

Appunti di Fondamenti di Comunicazioni e Internet

Canale 2 · Reti · Prof. Marco Polverini

Capitolo 6: Trasferimento dati affidabile: stop and wait

Autore: Tommaso C.

Appunti tratti dalle lezioni in aula e dal materiale del corso

Argomenti: Reliable data transfer · End-to-end e hop by hop
Stop and wait: riscontri, timeout, numeri di sequenza · Macchina a stati finiti
Efficienza e prodotto banda-ritardo · Protocolli con pipeline

Indice

1	Il trasferimento dati affidabile	1
2	Dove si realizza: end-to-end e hop by hop	1
2.1	Perché servono tutti e due	2
3	Il protocollo stop and wait	2
3.1	Codice, riscontro e timeout	2
3.2	Duplicati e numeri di sequenza	3
3.3	Numeri 0 e 1	5
3.4	Macchina a stati finiti	5
4	Efficienza dello stop and wait	6
5	Protocolli con pipeline	8
6	Esercizi	10
7	Soluzioni	12

1 Il trasferimento dati affidabile

Quando il ricevitore scopre che un frame ha errori (Capitolo 5) ha tre strade: scartarlo, correggerlo, se il codice lo permette, oppure recuperarlo con una procedura di **trasferimento dati affidabile**. Correggere può correggere male, e scartare perde il dato.

Definizione – Trasferimento dati affidabile

Il **trasferimento dati affidabile** (*reliable data transfer*, **RDT**) consegna i dati **senza errori**, **una sola volta** e **nell'ordine di invio**, anche se il canale è inaffidabile.

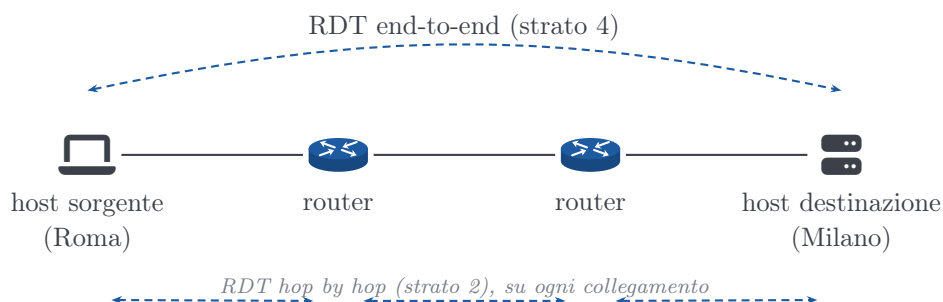
Un canale inaffidabile può fare quattro cose, le stesse del servizio best effort (Capitolo 4):

Evento	Che cosa succede	Esempio
errore	arrivano bit diversi da quelli trasmessi	trasmesso 110, ricevuto 100
perdita	il messaggio non arriva mai	un guasto lungo il collegamento
duplicato	lo stesso messaggio arriva due volte	M e subito dopo ancora M
fuori sequenza	l'ordine di arrivo è invertito	M_2 , generato dopo, arriva prima di M_1

2 Dove si realizza: end-to-end e hop by hop

Definizione – RDT end-to-end e hop by hop

- L'**RDT end-to-end** è realizzato solo dai sistemi terminali, sorgente e destinazione, nello **strato di trasporto** (strato 4). Lo fa TCP.
- L'**RDT hop by hop** (salto dopo salto) è realizzato su ogni collegamento, dallo **strato di collegamento** (strato 2).



2.1 Perché servono tutti e due

Il progettista di un'applicazione non sa quali collegamenti attraverseranno i suoi dati: sa solo di che servizi ha bisogno. Se gli serve l'RDT lo chiede allo strato di trasporto, quindi lo ottiene end-to-end. Per questo HTTP è trasportato da TCP.

Ripetere l'RDT a più livelli costa: è lavoro in più e rallenta il sistema. Conviene farlo anche sui collegamenti solo dove serve.

