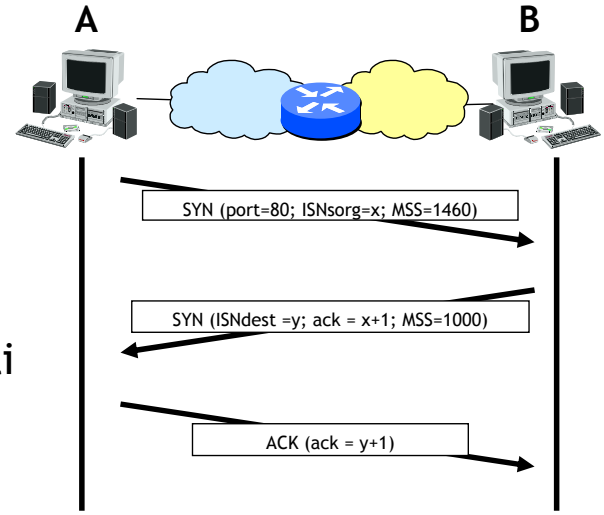


Instaurazione della connessione

- L'instaurazione della connessione serve per scambiarsi i dati relativi alla comunicazione
  - numero di porta (applicazione), numero iniziale di sequenza, dimensione della finestra, MSS, varie opzioni, ...
- Il canale di comunicazione aperto è bidirezionale
  - entrambe le stazioni possono trasferire dati
- Esempi
  - la stazione A sta visitando un sito www le cui pagine risiedono sulla stazione B
    - A apre una connessione verso B
    - alla fine della procedura di apertura della connessione, A manda un segmento con la richiesta di una pagina
      - la richiesta viene riscontrata dalla stazione B indicando la corretta ricezione
    - sulla stessa connessione, B invia la pagina ad A
      - ad ogni segmento ricevuto, A manda un ack a B
  - la stazione A invia una mail alla stazione B (server di posta)
    - A apre una connessione verso B
    - alla fine della procedura di apertura di connessione, A invia la mail a B
      - ogni segmento ricevuto, B invia un riscontro ad A

Maximum Segment Size (MSS)

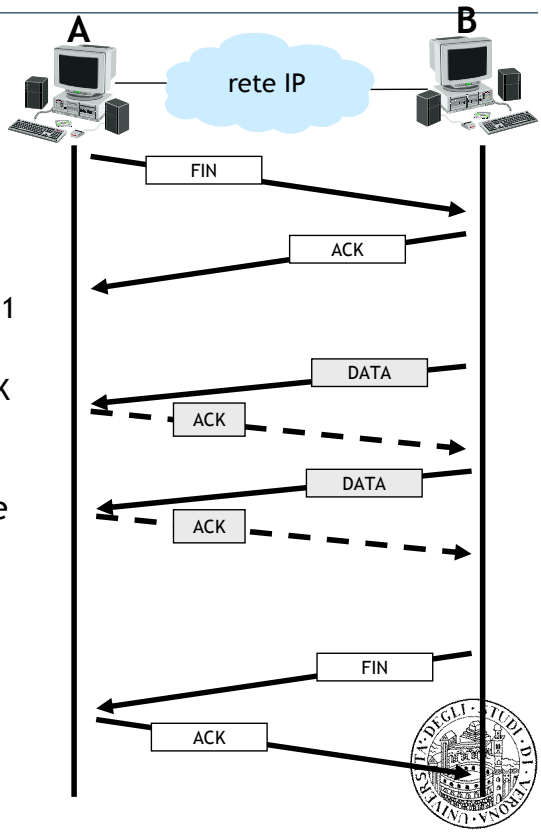
- Tra i vari parametri scambiati all'inizio di una connessione vi può essere anche la MSS
- La MSS consente ad una stazione di specificare la dimensione massima (espressa in numero di byte) del campo dati dei segmenti che è disponibile a ricevere
  - esistono meccanismi per far scoprire alla stazione 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



Gestione degli errori e delle perdite

Affidabilità

- Il TCP è un protocollo connection oriented affidabile, ovvero garantisce la corretta e ordinata consegna dei segmenti
- Il TCP dunque 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, il TCP si preoccupa della ritrasmissione dei segmenti
- 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
- Il 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
- Il RTO deve dunque essere calcolato dinamicamente di volta in volta
  - durante la fase di instaurazione della connessione
  - durante la trasmissione dei dati
- Il calcolo del 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

$$SRTT_{attuale} = (\alpha * SRTT_{precedente}) + ((1-\alpha) * RTT_{istantaneo})$$

$RTT_{istantaneo} \rightarrow$ misura di RTT sull' ultimo segmento

$SRTT_{precedente} \rightarrow$ stima precedente del valore medio di RTT

$SRTT_{attuale} \rightarrow$ stima attuale del valore medio di RTT

$\alpha \rightarrow$ coefficiente di peso $(0 < \alpha < 1)$

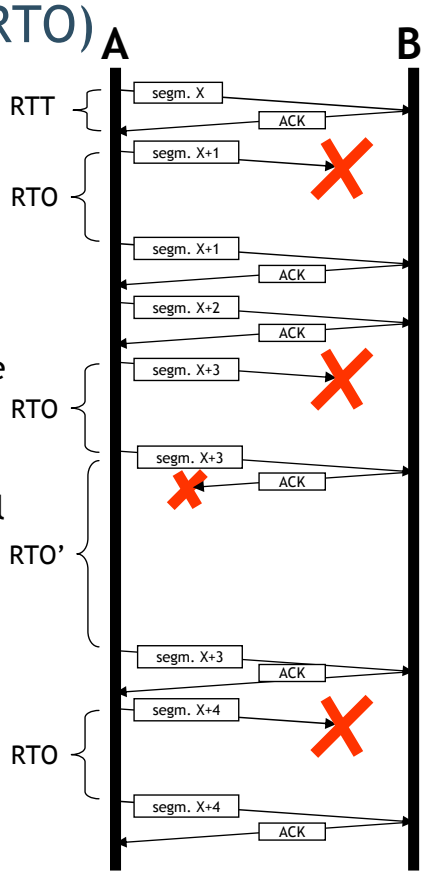
- Poiché RTT può variare anche molto in base alle condizioni della rete, il valore di RTT (SRTT, Smoothed RTT) utilizzato per il calcolo di RTO risulta una stima del valor medio di RTT sperimentato dai diversi segmenti
- Il parametro α è regolabile e, a seconda dei valori assunti, rende il peso della 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 = 0.9

Stima del Retransmission Time Out (RTO)

$$RTO = \beta * SRTT$$

$\beta \rightarrow$ delay variance factor (tipicamente = 2)

- La sorgente, dunque, attende fino a 2 volte il RTT medio (SRTT) 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
  - RTOretransmission = 2*RTO



Stima di RTT e RTO

- Le implementazioni attuali del TCP utilizzano algoritmi più evoluti per il calcolo di RTT e RTO, ma il principio dell'adattamento rispetto le 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?

