Reti: Esercizi TCP

Aymane Chabbaki

III semestre 2019/2020

Indice

1 Esame 12-6-2019

- Un client HTTP deve trasferire da un server un file di 56200 bytes.
- Gli host (client e server) hanno i seguenti indirizzi IP 128.122.15.171/26 e 128.122.15.16/24, la tecnologia usata per la trasmissione e l'accesso alla rete è Ethernet commutata a 100 Mbit/s dal lato client e 1 Gbit/s dal lato server.
- La rete Ethernet non è congestionata e si misura una latenza (ritardo tra l'inizio della trasmissione lato server e l'inizio della ricezione lato client) Te = 1.0 ms.
- L'overhead di Ethernet è pari a 36 byte (includendo tutti i campi e anche l'inter-packet-gap).

Domande:

1. La consegna dei pacchetti IP avviene in modo diretto o indiretto? Perché?
2. Che valore assume MSS (Maximum Segment Size di TCP) in assenza di opzioni per TCP e con il normale header IP (anche qui nessuna estensione o opzione viene usata)?
3. Quanti segmenti (e quindi pacchetti IP) verranno trasmessi sulla rete?
4. Che dimensione ha l'ultimo segmento dati della connessione?
5. Si mostrino i segmenti scambiati per l'apertura della connessione TCP conseguente al comando di "get" da parte del client FTP, scegliendo opportunamente le porte (port number) di TCP dal lato client e dal lato server.
6. Si mostri l'intero scambio di segmenti TCP per trasferire il file, calcolando il tempo di trasferimento e il throughput ottenuto a livello HTTP, cioè relativo ai byte utili per l'applicazione; la connessione viene chiusa dal server con un RST appena ricevuto l'ultimo ACK.

La rete perde il 12° segmento (contato ordinatamente nella segmentazione del file: i segmenti di apertura/chiusura e le ritrasmissioni non contano). Dato il basso valore di RTT, il Retransmission Timeout (RTO) di TCP è fissato al minimo ammesso dal sistema operativo: RTO = 120 ms.

7. Si mostri nuovamente l'intero scambio di segmenti tra client e server calcolando anche il tempo di trasmissione del file e il throughput come al punto 6.

Risposte:

1. La consegna dei pacchetti IP avviene in modo indiretto, in quanto i due indirizzi IP non appartengono alla stessa rete:

$$\text{IP } 128.122.15.171/26 : \begin{cases} \text{IP} & : 10000000.01111010.00001111.10101011 \ (128.122.15.171) \\ \text{SM} & : 11111111.11111111.11111111.11000000 \ (255.255.255.192) \\ \text{Net} & : 10000000.01111010.00001111.10000000 \ (128.122.15.128) \end{cases}$$

$$\text{IP } 128.122.15.16/24 : \begin{cases} \text{IP} & : 10000000.01111010.00001111.00010000 \ (128.122.15.16) \\ \text{SM} & : 11111111.11111111.11111111.00000000 \ (255.255.255.0) \\ \text{Net} & : 10000000.01111010.00001111.00000000 \ (128.122.15.0) \end{cases}$$

   Visto che si hanno due indirizzi di rete diversi (128.122.15.128 e 128.122.15.0) si ha che:

   - Il Client (/26) comunica indirettamente con il Server.
   - Il Server (/24) comunica direttamente con il Client in quanto la sua rete contiene anche la sottorete del Client.

   Questo è vero e funziona sono nel caso in cui Client e Server siano collegati ad uno Switch (se non ci fosse uno Switch, non riuscirebbero a comunicare).

2. MSS assume il valore:

$$\text{MSS} = 1500 - 20 - 20 = 1460 \text{ Byte}$$

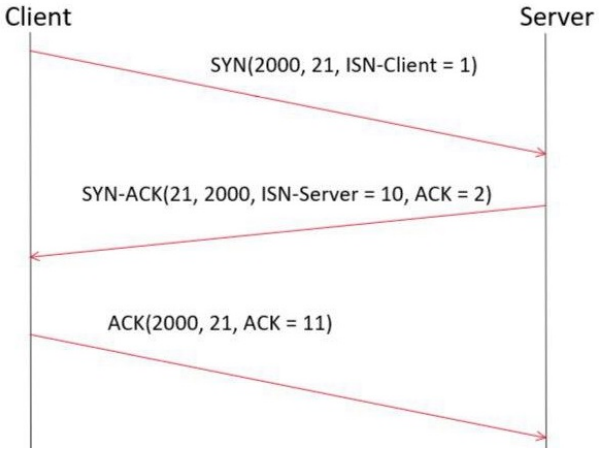
3. In rete vengono trasmessi (MSS = 1424 Byte per l'overhead):

$$N_s = \frac{56200 \text{ Byte}}{1424 \text{ Byte}} = 40 \text{ segmenti}$$

4. L'ultimo segmento ha dimensione:

$$56200 \text{ Byte} - (39 \cdot 1424) \text{ Byte} = 56200 \text{ Byte} - 55536 \text{ Byte} = 664 \text{ Byte}$$

5. TCP utilizza un meccanismo a 3 step per aprire una connessione tra sorgente e destinatario. Questa prevede che il client mandi un pacchetto speciale (con il flag SYN settato ad 1) al Server. Una volta ricevuto dal Server questo decide se accettare la domanda di connessione o rifiutarla. Se la accetta, allora risponde anch'esso con un pacchetto SYN (non è un semplice SYN in quanto oltre al flag SYN ha anche il flag ACK settato così da notificare che ha ricevuto la richiesta precedentemente inviata dal Client). A questo punto il Client invia un ACK per confermare la ricezione del messaggio del Server e finito questo procedimento, esiste una connessione TCP tra il Client e il Server.