	End-to-end (strato 4)	Hop by hop (strato 2)
Chi lo decide	chi progetta l'applicazione, che lo chiede al trasporto	la tecnologia del collegamento: Ethernet non lo offre, il 5G sì
Vantaggio	vale su qualunque percorso, anche se qualche collegamento non offre l'RDT	recupera subito, sul primo salto, dove gli errori sono più probabili
Svantaggio	il recupero è lento: serve un tempo di andata e ritorno fra i terminali, decine di millisecondi fra Roma e New York	ripetere il servizio su ogni collegamento appesantisce la rete

Sui collegamenti cablati, come la fibra, gli errori binari sono molto rari e l'RDT non serve. Sui collegamenti senza fili o via satellite sono frequenti.

3 Il protocollo stop and wait

Definizione – ARQ e stop and wait

Gli **ARQ** (*Automatic Repeat Request*) sono una famiglia di protocolli che realizza l'RDT sopra un collegamento inaffidabile. Il più semplice è lo **stop and wait**: il trasmettitore *A* invia un frame e **si ferma ad aspettare** il riscontro, e solo dopo invia il successivo. Il ricevitore è *B*.

3.1 Codice, riscontro e timeout

1. *A* non trasmette *M* ma una **parola di codice**, con bit di controllo (Capitolo 5): *B* può così rivelare gli errori.
2. Se la stringa ricevuta è una parola di codice, *B* **accetta** il frame.
3. *A* non può sapere che la ricezione è riuscita, se *B* non glielo dice. *B* invia un **riscontro** (**ACK**, *acknowledgement*), un messaggio **positivo**: arriva solo quando le cose vanno bene.

L'ACK è un frame a sé: ha un'intestazione e un codice che ne protegge dagli errori, ma nessun dato dell'utente.

Se il frame si perde, B non riceve nulla e non risponde, e A non vede mai arrivare un ACK: i due restano in stallo.

Definizione – Timeout di ritrasmissione

A ogni frame trasmesso A avvia un **timeout**, detto **RTO** (*Retransmission Timeout*). Se scade prima dell'arrivo dell'ACK, A **ritrasmette** il frame.

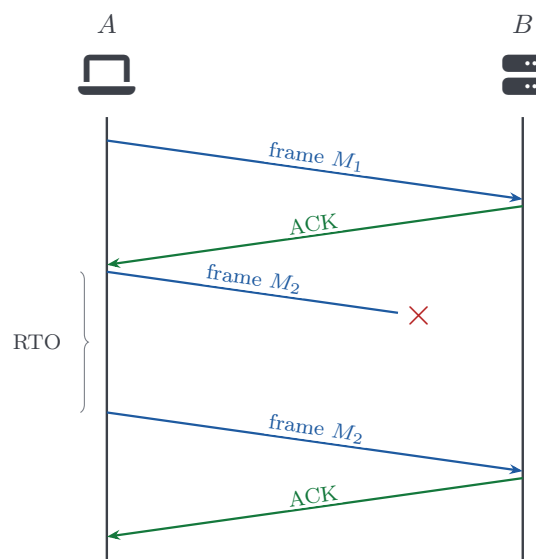


Figura 1: Un frame perso. B non riceve M_2 e non risponde. Allo scadere dell'RTO A ritrasmette M_2 .

3.2 Duplicati e numeri di sequenza

L'RTO può scadere anche se il frame è arrivato: il frame, o il suo ACK, è molto in ritardo. A ritrasmette un frame che B ha già accettato, e B , che non può saperlo, accetta anche la copia. Il ricevitore consegna così lo stesso messaggio due volte.

Definizione – Numero di sequenza

Un **numero di sequenza** è un identificativo univoco scritto nell'intestazione del frame, per esempio un contatore che aumenta di uno a ogni frame. Una ritrasmissione porta lo stesso numero del frame originale, quindi B riconosce il **duplicato**: lo **scarta**, e invia di nuovo l'ACK, perché quello che aveva mandato non ha sbloccato A .

Anche gli ACK portano un numero.