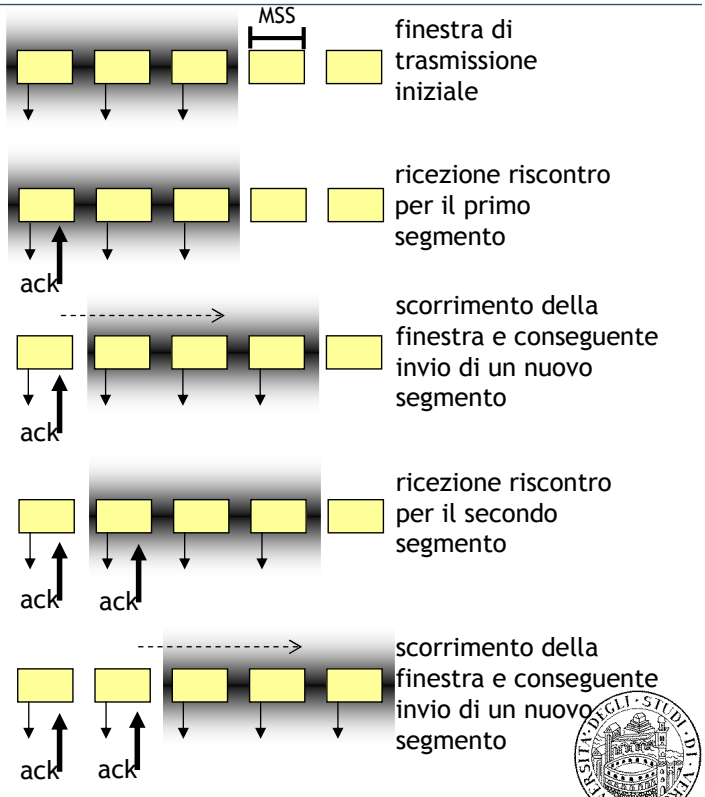
Controllo di flusso

Generalità

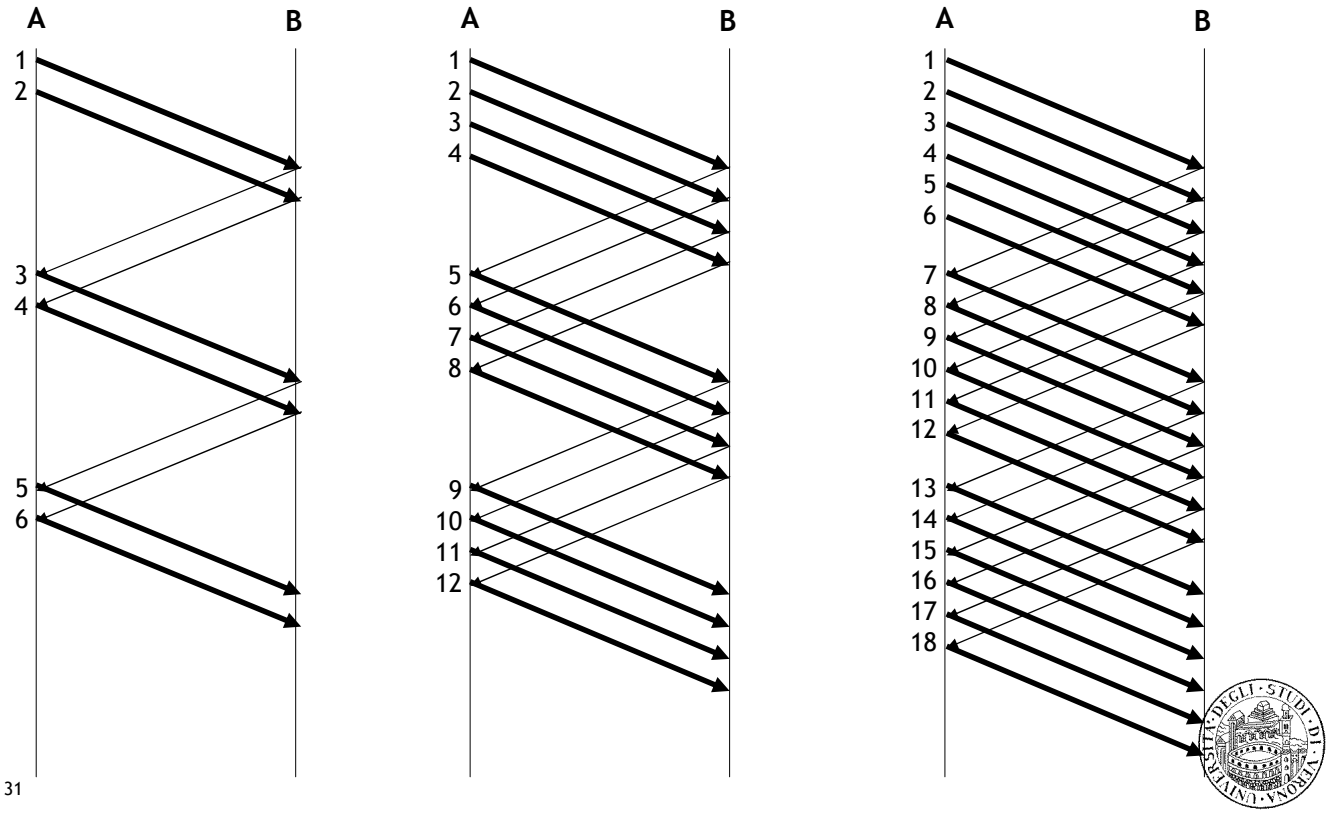
- Richiamiamo qui la differenza tra *controllo di flusso* e *controllo di congestione*
  - Controllo di flusso: ***azione preventiva*** finalizzata a limitare l'immissione di dati in rete a seconda della capacità end-to-end di questa
  - Controllo della congestione: azioni da intraprendere come ***reazione*** alla congestione di rete
- Problematiche associate al controllo di flusso:
  - com'è possibile da parte di una stazione determinare la capacità e lo stato della rete?
- Con la tecnica di *postive 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)



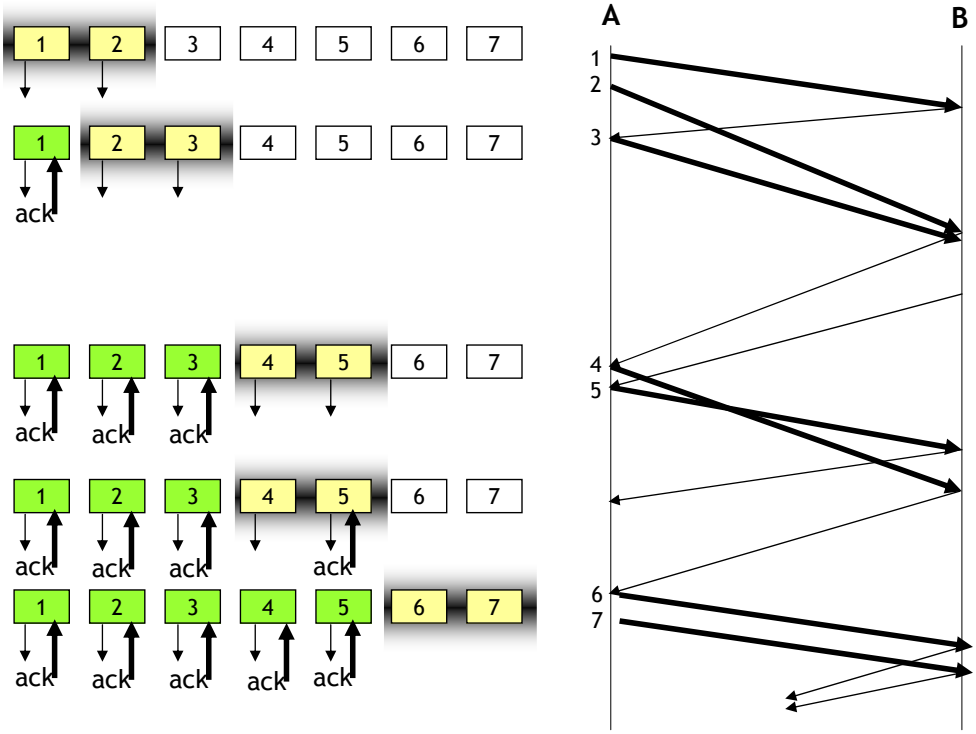
Influenza della dimensione della finestra



