TCP: Transport Control Protocol
Ottimizzazioni del protocollo

INFN-CNAF

Tiziana.Ferrari@cnaf.infn.it

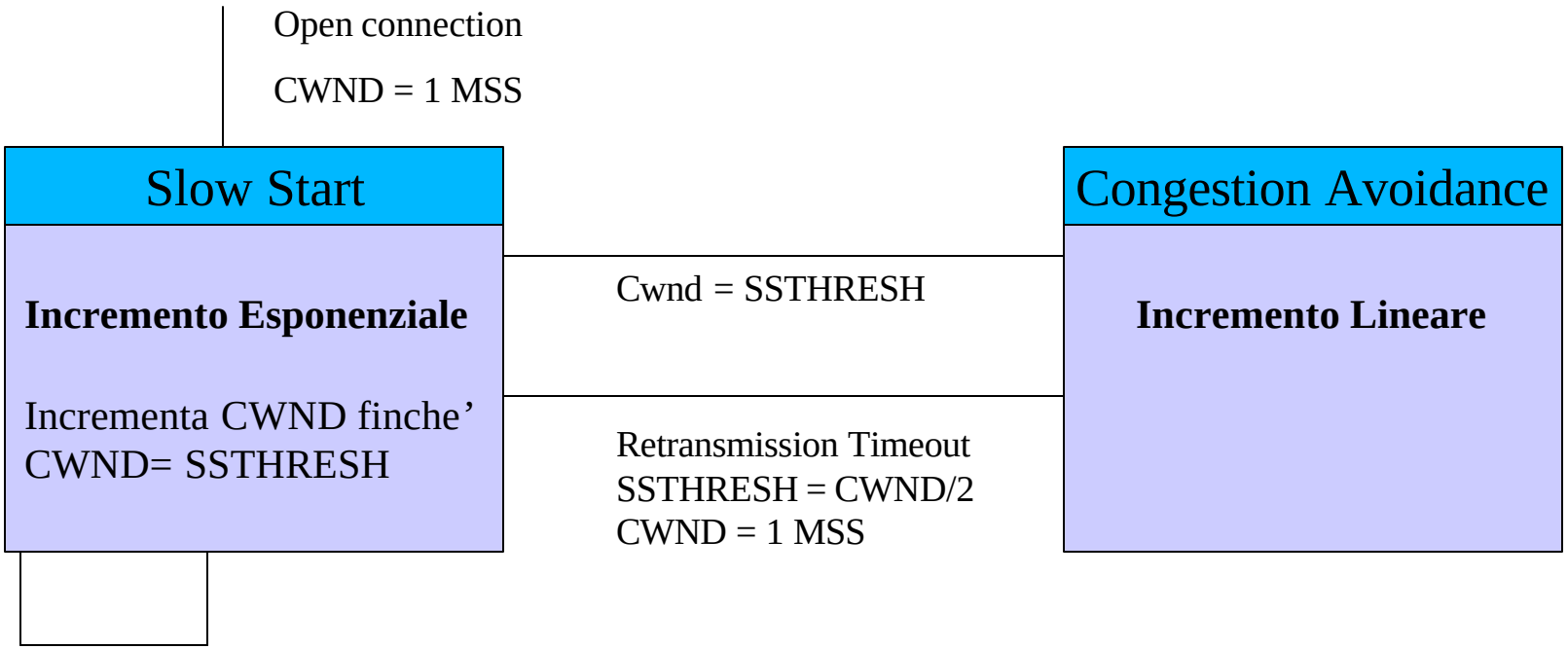
Sommario

- Introduzione
- Fast Recovery + Fast Retransmit
- TCP SACK
- Active Queue Management: Random Early Discard
- Early Congestion Notification
- Conclusioni

Introduzione

- Diverse ottimizzazioni proposte negli ultimi anni:
  - High Speed, Wireless, Link satellitari
- Congestion Control: migliorare il recupero dei pacchetti persi (la congestione si e' gia' verificata)
  1. Fast Retransmit
  2. Fast Recovery
  3. SACK
- Congestion Avoidance: prevenire la congestione riducendo il rate di trasmissione *prima* che essa si verifichi
  1. RED
  2. ECN
  3. (TCP Vegas)