Definizione – ACK numerato

L'ACK porta il numero di sequenza del **prossimo frame che B si aspetta** (*expected sequence number*). Dopo il frame 0 l'ACK porta 1. A **ignora** un ACK che non è quello che aspetta.

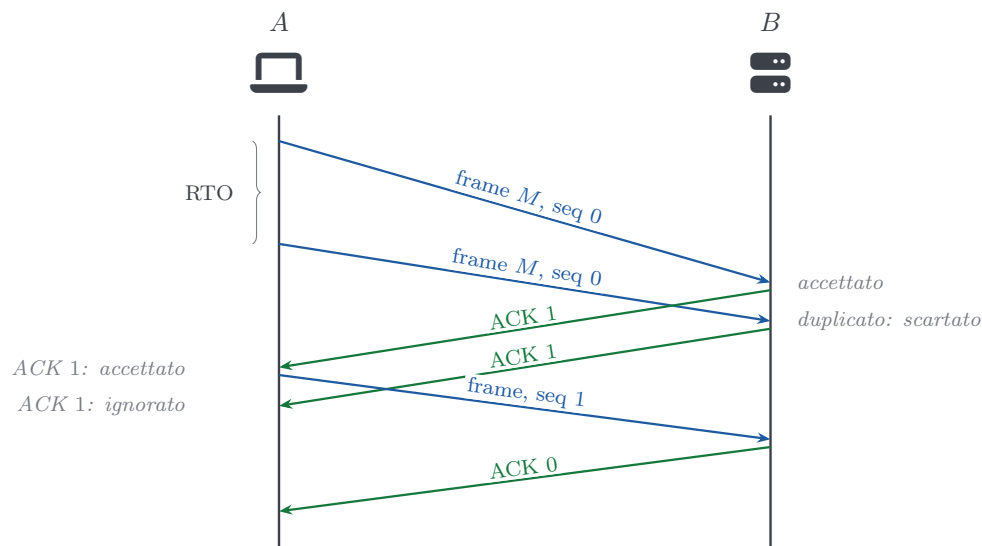


Figura 2: Il primo frame M è molto in ritardo e l'RTO scade: A ritrasmette M con lo stesso numero 0. B accetta il primo che arriva, scarta il secondo come duplicato e risponde a tutti e due con $ACK\ 1$. Il primo ACK sblocca A , che passa al frame 1 e ignora il secondo $ACK\ 1$, perché aspetta l' $ACK\ 0$.

Osservazione – Ricevere e consegnare

Ricevere un frame e consegnarlo allo strato superiore sono cose diverse: B riceve il duplicato, ma non ne consegna il contenuto.

Proposizione – I meccanismi dello stop and wait

Lo stop and wait si regge su cinque meccanismi:

- un **codice di rivelazione** degli errori, per accorgersi dei frame errati;
- i **riscontri positivi** (ACK), per sapere che la ricezione è riuscita;
- il **timeout** (RTO), per uscire dallo stallo quando frame o ACK si perdono;
- la **ritrasmissione**, per recuperare i frame persi o errati;
- i **numeri di sequenza**, per riconoscere duplicati e frame fuori sequenza.

3.3 Numeri 0 e 1

Definizione – Numerazione a un bit

Nello stop and wait basta usare i numeri di sequenza 0 e 1, alternati. C'è sempre al più un frame non riscontrato in giro, perché dopo ogni frame A si ferma: ritrasmette solo se scade l'RTO, e riprende solo con l'ACK. Se B riceve 0, poi 1 e poi di nuovo 0, sa che la catena è ripartita e che quel 0 è un frame nuovo.

Il protocollo non è però infallibile. Un frame può restare in rete molto più a lungo del previsto, e arrivare quando B si aspetta di nuovo lo stesso numero.