Considerazioni

- Se la dimensione della finestra è troppo piccola → sottoutilizzo la banda
- Se la dimensione è troppo grande → i riscontri potrebbero arrivare prima della conclusione della trasmissione della finestra
  - non c'è controllo di flusso (trasmissione continua)
- La trasmissione di un nuovo segmento dipende sempre dalla ricezione di un riscontro, quindi...
- ...con il metodo a finestra scorrevole, il tasso di immissione di nuovi segmenti dipende da:
  - condizione della rete
  - condizione del ricevente

Esempio di funzionamento



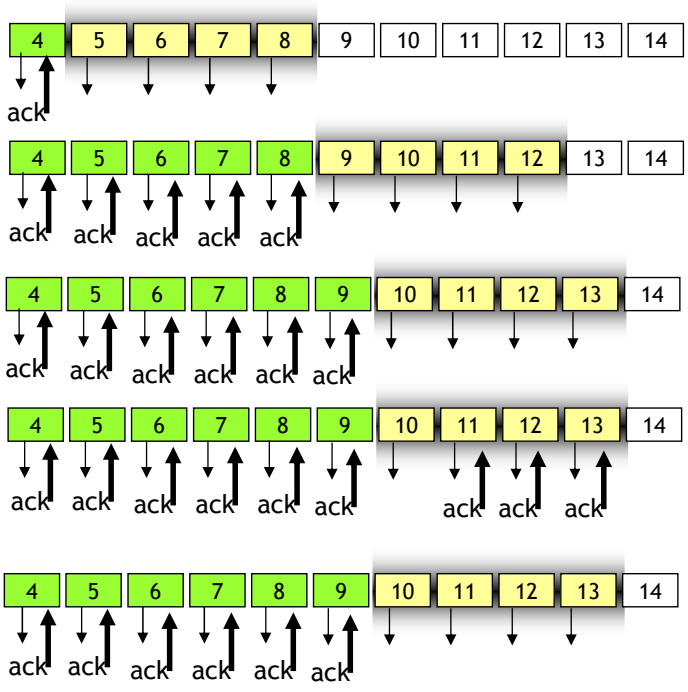
La finestra scorrevole nel TCP

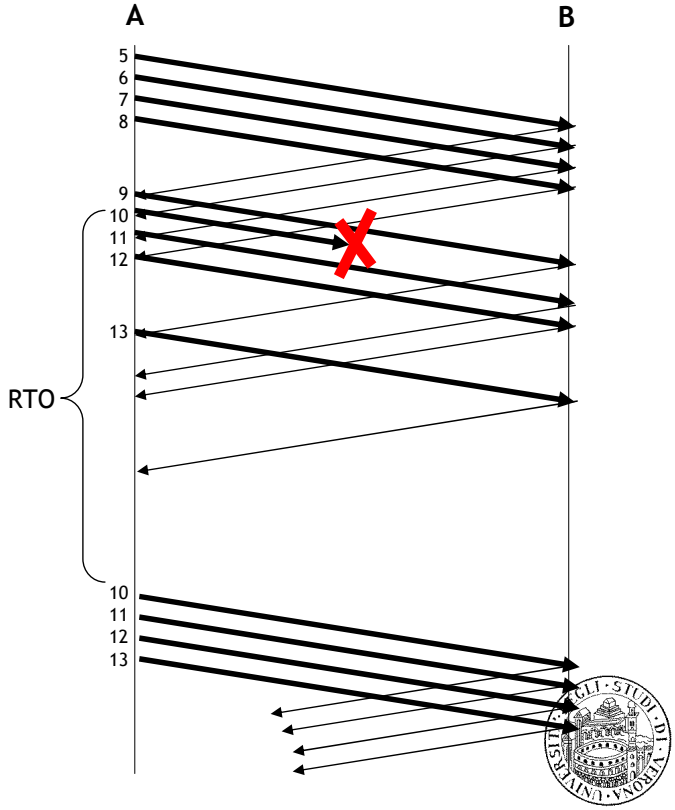
- I dati provenienti dall'applicazione vengono ordinati in una sequenza di byte numerati progressivamente (a partire dall'ISN, Initial Sequence Number)
- In base alla dimensione della MSS, tali byte sono suddivisi in segmenti
- Il meccanismo a finestra scorrevole è applicato alla sequenza di segmenti 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...

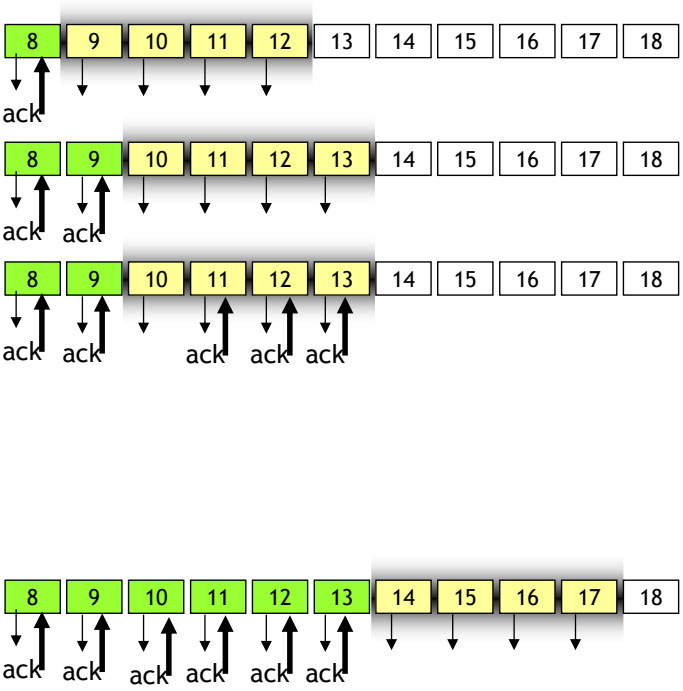
- In caso di perdita di un segmento, dopo lo scadere di un timeout (RTO), il TCP provvede a ritrasmetterlo
  - il comportamento risulta intuitivo se la finestra è pari a 1 segmento
- Problema: come si comporta il TCP quando il segmento appartiene ad una finestra di trasmissione (più segmenti trasmessi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