Il primo pacchetto SYN inviato dal Client contiene:

- Porta sorgente: 2000 (generato casualmente)
- Porta destinazione: 21 (porta FTP)
- Sequence Numerber: 1 (generato casualmente)

Il pacchetto SYN-ACK inviato dal Server contiene:

- Porta sorgente: 21 (porta FTP che fornisce il servizio richiesto)
- Porta destinazione: 2000 (porta del Client che ha richiesto il servizio)
- Sequence Numerber: 10 (generato casualmente)
- ACK: 2 (Sequence number inviato dal Client incrementato di 1 per specificare a quale pacchetto fa riferimento e ha ricevuto)

L'ultimo pacchetto inviato dal Client:

- Porta sorgente: 2000
- Porta destinazione: 21
- Sequence Numerber: 2
- ACK: 11 (Sequence number inviato dal Server incrementato di 1 per specificare a quale pacchetto fa riferimento e ha ricevuto)

6. Trasferimento file:

- Prima di poter disegnare il diagramma bisogna capire se si è nel caso di RTT $> T_t$ oppure $T_t >$ RTT:
  - RTT $= 2 \cdot T_e = 2 \cdot 1ms = 2ms$
  - $T_t = \dfrac{1500 \text{ Byte}}{100 \text{ Mbit/s}} = \dfrac{1500 \cdot 8 \text{ bit}}{100 \cdot 10^6 \text{ bit/s}} = 120 \cdot 10^{-6} \text{ bit} = 0.00012s = 0.12ms$
  - Dunque siamo nel caso in cui RTT $> T_t$
  - In un RTT vengono trasmessi al massimo:

$$\frac{\text{RTT}}{T_t} = \frac{2 \text{ ms}}{120\mu s} = 16.67 = 16 \text{ segmenti}$$

  - Questo non crea problemi (cioè i segmenti non vengono inviati back-to-back) in quanto al 6° RTT, i segmenti che vengono trasmessi sono solo 10.
  - Se dovessi inviare più di 10 segmenti, ad esempio 30, si ha che:
    - Dal segmento 32 al 48 si è in Slow Start "normale"
    - Dal 49 in poi, i segmenti vengono mandati back-to-back in quanto si satura completamente il numero di segmenti trasmissibili in un RTT, anche se si è ancora in Slow Start.

RTT	**CWND**	T_w
1	1	$[1]$
2	2	$[2, 3]$
3	4	$[4, 5, 6, 7]$
4	8	$[8, \ldots, 15]$
5	16	$[16, \ldots, 31]$
6	32	$[32, \ldots, 40]$

- Il trasferimento del file:

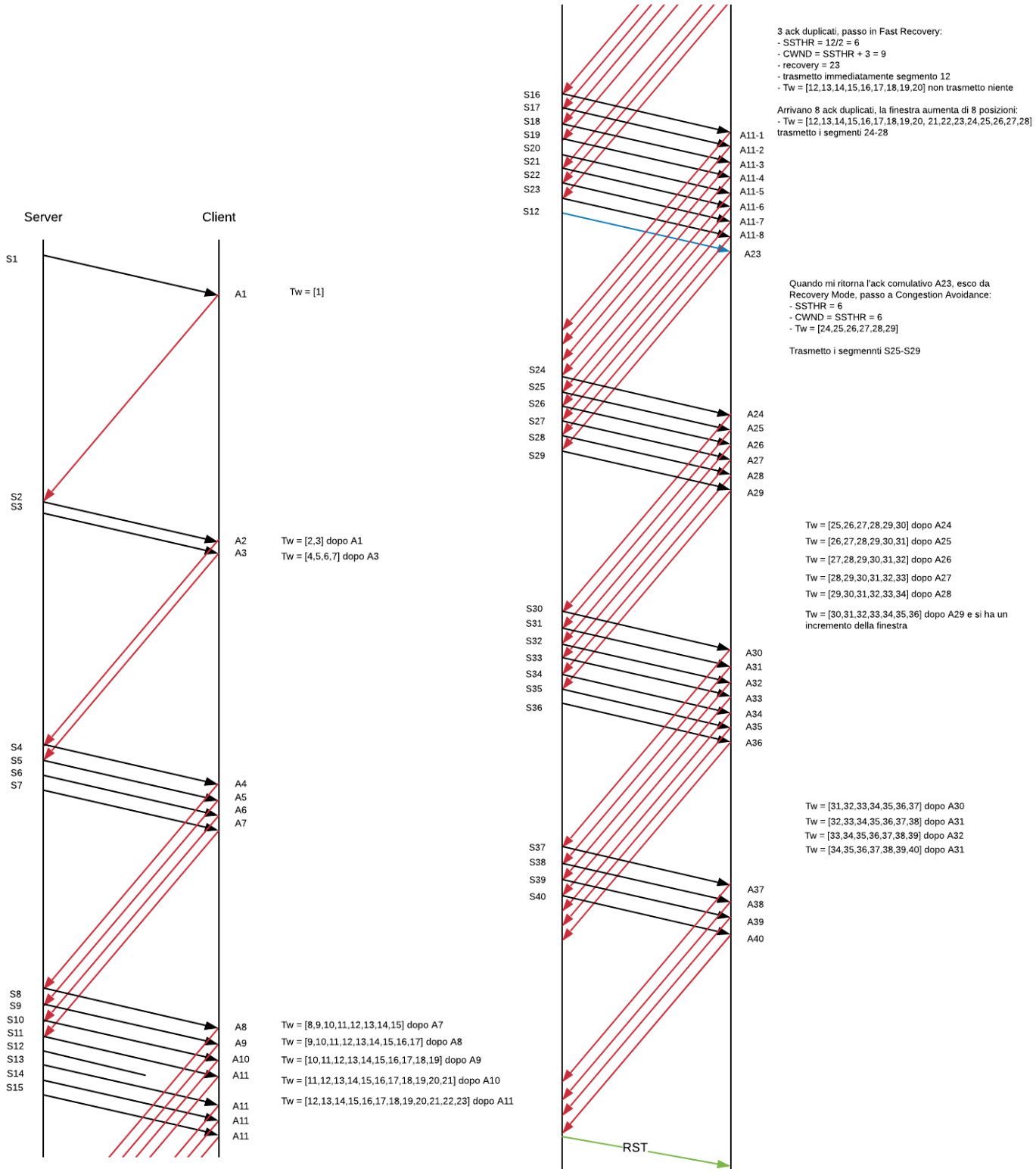
$$T_F = 7 \cdot RTT = 7 \cdot (2 \cdot T_e) = 14 \cdot 1\,ms = 14\,ms$$