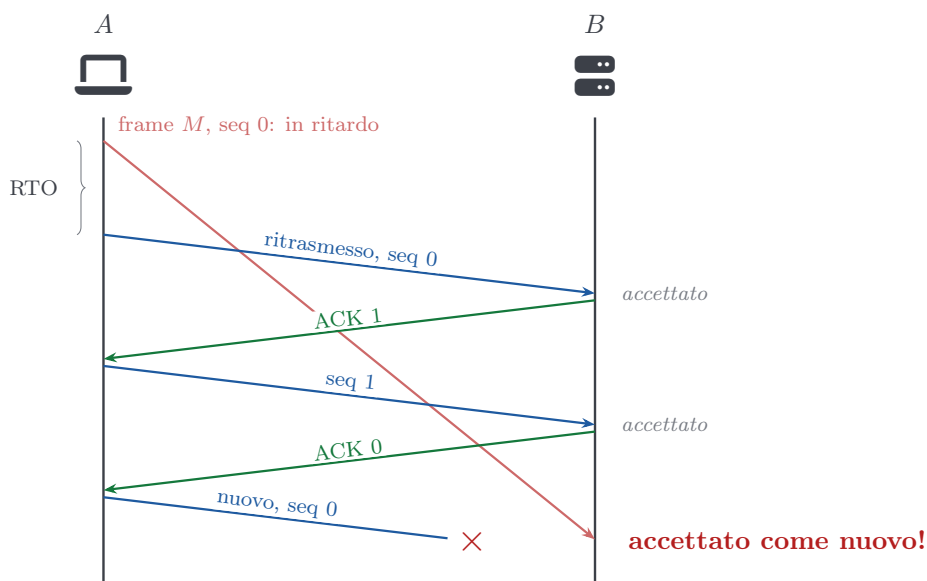


Figura 3: Il frame 0 originale resta in rete. La ritrasmissione e il frame 1 vanno a buon fine, e B aspetta di nuovo il numero 0. Il nuovo frame 0 si perde, e proprio allora arriva il vecchio: B lo accetta e lo consegna come se fosse quello nuovo.

Con più numeri di sequenza il rischio si riduce, ma si paga: il ricevitore ha bisogno di più memoria, e i numeri sono bit dell'intestazione, quindi overhead. Più piccolo è il campo, meno overhead.

3.4 Macchina a stati finiti

Definizione – Stati dello stop and wait

Lo stato del sistema è la coppia $(S_{\text{last}}, R_{\text{next}})$: il numero del frame che A sta trasmettendo e il numero che B si aspetta. Si parte da $(0, 0)$. Senza errori il sistema percorre quattro stati: ogni frame corretto e atteso fa passare B al numero successivo, ogni ACK atteso fa passare A al frame successivo.

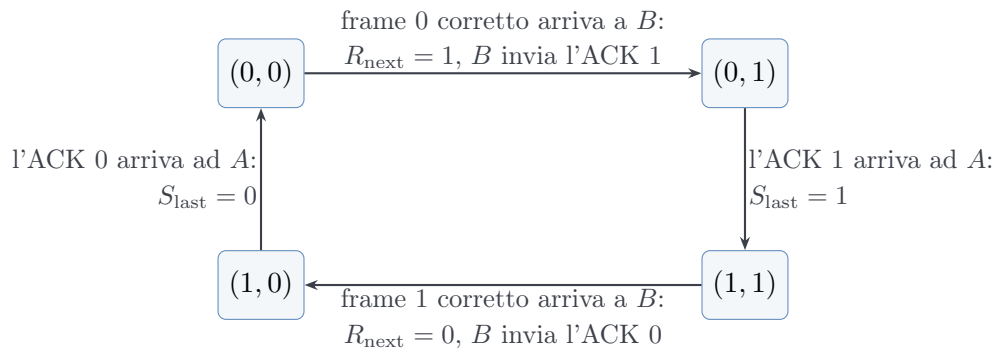


Figura 4: Gli stati $(S_{\text{last}}, R_{\text{next}})$ senza errori.

Negli altri casi lo stato non cambia. Se a B arriva un frame con errori o con un numero diverso da quello atteso, un duplicato, B lo scarta e invia l'ACK con R_{next} , cioè dice quale frame aspetta. Se ad A arriva un ACK diverso da quello atteso, A lo ignora.

