Controllo di congestione

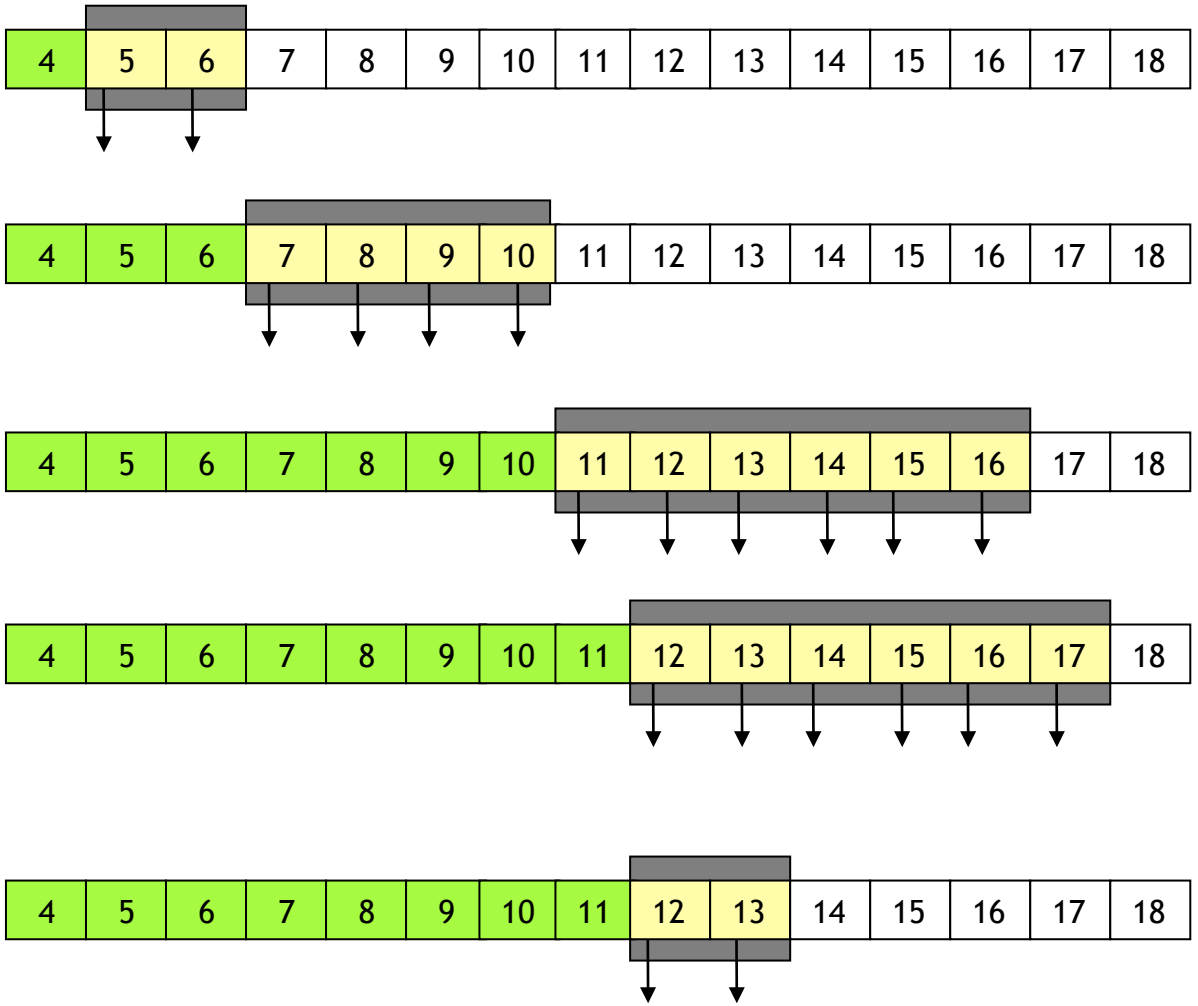
Generalità

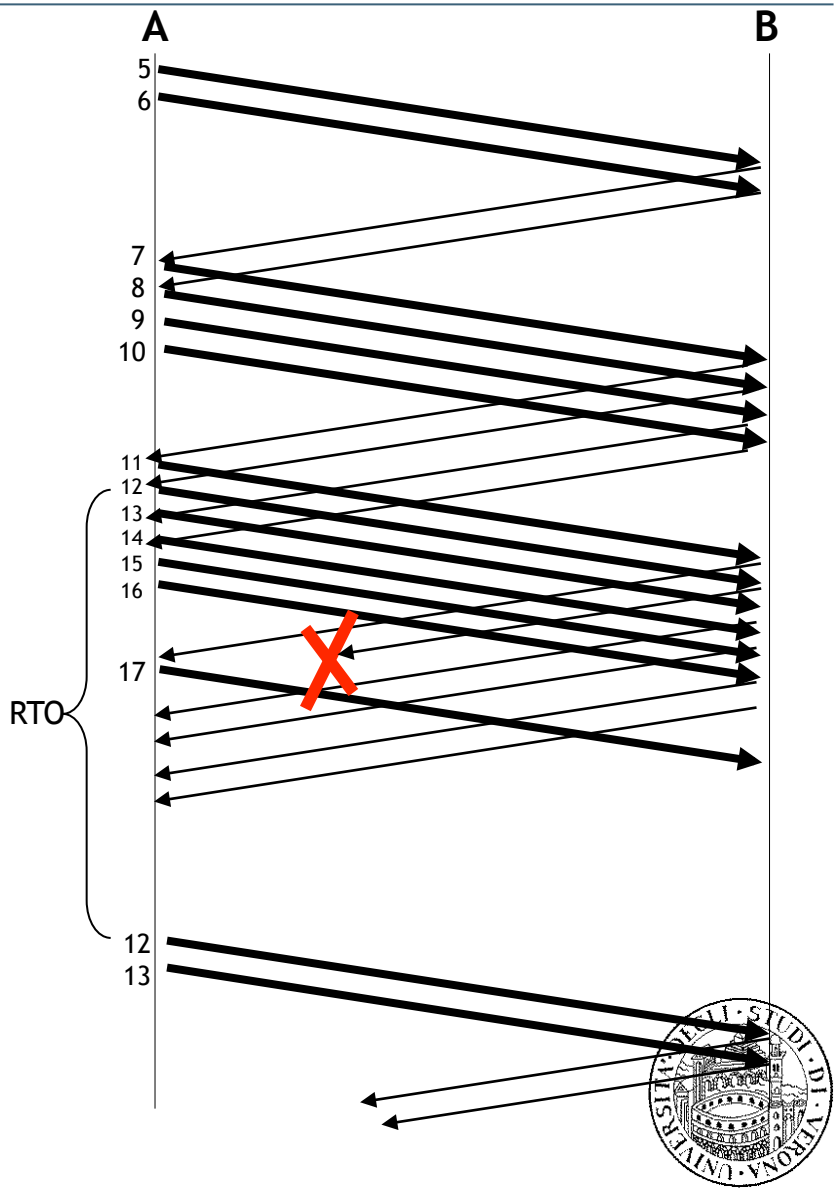
- In caso di congestione della rete, a causa dei buffer limitati degli apparati di rete, alcuni segmenti potrebbero venire persi
- La perdita dei segmenti e il relativo scadere del timeout di ritrasmissione è considerato un sintomo di congestione