- Throughput:

$$T_h = \frac{56200 \text{ Byte}}{T_F} = \frac{449600 \text{ bit}}{14ms} = \frac{449600 \text{ bit}}{0.014s} = 32114285.7 \text{ bit/s} = 30.6 \text{ Mbit/s}$$

7. Trasmissione con il 12° pacchetto perso:

   - RTO = $120ms$, SSTHR = 64 Kbyte = 45 segmenti.
   - Il trasferimento parte con Slow Start con CWND = 1. Dopo un RTT, la finestra di trasmissione raddoppia e possono partire i pacchetti S2 e S3. Dopo un altro RTT, la finestra di trasmissione raddoppia nuovamente e vengono inviati i pacchetti S3,S4,S5,S6,S7.
   - Al terzo RTT, la finestra di trasmissione è di 8 segmenti, i segmenti S8-S11 vengono trasmessi correttamente; il segmento S12 viene perso.
   - Vengono trasmessi successivamente i segmenti S13-S15, ma visto che il Client è fermo ancora al segmento S11 e si aspetta S12, manda 3 ACK duplicati.
   - Questo fa si che si passi in Fast Recovery (SSTHR = CWND/2 = 12/2 = 6, CWND = SSTHR + 3 = 9, Tw = [12,13,14,15,16,17], recovery = 23). Viene ritrasmesso il segmento S12 (non vengono trasmessi altri segmenti in quanto la finestra di trasmissione non contiene segmenti nuovi da trasmettere).
   - I segmenti S16-S23 vengono ricevuti dal Client, ma ritorna per ognuno di questi segmenti un ACK duplicato notificando al Server che è in attesa di S12. Questi 8 ACK duplicati fanno si che la dimensione della finestra di trasmissione aumenti di 8 (Tw = [12,...,28]). La finestra di trasmissione mi permette di trasmettere i segmenti S24-S28.
   - Il Client riceve il segmento S12 e trasmette un ACK cumulativo (A23) al Server, notificando che ha ricevuto correttamente fino al segmento S23.
   - Ricevuto questo ACK, il Server esce da Fast Recovery e passa a Congestion Avoidance (CWND = SSTHR = 6, Tw = [24,...,29]) e posso trasmettere il segmento S29.
   - La trasmissione continua con Congestion Avoidance fino alla trasmissione e recezione dell'ACK del segmento S40 (da notare che quando il Server riceve l'ACK A29, la dimensione finestra di trasmissione aumenta da 5 a 6).
   - Il Server una volta ricevuto l'ACK A40 relativo al segmento S40, trasmette un ultimo segmento (con il flag RST settato) così da terminare la connessione con il Client.


Server
Client
S1
A1
Tw = [1]
S2
S3
A2
A3
Tw = [2,3] dopo A1
Tw = [4,5,6,7] dopo A3
S4
S5
S6
S7
A4
A5
A6
A7
S8
S9
S10
S11
S12
S13
S14
S15
A8
A9
A10
A11
A11
A11
A11
Tw = [8,9,10,11,12,13,14,15] dopo A7
Tw = [9,10,11,12,13,14,15,16,17] dopo A8
Tw = [10,11,12,13,14,15,16,17,18,19] dopo A9
Tw = [11,12,13,14,15,16,17,18,19,20,21] dopo A10
Tw = [12,13,14,15,16,17,18,19,20,21,22,23] dopo A11
S16
S17
S18
S19
S20
S21
S22
S23
S12
A11-1
A11-2
A11-3
A11-4
A11-5
A11-6
A11-7
A11-8
A23
3 ack duplicati, passo in Fast Recovery:
- SSTHR = 12/2 = 6
- CWND = SSTHR + 3 = 9
- recovery = 23
- trasmetto immediatamente segmento 12
- Tw = [12,13,14,15,16,17,18,19,20] non trasmetto niente
Arrivano 8 ack duplicati, la finestra aumenta di 8 posizioni:
- Tw = [12,13,14,15,16,17,18,19,20, 21,22,23,24,25,26,27,28]
trasmetto i segmenti 24-28
Quando mi ritorna l'ack comulativo A23, esco da Recovery Mode, passo a Congestion Avoidance:
- SSTHR = 6
- CWND = SSTHR = 6
- Tw = [24,25,26,27,28,29]
Trasmetto i segmennti S25-S29
S24
S25
S26
S27
S28
S29
A24
A25
A26
A27
A28
A29
Tw = [25,26,27,28,29,30] dopo A24
Tw = [26,27,28,29,30,31] dopo A25
Tw = [27,28,29,30,31,32] dopo A26
Tw = [28,29,30,31,32,33] dopo A27
Tw = [29,30,31,32,33,34] dopo A28
Tw = [30,31,32,33,34,35,36] dopo A29 e si ha un incremento della finestra
S30
S31
S32
S33
S34
S35
S36
A30
A31
A32
A33
A34
A35
A36
Tw = [31,32,33,34,35,36,37] dopo A30
Tw = [32,33,34,35,36,37,38] dopo A31
Tw = [33,34,35,36,37,38,39] dopo A32
Tw = [34,35,36,37,38,39,40] dopo A31
S37
S38
S39
S40
A37
A38
A39
A40
RST