4 Efficienza dello stop and wait

Definizione – Efficienza

L'**efficienza** η è il rapporto fra il **tempo utile** e il **tempo totale** di un frame. Il tempo utile è il tempo che A passa a trasmettere il frame, $T_{\text{TX}} = L/C$. Il tempo totale va dal primo istante di trasmissione alla ricezione dell'ACK.

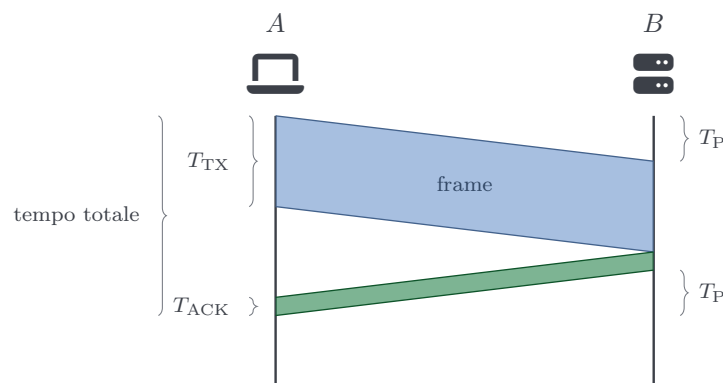


Figura 5: Un ciclo dello stop and wait. Il tempo utile è T_{TX} , quello totale va fino all'arrivo dell'ACK.

Il tempo totale comprende la trasmissione del frame, due volte la propagazione e la trasmissione dell'ACK. L'ACK è lungo L_H bit, con $T_{\text{ACK}} = L_H/C$: ha solo l'intestazione e il CRC.

Proposizione – Efficienza dello stop and wait

$$\eta = \frac{L/C}{L/C + 2T_P + L_H/C} = \frac{1}{1 + \frac{2CT_P}{L} + \frac{L_H}{L}}.$$

Con $L_H \ll L$ il termine dell'ACK si trascura. Con $\text{RTT} = 2T_P$,

$$\eta \simeq \frac{1}{1 + C \text{RTT}/L}.$$

L'RTT di questa formula è solo $2T_P$, senza i tempi di trasmissione: non è l'RTT del Capitolo 3.

L'ACK, inoltre, spesso non viaggia da solo: se la comunicazione è bidirezionale, B lo inserisce in un frame di dati diretto verso A (*feedback*), e il suo costo scompare.

Definizione – Prodotto banda-ritardo

Il **prodotto banda-ritardo** (*bandwidth-delay product*, BDP) è

$$\text{BDP} = C \cdot \text{RTT}.$$

Si misura in bit: sono i bit che possono stare in transito nel collegamento, come l'acqua in un tubo di sezione C e lunghezza RTT.

Lo stop and wait mette nel collegamento un solo frame alla volta. Se il BDP è piccolo rispetto a L il frame riempie quasi il tubo e l'efficienza è alta; se è grande il tubo resta quasi vuoto. Il BDP sta al denominatore di η .

Esempio – Efficienza al variare del ritardo e della capacità

Frame di $L = 10\,000$ bit e ACK di $L_H = 200$ bit. Con $v = 2 \times 10^8$ m/s (Capitolo 3) i ritardi corrispondono a collegamenti lunghi da $d = 100$ km a $d = 100\,000$ km. Per ogni RTT si calcola $\text{BDP} = C \cdot \text{RTT}$ e $\eta = 1/(1 + \text{BDP}/L + L_H/L)$:

d (km)	RTT	$C = 1$ Mbit/s		$C = 1$ Gbit/s	
		BDP (bit)	η	BDP (bit)	η
100	1 ms	10^3	89,3 %	10^6	0,990 %
1000	10 ms	10^4	49,5 %	10^7	0,0999 %
10 000	100 ms	10^5	9,07 %	10^8	0,0100 %
100 000	1 s	10^6	0,990 %	10^9	0,001 00 %

A 1 Gbit/s, la capacità d'accesso di una fibra domestica, l'efficienza è già dell'1% con $d = 100$ km: si usa un centesimo della capacità.