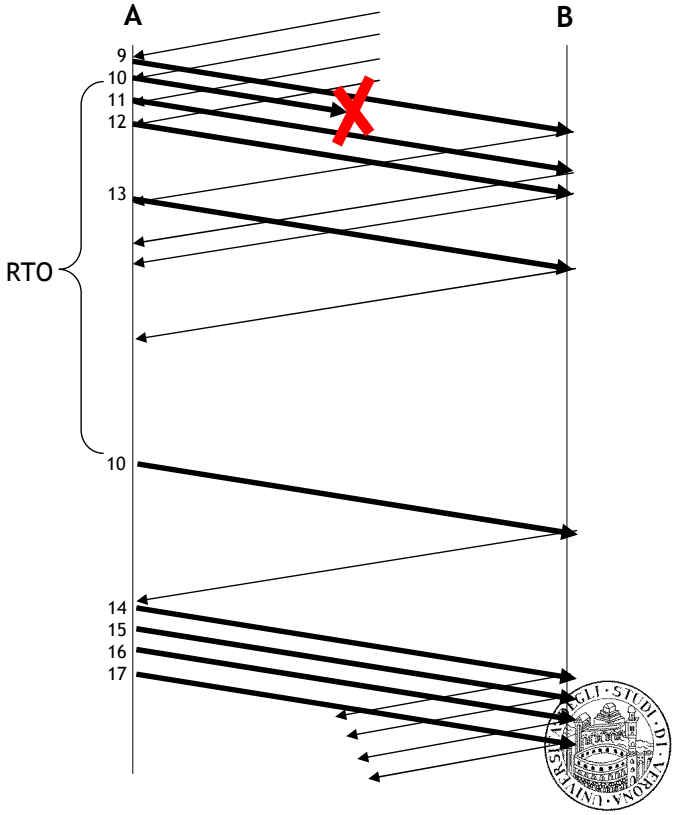
Go-back-N





Selective Repeat





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

- 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?

Controllo di congestione

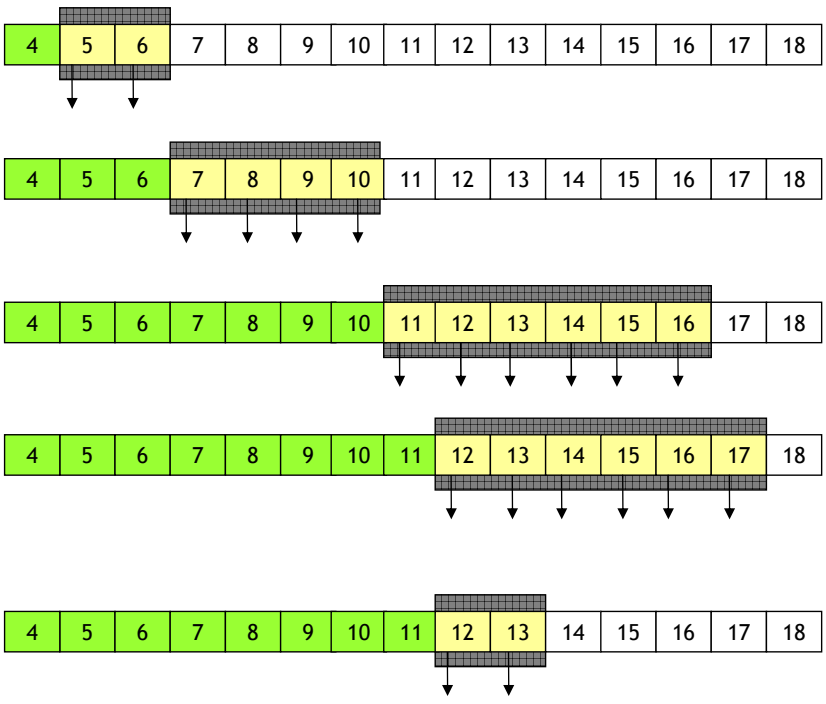
Generalità

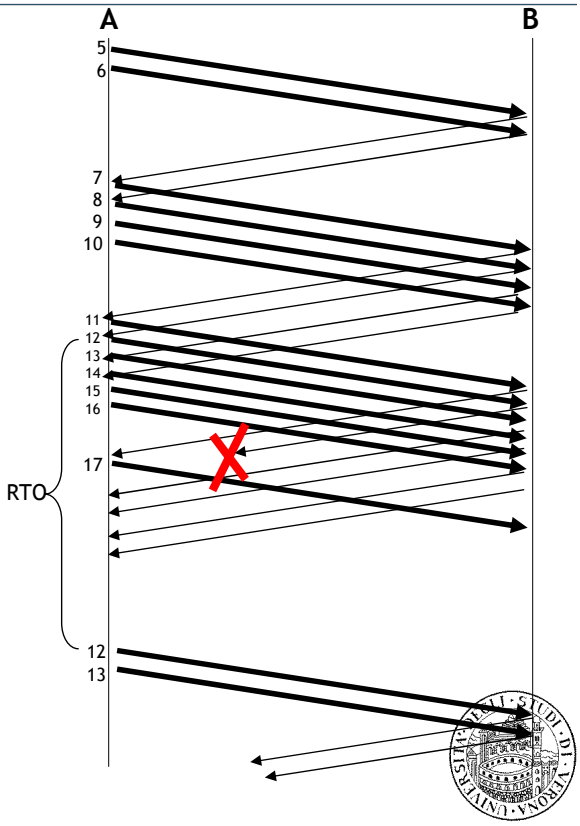
- In caso di congestione della rete, a causa dei buffer limitati degli apparati di rete, alcuni segmenti potrebbero venire persi
- La perdita dei segmenti e il relativo scadere del timeout di ritrasmissione è considerato un sintomo di congestione
  - il TCP non ha altri mezzi per conoscere lo stato della 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 la prevenzione delle situazioni di sovraccarico della rete

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





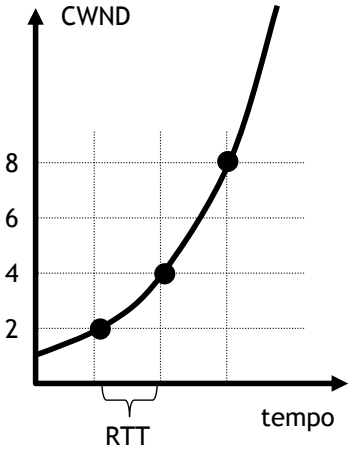
Algoritmi per il controllo della congestione

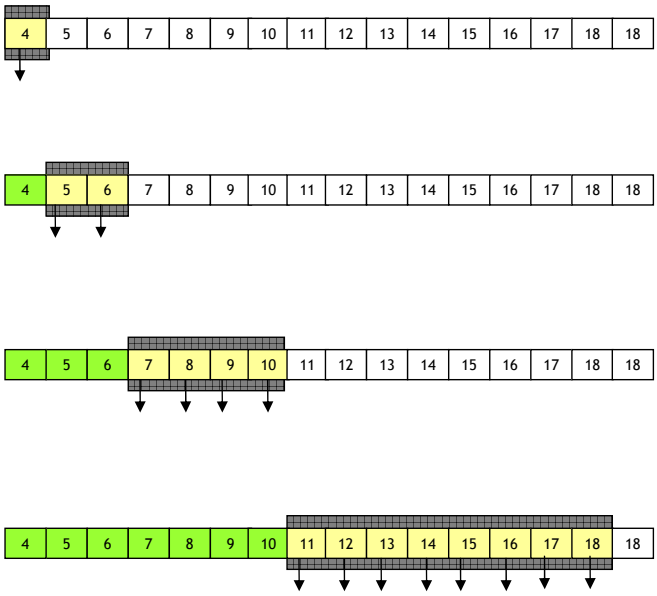
- Esistono diversi algoritmi che regolano la dimensione della finestra (CWND) al variare delle condizioni di rete
- I due algoritmi base sono:
  - Slow Start
  - Congestion Avoidance
- Altri algoritmi sono stati definiti per aumentare l'efficienza del TCP
  - Fast Retransmit
  - Fast Recovery
  - SACK