2 Esame 11-01-2019

- Consideriamo una applicazione web basata su http 1.1. Il browser del Client consente l'apertura di 2 connessioni TCP persistenti in parallelo.
- Client e Server sono connessi a una stessa subnet IP, realizzata su una LAN Ethernet a 100 Mbit/s.
- Il tempo di propagazione tra Client e Server, inclusi gli switch Ethernet, è approssimativamente costante e pari a $200\mu s$.
- Entrambe le Receiver Windows (RCWND) sono pari a 64 KByte.
- Durante una sessione, il Client richiede il trasferimento di 5 piccoli oggetti (file) di dimensioni rispettivamente:
  - O1 = 4500 Byte
  - O2 = 11400 Byte
  - O3 = 7800 Byte
  - O4 = 9400 Byte
  - O5 = 13200 Byte
- Si assuma che le connessioni TCP si comportino come se fossero appena state aperte.

Domande:

1. Come viene suddiviso il trasferimento dei 5 oggetti sulle 2 connessioni? Esiste una soluzione unica oppure sceglie il browser?
2. Mostrare in un diagramma lo scambio dei segmenti su una delle due connessioni a scelta, chiamiamola C.
3. La perdita di pacchetti su una connessione influenza il comportamento dell'altra?
4. Mostrare in un diagramma lo scambio dei segmenti sulla connessione C nel caso in cui la LAN scarti il 7° pacchetto trasmesso.
5. Calcolare il tempo di trasferimento totale dei cinque oggetti in assenza di perdite (bisogna tenere in conto entrambe le connessioni, ovviamente). Si trascurino gli header di http e altre informazioni che il server potrebbe aggiungere ai 5 file; si deve invece considerare l'overhead introdotto da TCP/IP e da Ethernet. Si consideri per quest'ultimo un header equivalente (incluso l'Inter Packet Gap) di 38 Byte.
6. Supponiamo ora che il sistema operativo consenta l'apertura di 1 sola connessione e non di 2; ricalcolare il tempo di trasmissione totale dei 5 oggetti.
7. Commentare alla luce dello scenario dato i risultati ottenuti ai punti 5 e 6.

Risposte:

1. Il trasferimento è gestito dal Browser, che decide autonomamente come trasferire i pacchetti usando le due connessioni a disposizione.
2. Dati utili per la risoluzione dell'esercizio:
   - I 5 file da trasferire hanno dimensione totale:
   $$D = (4500 + 11400 + 7800 + 9400 + 13200)\text{ Byte} = 46300\text{ Byte}$$
   - MSS = $1500 - 20 - 20 = 1460$ Byte
   - Il numero di segmenti da trasferire:
   $$S_t = \frac{46300\text{ Byte}}{1460\text{ Byte}} = 31.7 = 32\text{ segmenti}$$

- L'ultimo segmento ha dimensione:

$$46300 - (31 \cdot 1460) = 46300 - 45260 = 1040 \text{ Byte}$$

- Calcolo il tempo di trasmissione:

$$T_t = \frac{8 \cdot 1500 \text{ bit}}{100 \cdot 10^6 \text{ bit/s}} = 0.00012s = 120\mu s$$

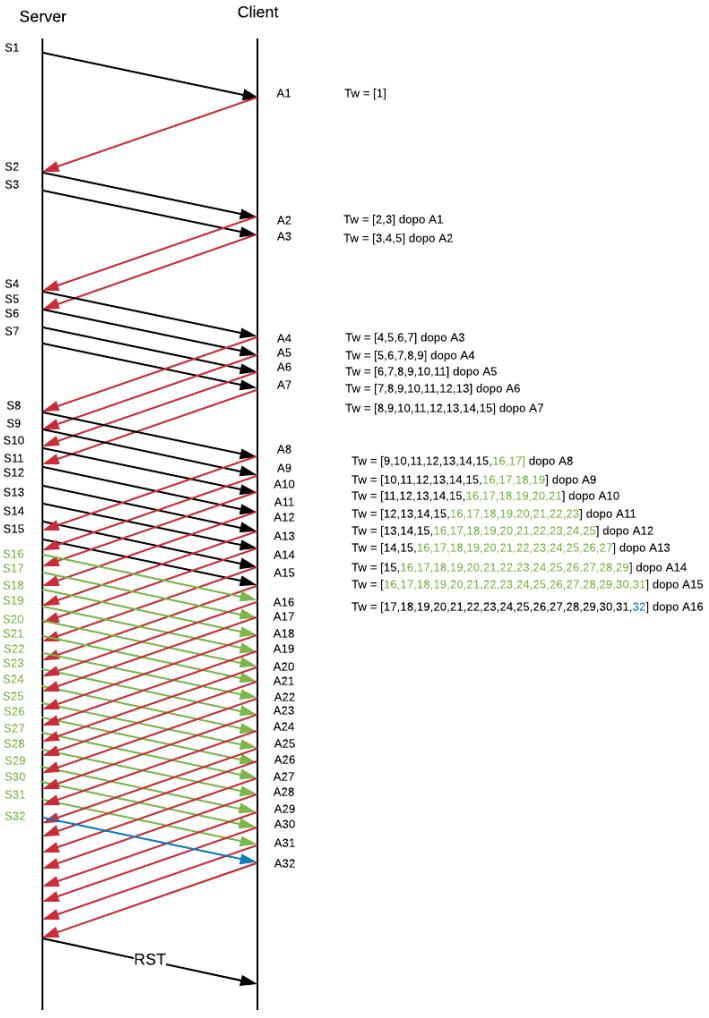
- Sapendo che il tempo di propagazione tra Client e Server è pari a $200\mu s$ e che:

$$\text{RTT} = 2 \cdot 200\mu s = 400\mu s$$

- Siamo nel caso in cui RTT $> T_t$:
  - In un RTT vengono trasmessi al massimo:

$$\frac{\text{RTT}}{T_t} = \frac{400\mu s}{120\mu s} = 3.33 = 3 \text{ segmenti}$$

  - RCWND $= \dfrac{64 \text{ KByte}}{1460 \text{ Byte}} = \dfrac{65536 \text{ Byte}}{1460 \text{ Byte}} = 45$ segmenti
  - SSTHR = RCWND = 45 segmenti
  - RTO $= 1s$
  - CWND $= 1$
  - Scambio dei pacchetti:

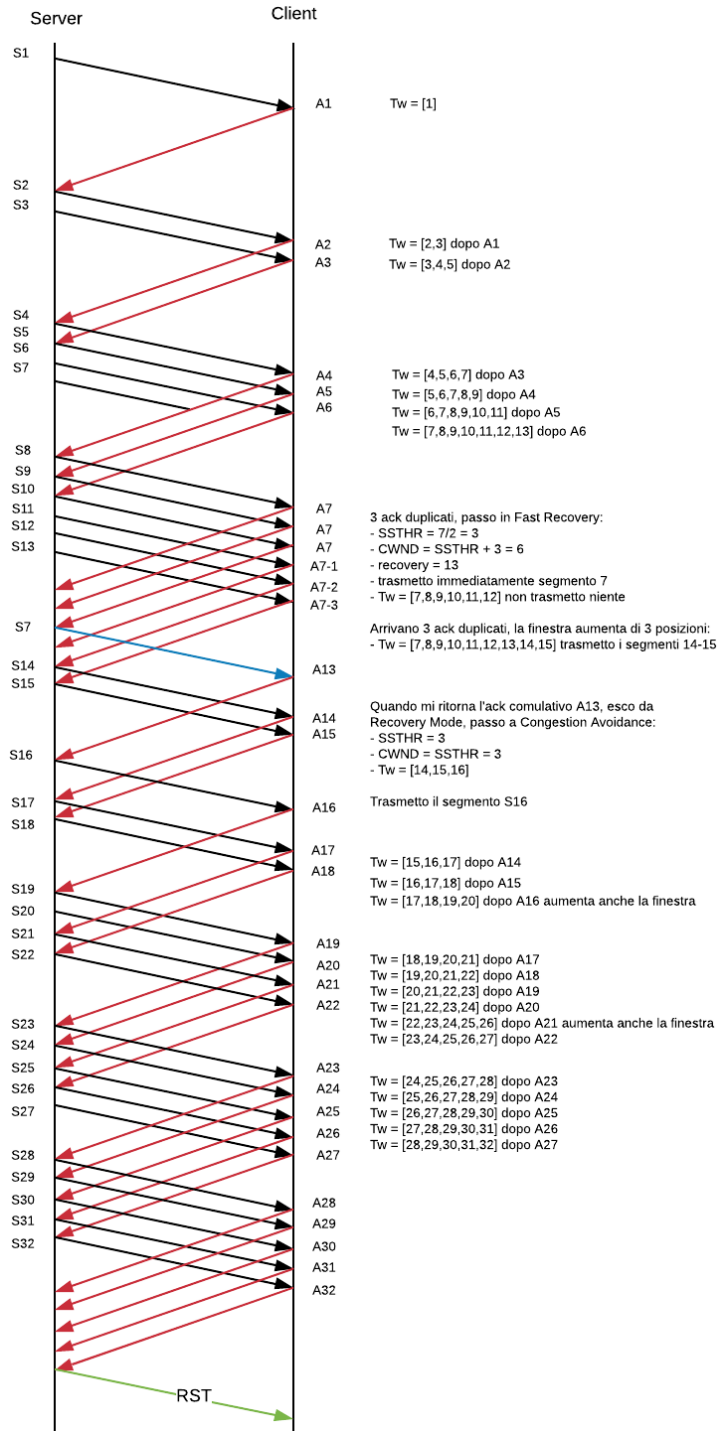


– Tempo di trasmissione:

$$T = \frac{8 \cdot (24 \cdot 1500 + 1080) \text{ bit}}{100 \cdot 10^6 \text{ bit/s}} + 3 \text{ RTT} = 2966.4\mu s + 1200\mu s = 4166.4\mu s = 4.2ms$$

3. Non influenza direttamente, quello che succede se una connessione perde pacchetti è che causa un ritardo all'altra connessione, in quanto il traffico non può transitare contemporaneamente nelle due connessioni, sennò si rischiano delle collisioni.