Con errori binari l'efficienza scende ancora: a parità di capacità, ritardo e lunghezza, un BER da 10^{-6} a 10^{-4} la riduce molto. Lo stop and wait mostra i principi del recupero degli errori, ma funziona male: c'è bisogno di protocolli con prestazioni più alte.

L'RTO non può durare meno del tempo totale: prima, l'ACK non può essere arrivato. Ogni frame perso, o ACK perso, costa quindi almeno un RTO in più: se il tempo totale di un ciclo è $T_{\text{TOT}} = T_{\text{TX}} + 2T_{\text{P}} + T_{\text{ACK}}$ e si ritrasmette k volte, n frame richiedono

$$n T_{\text{TOT}} + k \text{ RTO}.$$

5 Protocolli con pipeline

Lo stop and wait si ferma dopo ogni frame, e per questo tempo utile e totale distano. Un frame molto lungo non risolve: la lunghezza non la decide chi trasmette, e un frame lungo è più vulnerabile agli errori binari. Si può invece **non fermarsi**.

Definizione – Protocolli con pipeline

Un protocollo **con pipeline** ammette sul collegamento più frame **in volo**, cioè trasmessi e non ancora riscontrati, ciascuno con il proprio numero di sequenza. Dopo avere trasmesso N frame il trasmettitore aspetta i riscontri; N è un parametro che si può impostare.

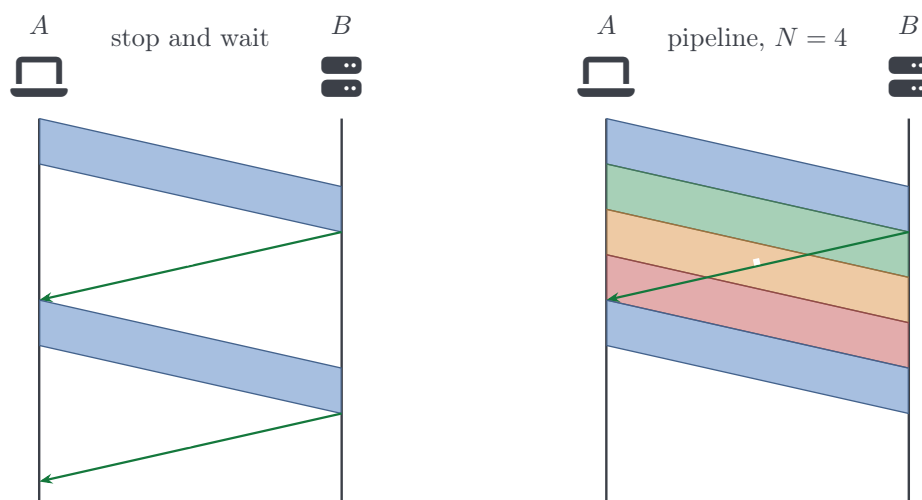


Figura 6: Stop and wait: un frame alla volta nel collegamento. Pipeline con $N = 4$: i frame si seguono senza pause, e il primo ACK arriva quando l'ultimo dei quattro finisce di essere trasmesso.

Proposizione – Efficienza con pipeline

Con N frame per ciclo l'efficienza è N volte quella dello stop and wait, ma non supera 1. Trascurando l'ACK,

$$\eta = \min \left(1, \frac{N L / C}{L / C + \text{RTT}} \right) = \min \left(1, \frac{N}{1 + C \text{RTT} / L} \right).$$

Il numero minimo di frame che dà $\eta = 1$, cioè una trasmissione senza interruzioni, è

$$N_{\min} = \left\lceil 1 + \frac{C \text{RTT}}{L} \right\rceil$$

Se si tiene conto anche dell'ACK basta contare il ciclo completo: la trasmissione è senza interruzioni con $N \geq T_{\text{TOT}}/T_{\text{TX}}$, e

$$N_{\min} = \left\lceil \frac{T_{\text{TX}} + 2 T_{\text{P}} + T_{\text{ACK}}}{T_{\text{TX}}} \right\rceil = \left\lceil \frac{1}{\eta_{\text{SW}}} \right\rceil,$$

dove η_{SW} è l'efficienza dello stop and wait. Si applica agli Esercizi 2 e 3.