Richiamo: Slow Start e Congestion Avoidance



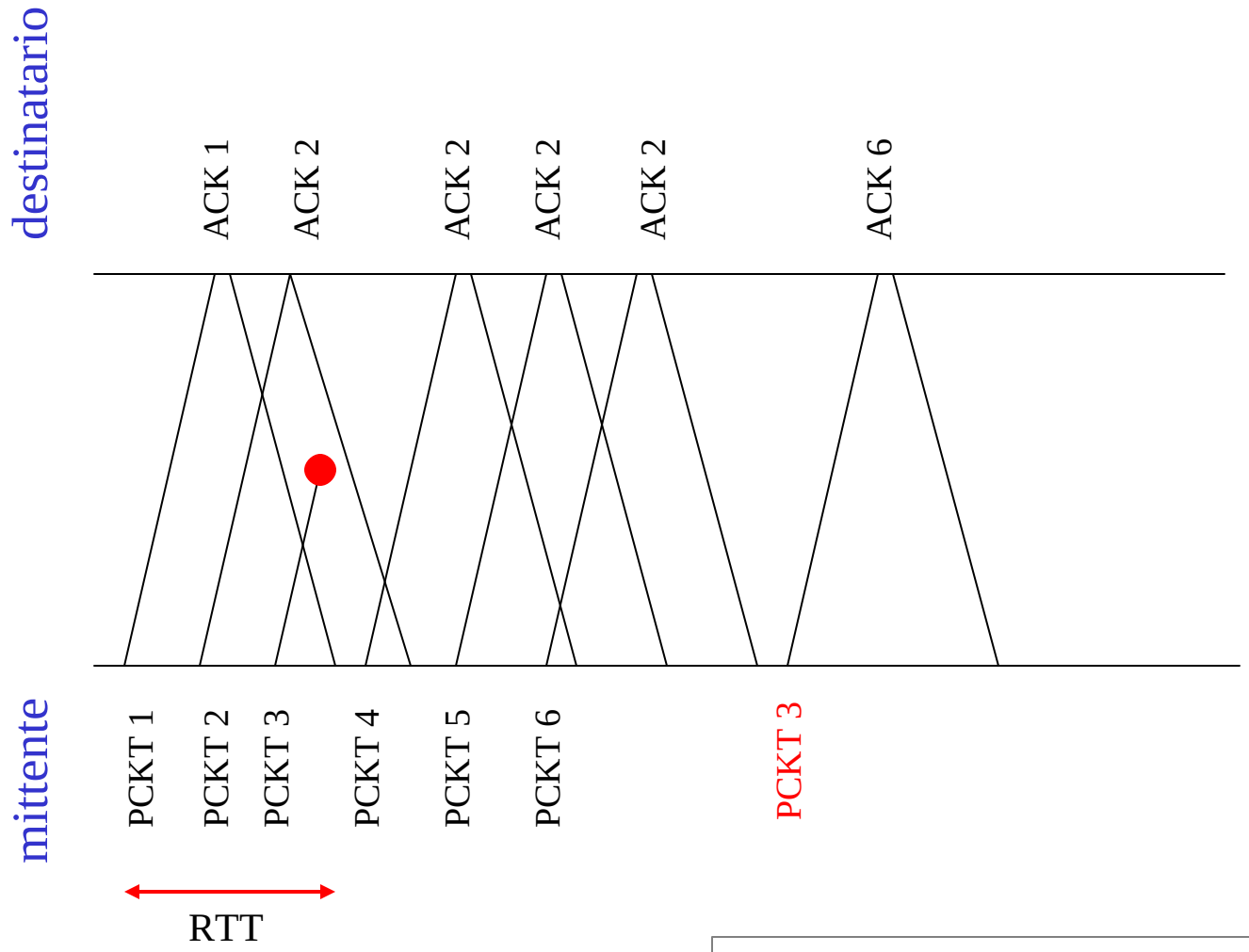
Incremento Esponenziale: CWND = CWND +MSS

Incremento lineare: CWND = CWND + (MSS * MSS) / CWND

1. Fast Retransmit

- OBIETTIVO:
  - la ritrasmissione di un pacchetto perso viene attivata senza dover attendere la scadenza del'RTY Time Out. Infatti il periodo di inattivita` durante lo scadere del timeout causa un sottoutilizzo del canale trasmissivo.
- MECCANISMO:
  - Come nel TCP standard, nel caso che un pacchetto venga ricevuto fuori ordine (per via di un cambiamento nell'ordine di consegna dei segmenti TCP al nodo ricevente o a causa della perdita di uno o piu` segmenti TCP), il ricevente rispedisce un ACK duplicato.
  - Quando il mittente rileva un ACK duplicato, puo' dedurre che l'altro end-node ha ricevuto un segmento fuori ordine e che probabilmente il pacchetto mancante e' stato perso. Per questo motivo effettua la ritrasmissione singola del segmento atteso.
  - Poiche' il segmento mancante potrebbe essere stato oggetto di un ritardo, il mittente aspetta 3 ACK duplicati prima di procedere con la ritrasmissione. (L'assunzione e' che in caso di permutazione nell'ordine di consegna, il numero di ack duplicati sia contenuto, mentre in caso di vera e propria perdita tale numero sia piu' elevato)

Scenario con Fast Retransmit



Congestion Window = 4 PCKT

2. Fast Recovery

- OBIETTIVO:
  - Ottimizzazione delle prestazioni di TCP in caso di moderata congestione. Infatti, ogni qual volta un timeout scade, il protocollo TCP standard pone CWND = 1 e passa nello stato slow start. Questo tipo di comportamento risulta eccessivamente aggressivo in caso di perdite sporadiche ed isolate di segmenti TCP.
- MECCASNISMO:
  - Dopo la ritrasmissione del segmento mancante, non si passa in slow start, ma piuttosto si rimane in congestion avoidance:
- Fast retransmit e fast recovery sono normalmente implementati insieme

Fast Recovery (cont)

- Sequenza dell'algoritmo (*):
  1. alla ricezione del terzo ack duplicato:

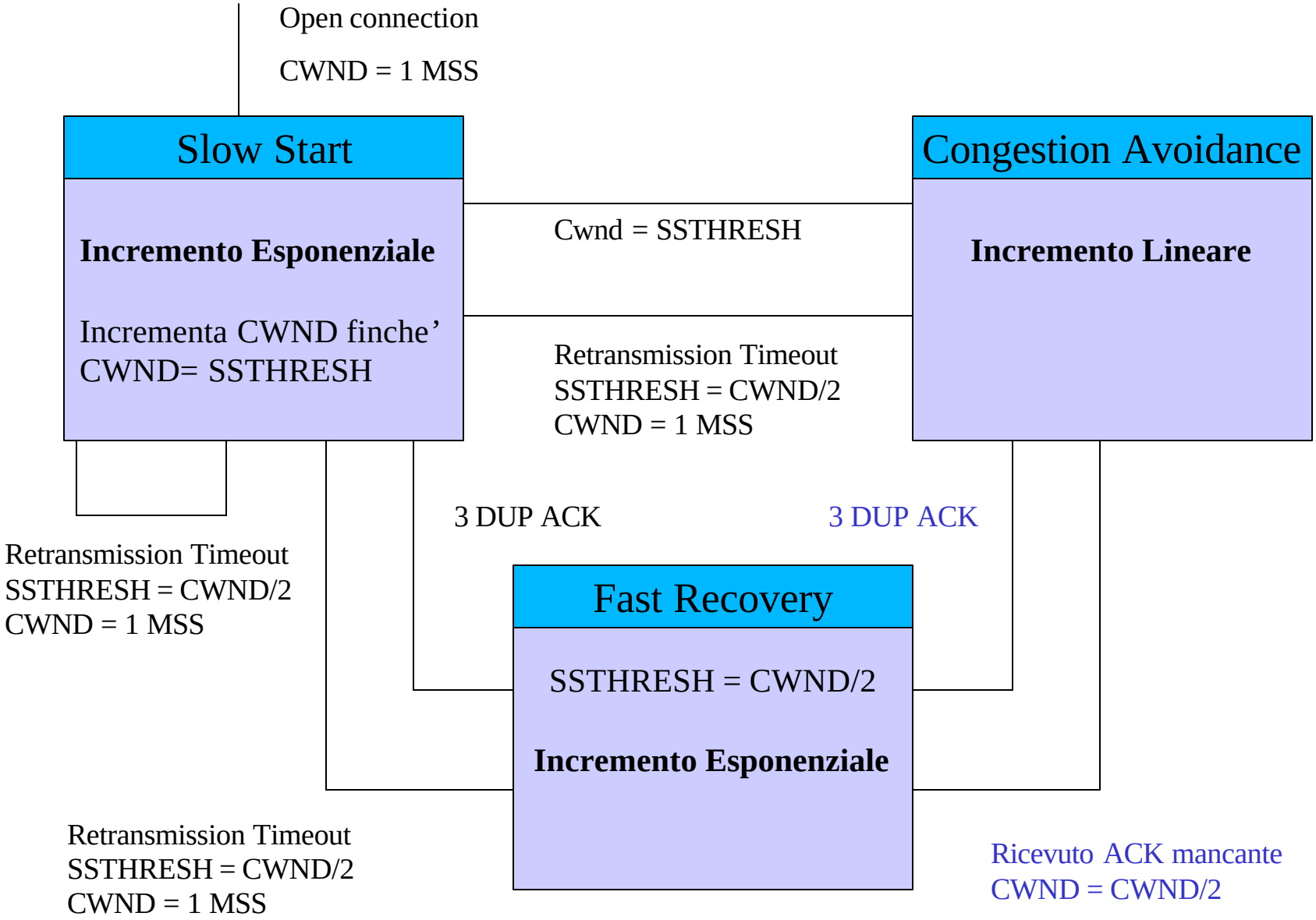
     ssthresh:= max (2*MSS, cwnd /2);
  2. Il segmento mancante viene ritrasmesso (fast retransmit)
  3. Si assume che i tre ack duplicati corrispondano a tre segmenti correttamente ricevuti: cwnd := ssthresh + 3*MSS
  4. Al primo ack ricevuto che garantisce un avanzamento della congestion window (non si tratta di un ack duplicato):

     cwnd := ssthresh

     Si assume che take ack sia quello ottenuto grazie alla ritrasmissione effettuata al passo 2. L'ack number contenuto in tale ack dovrebbe includere tutti i segmenti intermedi inviati dopo il segmento perso e prima della ricezione del primo ack duplicato.

