1. Introduzione alle reti

1.1 Rete di accesso

- Rete che connette fisicamente un sistema al suo **edge router** (router di bordo)
  - Primo router sul percorso dal sistema d'origine a un qualsiasi altro sistema di destinazione collocato al di fuori della stessa rete di accesso

1.2 Mezzi trasmissivi

- **Mezzi vincolati**
  - Onde contenute in un mezzo fisico
- **Mezzi non vincolati**
  - Onde si propagano nell'atmosfera e nello spazio esterno

1.3 Protocollo

Definisce

- **Formato, Ordine** dei messaggi **ricevuti** e **inviati**
- **Azioni adottate** su trasmissione e ricezione di dati

1.4 Modelli (Accesso alla rete)

- **Client/Server**
  - Host client richiedono e ricevono servizi da **server**
- **Peer-Peer**
  - Minimo o nessun uso di server

1.5 Commutazione di pacchetto (Packet-switching)

- Gli host dividono i messaggi delle applicazioni in **pacchetti**
- Pacchetti rilanciati in rete da un **router** al successivo
- Ogni pacchetto viene trasmesso utilizzando la **piena capacità trasmissiva del link**

1.5.1 Trasmissione "Store-and-Forward"

Il commutatore di pacchetto (router) deve ricevere **l'intero pacchetto** prima di poterne cominciare a trasmettere sul collegamento in uscita il primo bit

$$\frac{L}{R}$$

secondi per trasmettere (**push out**) pacchetti di **L bit** su un link da **R bps**

L -> bit per pacchetto [bit]

R -> bit per secondo [bit/s]

- *Esempio*

L = 7,5 Mbit

R = 1,5 Mbit/sec

$$\frac{L}{R} = \frac{7{,}5\ Mbit}{1{,}5\ Mbit/s} = 5\ \text{sec}$$

- ***Ritardo End2End***

$$N\left(\frac{L}{R}\right)$$

***N** = Numero di collegamenti tra Sorgente e Destinazione (N-1 ROUTER)*

- *Esempio*

*N = **2 Collegamenti**, L = 7,5 Mbit, R = 1,5 Mbit/sec*

$$N\left(\frac{L}{R}\right) = 2\left(\frac{7{,}5\ Mbit}{1{,}5\ Mbit/s}\right) = 2(5\ sec) = 10\ sec$$

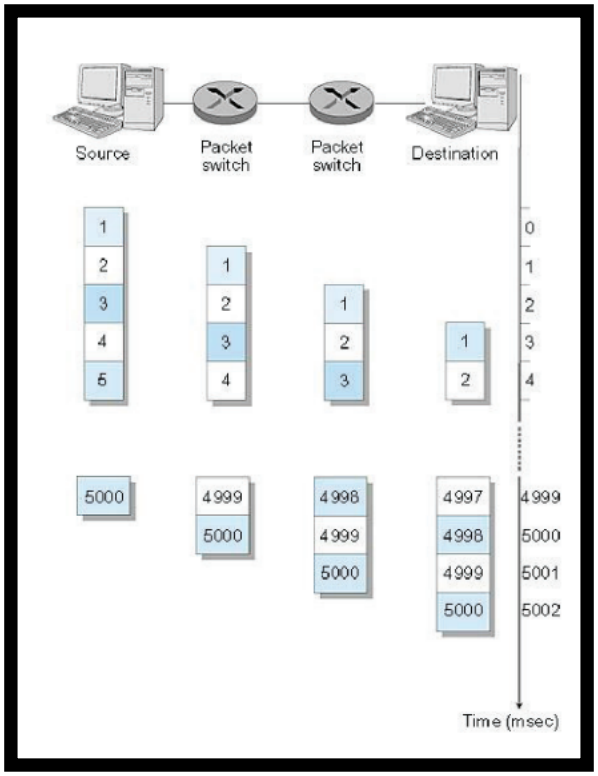
1.5.2 Segmentazione del Messaggio

Pipelining: Ogni link lavora in **parallelo**

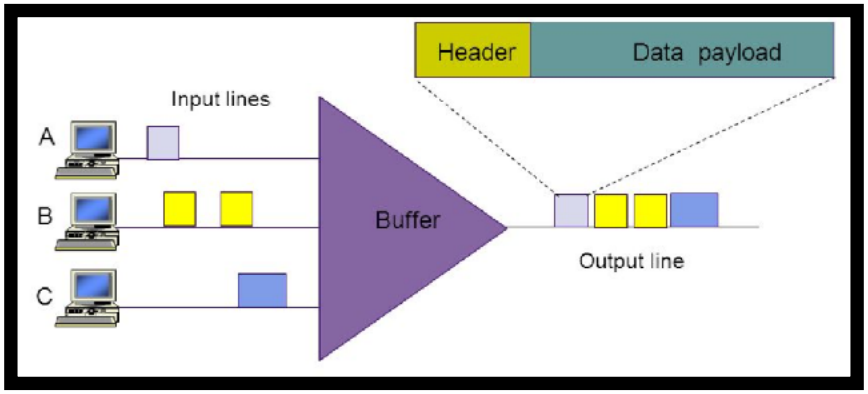
Esempio:

- *5000 pacchetti*
- *L = 1500 bit*
- *TT per pacchetto = 1 ms*

Pipelining: *Invece che impiegare 15 secondi impiega 5,002 secondi*



- **Multiplazione Statistica**



- Sequenza dei pacchetti da A, B, C <u>non ha uno schema prefissato</u>

- **Multiplazione Time-Division (TDM)**
  - Ogni host ottiene ottiene a turno l'uso esclusivo del canale di output

1.6 Reti a commutazione di pacchetto

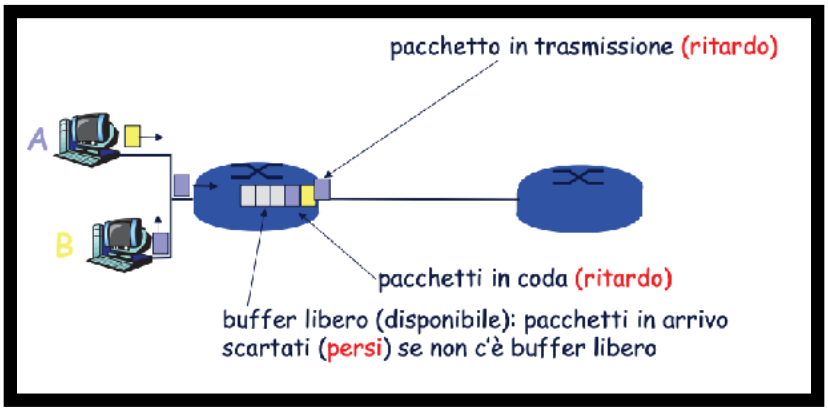
1.6.1 Forwarding

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

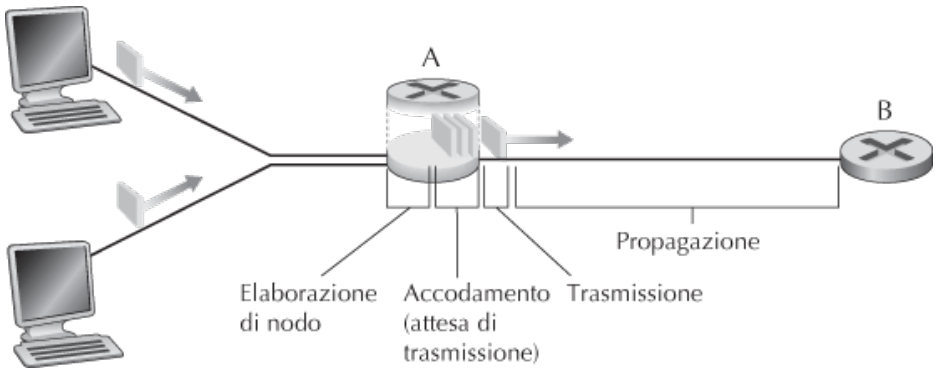
Tipo di rete	Salto successivo	Percorso	Note
Datagram	L'indirizzo destinazione contenuto nel pacchetto determina il salto successivo	Può variare durante la sessione	
Circuito virtuale	Ogni pacchetto porta un tag (**Virtual Circuit ID**) che determina il salto successivo	Fissato per tutta la sessione	**I router mantengono lo stato per ogni collegamento**

1.6.2 Ritardi e Perdite

- **Tasso di arrivo** dei pacchetti nel link supera (temporaneamente) la capacità di output del link
- **Accodarsi** dei pacchetti nei **buffer** dei router che aspettano il loro turno



- Ci sono **quattro sorgenti** che determinano il ritardo del pacchetto



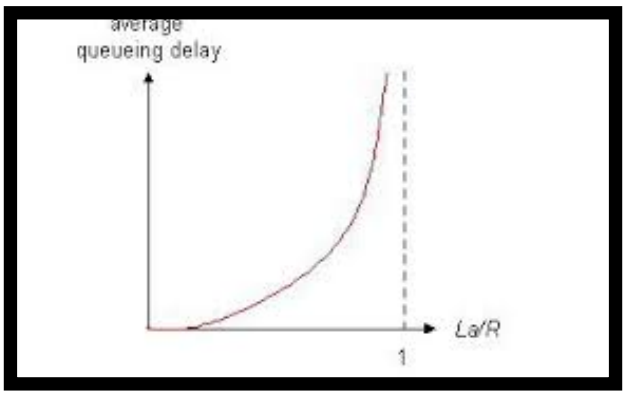
  - d_{proc}: **Tempo di elaborazione (processing delay)**
    - Tempo richiesto per esaminare l'intestazione del pacchetto e per determinare dove dirigerlo
  - d_{queue}: **Tempo di coda (queuing delay)**
    - Tempo di attesa al link di output per la trasmissione
    - Dipende dal numero di pacchetti precedentemente arrivati nel router
  - d_{trans}: **Tempo di trasmissione (transmission delay)**
    - Quantità di tempo impiegata dal router per trasmettere in uscita il pacchetto
      - L = Lunghezza pacchetto [bit]
      - R = Link bandwidth [bit/s]
      - $\frac{L}{R}$ = Tempo per inviare il pacchetto nel link
  - d_{prop}: **Tempo di propagazione (propagation delay)**
    - Quantità di tempo richiesta per la propagazione di un bit da un router a quello successivo
    - d = Lunghezza del link fisico [m]
    - v = Velocità di propagazione nel mezzo (~$2 \times 10^8\ m/s$)
    - $\frac{d}{v}$ = Ritardo di propagazione

- **Ritardo di accodamento**

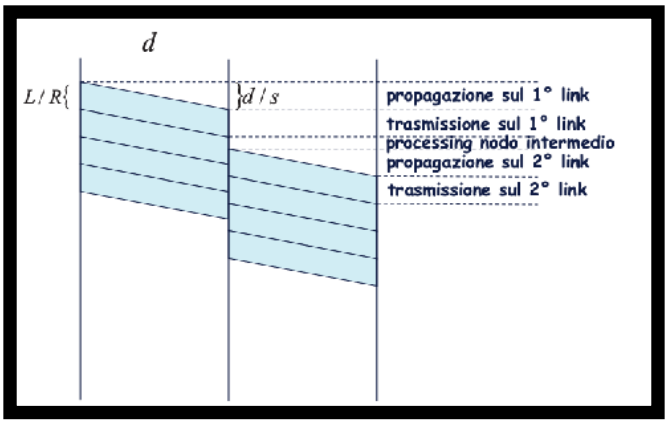
Supponiamo che N pacchetti giungano simultaneamente ogni (L/R)N secondi. Il primo pacchetto trasmesso non subisce ritardo di accodamento; il secondo presenta un ritardo di L/R secondi.