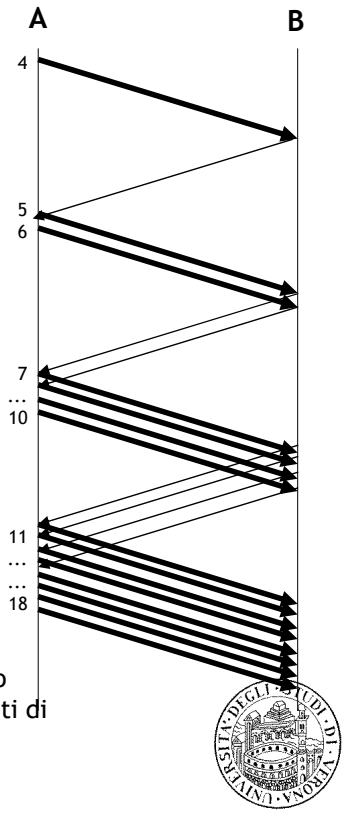
Slow Start e Congestion Avoidance

- 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

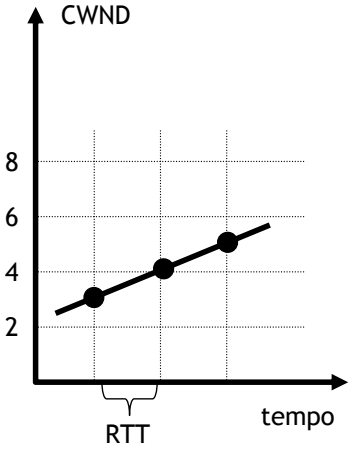


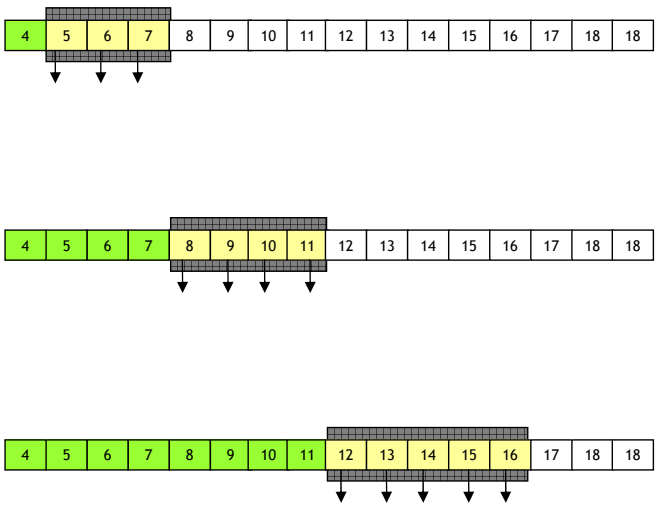


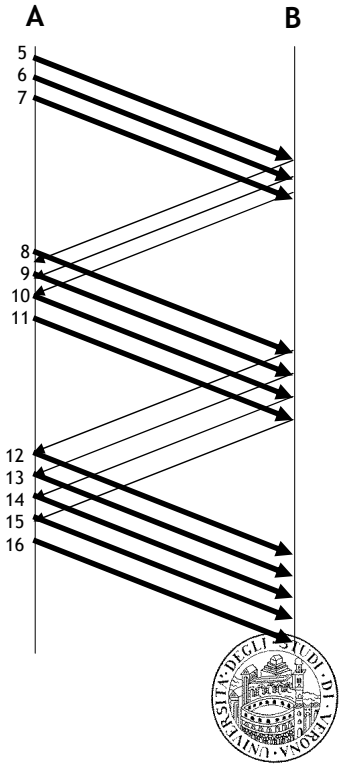


NOTA: per semplificare la rappresentazione grafica, si assume che i segmenti vengano generati e trasmessi tutti nello stesso istante e i corrispettivi riscontri vengano ricevuti di conseguenza tutti insieme dopo un tempo pari a RTT (supposto costante)

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

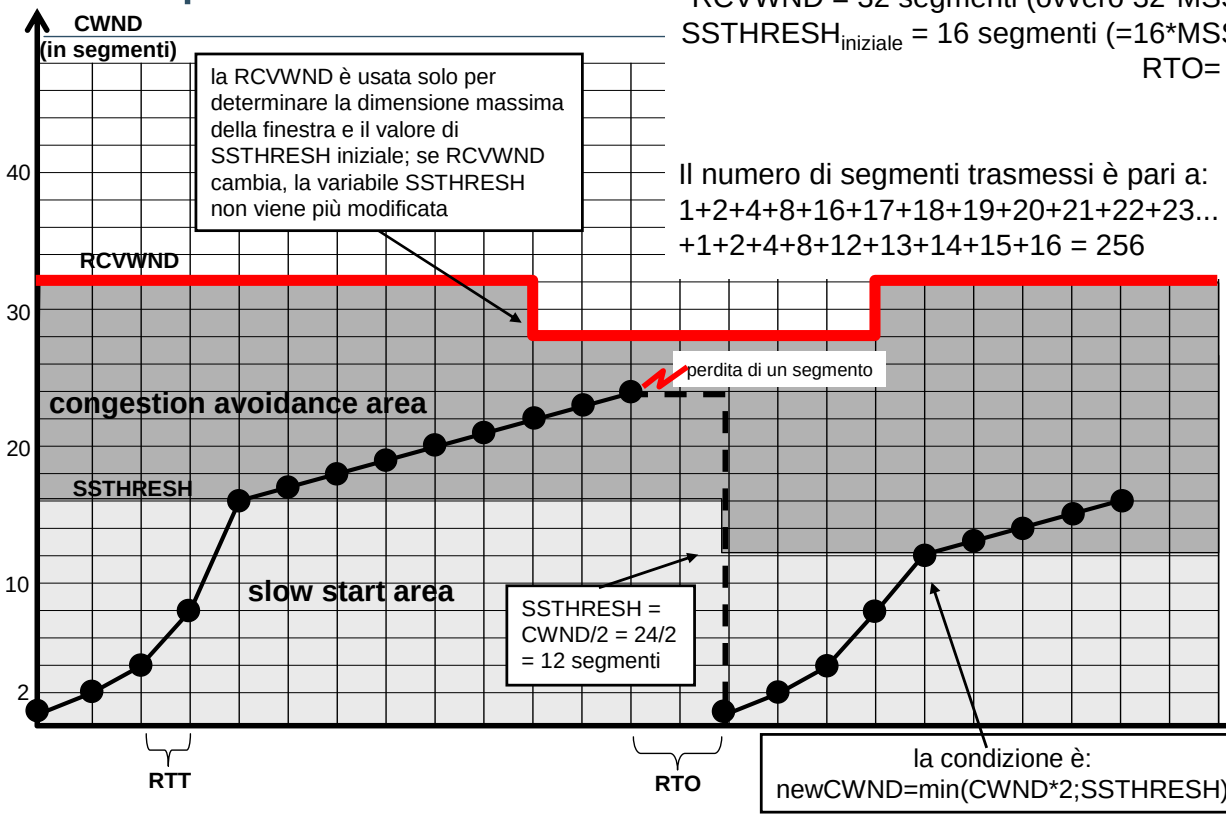
- L'algoritmo che regola la dimensione della CWND è il seguente:
  - all'inizio della trasmissione si pone
    - CWND = 1 segmento (ovvero un numero di byte pari a MSS)
    - SSTHRESH = RCVWND oppure SSTHRESH = RCVWND / 2 (dipende dalle implementazioni)
  - la CWND evolve secondo l'algoritmo di Slow 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