4. Nel caso venga perso il 7° pacchetto:



5. Complicato e non visto a lezione.

6. Il tempo è uguale a quello calcolato nell'esercizio 2.

3 Esame 08-01-2018

- Un client HTTP deve trasferire da un server un file di 42800 Bytes.
- Gli host (client e server) hanno i seguenti indirizzi IP 130.122.15.171/24 e 130.122.1.4/24, la tecnologia usata per la trasmissione e l'accesso alla rete è Ethernet commutata a 100 Mbit/s dal lato client e 1 Gbit/s dal lato server.
- La rete Ethernet non è congestionata e si misura una latenza (ritardo tra l'inizio della trasmissione lato server e l'inizio della ricezione lato client) Te = $1.2ms$.
- L'overhead di Ethernet è pari a 36 Byte (includendo tutti i campi e anche l'inter-packet-gap).

Domande:

1. La consegna dei pacchetti IP avviene in modo diretto o indiretto? Perché?
2. Che valore assume MSS (Maximum Segment Size di TCP) in assenza di opzioni per TCP e con il normale header IP (anche qui nessuna estensione o opzione viene usata)?
3. Quanti segmenti (e quindi pacchetti IP) verranno trasmessi sulla rete?
4. Che dimensione ha l'ultimo segmento dati della connessione?
5. Si mostrino i segmenti scambiati per l'apertura della connessione TCP conseguente al comando di "get" da parte del client FTP, scegliendo opportunamente le porte (port number) di TCP dal lato client e dal lato server.
6. Si mostri l'intero scambio di segmenti TCP per trasferire il file, calcolando il tempo di trasferimento e il throughput ottenuto a livello HTTP; la connessione viene chiusa dal server con un RST appena ricevuto l'ultimo ACK.

La rete perde il 15° segmento (contato ordinatamente nella segmentazione del file: i segmenti di apertura/chiusura e le ritrasmissioni non contano). Dato il basso valore di RTT, il Retransmission Timeout (RTO) di TCP è fissato al minimo ammesso dal sistema operativo: RTO = $180ms$.

7. Si mostri nuovamente l'intero scambio di segmenti tra client e server calcolando anche il tempo di trasmissione del file e il throughput come al punto 6.

Risposte:

1. La consegna dei pacchetti IP avviene in modo indiretto, in quanto i due indirizzi IP non appartengono alla stessa rete:

$$\text{IP } 130.122.15.171/24 : \begin{cases} \text{IP} & : 10000010.01111010.00001111.10101011 \ (130.122.15.171) \\ \text{SM} & : 11111111.11111111.11111111.00000000 \ (255.255.255.0) \\ \text{Net} & : 10000010.01111010.00001111.00000000 \ (130.122.15.0) \end{cases}$$

$$\text{IP } 130.122.1.4/24 : \begin{cases} \text{IP} & : 10000010.01111010.00000001.00000100 \ (130.122.1.4) \\ \text{SM} & : 11111111.11111111.11111111.00000000 \ (255.255.255.0) \\ \text{Net} & : 10000010.01111010.00000001.00000000 \ (130.122.1.0) \end{cases}$$

   Visto che si hanno due indirizzi di rete diversi (130.122.1.0 e 130.122.15.0) allora Client e Server non comunicano direttamente, ma dovranno comunicare tramite un router.

2. In assenza di opzioni per TCP e con il normale header IP, MSS assume valore:

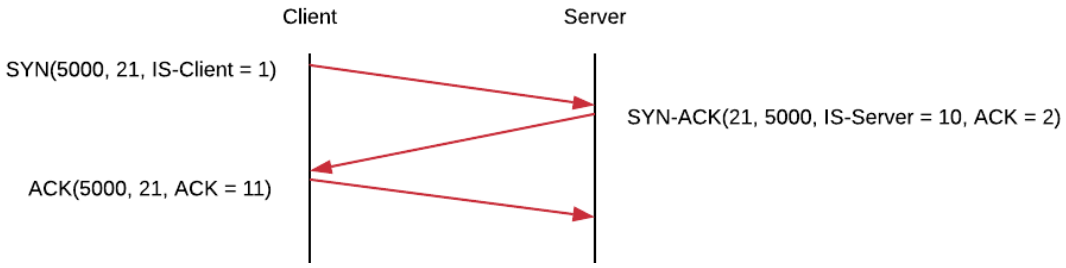
$$\text{MSS} = \text{MTU} - 20 \text{ Byte} - 20 \text{ Byte} = 1500 \text{ Byte} - 40 \text{ Byte} = 1460 \text{ Byte}$$

3. Visto che abbiamo un overhead di 36 Byte, si ha che MSS = 1424 Byte; dunque il numero di segmenti che verranno trasmessi è:

$$S_F = \frac{42800 \text{ Byte}}{1424 \text{ Byte}} = 30.05 = 31 \text{ segmenti}$$

4. L'ultimo pacchetto ha dimensione: $42800 - (30 \cdot 1424) = (42800 - 42720) = 80$ Byte

5. TCP utilizza un meccanismo a 3 step per aprire una connessione tra sorgente e destinatario. Questa prevede che il client mandi un pacchetto speciale (con il flag SYN settato ad 1) al Server. Una volta ricevuto dal Server questo decide se accettare la domanda di connessione o rifiutarla. Se la accetta, allora risponde anch'esso con un pacchetto SYN (non è un semplice SYN in quanto oltre al flag SYN ha anche il flag ACK settato così da notificare che ha ricevuto la richiesta precedentemente inviata dal Client). A questo punto il Client invia un ACK per confermare la ricezione del messaggio del Server e finito questo procedimento, esiste una connessione TCP tra il Client e il Server.