L'n-esimo pacchetto ha un ritardo di accodamento di (n-1)L/R secondi

- L = Lunghezza pacchetto [bit]
- R = Link bandwidth [bit/s]
- a = Tasso medio di arrivo dei pacchetti al router [pacchetti/s]



- $\frac{L \times a}{R}$ = **Intensità di traffico**
  - $\frac{L \times a}{R}$ ~ 0 —> **Ritardo medio piccolo**
  - $\frac{L \times a}{R}$ ~ 1 —> **Ritardo grande**
  - $\frac{L \times a}{R}$ > 1 —> **Ritardo medio infinito** (più traffico in arrivo di quanto possa essere servito)



1.7 Perdita di pacchetti

Si ha quando:

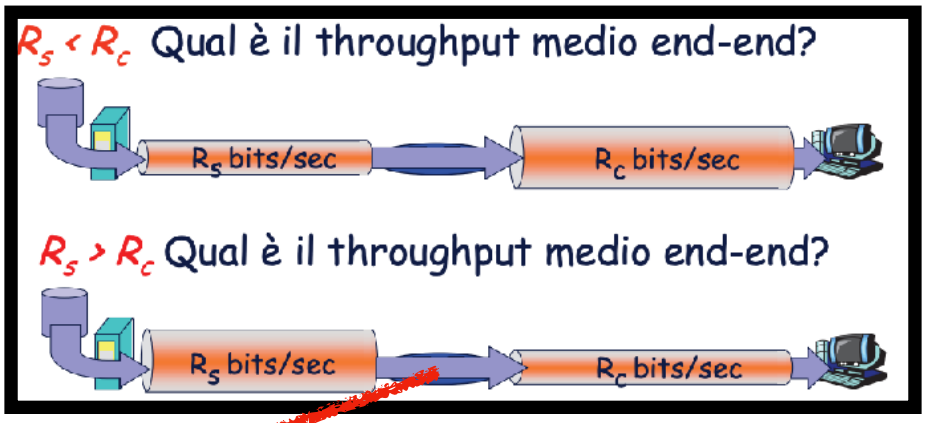
- **Coda (buffer)** nel router prima del link ha **capacità finita**
- Quando un pacchetto arriva su una **coda piena**, questo viene **scartato**
  - Il **pacchetto perso** può essere ritrasmesso dal nodo precedente o dal terminale, o non essere ritrasmesso del tutto

1.8 Throughput (tra A e B)

- **Throughput istantaneo** in ogni istante di tempo è la velocità [bps] alla quale B sta ricevendo il file
- **Throughput medio** di un file di *F bit* che richiede *T secondi* è di ***F/T bps***

In una rete con due collegamenti il throughput è dato da ***min(Rs, Rc)***

In una rete con N collegamenti il throughput con velocità date da ***(R1, R2, R3, …, Rn)*** *è dato da* ***min(R1, R2, R3, …, Rn)***



- **Bottleneck Link**
  - Link sul percorso che vincola throughput end-end

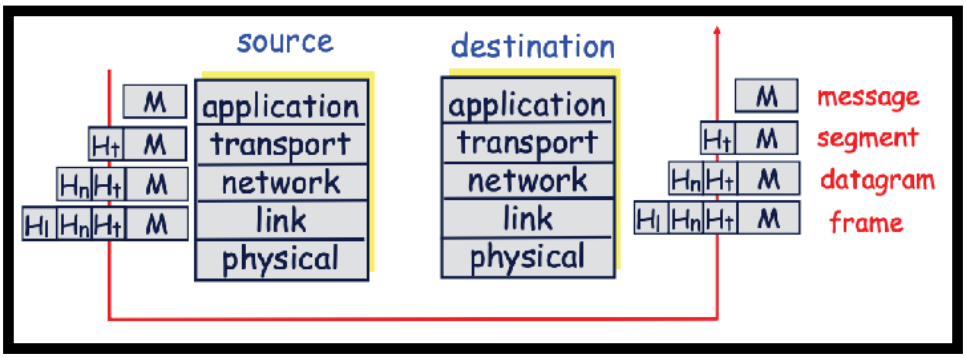
1.9 Stratificazione dei protocolli e dati

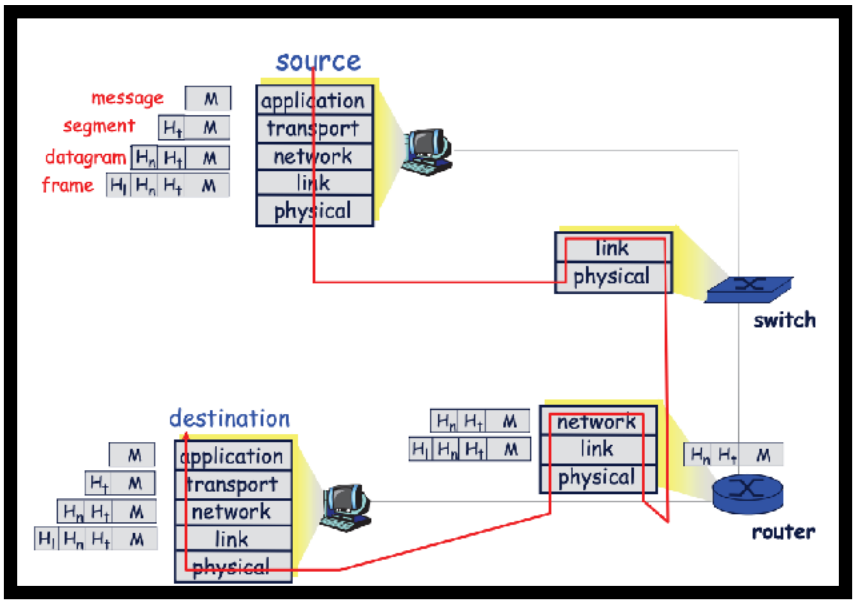
1.9.1 Internet Protocol Stack

- **Applicazione**: sostiene le applicazioni di rete
- **Trasporto**: trasferimento dati host-host
- **Rete**: routing di datagrammi da sorgente a destinazione
- **Link**: data transfer tra elementi vicini della rete
- **Fisico**: bit su cavo

1.9.2 Incapsulamento

- Ogni strato riceve i dati dallo stato soprastante
  - Aggiunge **header di informazioni** per creare una nuova unità dati
  - Passa nuova unità allo strato sottostante





1.10 Esercizi

- *Quale dei seguenti strati della pila protocollare di Internet compiono il processo di instradamento durante il percorso tra host mittente e destinatario*
  - **Strato di Rete**
- *Se la velocità di propagazione del segnale nel mezzo è di 250 milioni di metri al secondo, l'ampiezza di banda del link è di 25 kbps e la dimensione del pacchetto è di 100 bit, calcolare la distanza a cui il ritardo di propagazione è uguale a quello di trasmissione (si trascurino tutte le altre possibili cause di ritardo):*
  - v = 250 milioni di metri al secondo = 250.000.000 m/s
  - R = 25 kbps = 25.000 bit/s
  - L = 100 bit

$$\left(\frac{L}{R}\right) = \left(\frac{d}{v}\right)$$

$$d = v \times \left(\frac{L}{R}\right) = 250.000.000\ \frac{m}{s} \times \left(\frac{100\ bit}{25.000\ \frac{bit}{s}}\right) = 1.000.000\ m = 1.000\ km$$

- *Un programma client è un programma che*
  - **Inizia la conversazione**
- *Secondo il meccanismo di "store-and-forward", un messaggio di 7,5 Mbit deve passare per 4 router e 5 link ciascuno con ampiezza di banda 1,5 Mbit/s per andare da host sorgente a destinazione. Quanto tempo impiega la trasmissione <u>in assenza di frammentazione del messaggio</u>?*
  - **Secondo il meccanismo di store-ad forward, il file <u>non frammentato</u> deve arrivare al router successivo di ogni link completamente prima di proseguire il cammino, quindi sarà necessario calcolare il tempo di trasmissione per ogni link (L/R) e moltiplicarlo per il numero di link**

$$T = N \times \left(\frac{L}{R}\right) = 5 \times \left(\frac{7{,}5\ Mbit}{1{,}5\frac{Mbit}{s}}\right) = 25\ s$$

- *Secondo il meccanismo di "store-and-forward", un messaggio di 7,5 Mbit deve passare per 4 router e 5 link ciascuno con ampiezza di banda 1,5 Mbit/s per andare da host sorgente a destinazione. Quanto tempo impiega approssimativamente (a meno di errore inferiore allo 0,1%) il trasferimento <u>se il messaggio è frammentato in 5.000 pacchetti da 1.500 bit l'uno</u>?*

  - **Secondo il meccanismo di store-and-forward il <u>file frammentato</u> non deve necessariamente aspettare l'arrivo di tutti i suoi pacchetti prima di essere trasferito al router successivo, dobbiamo quindi calcolare l'istante di arrivo dell'ultimo pacchetto.**
  - **T1 è il tempo impiegato dal primo pacchetto a raggiungere il primo router**

$$T1 = \left(\frac{L}{R}\right) = \left(\frac{0{,}0015\ Mbit}{1{,}5\frac{Mbit}{s}}\right) = 1\ ms$$

  - **Quindi per arrivare in fondo ha bisogno di 5 ms (5 link * 1 ms)**
  - **L'ultimo pacchetto dei 5000 da trasferire sarà trasferito quindi all'istante "5000*1 ms" —> 5000 ms —> 5 s (dopo 5 secondi che sarà partito il primo pacchetto), e arriverà a destinazione 5 ms dopo (il tempo di oltrepassare i 5 link)**
  - **Il tempo totale di trasmissione (transmission delay) sarà dato da 5s + 5ms, ovvero 5,005 secondi**

- *In una rete con ampiezza di banda R=100 Mbps, circolano pacchetti della dimensione di L=1500 byte. A quale numero minimo di pacchetti per secondo il ritardo di coda è senz'altro dominante sui ritardi elaborazione, propagazione e trasmissione?*

  - **Il ritardo di coda è dato da (L*a)/R, e se superiore a 1 i pacchetti vengono persi, se già è intorno all'1 è significativo e dominante**
  - **Dati L e R possiamo estrapolare a, ponendo l'equazione = 1**

$$\frac{L \times a}{R} = 1 \text{ —> } \frac{12.000\ bit \times a}{100.000.000\ bit/s} = 1$$

$$a = \frac{R}{L} = \frac{100.000.000\ bit/s}{12.000\ bit} = 8.333{,}333...\ pac/sec$$

Ovvero il limite di velocità a cui possono arrivare i pacchetti senza incorrere in una packet-loss

- *Se la velocità di propagazione del segnale nel mezzo è di 300 milioni di metri al secondo, l'ampiezza di banda del link è di 12 Mbps e la dimensione del pacchetto è di 1.500 byte, calcolare la distanza a cui il ritardo di propagazione è uguale a quella di trasmissione*
  - **Ritardo di trasmissione = (L/R)**
  - **Ritardo di propagazione = (d/v)**

$$\frac{L}{R} = \frac{d}{v} \text{ —> } \frac{12.000\ bit}{12.000.000\ bit/s} = \frac{d}{300.000.000\ m/s}$$

$$d = v \times \frac{L}{R} = 300.000.000\ \frac{m}{s} \times \frac{12.000\ bit}{12.000.000 \frac{bit}{s}} = 300.000\ m = 300\ km$$

- *Se il ritardo di propagazione di un pacchetto di 1500 byte è di 60 μs, quale deve essere l'ampiezza di banda per avere un ritardo di trasmissione uguale?*
  - **Ritardo di propagazione = (d/v)**
  - **Ritardo di trasmissione = (L/R) —> R = Ampiezza di banda**

$$\frac{L}{R} = \frac{d}{v} \text{ —> } \frac{12.000\ bit}{R} = 60\,\mu s$$

$$R = \frac{L}{\frac{d}{v}} = \frac{12.000\ bit}{0{,}00006\ s} = 200.000.000\ \frac{bit}{s} = 200\ \frac{Mbit}{s}$$

- *Se la somma dei ritardi di trasmissione e di propagazione è di 13 μs per un pacchetto di 1500 byte, su un tratto di 900 m, sapendo che il segnale si propaga alla velocità di 300.000 km/s, calcolare l'ampiezza di banda del collegamento*
  - **d/v + L/R = 0,000013 s**
  - **L = 1500 byte = 12.000 bit**
  - **d = 900 m**
  - **v = 300.000.000 m/s**

$$\frac{d}{v} + \frac{L}{R} = 0{,}000013\ s$$

- $\frac{900\ m}{300.000.000\frac{m}{s}} + \frac{12.000\ bit}{R} = 0{,}000013\ s$
- $0{,}000003\ s + \frac{12.000\ bit}{R} = 0{,}000013\ s$
- $\frac{12.000\ bit}{R} = 0{,}000013\ s - 0{,}000003\ s$

$$R = \frac{12.000\ bit}{0{,}000010\ s} = 1.200.000.000\ \frac{bit}{s} = 1{,}2\ Gbps$$

- *Se la velocità di propagazione del segnale nel mezzo è di 300 milioni di metri al secondo, l'ampiezza di banda nel link è di 1 Gbps e la dimensione del pacchetto è di 500 byte, calcolare la distanza a cui il ritardo di propagazione è uguale a quello di trasmissione*
  - **v = 300.000.000 m/s**
  - **R = 1 Gbps = 1.000.000.000 bit/s**
  - **L = 500 byte = 6.000 bit**

$$\frac{L}{R} = \frac{d}{v} \text{ -> } \frac{6.000\ bit}{1.000.000.000\ bit/s} = \frac{d}{300.000.000\ m/s}$$

$$d = v \times \frac{L}{R} = 300.000.000\ \frac{m}{s} \times \frac{4.000\ bit}{1.000.000.000\frac{bit}{s}} = 1.200\ m = 1{,}2\ km$$