Riepilogo

- **RDT**: consegna senza errori, una volta sola e in ordine, su un canale che può sbagliare, perdere, duplicare e disordinare.
- **End-to-end e hop by hop**: il primo è nel trasporto (TCP) e serve sempre a chi lo chiede; il secondo è nello strato 2, solo dove gli errori sono frequenti.
- **Stop and wait**: un frame alla volta; ACK positivo; se l'RTO scade si ritrasmette. Meccanismi: codice, ACK, RTO, ritrasmissione, numeri di sequenza.
- **Numeri**: il frame porta il suo numero, l'ACK il prossimo atteso. Bastano 0 e 1, perché c'è un solo frame in volo; B scarta i duplicati e rinvia l'ACK.
- **Stati**: $(S_{\text{last}}, R_{\text{next}})$, quattro stati senza errori.
- **Efficienza**: $\eta = 1/(1 + 2CT_P/L + L_H/L) \simeq 1/(1 + \text{BDP}/L)$, con $\text{BDP} = C \cdot \text{RTT}$.
- **Tempo con perdite**: $nT_{\text{TOT}} + k \text{RTO}$; l'RTO è almeno T_{TOT} .
- **Pipeline**: N frame in volo, $\eta = \min(1, N\eta_{\text{SW}})$, $N_{\text{min}} = \lceil 1/\eta_{\text{SW}} \rceil$.

6 Esercizi

Esercizio 1 – Appello del 11/02/2025, Canale 1, Prof.ssa Francesca Cuomo, esercizio, adattato

Si consideri un collegamento tra due terminali A e B con velocità di trasmissione sul link di 10 kbit/s. Il terminale A deve inoltrare a B 7 trame, tutte della stessa dimensione (1000 B), che impiegano un tempo di trasmissione T . Il ritardo di propagazione sul collegamento è $\tau = 2$ s in ognuna delle due direzioni. Si considerano trascurabili i tempi di trasmissione dei riscontri.

I terminali adottano un meccanismo di controllo d'errore basato su stop and wait. Si assuma che il timeout associato a una trama sia pari al tempo minimo necessario per ricevere il relativo riscontro.

Supponendo che nel trasferimento delle 7 trame da A a B la 3^a trama venga persa, si calcoli:

1. il tempo necessario per trasferire le 7 trame, cioè il tempo dall'invio del primo bit della prima trama al riscontro dell'ultima trama;
2. l'efficienza di utilizzo del collegamento;
3. volendo massimizzare l'efficienza, si può adottare un meccanismo Go-Back-N: si calcoli la dimensione ottima N della finestra di trasmissione.

Soluzione a pagina 12 →

Esercizio 2 – Appello del 12/01/2023, Canale 1, Prof.ssa Francesca Cuomo, domanda 7

Si consideri un protocollo di controllo d'errore di tipo Go-Back-N. Sia: 1) il ritardo di propagazione pari a $1/3$ del tempo di trasmissione di una trama; 2) il tempo di trasmissione dei riscontri pari a $2/3$ del tempo di trasmissione di una trama e l'elaborazione nei nodi trascurabile. Quanto deve essere la dimensione minima della finestra in emissione W_s affinché, se non si verificano errori, il trasmettitore sia in grado di trasmettere trame continuamente (senza soluzione di continuità)?

- a. $W_s = 3$
- b. $W_s = 4$
- c. $W_s = 2$

Soluzione a pagina 12 →

Esercizio 3 – Appello del 08/02/2023, Canale 1, Prof.ssa Francesca Cuomo, domanda 12

Si consideri un protocollo di controllo d'errore di tipo Go-Back-N. Sia: 1) il ritardo di propagazione pari a $1/5$ del tempo di trasmissione di una trama; 2) il tempo di trasmissione dei riscontri pari a $3/5$ del tempo di trasmissione di una trama e l'elaborazione nei nodi trascurabile. Quanto deve essere la dimensione minima della finestra in emissione W_s affinché, se non si verificano errori, il trasmettitore sia in grado di trasmettere trame continuamente (senza soluzione di continuità)?