(*) si assume che cwnd sia espresso in numero di MSS

Fast Recovery (cont)

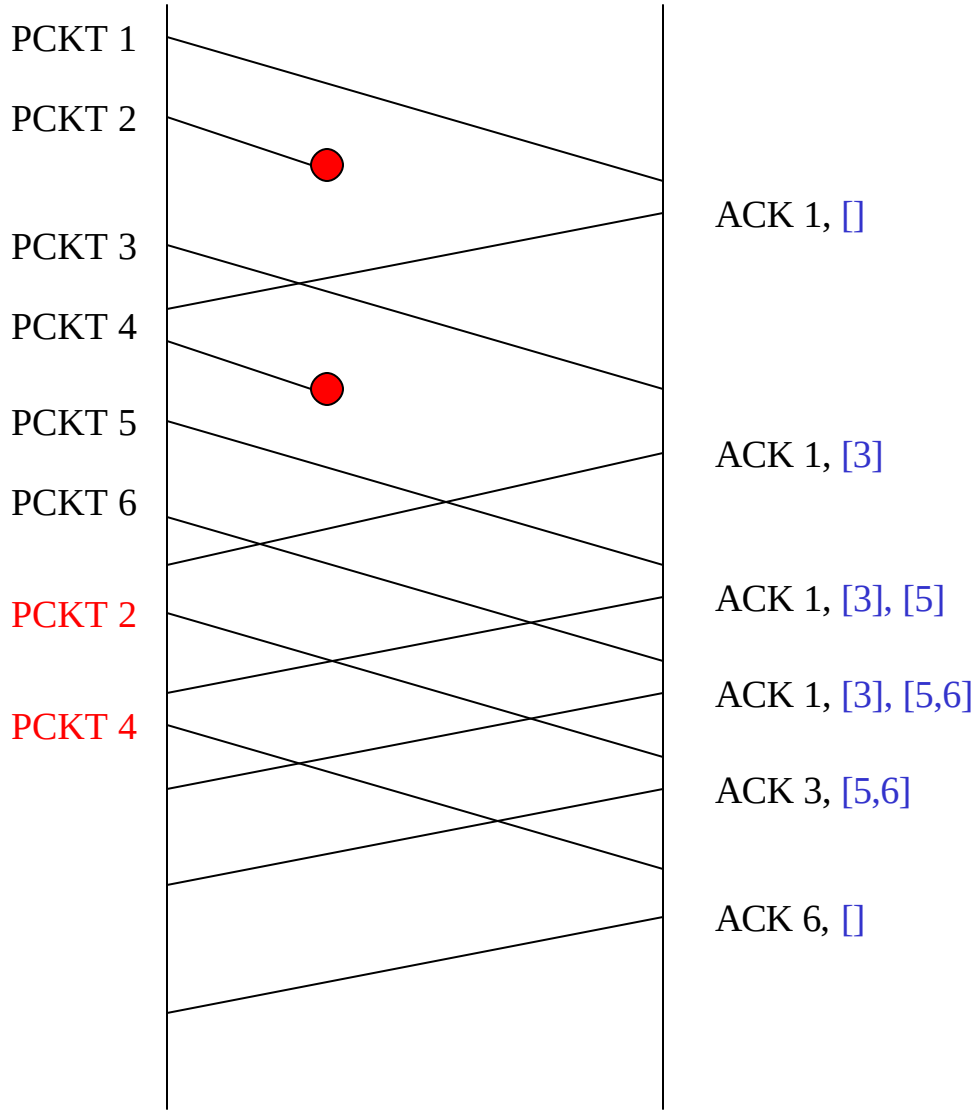


3. Selective Acknowledgment (SACK)

- TCP puo' sperimentare scarse prestazioni quando si verificano perdite multiple in una singola finestra di spedizione, poiche' in questo caso l'utilizzo ripetuto del protocollo fast recovery comporta comunque delle interruzioni della trasmissione: il mittente deve infatti ogni volta attendere la ricezione di 3 ack duplicati (o nel caso peggiore lo scadere di un timeout).
- OBIETTIVO:
  - Informare il mittente in modo esplicito dei pacchetti che NON sono stati ricevuti correttamente attraverso un meccanismo di Acknowledgement potenziato (attraverso il campo option dell'header TCP)
  - Al contrario, il protocollo TCP standard utilizza un semplice meccanismo di ACK cumulativo, il quale permette al mittente di notificare al massimo della perdita di un singolo messaggio.
  - Il mittente e' autorizzato a ritrasmettere in ordine sequenziale solo i pacchetti che sono stati persi

- La possibilita' di utilizzare l'opzione SACK viene negoziata all'apertura della connessione tramite il pacchetto SYN
- L'opzione SACK e' spedita dal ricevente per informare il mittente dei blocchi non contigui di dati ricevuti ed e' utilizzata solo in caso di presenza di blocchi di dati ricevuti in modo non contiguo (a causa della paerdita di messaggi TCP o alla ricezione in un ordine diverso da quello di invio.

Esempio: trasmissione di messaggi SACK



Nota: questo esempio mostra il comportamento del ricevente TCP che implementa il protocollo SACK, su ricezione di una data sequenza di messaggi TCP (e non rispecchia il reale meccanismo di congestion avoidance/control adottato dalla sorgente su ricezione di messaggi SACK e di ACK duplicati.