  - il TCP non ha altri mezzi per conoscere lo stato della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


4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
4 5 6 7 8 9 10 11 12 13 14 15 16 17 18


A
B
5
6
7
8
9
10
11
12
13
14
15
16
17
RTO
12
13

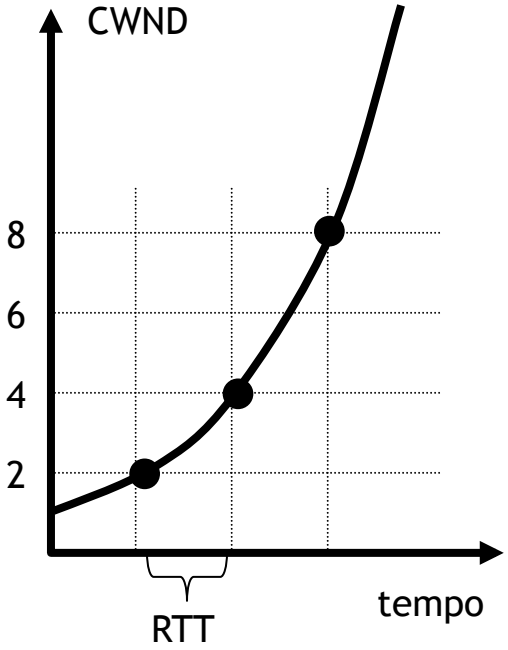
Algoritmi per il controllo della congestione