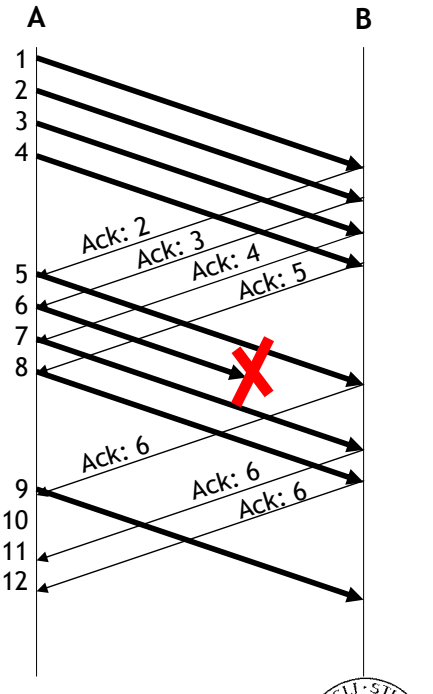
Il numero di segmenti trasmessi è pari a:
1+2+4+8+16+17+18+19+20+21+22+23...
+1+2+4+8+12+13+14+15+16 = 256

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 piu' 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)

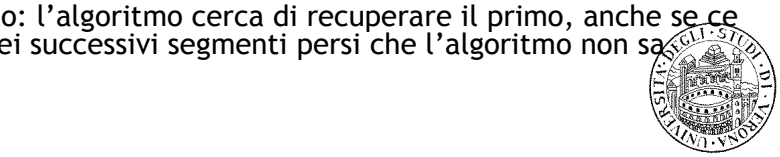
ACK duplicati

- 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 e' ancora arrivato
  - la ricezione di un numero sufficientemente alto di ACK duplicati puo' essere interpretata come forte di indicazione che e' avvenuta una perdita

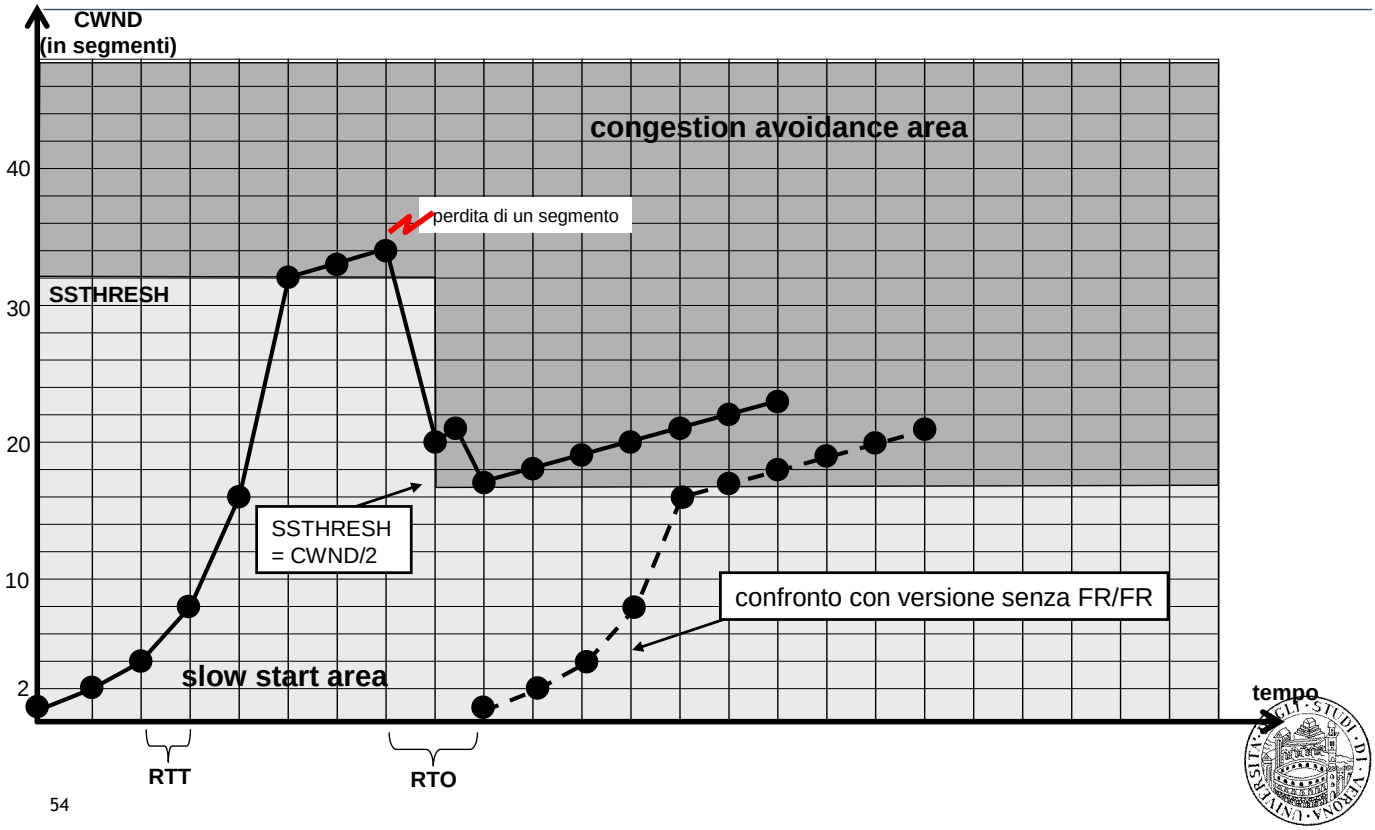


Fast Retransmit – Fast Recovery: algoritmo

- 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 tramissione 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



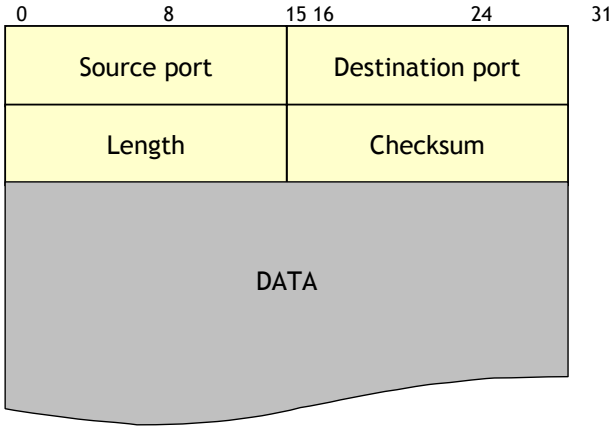
Riassunto

- Il TCP è un protocollo di trasporto connection oriented affidabile e svolge le seguenti funzioni:
  - indirizzamento a livello applicativo / multiplazione / demultiplazione
    - utilizza il concetto di porta per indirizzare le diverse applicazioni;
  - instaurazione, gestione e rilascio delle connessioni
    - utilizza il concetto di socket per identificare una connessione;
    - si preoccupa di gestire la connessione, ovvero lo scambio di informazioni necessarie per concordare l'attivazione di un canale di comunicazione (INS, MSS, Window, ...)
  - recupero degli errori
    - gestisce il recupero dei segmenti errati o persi utilizzando un timeout di ritrasmissione; RTT e RTO vengono calcolati dinamicamente
    - una volta recuperati gli errori, il TCP può effettuare la consegna ordinata dei segmenti all'applicazione
  - Controllo di flusso
    - implementa un controllo di flusso a finestra scorrevole (con dimensione della finestra variabile) come controllo del tasso di immissione di segmenti in rete
  - Controllo della congestione
    - reagisce alla congestione (scadere del RTO) diminuendo l'ampiezza della finestra di trasmissione secondo algoritmi noti