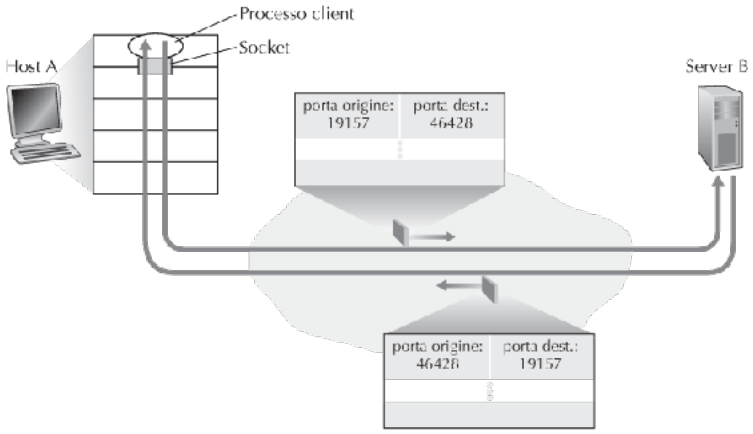
2. Livello di Trasporto

2.1 Introduzione e servizi a livello di trasporto

- Un protocollo a livello di trasporto mette a disposizione una **comunicazione logica** tra processi applicativi di host differenti
- Per comunicazione logica si intende che tutto proceda come se **gli host che eseguono i processi fossero direttamente collegati**
- Il livello di trasporto converte i messaggi che riceve da un processo applicativo in pacchetti a livello di trasporto, i **segmenti**
- In Internet esistono due **protocolli a livello di trasporto**
  - **TCP (Transmission Control Protocol)**
    - *Trasferimento affidabile*
    - *Controllo di congestione:* permette alle proprie connessioni di attraversare un collegamento di rete congestionato in modo da **condividere equamente la larghezza di banda** del collegamento stesso
  - **UDP (User Datagram Protocol)**
    - *Trasferimento non affidabile*
    - *Le applicazioni che usano UDP possono spedire a qualsiasi velocità desiderata e per tutto il tempo voluto*
- Protocollo IP (rete) —> ***Best-Effort***
  - *IP non garantisce la consegna dei pacchetti tra due host* —> ***Servizio non affidabile***
- **I protocolli di trasporto TCP e UDP** estendono il servizio di connessione logica dato da IP “tra sistemi periferici” a quello di **“tra processi in esecuzione su sistemi periferici”** —> ***(Da host-to-host a process-to-process)***
  - **MULTIPLEXING E DEMULTIPLEXING a livello di trasporto**

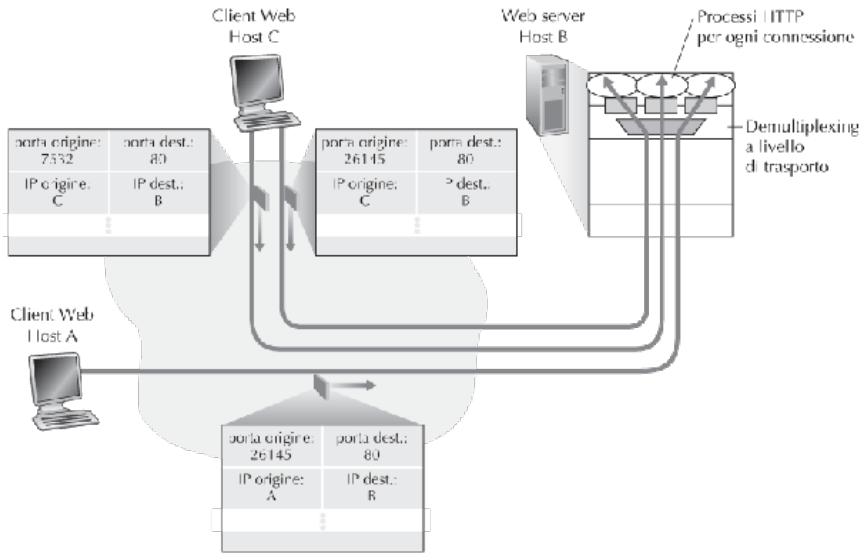
2.2 Multiplexing e Demultiplexing

- I processi in rete comunicano attraverso le **socket**, attraverso le quali i dati fluiscono dalla rete al processo e viceversa
  - **Demultiplexing**
    - Trasporto dati dei **segmenti** verso la giusta socket
      - L'host riceve *datagrammi IP*
        - Ogni datagramma ha un *indirizzo IP sorgente* e un *indirizzo IP destinazione*
        - Ogni datagramma porta **un solo segmento** dello strato di trasporto
        - Ogni segmento ha *numero di porta sorgente* e *numero di porta destinazione*
      - L'host usa **indirizzo IP e numeri di porta** per indirizzare il segmento alla socket appropriata
  - **Multiplexing**
    - Radunare frammenti di dati da diverse socket sull'host origine e **incapsulare** ognuno con intestazioni a livello di trasporto e **passarli al livello di rete**
- ***Multiplexing e Demultiplexing senza connessione (UDP)***
  - **Socket UDP identificata da:**
    - Indirizzo IP
    - Numero di porta
  - *Di conseguenza, se due segmenti UDP presentano diversi indirizzi IP e/o diversi numeri di porta di origine, ma hanno lo stesso indirizzo IP e lo stesso numero di porta destinazione,* ***saranno diretti allo stesso processo di destinazione*** *tramite la medesima socket*



- *Quando un host riceve un segmento UDP:*
  - Controlla numero di porta destinazione del segmento
  - Dirige segmento UDP alla socket con quel numero di porta
- *Datagrammi IP con stesso numero porta destinazione, ma differenti indirizzi IP sorgente e/o numeri di porta sorgente* ***vengono indirizzati alla stessa socket***

- ***Multiplexing e Demultiplexing orientati alla connessione (TCP)***
  - **Socket TCP identificata da:**
    - Indirizzo IP origine
    - Numero di porta origine
    - Indirizzo IP destinazione
    - Numero di porta destinazione
  - **Demultiplexing (orientato alla connessione)**
    - *Due segmenti TCP in arrivo, aventi indirizzi IP di origine o numeri di porta di origine diversi,* ***vengono diretti a due socket differenti****, anche a fronte di indirizzi IP e porta destinazione uguale*



2.3 Trasporto non orientato alla connessione: UDP

- **Senza connessione**
  - Nessun *handshaking* tra sender e receiver UDP
  - Ogni segmento UDP trattato indipendentemente dagli altri
- **Vantaggi**
  - Non occorre stabilire la connessione *(potrebbe causare ritardi)*
  - Nessuno stato di connessione
  - Minor spazio usato per l'intestazione del pacchetto
  - Nessun controllo di congestione: un pacchetto può essere sempre inviato
- *Le applicazioni **possono** ottenere un **trasferimento di dati affidabile** anche con UDP se l'affidabilità è insita nell'**applicazione** stessa*

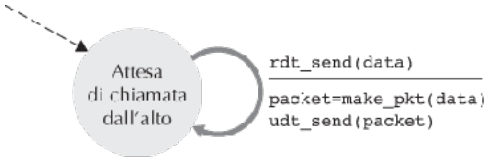
2.3.1 Checksum

- Serve per **rilevare gli errori**, viene utilizzato per determinare se i bit del segmento UDP sono stati alterati durante il loro trasferimento da sorgente a destinazione
- *Sender*
  - Effettua il complemento a 1 sella somma di tutte le parole da 16 bit nel segmento, e l'eventuale riporto viene sommato al primo bit —> Il risultato viene posto nel campo checksum del segmento UDP
- *Receiver*
  - Calcola la checksum del segmento ricevuto
  - Controlla se la checksum calcolata è uguale al valore estratto dal campo di checksum del segmento UDP ricevuto

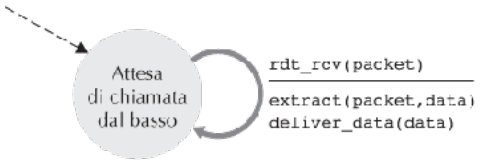
2.4 Principi del trasferimento di dati affidabile

- Nessun bit corrotto
- Nessun bit perso
- Tutti i bit consegnati nell'ordine di invio

2.4.1 Trasferimento dati affidabile su canale affidabile: *rdt 1.0*



a. rdt1.0: lato mittente

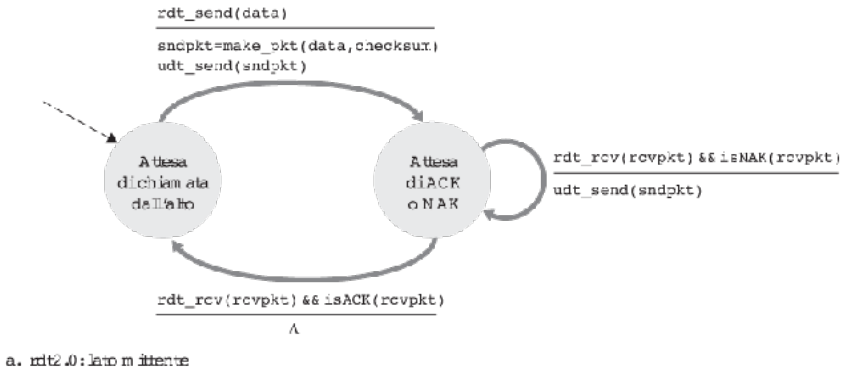


b. rdt1.0: lato ricevente

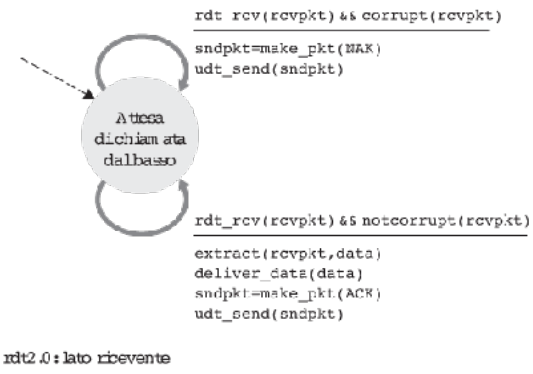
- *Mittente*
  - Accetta dati dal livello superiore
  - Crea un pacchetto contenente dati
  - Lo invia sul canale
- *Ricevente*
  - Raccoglie i pacchetti dal sottostante canale
  - Estrae i dati dai pacchetti
  - Li passa al livello superiore

2.4.2 Trasferimento dati affidabile su un canale con errori sul bit: *rdt 2.0*

- Notifiche di **acknowledgement positive e negative**
  - Protocolli **ARQ** *(Automatic Repeat Request)*
    - ***Rilevamento dell'errore***
    - ***Feedback del destinatario***
      - *ACK e NAK*
    - ***Ritrasmissione***



a. rdt2.0: lato mittente



b. rdt2.0: lato ricevente

- *Mittente*
  - Attesa dati da raccogliere livello superiore
  - Crea pacchetto contenente i dati da inviare, **insieme al checksum**
  - Spedisce il pacchetto
  - **Attende un ACK o un NAK dal destinatario**
    - Se riceve un ACK, il protocollo ritorna allo stato di attesa iniziale
    - Se riceve un NAK, il protocollo ritrasmette l'ultimo pacchetto e rimane in attesa di un altro riscontro dalla ritrasmissione