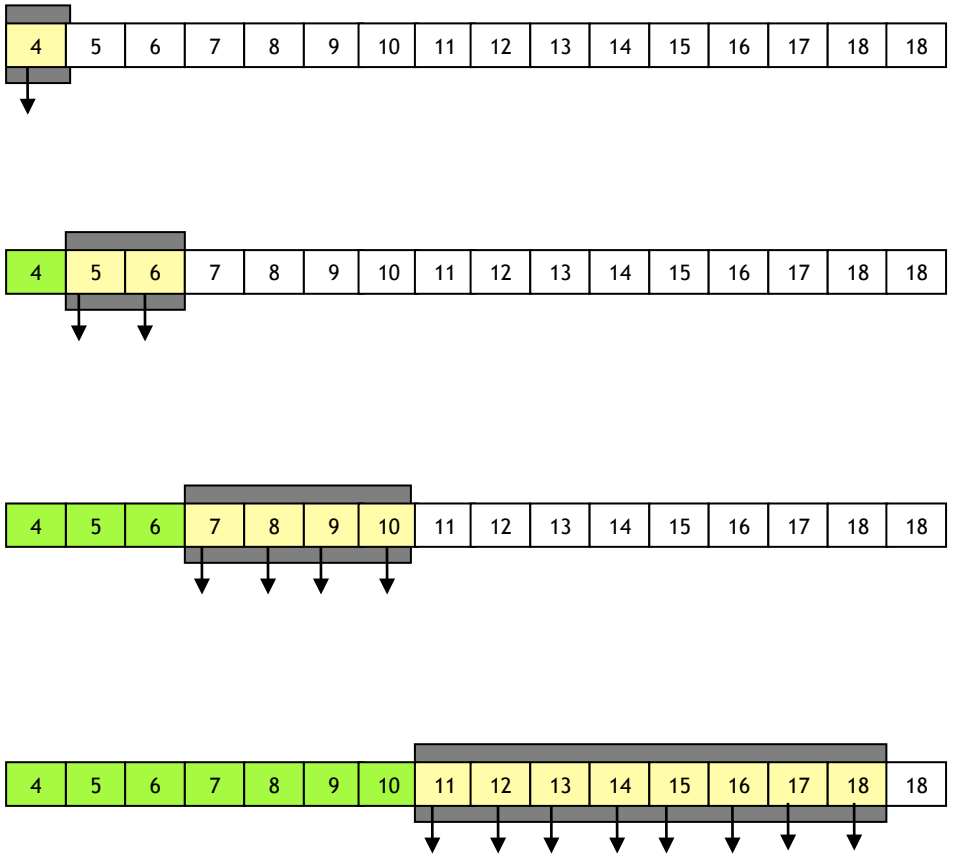
- Esistono diversi algoritmi che regolano la dimensione della finestra (CWND) al variare delle condizioni di rete
- I due algoritmi base sono:
  - Slow Start
  - Congestion Avoidance
- Altri algoritmi sono stati definiti per aumentare l'efficienza del TCP
  - Fast Retransmit
  - Fast Recovery
  - SACK