Il protocollo UDP

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 nel campo dati



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' compito dei livelli superiori (applicazioni) la verifica della perdita dei messaggi, consegna in ordine, controllo di flusso, ...
- E' utilizzato per il supporto di transazioni semplici tra applicativi e per le applicazioni multimediali

Verifica contenuti

Domande generali

- In quali tipi di nodi di rete è implementato il livello d i trasporto?
- A cosa servono i numeri di porta?
- A cosa servono i numeri di porta considerati noti (well known ports)?
- Cos'è il socket e come si identifica?
- Che tipo di servizio viene offerto all'applicazione da parte del protocollo TCP?
- Cosa vuol dire che il TCP è un protocollo connection-oriented affidabile?
- Che differenza c'è tra controllo di flusso e controllo di congestione?
- Che meccanismi usa il TCP per il controllo di flusso? E per il controllo della congestione?

Domande generali

- Per cosa viene utilizzato il campo Window dell'header TCP?
- Per cosa viene utilizzato il campo SYN dell'header TCP?
- A cosa serve l'opzione MSS dell'header quando viene usata?
- Come viene stimato il timeout di ritrasmissione del TCP?
- Che cos'è la CWND?
- Descrivere come evolve la CWND
  - Come si distingue tra le fasi di Slow Start e Congestion Avoidance?
  - Come viene variata la CWND in Slow Start?
  - Come viene variata la CWND in Congestion Avoidance?
  - Come vengono fissate la CWND e la SSTHRESH allo scadere di un timeout?
- Descrivere il comportamento in caso di perdita singola qualora sia implementata la variante Fast Retransmit e Fast Recovery
- Che tipo di servizio viene offerto all'applicazione da parte del protocollo UDP? Che funzioni implementa?

Note sulle modalità di svolgimento degli esercizi sul TCP

Ipotesi generali:

- si ragiona sempre per
  - numero di segmenti,
    - occorre fare la conversione byte → numero di segmenti, attraverso la MSS
  - multipli di RTT
    - occorre fare la conversione secondi → RTT, in base al valore del RTT
- L'intervallo RTT è considerato un intervallo chiuso:
  - i segmenti vengono inviati contemporaneamente all'inizio di ciascun intervallo e con tempo di trasmissione trascurabile
  - tali segmenti vengono ricevuti e riscontrati simultaneamente dal ricevitore
  - i riscontri (ACK) arrivano simultaneamente al termine dell'intervallo lungo RTT

Evoluzione della finestra di trasmissione:

- L'evoluzione della CWND segue le seguenti regole
  - IF (CWND_old < SSTHRESH)
    - CWND_new = min ((CWND_old + numero_ACK); SSTHRESH; RCVWND)
  - ELSE
    - CWND_new = min((CWND_old + (numero_ACK/CWND_old)); RCVWND)
- In caso di errori sul canale:
  - si attendere lo scadere del timeout (RTO)
    - le istruzioni per determinare il RTO vengono date nel testo dell'esercizio
  - si pone
    - SSTHRESH = max((CWND/2); 2)
    - CWND_new = 1 segmento
  - ATTENZIONE: i periodi di "rete fuori uso" sono intervalli aperti; durante tali periodi tutti i segmenti in transito vengono persi

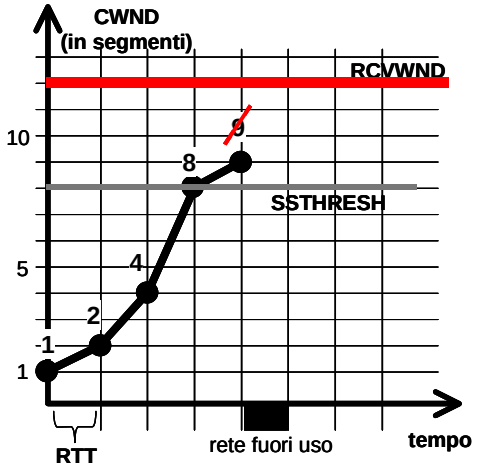
Obiettivo degli esercizi:

- disegnare l'evoluzione della trasmissione fino a fine trasmissione, determinando:
  - CWND
  - SSTHRESH
  - segmenti inviati ad ogni RTT
- NOTA: la trasmissione ha fine solo quando sono stati ricevuti tutti i riscontri degli ultimi segmenti inviati
- ATTENZIONE: i segmenti inviati e la dimensione della CWND non sempre coincidono

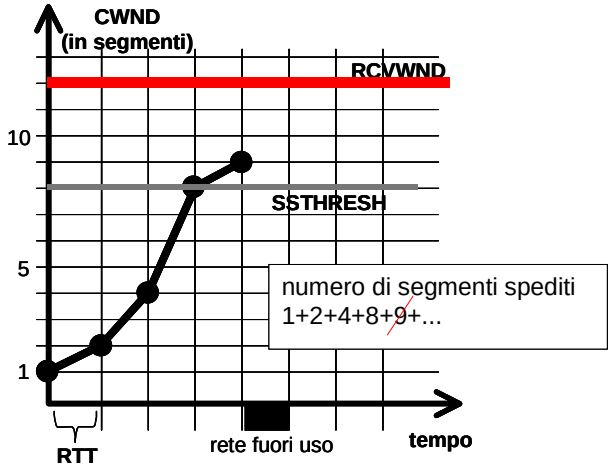
Notazione utilizzata:

- il grafico dell'evoluzione della trasmissione ha come unità di misura:
  - sull'asse x → il Round Trip Time (RTT)
  - sull'asse y → la CWND (espressa in numero di segmenti)
- i punti sul grafico rappresentano dunque il valore della CWND per ogni RTT
  - per chiarezza, in genere si deve esplicitare il valore assunto dalla CWND a fine trasmissione (gli altri valori sono desunti dal grafico)
- gli effettivi segmenti inviati vengono indicati con un numero posto al di sopra di ciascun punto del grafico
  - in caso di segmenti inviati e persi, la notazione prevede di barrare tale numero
- SSTHRESH e RCVWND sono linee che si devono distinguere dal resto del grafico e tra loro
  - la differenziazione può essere effettuata tramite l'utilizzo di linee di maggior spessore, tratteggiate, colorate...