Il primo pacchetto SYN inviato dal Client contiene:

- Porta sorgente: 5000 (generato casualmente)
- Porta destinazione: 21 (porta FTP)
- Sequence Numerber: 1 (generato casualmente)

Il pacchetto SYN-ACK inviato dal Server contiene:

- Porta sorgente: 21 (porta FTP che fornisce il servizio richiesto)
- Porta destinazione: 5000 (porta del Client che ha richiesto il servizio)
- Sequence Numerber: 10 (generato casualmente)
- ACK: 2 (Sequence number inviato dal Client incrementato di 1 per specificare a quale pacchetto fa riferimento e ha ricevuto)

L'ultimo pacchetto inviato dal Client:

- Porta sorgente: 5000
- Porta destinazione: 21
- Sequence Numerber: 2
- ACK: 11 (Sequence number inviato dal Server incrementato di 1 per specificare a quale pacchetto fa riferimento e ha ricevuto)

6. Trasferimento file:

   - Prima di poter disegnare il diagramma bisogna capire se si è nel caso di RTT $> T_t$ oppure $T_t >$ RTT:
     - RTT $= 2 \cdot T_e = 2 \cdot 1.2ms = 2.4ms$
     - $T_t = \dfrac{1500 \text{ Byte}}{100 \text{ Mbit/s}} = \dfrac{1500 \cdot 8 \text{ bit}}{100 \cdot 10^6 \text{ bit/s}} = 120 \cdot 10^{-6} \text{ bit} = 0.00012s = 0.12ms$
     - Dunque siamo nel caso in cui RTT $> T_p$
       * In un RTT vengono trasmessi al massimo:

$$\frac{\text{RTT}}{T_t} = \frac{2.4ms}{0.12ms} = 20 \text{ segmenti}$$

RTT	CWND	T_w
1	1	[1]
2	2	[2, 3]
3	4	[4, 5, 6, 7]
4	8	[8, . . . , 15]
5	16	[16, . . . , 31]

- Il trasferimento del file:

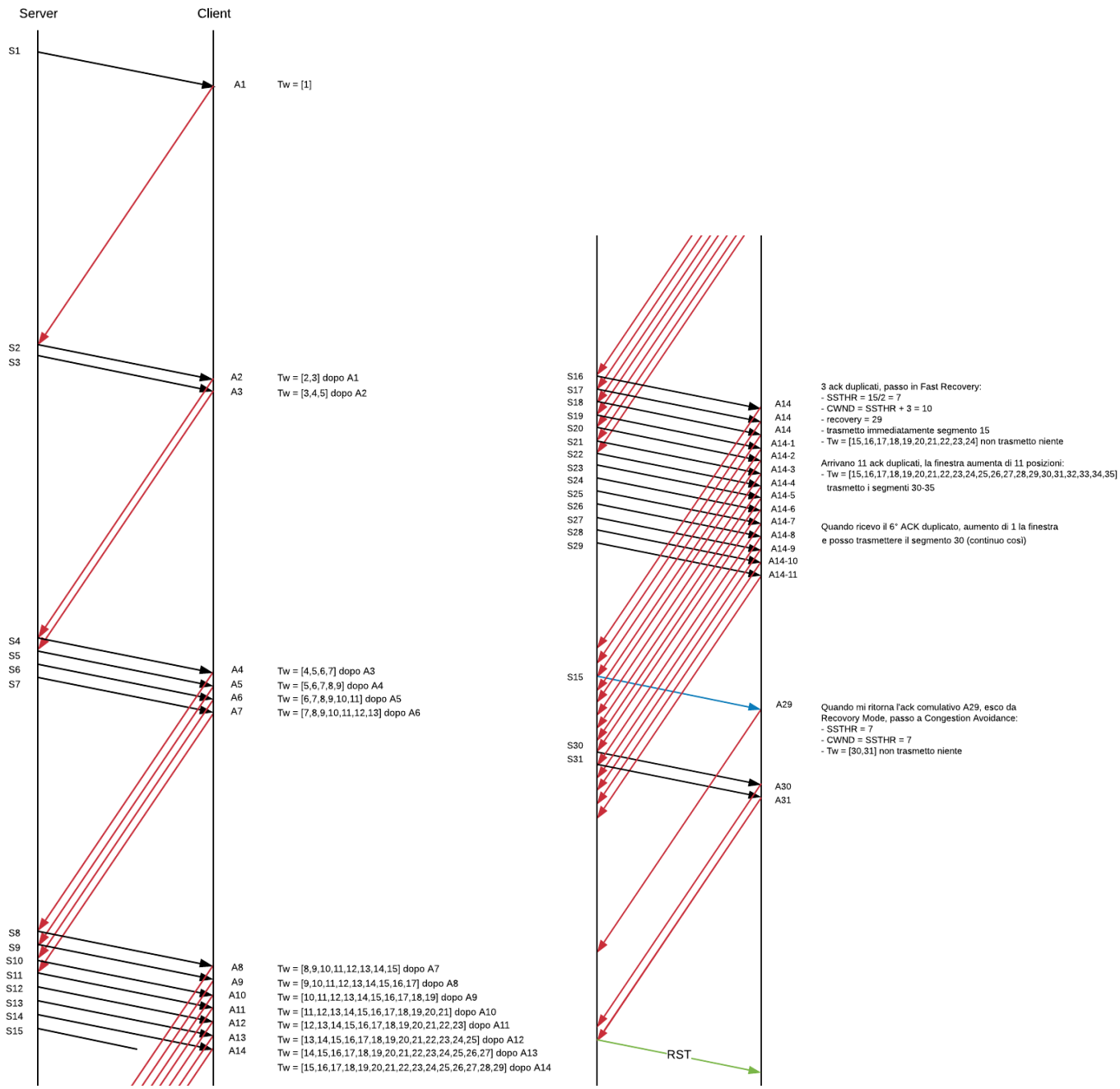
$$T_F = 6 \cdot RTT = 6 \cdot (2 \cdot T_e) = 12 \cdot 1.2\,ms = 14.4\,ms$$