Comportamento del nodo mittente

- Il comportamento di un mittente TCP su ricezione di un messaggio SACK deve rispettare le seguenti regole:
  - Non si edve procedere alla ritrasmissione di dati se un solo SACK è ricevuto (questo per rendere il protcollo più robusto in caso di riordino di pacchetti).
  - Il principio di ritrasmissione allo scadere di un timeout deve essere rispettato, così come il meccanismo di congestion control e avoidance.
  - Uno o più messaggio NON appartenenti ad un blocco SACK possono essere ri-inviati, se il parametro cwnd lo permette (a differenza di Fast retransmit)
- In “esempio del meccanismo di acknowledgement”:
  - il mittente provvede alla tramissione di un messaggio mancante su ricezione del secondo SACK duplicato. Questo consente di partire con la ritrasmissione prima ancora di ricevere 3 ACK duplicati (come in fast restransmit).
  - Viene inviato un nuovo messaggio solo su ricezione di un nuovo SACK.

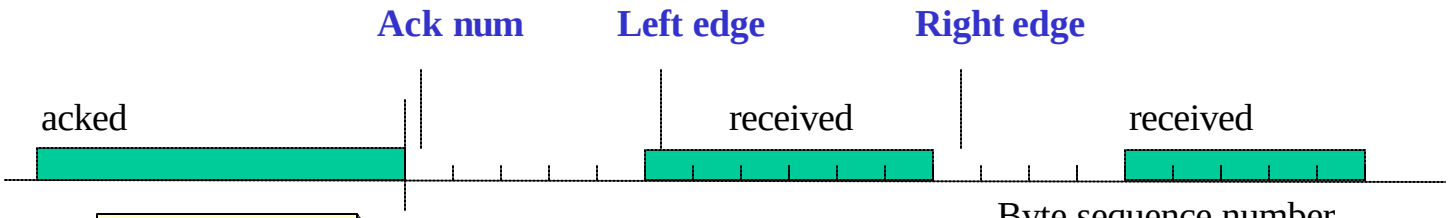
Formato dell'opzione SACK

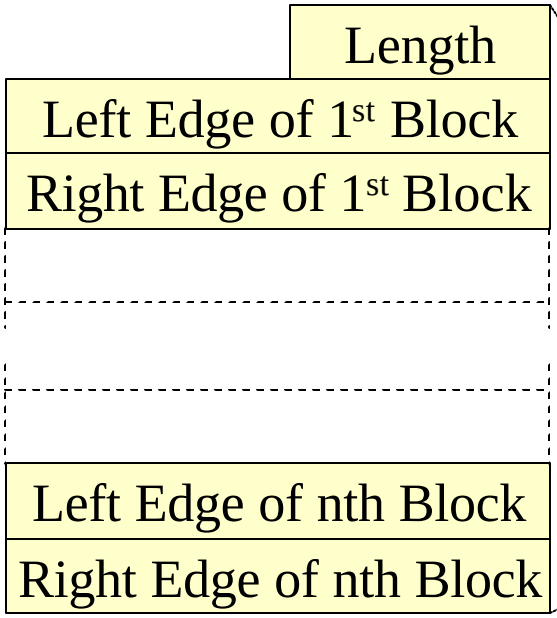
Block: blocco contiguo di byte ricevuti correttamente dal ricevente

Length: numero dei blocchi non contigui contenuti nel campo "option"

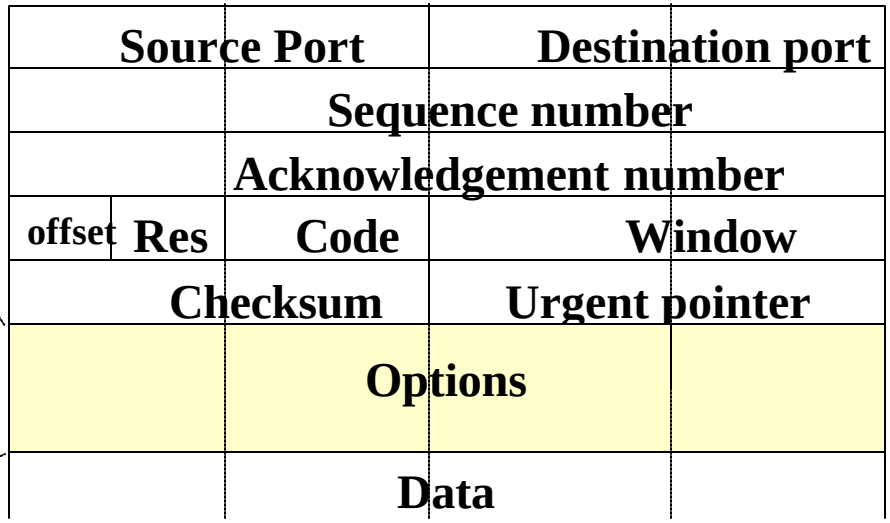
Left Edge: il numero di sequenza (32 bit) del primo byte del blocco

Right Edge: il minimo numero di sequenza (32 bit) di byte non appartenente al blocco





Formato dell'opzione SACK



Formato del header TCP

Numero max di blocchi

- Data offset: è il campo dell'intestazione TCP che indica il numero di word di 32 bit contenute nell'header TCP. Essendo tale campo di 4 bit il numero massimo di word che una intestazione può contenere è pari a 15.
- Lunghezza minima dell'intestazione si registra quando il campo "options" è assente. La lunghezza minima è pari a 5 word (20 byte). Qundi la lunghezza massima dell'opzione SACK è pari a 10 word (40 byte).
- La lunghezza dell'opzione SACK è pari a 2 + 8*n dove n è il numero di blocchi contenuti nell'opzione, quindi in numero massimo di blocchi che possono essere inclusi è pari a 4. In presenza di altre opzioni, il numero reale di blocchi può essere inferiore.