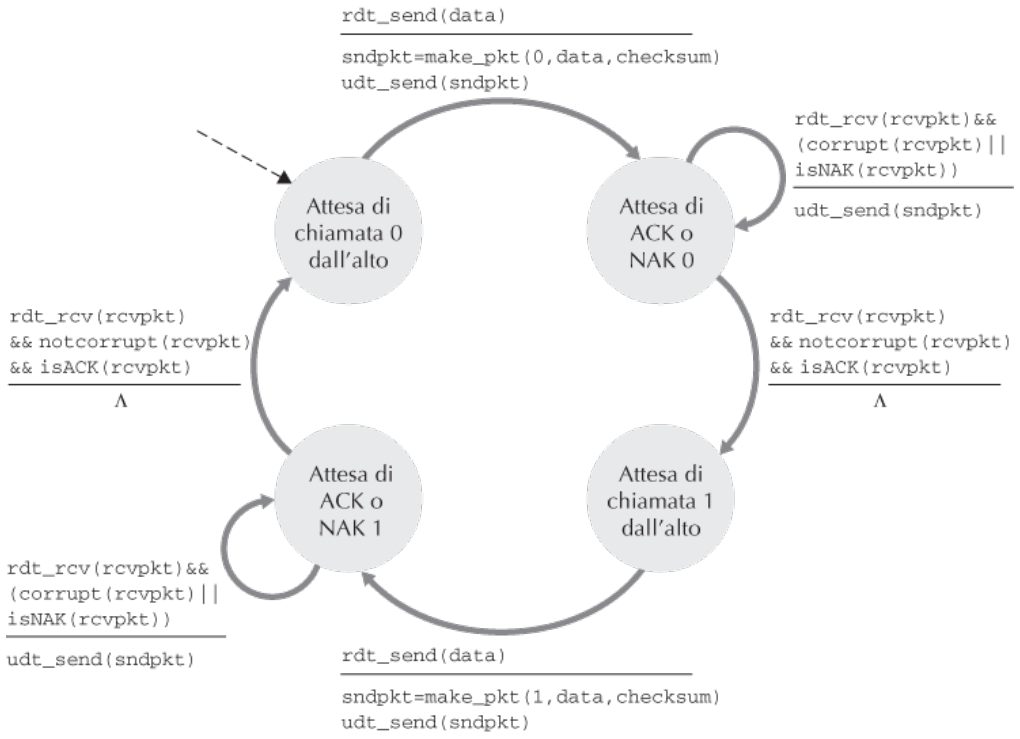
- *Destinatario*
  - All’arrivo del pacchetto, il destinatario risponde con un ACK o un NAK, a seconda che il pacchetto sia corrotto o meno (**in base alla checksum**)

NB: *Quando il mittente è il stato di attesa di ACK o NAK,* ***non può*** *recepire dati dal livello superiore —>* **Protocolli stop-and-wait**

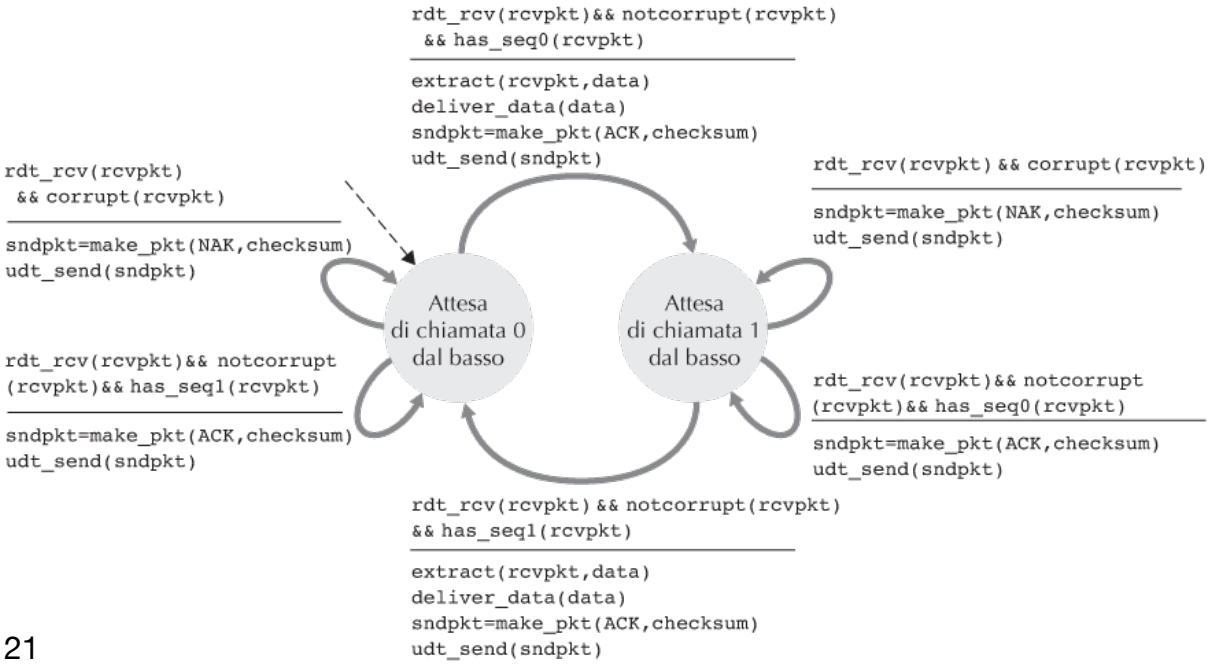
Esiste una possibilità che i pacchetti ACK e NAK possano a loro volta essere corrotti

Viene introdotto quindi l'obbligo per il mittente di "numerare i pacchetti", utilizzando un solo bit perché stiamo ipotizzando che non esista perdita di pacchetti (0 e 1), che permette al destinatario di stabilire se il pacchetto sia una ritrasmissione o meno

Mittente rdt 2.1

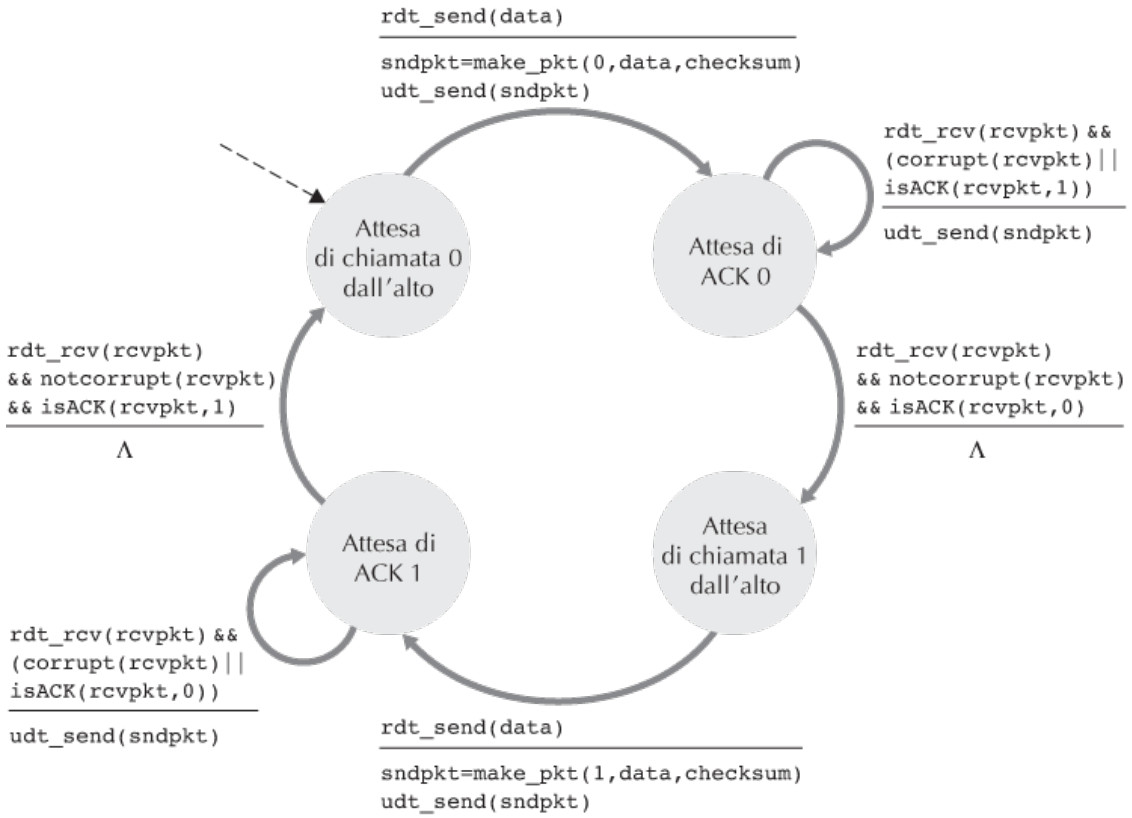


Destinatario rdt 2.1

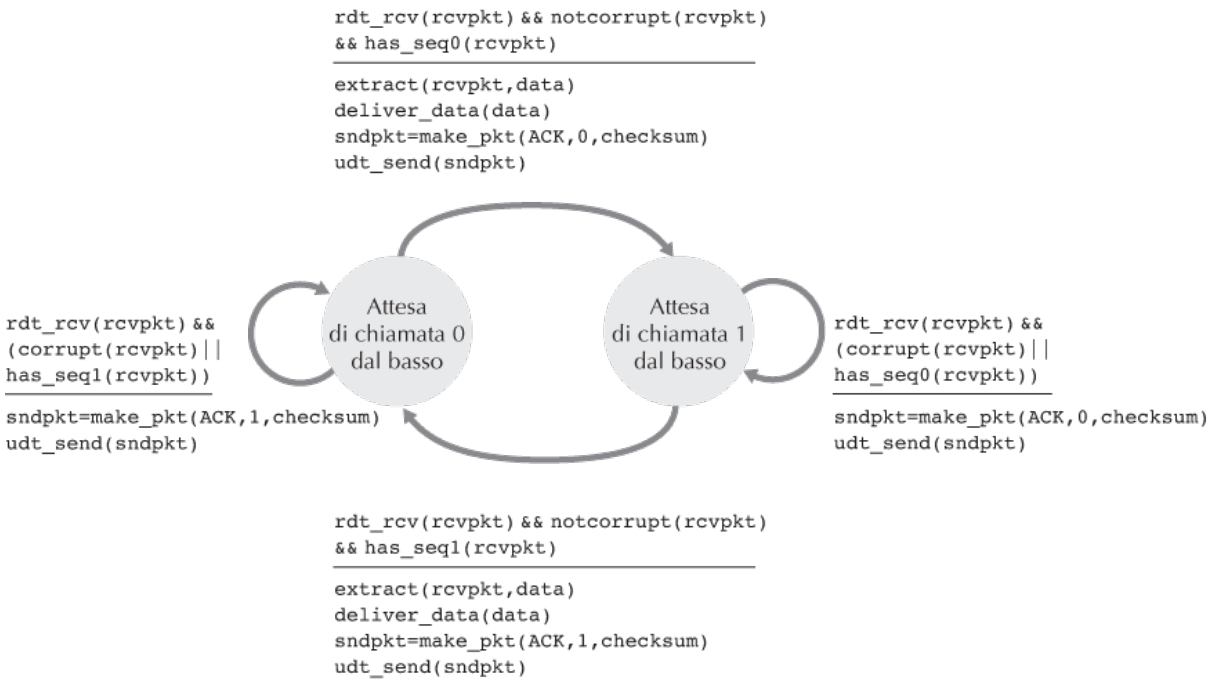


rdt 2.2 (inserimento del bit di sequenza nell'ACK destinatario)

- *Mittente*



- *Destinatario*



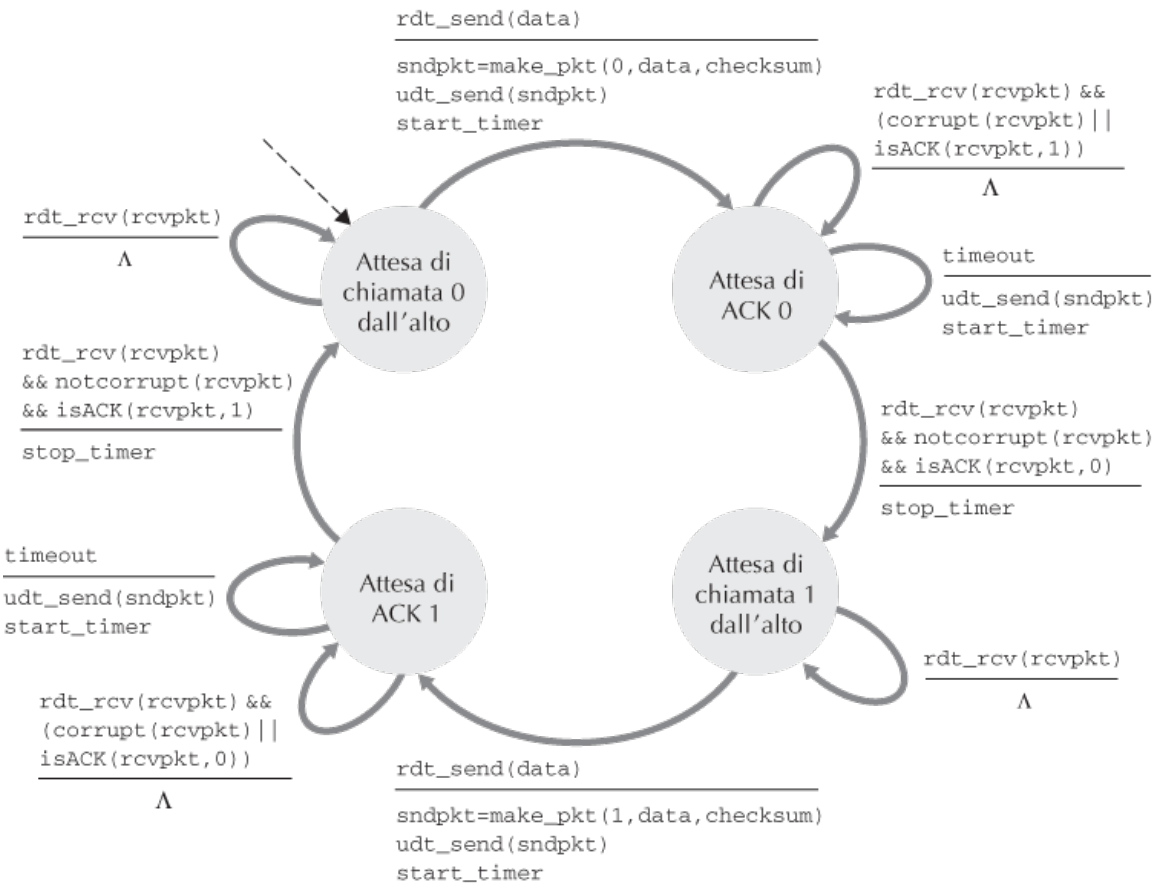
2.4.3 Trasferimento dati affidabile su canale con perdite ed errori sui bit: *rdt 3.0*