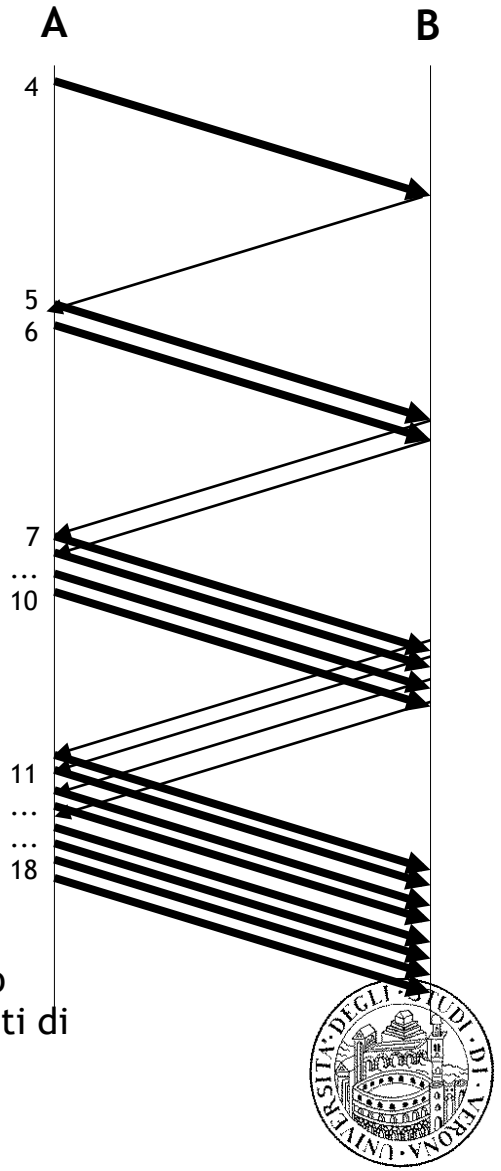
Slow Start e Congestion Avoidance

- 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

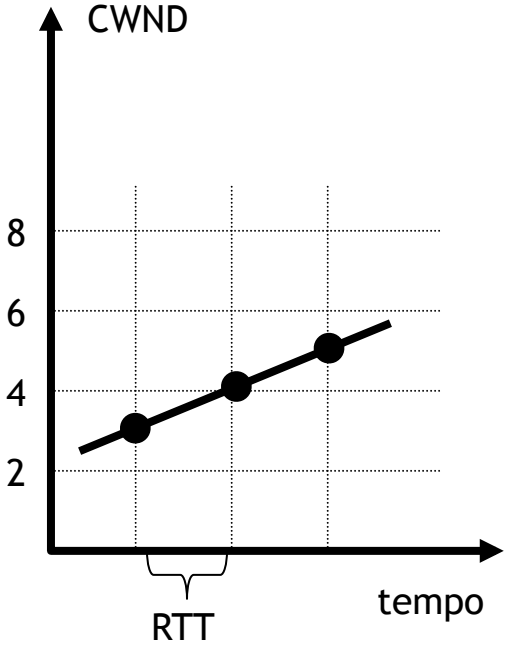


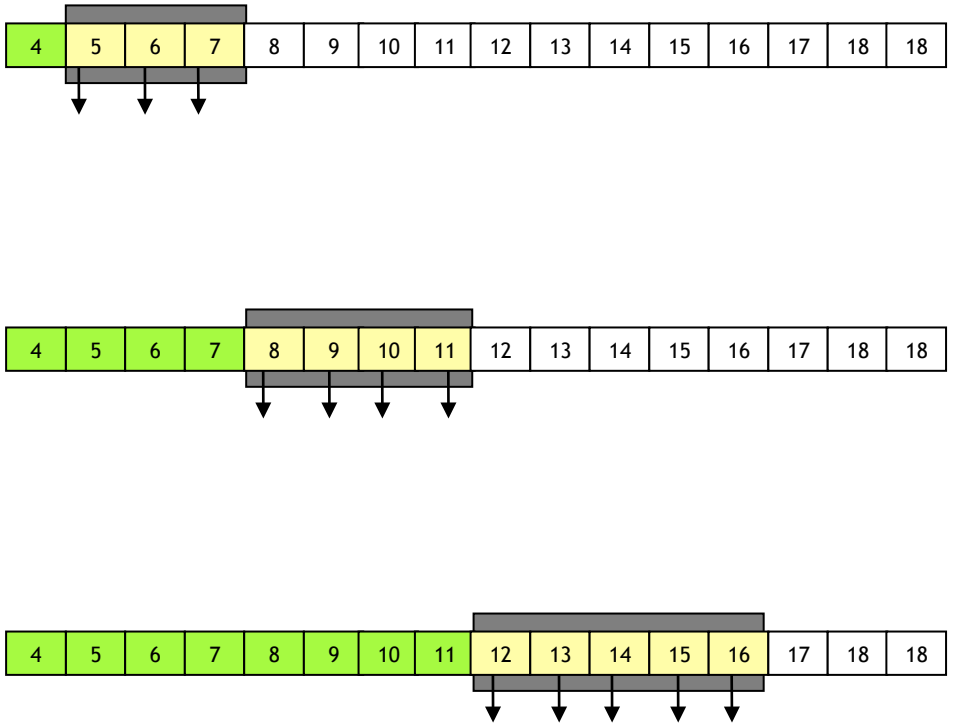


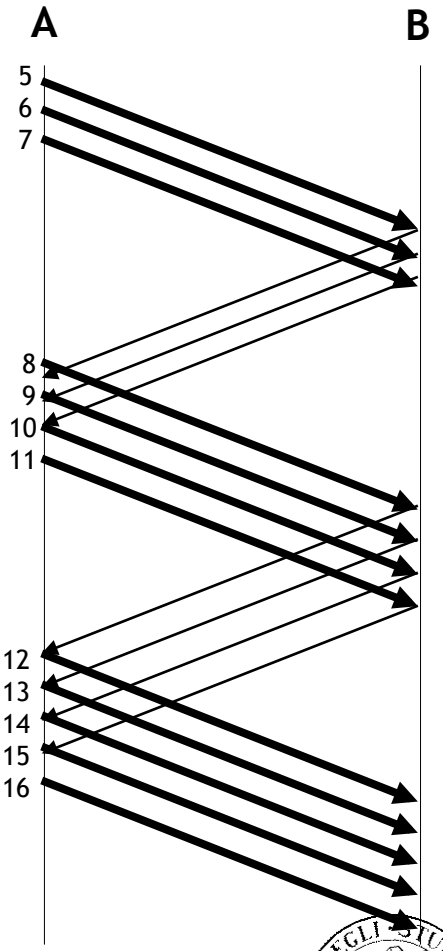


NOTA: per semplificare la rappresentazione grafica, si assume che i segmenti vengano generati e trasmessi tutti nello stesso istante e i corrispettivi riscontri vengano ricevuti di conseguenza tutti insieme dopo un tempo pari a RTT (supposto costante)