SACK: Esempi di *spedizione* di messaggi SACK

- Esempio 1: The first 4 segments are received but the last 4 are dropped.
  - The data receiver will return a normal TCP ACK segment acknowledging sequence number 7000, with no SACK option.

- Esempio 2: The first segment is dropped but the remaining 7 are received. Upon receiving each of the last seven packets, the data receiver will return a TCP ACK segment that acknowledges sequence number 5000 and contains a SACK option specifying one block of queued data:

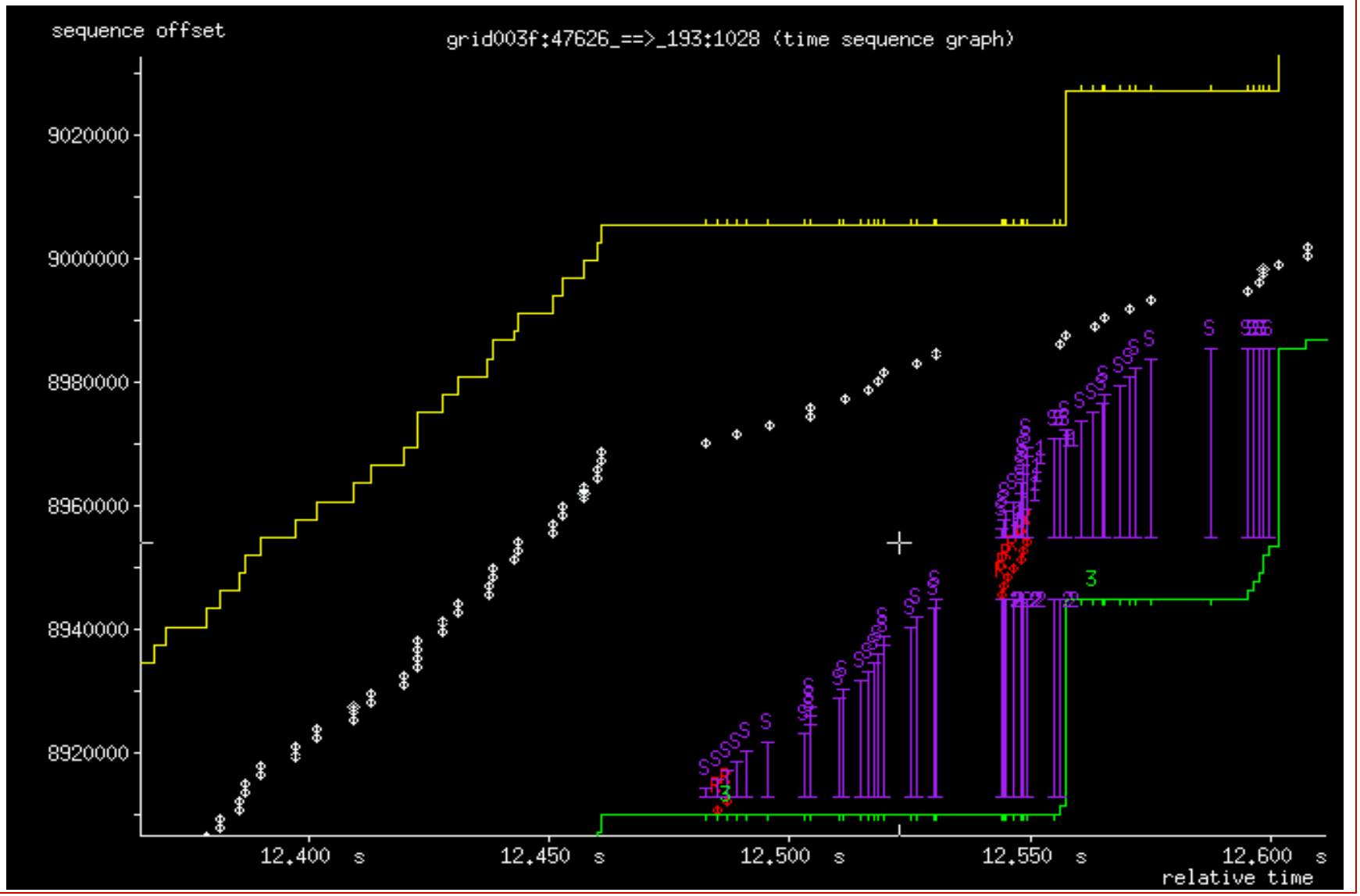
Triggering Segment	ACK	Left Edge	Right Edge
5000 (lost)			
5500	5000	5500	6000
6000	5000	5500	6500
6500	5000	5500	7000
7000	5000	5500	7500
7500	5000	5500	8000
8000	5000	5500	8500
8500	5000	5500	9000

SACK: Esempi di *spedizione* di messaggi SACK (cont)