- a. $W_s = 3$
- b. $W_s = 4$
- c. $W_s = 2$

Soluzione a pagina 13 →

7 Soluzioni

Soluzione dell'Esercizio 1 – Appello del 11/02/2025, Canale 1, Prof.ssa Francesca Cuomo, esercizio, adattato

1. *Tempo di trasferimento.* Una trama è lunga 8000 bit, quindi

$$T = T_{\text{TX}} = \frac{8000 \text{ bit}}{10 \text{ kbit/s}} = 0,8 \text{ s}.$$

Gli ACK sono trascurabili ($T_{\text{ACK}} = 0$), quindi il ciclo è

$$T_{\text{TOT}} = T + 2\tau = 0,8 \text{ s} + 4 \text{ s} = 4,8 \text{ s},$$

e il timeout, il tempo minimo per ricevere l'ACK, vale $\text{RTO} = T_{\text{TOT}}$. Le 7 trame richiedono 7 cicli. La terza trama persa costringe ad aspettare un RTO e a ritrasmetterla, cioè $k = 1$:

$$7T_{\text{TOT}} + 1 \cdot \text{RTO} = 8 \cdot 4,8 \text{ s} = \boxed{38,4 \text{ s}}.$$

2. *Efficienza.* Il tempo utile è quello di trasmissione delle 7 trame, $7T = 5,6 \text{ s}$, e il tempo totale è $38,4 \text{ s}$:

$$\eta = \frac{7T}{T_{\text{tot}}} = \frac{5,6 \text{ s}}{38,4 \text{ s}} \simeq \boxed{0,146}.$$

3. *Finestra ottima.* Il tempo di andata e ritorno della propagazione è $\text{RTT} = 2\tau = 4 \text{ s}$ e $L/C = T = 0,8 \text{ s}$, quindi

$$N_{\min} = \left\lceil 1 + \frac{C \text{RTT}}{L} \right\rceil = \left\lceil 1 + \frac{4 \text{ s}}{0,8 \text{ s}} \right\rceil = \lceil 1 + 5 \rceil = \boxed{6}.$$

Con $N = 6$ il trasmettitore non si ferma mai: sei trame occupano $4,8 \text{ s}$, proprio il ciclo di una trama.

[← Torna all'Esercizio 1, pagina 10](#)

Soluzione dell'Esercizio 2 – Appello del 12/01/2023, Canale 1, Prof.ssa Francesca Cuomo, domanda 7

Sia T il tempo di trasmissione di una trama. Allora $T_P = T/3$ e $T_{\text{ACK}} = 2T/3$, e l'elaborazione è trascurabile. Il ciclo è

$$T_{\text{TOT}} = T + 2 \cdot \frac{T}{3} + \frac{2T}{3} = \frac{7}{3} T.$$

La trasmissione è continua se la finestra copre il ciclo:

$$W_s \geq \frac{T_{\text{TOT}}}{T} = \frac{7}{3} \simeq 2,33, \quad W_s = \lceil 7/3 \rceil = 3.$$

Con $W_s = 2$ il trasmettitore si fermerebbe, perché $2 < 7/3$; con 4 la finestra sarebbe più grande del necessario. Risposta: **a**.

← Torna all'Esercizio 2, pagina 11

Soluzione dell'Esercizio 3 – Appello del 08/02/2023, Canale 1, Prof.ssa Francesca Cuomo, domanda 12

Con $T_P = T/5$ e $T_{ACK} = 3T/5$ il ciclo è

$$T_{TOT} = T + 2 \cdot \frac{T}{5} + \frac{3T}{5} = 2T,$$

quindi $W_s \geq T_{TOT}/T = 2$, e il minimo è $W_s = \lceil 2 \rceil = 2$: due trame occupano $2T$, esattamente il ciclo, e il trasmettitore non si ferma. Risposta: **c**.

← Torna all'Esercizio 3, pagina 11