- Throughput:

$$T_h = \frac{42800 \text{ Byte}}{T_F} = \frac{342400 \text{ bit}}{14.4ms} = \frac{342400 \text{ bit}}{0.0144s} = 23777777.7 \text{ bit/s} = 22.7 \text{ Mbit/s}$$

7. Trasmissione con il 15° pacchetto perso:

- RTO = $180ms$
- SSTHR = 64 Kbyte = 44 segmenti.
- Il trasferimento parte con Slow Start con CWND = 1.
- Dopo un RTT, la finestra di trasmissione raddoppia e vengono trasmessi i segmenti S2 e S3.
- Dopo un altro RTT, la finestra raddoppia nuovamente e vengono inviati i pacchetti S3,S4,S5,S6,S7.
- Al terzo RTT, la finestra è di 8 segmenti e l'ultimo pacchetto che viene trasmesso è il 15°, che viene perso.
- Gli ACK dei segmenti trasmessi prima del pacchetto S15 vengono ricevuti dal Client, che aumenta la finestra di trasmissione (CWND = [15-29]) ma il Server non riceve l'ACK del segmento A15.
- Il Server trasmette i segmenti non inviati della finestra di trasmissione e dopo un RTT inizia a ricevere gli ACK dal Client.
- Visto che il segmento S15 non è stato ricevuto dal Client, il Server riceve 3 ACK duplicati dei segmenti S16-S18 (viene notificato del fatto che il Client ha ricevuto fino al S14 pacchetto).
- TCP entra in fase Fast recovery (dimezza SSTHR, inizializza CWND a SSTHR e la incrementa di 3, salva nella variabile recovery A29 che è l'ultimo pacchetto che il Server ha inviato).
- Viene trasmesso il pacchetto perso (S15); la nuova finestra non permette la trasmissione di nuovi segmenti (sono già stati trasmessi tutti).
- Vengono ricevuti dal Server 11 ACK duplicati (A20-A29), seguendo l'algoritmo di Fast Retransmit, la finestra del Server viene incrementa di 11 posizioni (Tw = [15,...,31]).
- Visto che la finestra lo permette, il Server è abilitato a trasmettere i segmenti S30 e S31.
- Quando il Client riceve S15, manda un ACK cumulativo notificando al Server che ha ricevuto tutto fino al segmento S29.
- Una volta ricevuto questo ACK, il Server esce dalla modalità Fast Recovery e passa a Congestion Avoidance (setta CWND = SSTHR = 7).
- La nuova finestra non permette al Server di inviare nessun nuovo segmento, in quanto sono già stati tutti trasmessi; dunque aspetta gli ACK dei segmenti che ha trasmesso, così per poter chiudere la connessione.
- Ricevuti gli ultimi ACK, il Server manda un ultimo segmento, con il flag RST settato, così chiude la connessione con il Client.


Server
Client
S1
A1
Tw = [1]
S2
S3
A2
Tw = [2,3] dopo A1
A3
Tw = [3,4,5] dopo A2
S4
S5
S6
S7
A4
Tw = [4,5,6,7] dopo A3
A5
Tw = [5,6,7,8,9] dopo A4
A6
Tw = [6,7,8,9,10,11] dopo A5
A7
Tw = [7,8,9,10,11,12,13] dopo A6
S8
S9
S10
S11
S12
S13
S14
S15
A8
Tw = [8,9,10,11,12,13,14,15] dopo A7
A9
Tw = [9,10,11,12,13,14,15,16,17] dopo A8
A10
Tw = [10,11,12,13,14,15,16,17,18,19] dopo A9
A11
Tw = [11,12,13,14,15,16,17,18,19,20,21] dopo A10
A12
Tw = [12,13,14,15,16,17,18,19,20,21,22,23] dopo A11
A13
Tw = [13,14,15,16,17,18,19,20,21,22,23,24,25] dopo A12
A14
Tw = [14,15,16,17,18,19,20,21,22,23,24,25,26,27] dopo A13
Tw = [15,16,17,18,19,20,21,22,23,24,25,26,27,28,29] dopo A14
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
A14
A14
A14
A14-1
A14-2
A14-3
A14-4
A14-5
A14-6
A14-7
A14-8
A14-9
A14-10
A14-11
3 ack duplicati, passo in Fast Recovery:
- SSTHR = 15/2 = 7
- CWND = SSTHR + 3 = 10
- recovery = 29
- trasmetto immediatamente segmento 15
- Tw = [15,16,17,18,19,20,21,22,23,24] non trasmetto niente
Arrivano 11 ack duplicati, la finestra aumenta di 11 posizioni:
- Tw = [15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35]
trasmetto i segmenti 30-35
Quando ricevo il 6° ACK duplicato, aumento di 1 la finestra
e posso trasmettere il segmento 30 (continuo così)
S15
A29
Quando mi ritorna l'ack comulativo A29, esco da
Recovory Mode, passo a Congestion Avoidance:
- SSTHR = 7
- CWND = SSTHR = 7
- Tw = [30,31] non trasmetto niente
S30
S31
A30
A31
RST