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

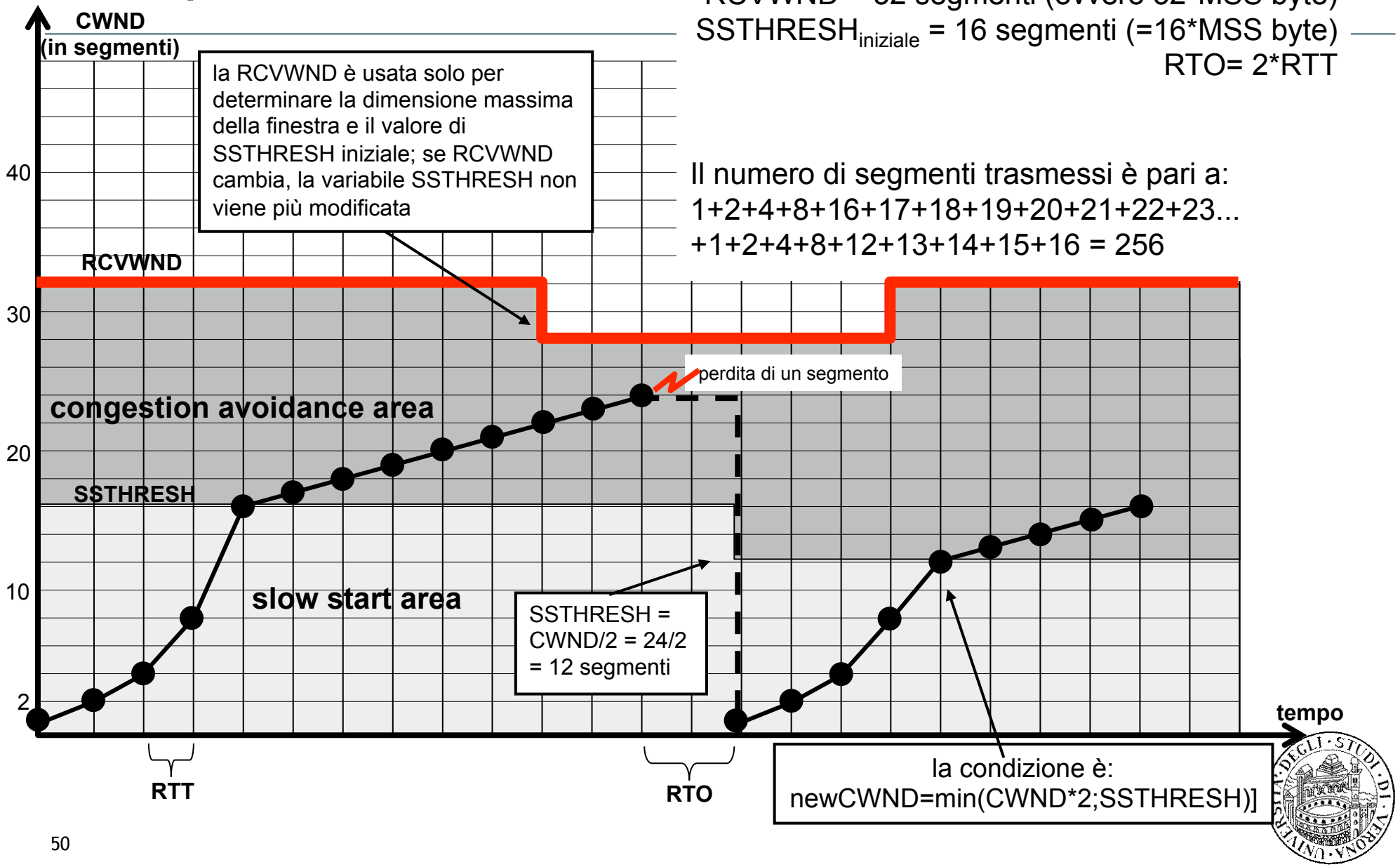
- L'algoritmo che regola la dimensione della CWND è il seguente:
  - all'inizio della trasmissione si pone
    - CWND = 1 segmento (ovvero un numero di byte pari a MSS)
    - SSTHRESH = RCVWND oppure SSTHRESH = RCVWND / 2 (dipende dalle implementazioni)
  - la CWND evolve secondo l'algoritmo di Slow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



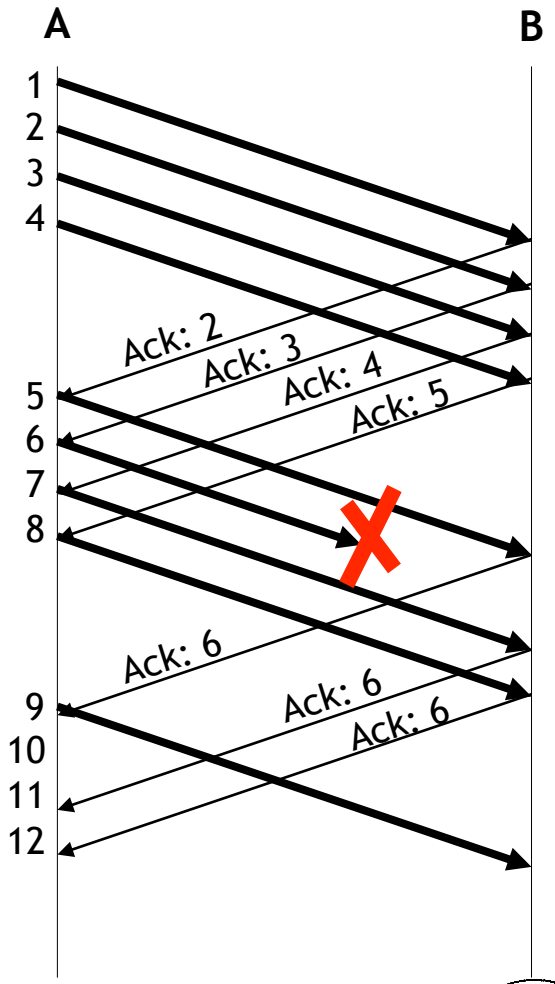
Il numero di segmenti trasmessi è pari a:
1+2+4+8+16+17+18+19+20+21+22+23...
+1+2+4+8+12+13+14+15+16 = 256