Viene introdotto un timer (countdown timer) che il mittente inizializza ogni volta che invia un pacchetto

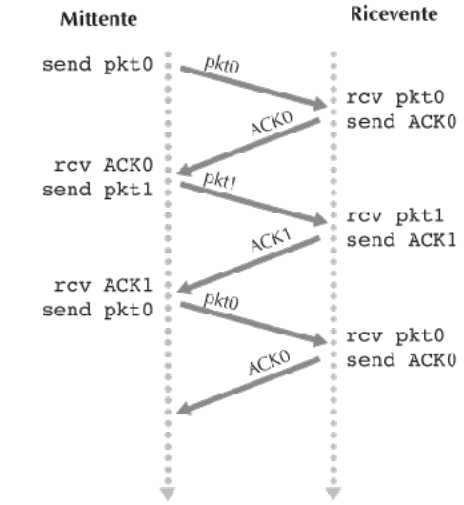
Di conseguenza il mittente dovrà essere in grado di:

- **Inizializzare il timer**
- **Rispondere ad un interruppe generato dal timer**
- **Fermare il timer**

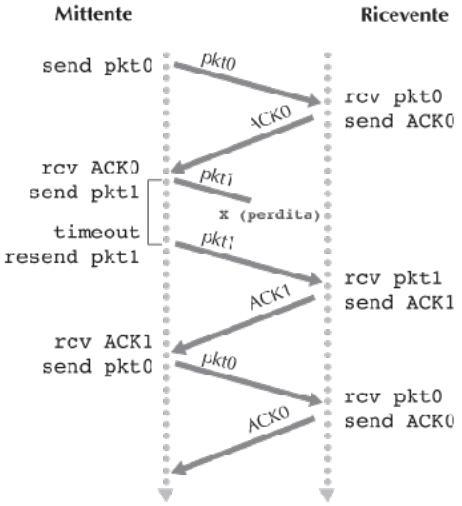
- *Mittente rdt 3.0*



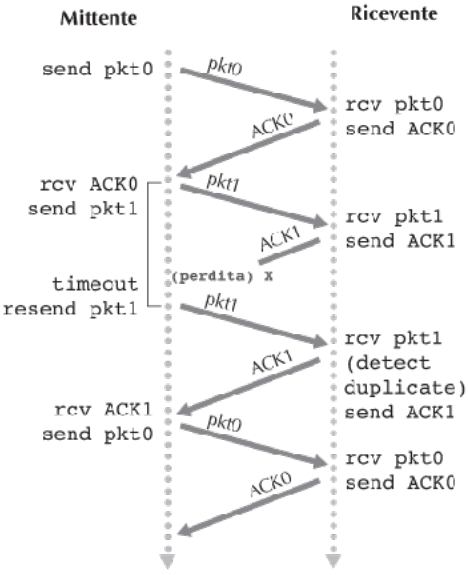
Operazioni di rdt 3.0



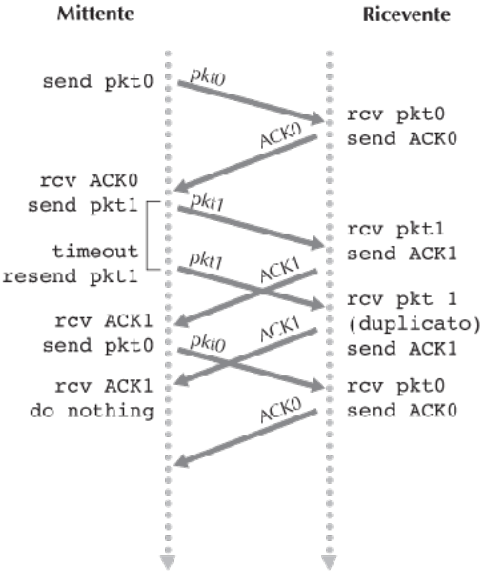
a. Operazioni senza perdite



b. Perdita di pacchetto



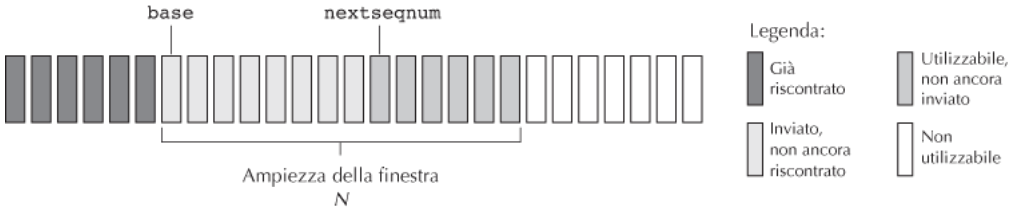
c. Perdita di ACK



d. Timeout prematuro

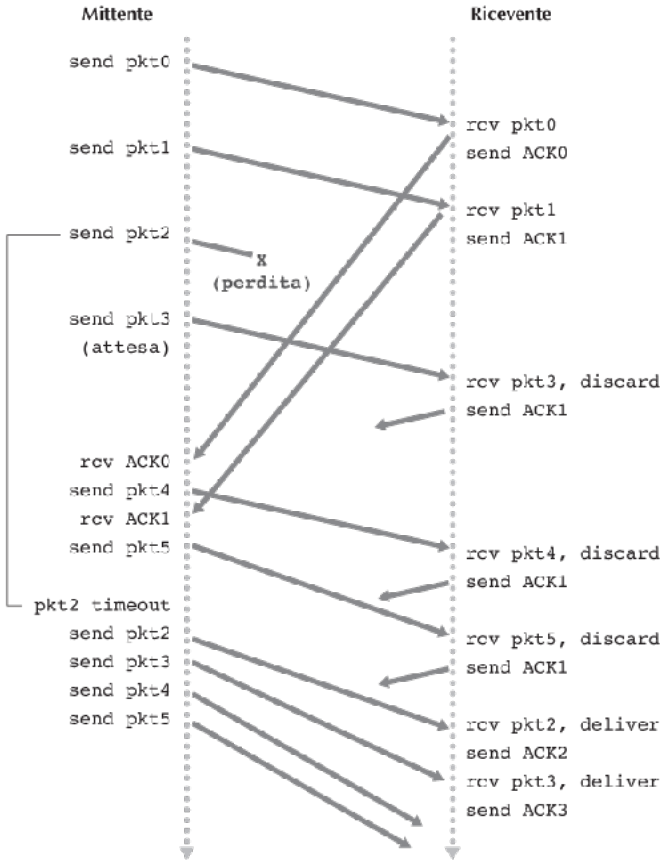
2.5 Protocolli con pipeline (*non stop-and-wait)*

2.5.1 GO-BACK-N

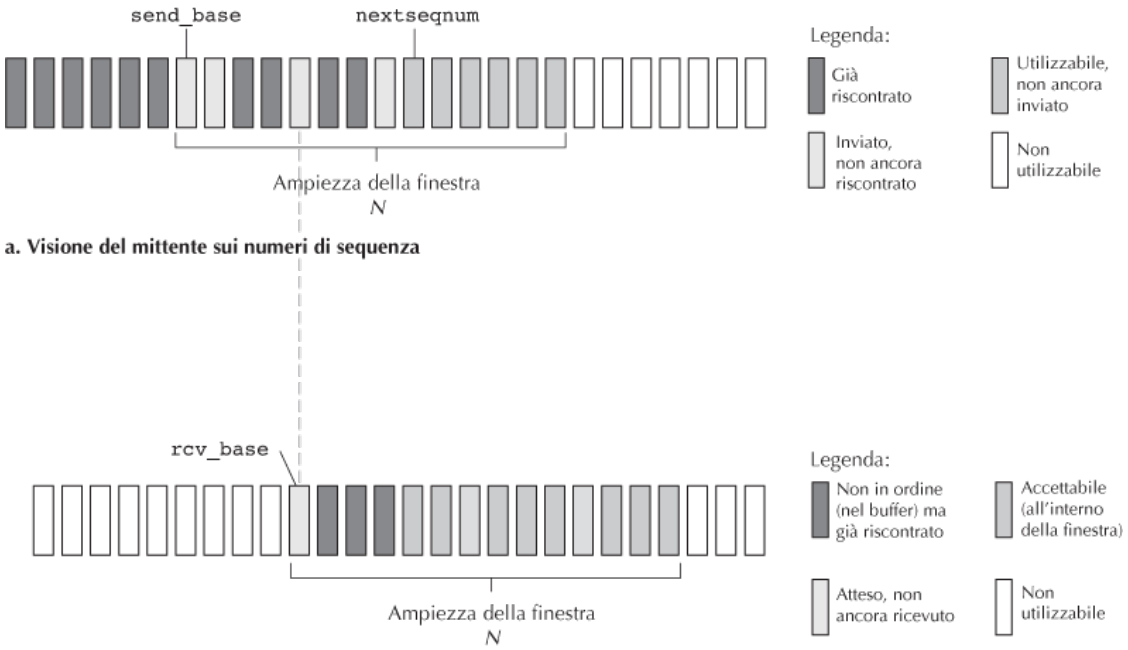


- Sender può avere fino a N pacchetti in volo senza riscontro
- Receiver invia solo **ACK cumulativi**
  - **Invia ACK dell'ultimo pacchetto ricevuto correttamente se arriva un pacchetto fuori sequenza** *(aspetto pacchetto n, arriva pacchetto n+1, invio ack(n))*
- Sender ha **timer** per il pacchetto più vecchio senza ACK

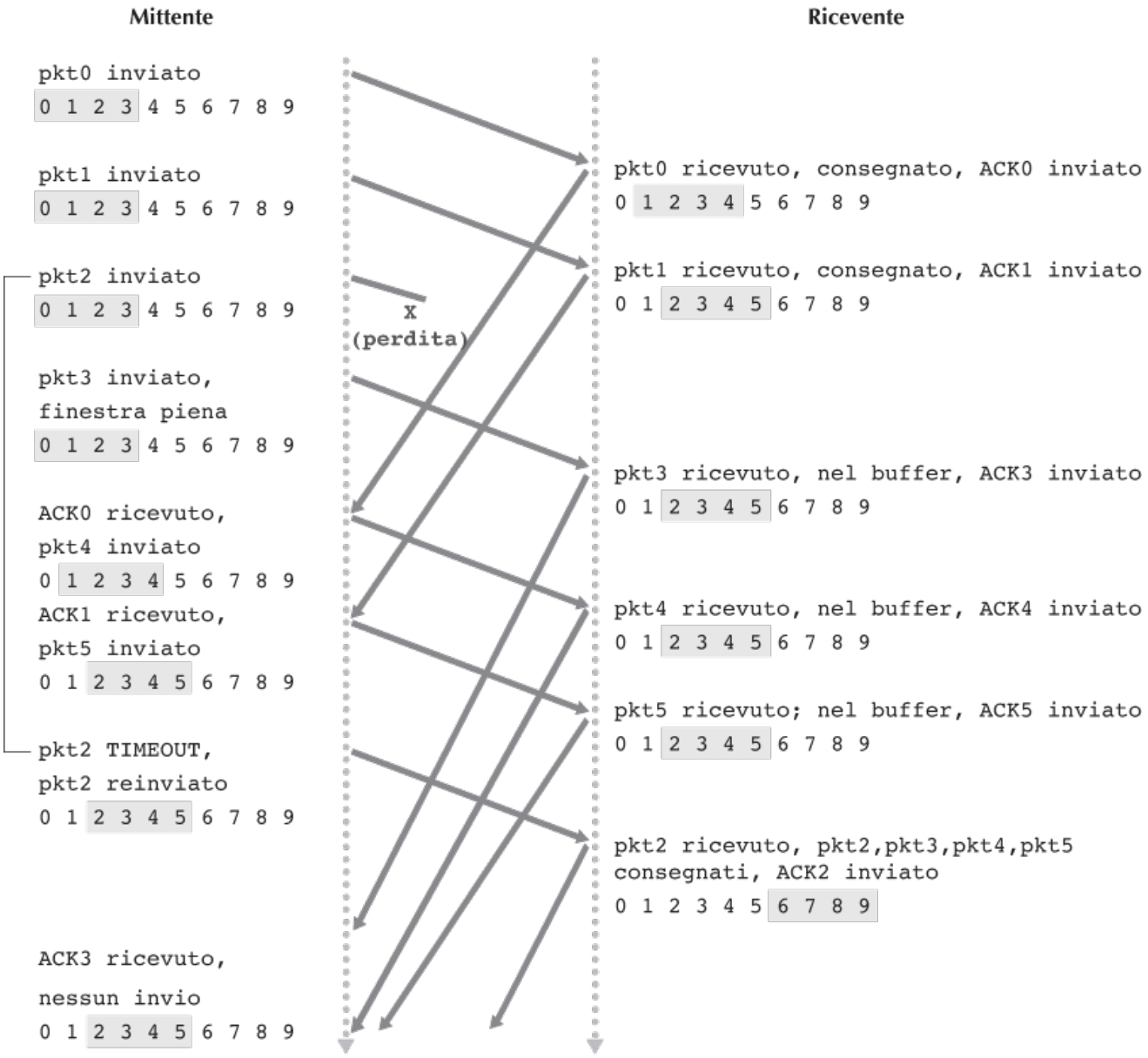
In generale, se un pacchetto con numero di sequenza **n** viene ricevuto correttamente ed è in ordine il destinatario manda un **ACK** per quel pacchetto e consegna i suoi dati al livello superiore. **In tutti gli altri casi, il destinatario scarta il pacchetto e rimanda un ACK per il pacchetto in ordine ricevuto più di recente**. Quando il timer del pacchetto perduto scade, il mittente ritrasmette tutti i pacchetti partendo da quello che corrisponde all'ultimo ACK ricevuto.