NOTAZIONE 1



NOTAZIONE 2



Reti di Calcolatori: Esercizi sul TCP

1) Comportamento base

Un'applicazione A deve trasferire 96 kbyte all'applicazione B utilizzando il protocollo TCP. Si supponga che la connessione sia già stata instaurata. Le variabili note sono le seguenti:

- MSS concordata pari a 1000 byte;
- RCVWND annunciata pari a 32 Kbyte;
  - SSTHRESH = RCVWND/2;
- RTT costante pari a 0.5 secondi;
- primo RTO= 2*RTT;
  - perdite sequenziali RTO_new=2*RTO_old

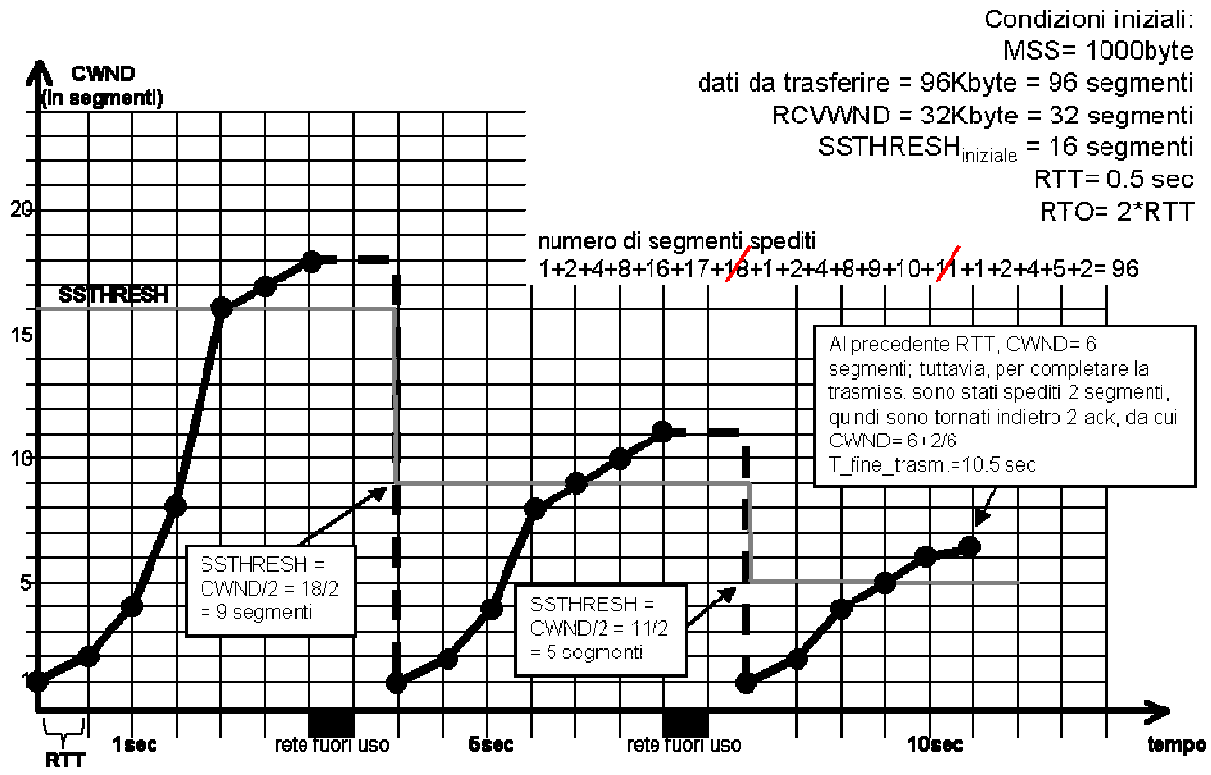
Si supponga che la rete vada fuori uso

- t1=3 sec x 0.5 sec
- t2=7 sec x 0.5 sec

Determinare

- andamento della CWND e valore finale della CWND e della SSTHRESH
- tempo di trasferimento

Soluzione



2) Perdite consecutive

Un'applicazione A deve trasferire 46.5 kbyte all'applicazione B utilizzando il protocollo TCP. Si supponga che la connessione sia già stata instaurata. Le variabili note sono le seguenti:

- MSS concordata pari a 1500 byte;
- RCVWND annunciata pari a 24 Kbyte;
  - SSTHRESH = RCVWND/2;
- RTT costante pari a 0.5 secondi;
- primo RTO= 2*RTT;
  - perdite sequenziali RTO_new=2*RTO_old

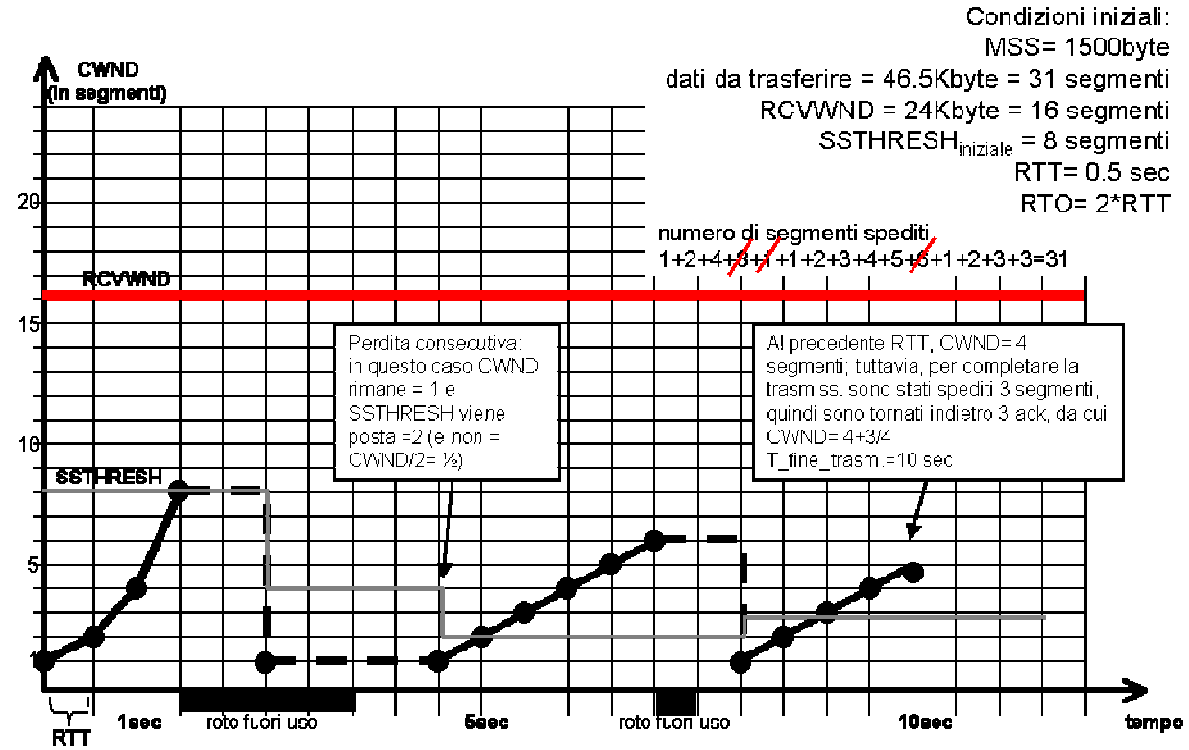
Si supponga che la rete vada fuori uso

- t1=1.5 sec x 2 sec
- t2=7 sec x 0.5 sec

Determinare

- andamento della CWND e valore finale della CWND e della SSTHRESH
- tempo di trasferimento

Soluzione



3) Raggiungimento RCVWND

Un'applicazione A deve trasferire 104 kbyte all'applicazione B utilizzando il protocollo TCP. Si supponga che la connessione sia già stata instaurata. Le variabili note sono le seguenti:

- MSS concordata pari a 1200 byte;
- RCVWND annunciata pari a 24 Kbyte;
  - SSTHRESH = RCVWND;
- RTT costante pari a 0.5 secondi;
- primo RTO= 2*RTT;
  - perdite sequenziali RTO_new=2*RTO_old

Si supponga che la rete vada fuori uso

- t1=3.5 sec x 0.5 sec
- t2=6.5 sec x 4 sec

Determinare

- andamento della CWND e valore finale della CWND e della SSTHRESH
- tempo di trasferimento

Soluzione