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

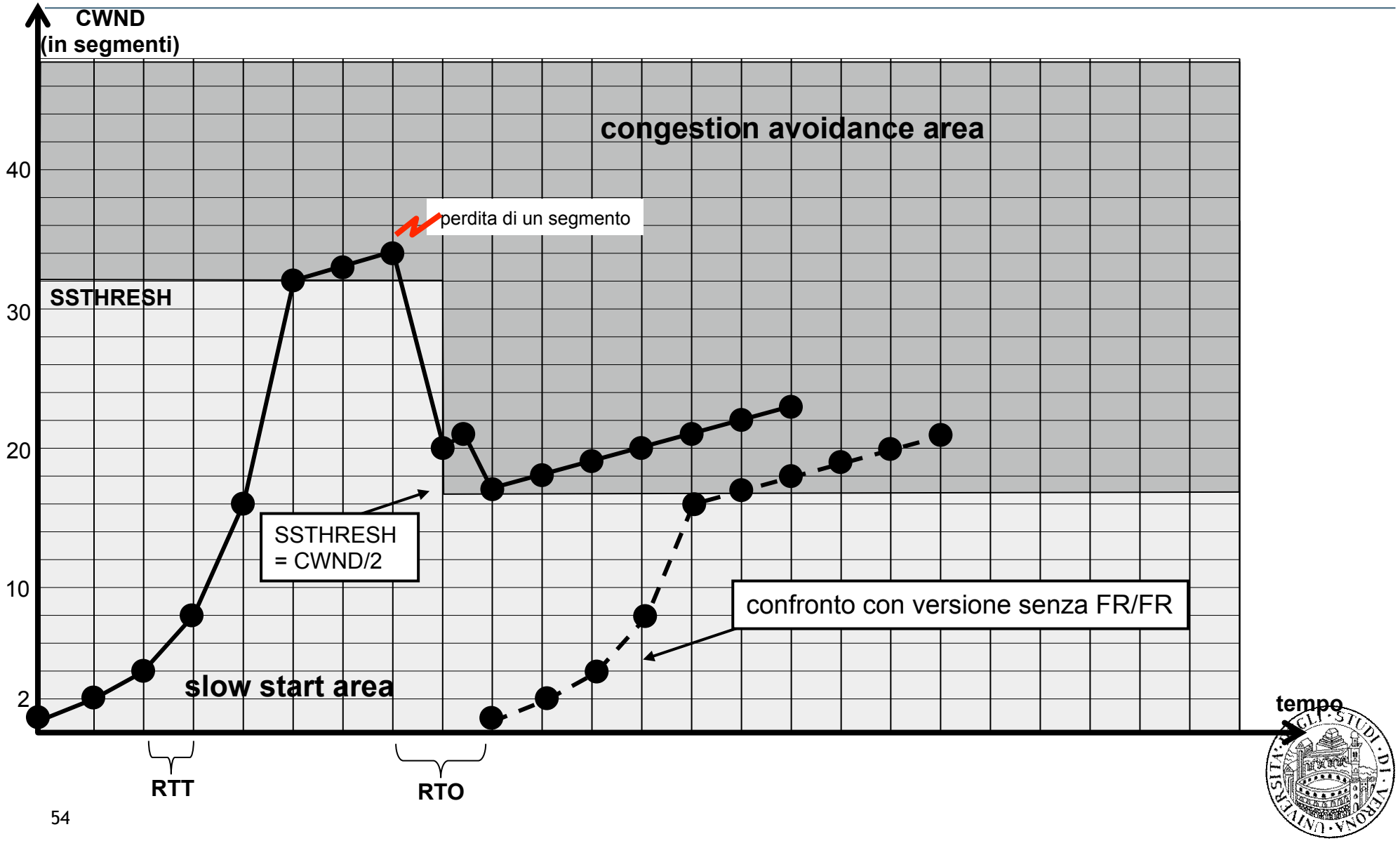
- Negli ACK il campo ***Ack Number*** contiene il successivo numero di sequenza che ci si aspetta arrivi
  - questo dice implicitamente che tutti i segmenti precedenti sono stati ricevuti correttamente
- Se un segmento arriva fuori sequenza, la destinazione invia un ACK indicando il numero di sequenza 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