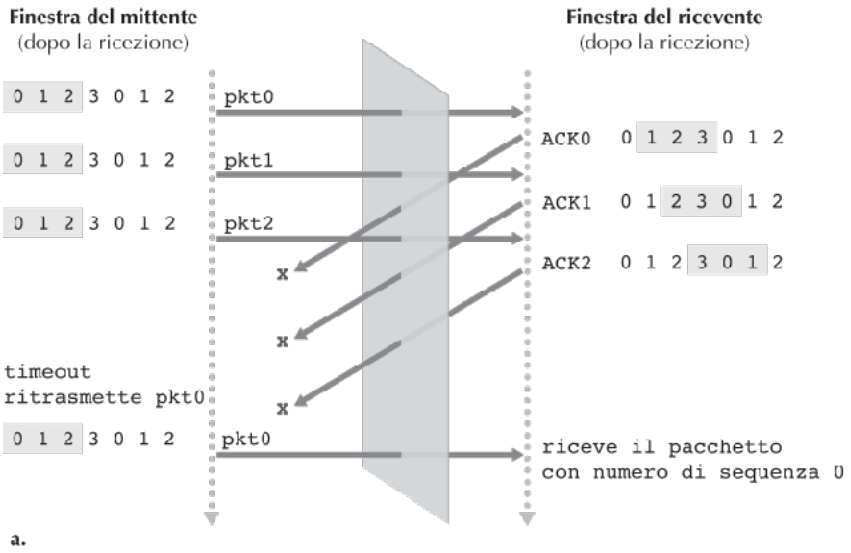
2.5.2 Ripetizione selettiva



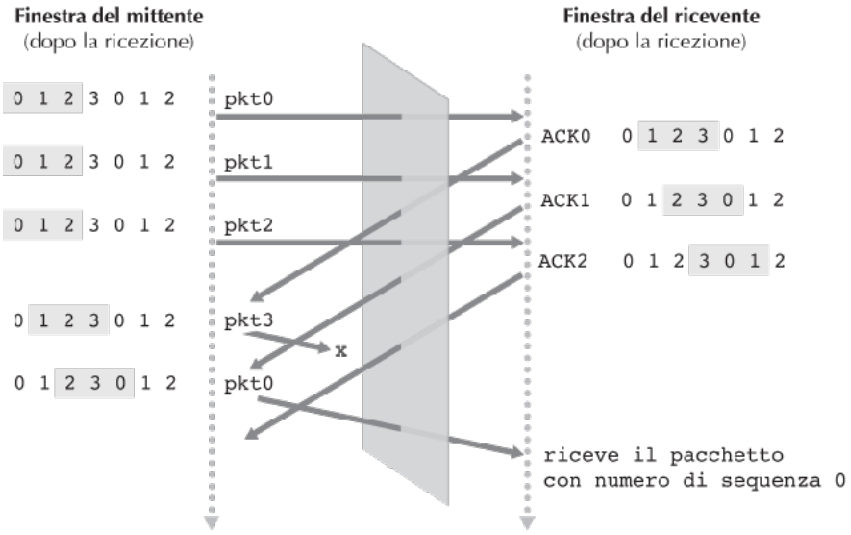
- *Mittente*
  - Se **disponibile** un **posto in finestra**, **invia pacchetto**
  - Se va in **timeout** un pacchetto lo ri-invia
  - Se riceve un **ACK**:
    - Se era **il più piccolo pacchetto senza avere ancora ricevuto un ACK**, **avanza la base della finestra fino al successivo pacchetto senza ACK**
- *Destinatario*
  - Riceve pacchetto **in finestra**
    - **Invia ACK**
    - **Controlla numero di sequenza pacchetto**
      - Se in ordine: **consegna**, **avanza finestra al successivo pacchetto non ricevuto**
      - Se non in ordine: **salva nel buffer il pacchetto in attesa degli precedenti mancanti**
  - Riceve pacchetto **non in finestra**:
    - **Invia ugualmente ACK del pacchetto**


Mittente
Ricevente
pkt0 inviato
0 1 2 3 4 5 6 7 8 9
pkt1 inviato
0 1 2 3 4 5 6 7 8 9
pkt2 inviato
0 1 2 3 4 5 6 7 8 9
X
(perdita)
pkt3 inviato,
finestra piena
0 1 2 3 4 5 6 7 8 9
ACK0 ricevuto,
pkt4 inviato
0 1 2 3 4 5 6 7 8 9
ACK1 ricevuto,
pkt5 inviato
0 1 2 3 4 5 6 7 8 9
pkt2 TIMEOUT,
pkt2 reinviato
0 1 2 3 4 5 6 7 8 9
ACK3 ricevuto,
nessun invio
0 1 2 3 4 5 6 7 8 9
pkt0 ricevuto, consegnato, ACK0 inviato
0 1 2 3 4 5 6 7 8 9
pkt1 ricevuto, consegnato, ACK1 inviato
0 1 2 3 4 5 6 7 8 9
pkt3 ricevuto, nel buffer, ACK3 inviato
0 1 2 3 4 5 6 7 8 9
pkt4 ricevuto, nel buffer, ACK4 inviato
0 1 2 3 4 5 6 7 8 9
pkt5 ricevuto; nel buffer, ACK5 inviato
0 1 2 3 4 5 6 7 8 9
pkt2 ricevuto, pkt2,pkt3,pkt4,pkt5
consegnati, ACK2 inviato
0 1 2 3 4 5 6 7 8 9

Attenzione ai casi particolari!



a.



b.

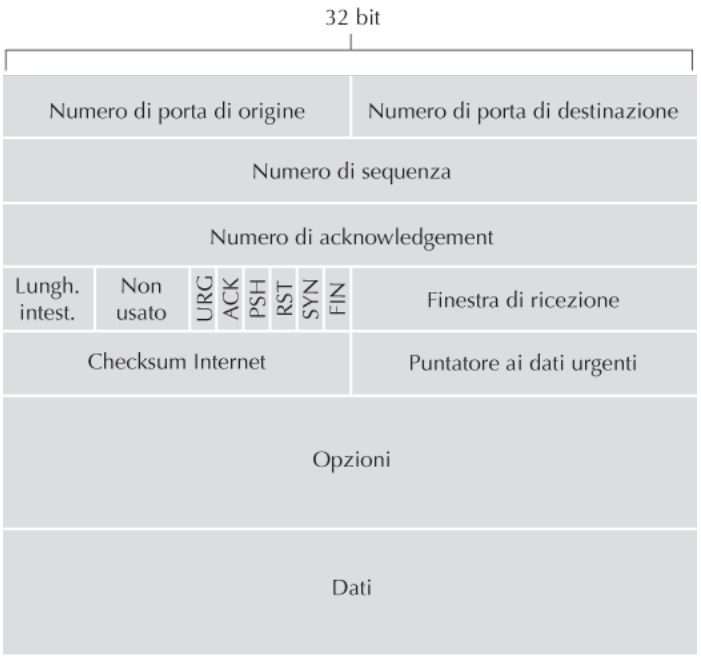
2.6 Trasporto orientato alla connessione: TCP

2.6.1 Connessione TCP

- **Orientato alla connessione**
  - Prima di effettuare lo scambio dei dati i processi effettuano un **handshake** *(three-way handshake)*
- Servizio **full-duplex**
- Connessione **punto a punto**
- *Presenza di un **buffer di invio** (riservato durante l'handshake)*

- **MSS** *(Maximum segment size)*
  - Massima quantità di dati prelevabili e posizionabili in un segmento
- **MTU** *(Maximum transmission unit)*
  - Lunghezza del frame più grande

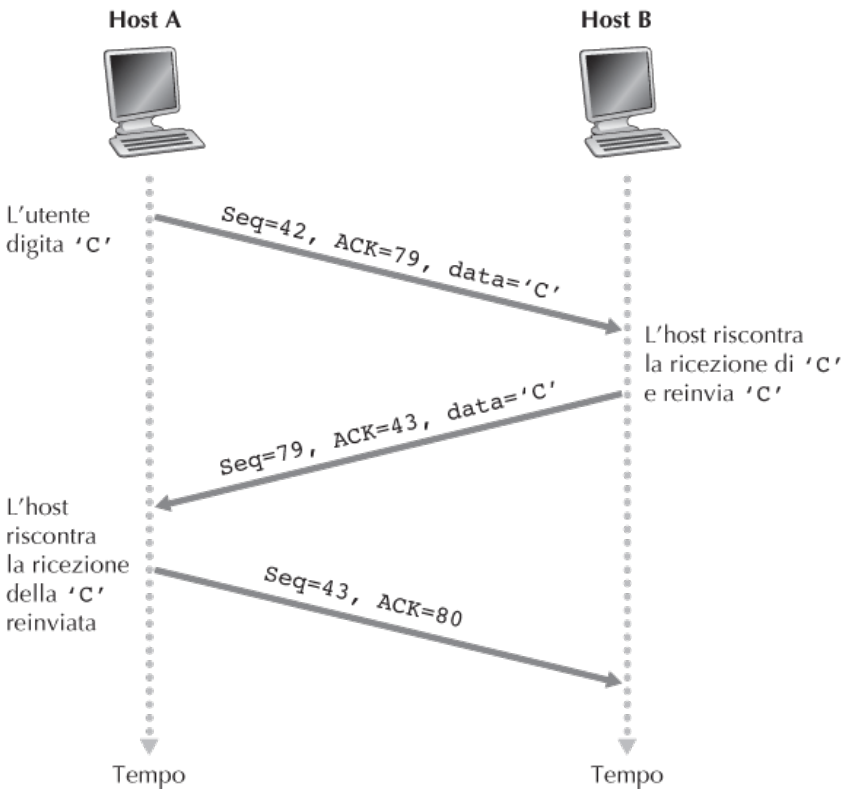
2.6.2 Struttura segmenti TCP



- **Numero di sequenza**
- **Numero di ACKnowledgement**
- **Finestra di ricezione**
  - Controllo di flusso
- **Lunghezza intestazione**
- **Options**
- **Flag**

2.6.3 Numeri di sequenza e numeri di acknowledgement

- TCP vede i dati come un flusso di byte non strutturati, ma ordinati
- **Numero di sequenza per ogni segmento**:
  - Numero nel flusso di byte del **primo byte del segmento**
    - *Il primo numero di sequenza sia per il server sia per il client è assegnato* ***casualmente***
  - Ogni numero di sequenza viene inserito nel campo numero di sequenza dell'intestazione del segmento TCP appropriato
- **Numero di acknowledgement per ogni segmento:**
  - Il numero di acknowledgement che l'host A scrive nei propri segmenti è il **numero di sequenza del byte successivo che l'host A attende dall'host B**
  - ACK **cumulativi**
    - TCP effettua ACK solo dei byte fino al primo byte mancante nel flusso
- *I segmenti* ***fuori sequenza*** *vengono gestiti in due modi:*
  - Il destinatario **scarta** i segmenti non ordinati
  - Il destinatario **mantiene** i byte non ordinati e attende quelli mancanti per colmare i vuoti