- Esempio 3: The 2nd, 4th, 6th, and 8th (last) segments are dropped.
  - The data receiver ACKs the first packet normally. The third, fifth, and seventh packets trigger SACK options as follows:

Triggering Seg.	ACK	1st Blck Left Right	2nd Blck Left Right	3rd Blck Left Right Edge
5000	5500			
5500 (lost)				
6000	5500	6000 6500		
6500 (lost)				
7000	5500	7000 7500	6000 6500	
7500 (lost)				
8000	5500	8000 8500	7000 7500	6000 6500
8500 (lost)				

SACK: esempio del meccanismo di acknowledgement



Medoti di Congestion Avoidance avanzati

- Obiettivo: prevenire la congestione riducendo il rate di trasmissione prima che essa si verifichi
- Il meccanismo di controllo della congestione di TCP considera la rete come una scatola nera
- Usare la perdita dei pacchetti come la sola indicazione della congestione non e' sufficiente per applicazioni sensibili alla perdita e al ritardo dei pacchetti (e.g. applicazioni multimediali)
- Due strategie:
  - Router-Centric: ogni router monitora la lunghezza della propria coda e , prima che la congestione si verifichi, notifica esplicitamente o implicitamente gli end-node (e.g. ECN)
  - Host-Centric: gli end-point osservano il numero di pacchetti che riescono ad arrivare a destinazione e adattano il rate di conseguenza (e.g. TCP Vegas)

Router-Centric: Active Queue Management

- Le perdite dei pacchetti in caso di congestione sono causate dalla saturazione dei buffer all'interno dei router
- Drop Tail:
  - il router in caso di overflow elimina i pacchetti eccedenti in modo indiscriminato, non distinguendo tra i diversi flussi di pacchetti
  - sincronizzazione delle perdite tra i flussi
- Active Queue Management: gestione piu' sofisticata delle code

4. Random Early Detection (RED)

- I meccanismi di AQM differiscono in base
  - Metodo utilizzato per rilevare l'imminente congestione
  - Tipo di azione utilizzata per notificare la congestione
- Nel caso in cui la coda eccede una certa soglia, i pacchetti in arrivo vengono scartati in base ad una probabilita' P

SampleLen: lunghezza istantanea della coda
AvgLen: lunghezza media della coda
Weight: parametro compreso tra 0 e 1 che determina il peso del valore istantaneo rispetto alla media finora calcolata

AvgLen = (1 - Weight) * AvgLen + Weight * SampleLen

RED (cont)

If AvgLen <= MinThreshold

queue the packet

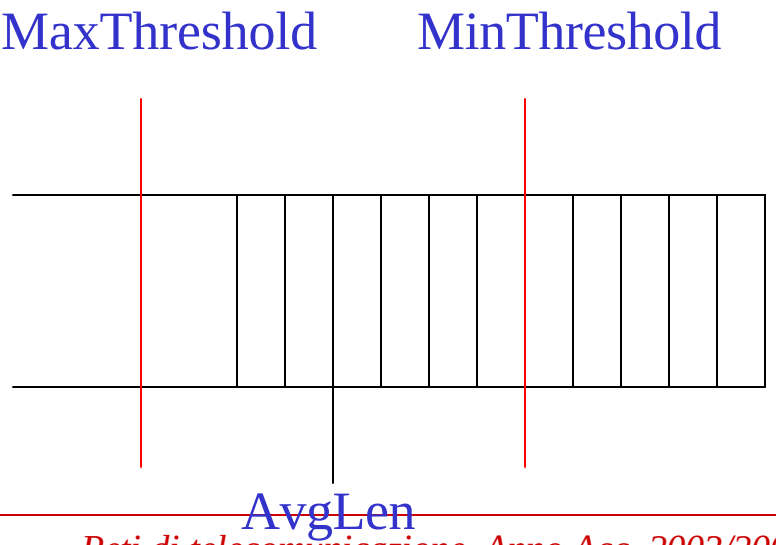
Else if MinThreshold < AvgLen < MaxThreshold