NB: *I caratteri ASCII sono grandi 1 byte*

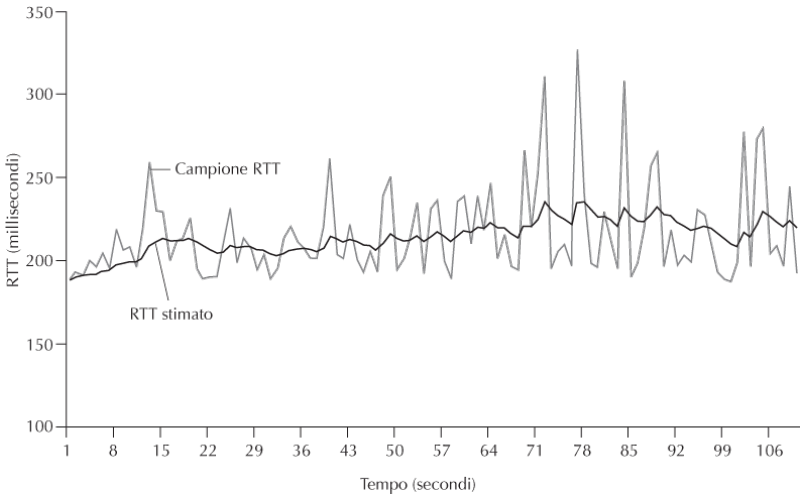
Particolarità: *Il riscontro dei dati dal client al server viene trasportato in un segmento che a sua volta trasporta dati dal server al client —> L'acknowledgement è* ***piggybacked*** *sul segmento dati dal server al client*

2.6.4 Timeout e stima del tempo di andata e di ritorno

- Il **timeout** dovrebbe essere **più grande** del tempo di andata e ritorno sulla connessione *(RTT)*
- RTT di un segmento —> **SampleRTT**
  - Quantità di tempo che intercorre tra l'istante di invio del segmento e quello di ricezione dell'acknowledgement
- **TCP effettua una sola misurazione di SampleRTT alla volta**
  - In ogni istante di tempo, SampleRTT viene valutato per uno solo dei segmenti trasmessi e per cui non si è ancora ricevuto acknowledgement
- TCP **non calcola mai** il SampleRTT per i segmenti **ritrasmessi**

- Media ponderata dei valori di SampleRTT —> **EstimatedRTT**
  - Quando si ottiene un nuovo SAmpleRTT, TCP aggiorna EstimatedRTT secondo la formula

$$EstimatedRTT = [(1-\alpha) \times EstimatedRTT] + [\alpha \times SampleRTT]$$



  - Tale media attribuisce maggiore importanza ai campioni recenti rispetto a quelli vecchi —> **media mobile esponenziale ponderata**
- Variabilità di SampleRTT —> **DevRTT**
  - Stima della variabilità di RTT secondo la formula

$$DevRTT = (1-\beta) \times DevRTT + \beta \times |SampleRTT - EstimatedRTT|$$

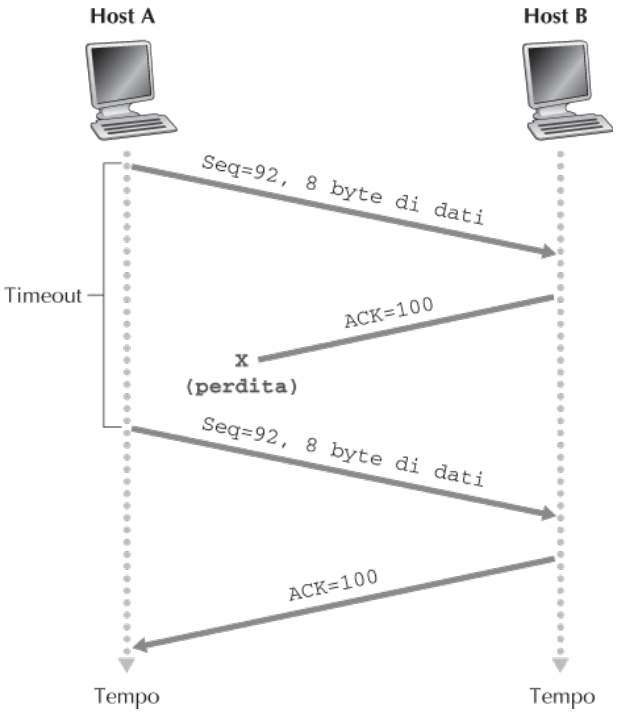
<u>Da qui si riesce a stabilire un valore di timeout per ogni pacchetto trasmesso</u>

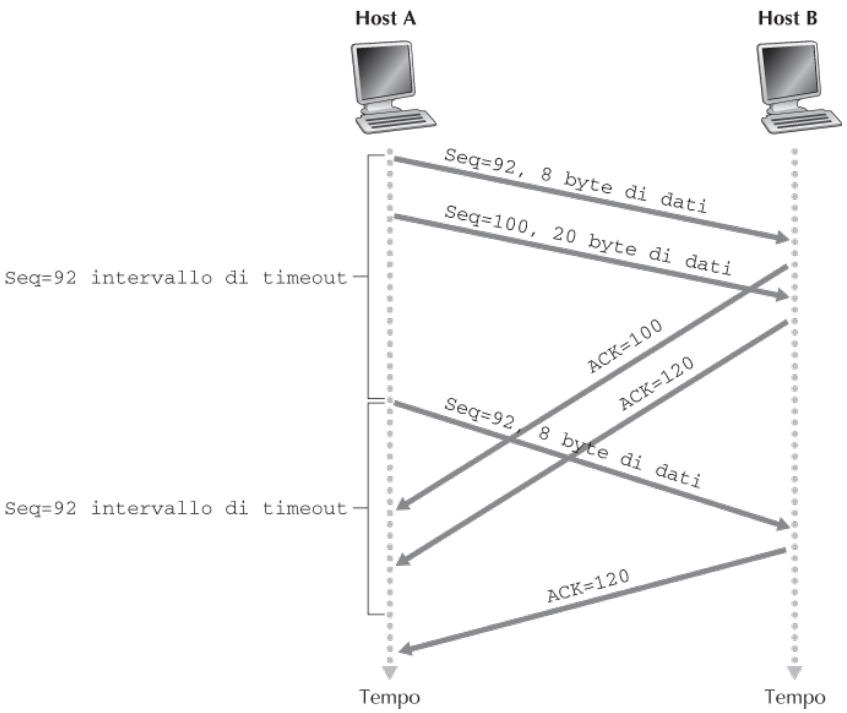
$$TimeoutInterval = EstimatedRTT + 4 \times DevRTT$$

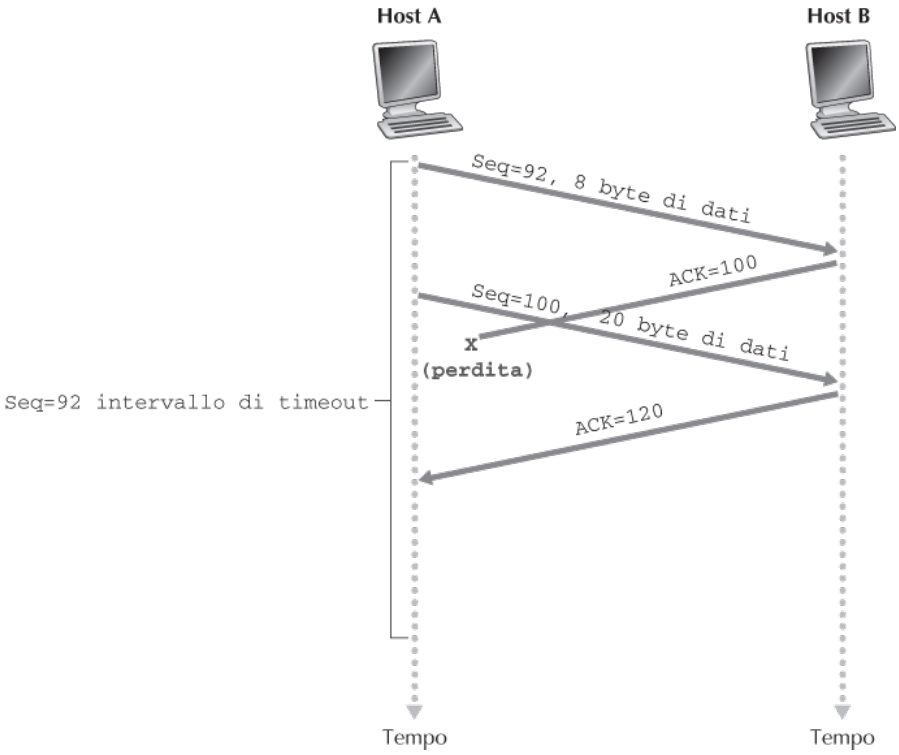
- Viene raccomandato n valore iniziale di TimeoutInterval pari a **1 secondo**
- **Quando si verifica un timeout, TimeoutInterval viene raddoppiato** per evitare un **timeout prematuro**

Quando viene ricevuto un segmento ed EstimatedRTT viene aggiornato, TimeoutInterval viene ricalcolato secondo la formula precedente

Scenari particolari di ritrasmissione





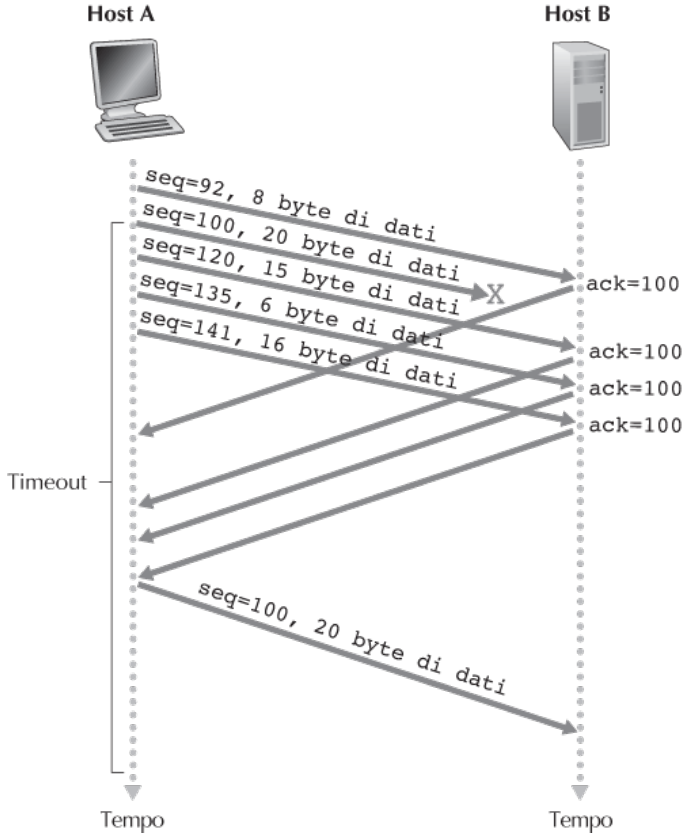


2.6.5 Ritrasmissione rapida

*Il mittente può rilevare la perdita di pacchetti prima che si verifichi un timeout grazie agli **ACK duplicati** relativi a un segmento il cui ACK è stato ricevuto dal mittente*

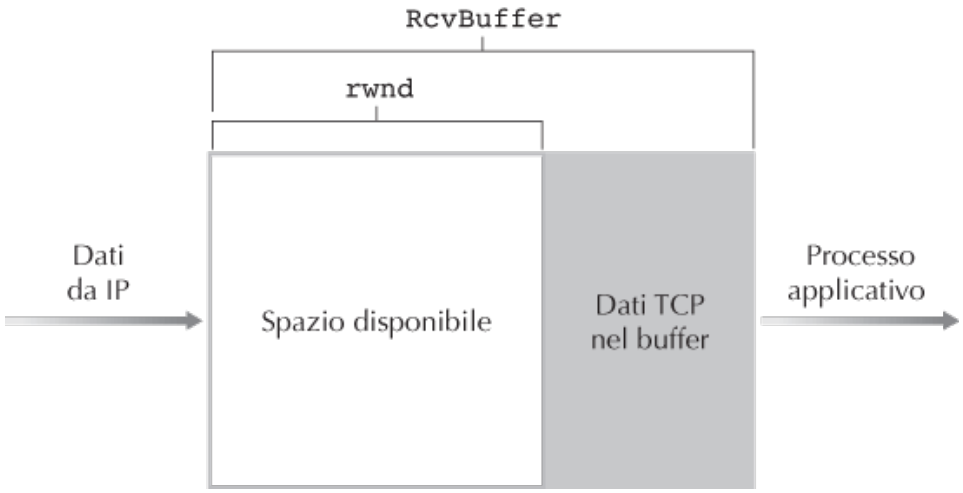
*Quando il destinatario TCP riceve un segmento con numero di sequenza superiore al successivo numero di sequenza atteso e in ordine, **rileva un buco**, ossia un **segmento mancante***