calculate probability P

drop the arriving packet with probability P

Else if MaxThreshold <= AvgLen

drop the arriving packet



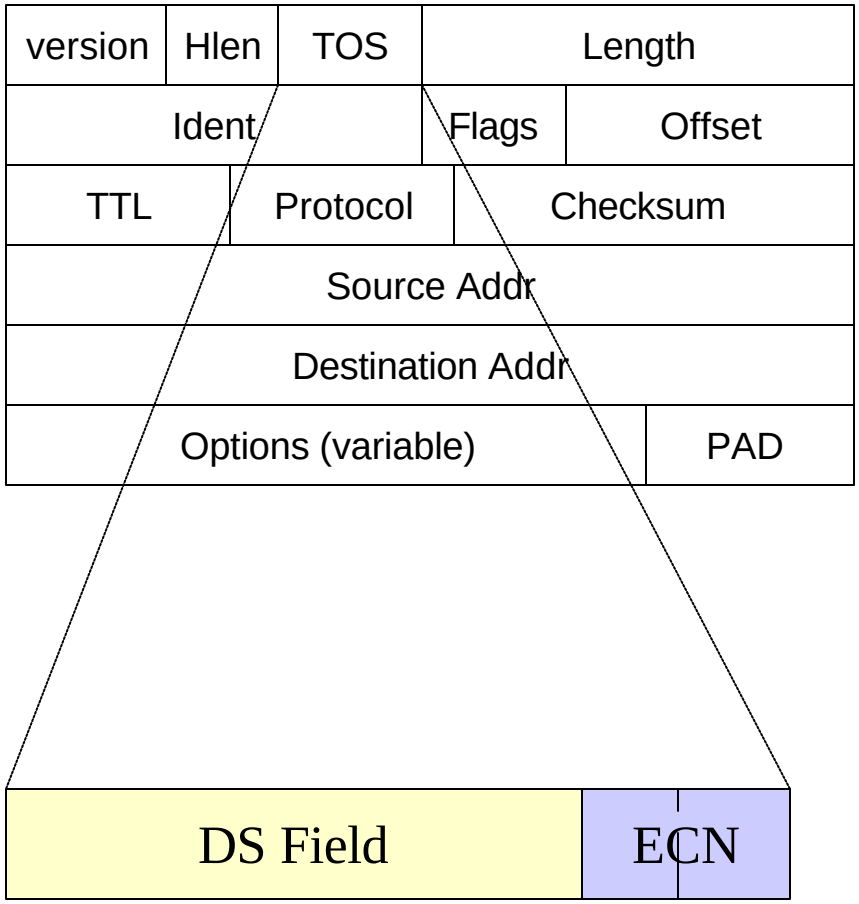
5. Explicit Congestion Notification

- La presenza di congestione viene effettuata anche in reale assenza di pacchetti persi
- Comporta una modifica sia del software di gestione dei router sia degli stack negli end-node
- L'indicazione del superamento di una soglia di allerta da parte del router viene effettuato marcando i pacchetti
- Il mittente puo' ridurre il rate di trasmissione senza subire una degradazione della prestazione dovuta alla perdita e alla conseguente ritrasmissione di un pacchetto
- ECN non richiede che tutti i router intermedi debbano neccesariamente supportare il protocollo, in altre parole, ECN può essere implementato in modo incrementale.
- ECN richiede:
  - Segnalazione a livello IP (per notificare che un dato nodo è in grado di settare i 2 bit ECN nell'intestazione IP, e per permettere ad un router di segnalare uno stato di cogestione imminente)
  - Segnalazione a livello TCP (per comunicare al mittente TCP che un messaggio ha sperimentato congestione per comunicare al ricevente TCP che il parametro cwnd è stato ridotto in risposta alla notifica del destinatario)

ECN: riduzione della congestion window

- Meccanismo di reazione alla notifica di presenza di congestione:
  - Il mittente adotta il meccanismo di fast recovery (TTHRESH = ½ CWND; CWND' = ½ CWND)
  - Il parametro CWND è ridotto solo una volta, i successivi messaggi SACK non generano ulteriori modifiche del parametro. Una volta ridotto, il parametro non viene più dimezzato fin tanto che il numero di byte pari al valore di di CWND appena prima della ricezione del messaggio SACK, non riceve il corrispondente ACK.
  - Il parametro CWND non viene modificato nel caso in cui esso sia già stato ridotto recentemente (nel precedente intervallo di tempo pari al RTT)

ECN Codepoint



Campo TOS header IP

TOS: Type Of Service

DS Field: Differentiated Service

ECN: Explicit Congestion Notification

ECN Codepoint

- Il campo CE del header IP e’ di 2 bit
- 10, 01
  - usati dal mittente pe indicare che il protocollo di trasporto puo’ utilizzare l’opzione ECN
  - i router devono trattarli come equivalenti
  - consentono al mittente di verificare che i codici non vengano cancellati dai router
- 00 : indica un pacchetto che non usa ECN
- 11 : utilizzato dal router per notificare la presenza di congestione

Conclusioni

- Il successo di Internet si basa sulla capacita' di TCP di assicurare un servizio di trasporto efficiente, robusto e adattivo
- Le esigenze delle applicazioni e degli ambienti attuali sono molto diverse rispetto a quelle che hanno portato alla definizione dei sui mecchanismi originali
  - Wires Vs Wireless
  - Durata sessione
  - Reti ad alta velocita'
- Fairness: modifiche che portano ad un comportamento troppo aggressivo da parte del mittente porterebbe al collasso della rete

Bibliografia

1. RFC 2001 (1997): TCP SLOW START, CONGESTION AVOIDANCE, FAST RETRANSMIT, FAST RECOVERY
2. RFC 1072: TCP Extensions for Long-Delay Paths
3. RFC 2018 (1996): TCP SELECTIVE ACKNOWLEDGMENT OPTIONS
4. S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, 1(4), August 1993.
5. RFC 3168 (2001): The Addition of Explicit Congestion Notification (ECN) to IP
6. TCP VEGAS: END TO END CONGESTION AVOIDANCE ON A GLOBAL INTERNET (L.S.Brakmo and L.L.Peterson, 1995)
7. END TO END CONGESTION DETECTION AND AVOIDANCE IN WIDE AREA NETWORKS (L.S.Brakmo, 1996)