- Se il mittente TCP riceve **3 ACK duplicati** per lo stesso dato, considera questo evento come indice che il segmento che segue è andato perduto
  - **Il mittente TCP effettua una ritrasmissione rapida**
    - *Rispedisce il segmento mancante **prima che scada il timer***



2.6.6 Controllo di flusso

- TCP offre un **servizio di controllo di flusso** alle proprie applicazioni per evitare che il mittente saturi il buffer del ricevente
  - **Confronto** su **frequenza di invio del mittente** con **frequenza di lettura del destinatario**
- I mittenti TCP possono anche essere rallentati dalla congestione nella rete IP
  - **Controllo di congestione**
- **Finestra di ricezione**
  - Fornisce al mittente **un'indicazione dello spazio libero disponibile nel buffer del destinatario**
  - TCP full-duplex —> *Entrambi i mittenti mantengono finestre di ricezione diverse*
    - **LastByteRead**
      - Numero dell'**ultimo byte** nel flusso di dati **che il processo applicativo in B ha letto dal buffer**
    - **LastByteRcvd**
      - Numero dell'**ultimo byte**, nel flusso di dati, **che proviene dalla rete** e **che è stato copiato nel buffer di ricezione di B**
  - TCP non può mandare in overflow il buffer allocato
    - $LastByteRcvd - LastByteRead \leq RcvBuffer$
  - **Finestra di ricezione = Quantità disponibile nel buffer**
    - $rwnd = RcvBuffer - [LastByteRcvd - LastByteRead]$



- **<u>L'Host B comunica all'host A quanto spazio disponibile sia presente nel buffer della connessione</u>** (rwnd) —>
  - **Scrive** il **valore corrente di rwnd** nel campo apposito dei segmenti che manda ad A
  - L'Host B **inizializza rwnd** con il valore di **RcvBuffer**
- **<u>L'Host A tiene traccia di due variabili</u>** —>
  - **LastByteSent**: *Ultimo byte mandato*
  - **LastByteAcked**: *Ultimo byte da cui si è ricevuto riscontro*
  - $(LastByteSent - LastByteAcked) \leq rwnd$
- **NB:** *Quando il processo applicativo in B svuota il buffer, TCP non invia nuovi segmenti con nuovi valori di rwnd —> Host A* ***bloccato***
  - L'Host A **deve continuare a inviare segmenti con un byte di dati quando la finestra di ricezione di B è 0.**

2.6.7 Gestione della connessione TCP

Handshake a tre vie

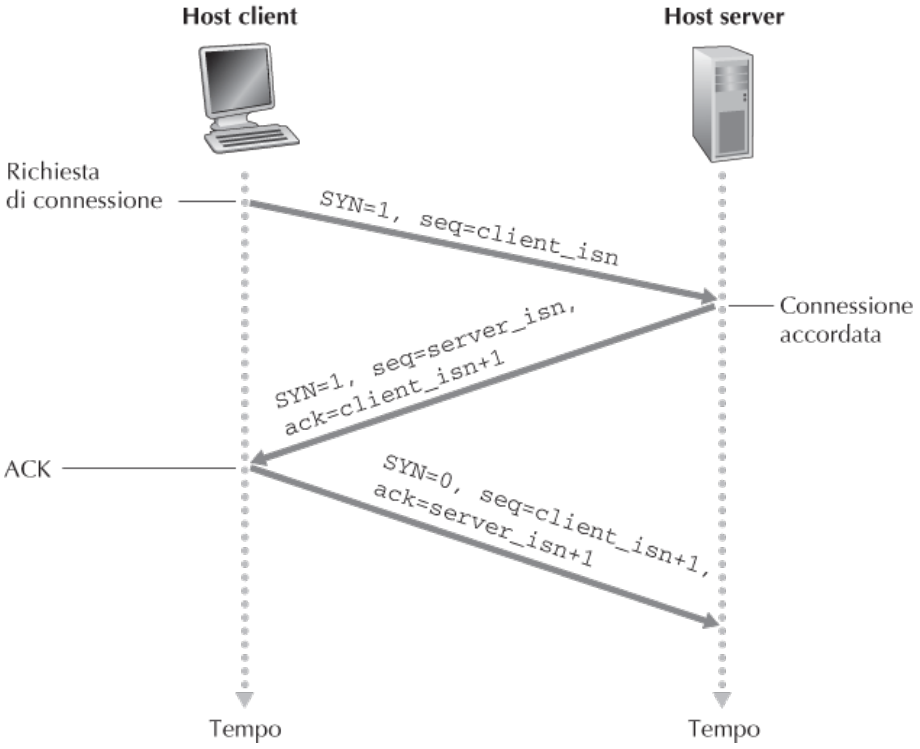
1. TCP **lato client invia** un segmento consentente il flag **SYN = 1** e **sceglie e pone nel segmento un numero di sequenza iniziale casuale** *(client_isn)*
2. Il **server estrae** il segmento, **alloca i buffer e le variabili TCP**, invia un segmento con

   - SYN = 1
   - ACK = client_isn + 1

   e **sceglie un proprio numero di sequenza iniziale** *(server_isn)* ponendolo nel campo di sequenza

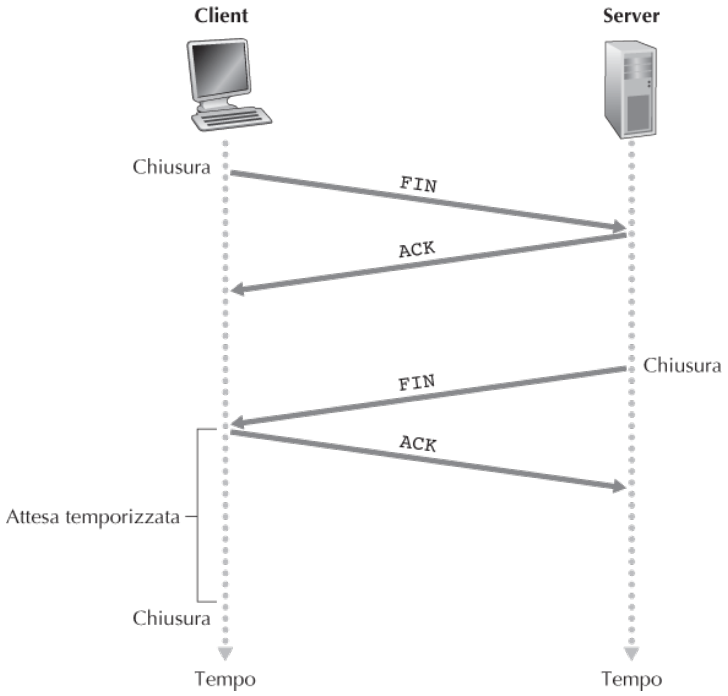
   SEGMENTO SYNACK

3. Alla ricezione del segmento SYNACK, anche il **client alloca buffer e variabili** alla connessione, invia un segmento con ACK = server_isn + 1



Chiusura collegamento

1. Il **client invia** un segmento con **FIN = 1**
2. Il **server estrae** il pacchetto, legge FIN = 1, **e spedisce** a sua volta un segmento con **FIN = 1**
3. Il **client invia** un segmento con un **ACK**



Se il destinatario TCP riceve su segmento TCPSYN con una porta di destinazione chiusa ***risponde con un segmento contenente il bit*** **RST = 1**

2.7 Controllo congestione TCP

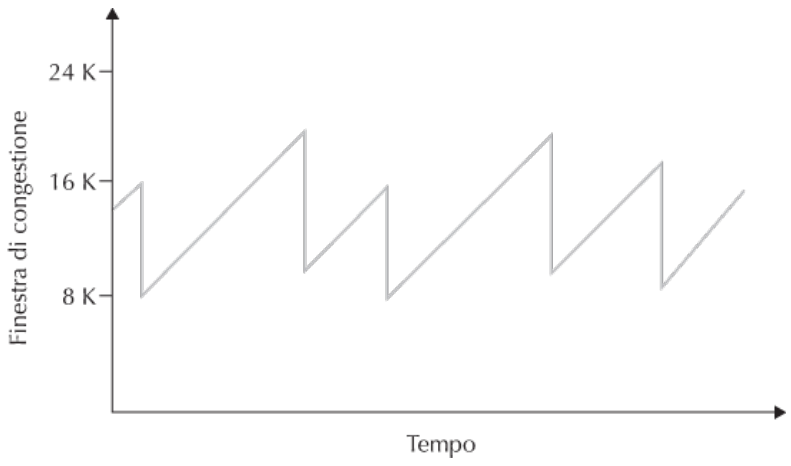
- Imporre a ciascun mittente un limite alla velocità d'invio sulla propria connessione in funzione della congestione di rete percepita
  - Se il mittente TCP si accorge di condizioni di scarso traffico di rete sul percorso per la destinazione, **incrementa il proprio tasso trasmissivo**
  - Se il mittente TCP si accorge di condizioni di alto traffico di rete sul percorso per la destinazione, **diminuisce il proprio tasso trasmissivo**
- **Finestra di congestione**
  - La quantità di dati che non hanno ancora ricevuto ACK inviata da un mittente non può eccedere il minimo tra i valori di **cwnd** e **rwnd**

$$LastByteSent - LastByteAcked \leq min\{cwnd, rwnd\}$$

- *Un segmento perso implica congestione, quindi i tassi di trasmissione del mittente TCP dovrebbe essere decrementi quando un segmento viene perso*
- *Un ACK determina che la rete sta consegnando i segmenti del mittente al ricevente e quindi il tasso di trasmissione del mittente può essere aumentato quando arriva un ACK non duplicato*
- *Rilevamento della larghezza di banda —> Quando viene rilevata la perdita viene percepito implicitamente il valore massimo della larghezza di banda*
- **Algoritmo di controllo di congestione TCP**
  - **Slow Start**
    - **Valore iniziale di MSS/RTT**
    - **Raddoppio della velocità trasmissiva ad ogni RTT**
    - Con un **timeout**
      - **cwnd = 1 MSS**
      - **ssthresh** (variabile di soglia, oltre la quale viene registrata una perdita) **= cwnd/2**
  - **Congestion Avoidance**
    - Quando viene raggiunta **ssthresh** la velocità trasmissiva **aumenta linearmente (1 MSS ogni RTT)**
    - Con un **timeout (Pacchetti persi)**

        - **cwnd = 1 MSS**
        - **ssthresh = cwnd/2**
    - Con **3 ACK duplicati (ACK perduti)**
        - **cwnd = cwnd/2**
        - **ssthresh = cwnd/2**
- **<u>Fast Recovery</u>**
    - **cwnd = cwnd + (1 MSS * ACK duplicato ricevuto)**
    - Con un **ACK**
        - **Congestion Avoidance**
    - Con un **Timeout**
        - **Slow Start**

- **Controllo congestione TCP —> AIMD (Additive-increase multiplicative-decrease)**



- **Throughput TCP**
  - **Generale**
    - **W =** Larghezza finestra [byte] quando si verifica la perdita = larghezza massima finestra
    - **L** = Tasso di perdita pacchetti [pacchetti/s]
    - **T** = Throughput connessione [bit/s]

$$T = \frac{3}{4}\frac{W}{RTT} \qquad\qquad T = \frac{3}{4}\frac{nL}{RTT}$$

  - **Perdita di pacchetti**
    - **T** = Throughput connessione [bit/s]
    - **L** = Tasso di perdita pacchetti [pacchetti/s]
    - **MSS** = Maximum Segment Size [bit]

$$T = \frac{1{,}22MSS}{RTT\sqrt{L}}$$

  - **Modellazione della latenza**
    - **<u>Finestra di congestione statica</u>**
      - **1° caso: Viene ricevuto un ACK prima che siano stati inviati tutti i pacchetti**

$$Latenza = 2RTT + \frac{O}{R}$$

      - **2° caso: Il server trasmette tutti i pacchetti previsti nella finestra prima di ricevere un ACK**

$$Latenza = 2RTT + \frac{O}{R} + (K-1) \times \left(\frac{S}{R} + RTT - \frac{W \times R}{R} \right)$$

    - *K = numero di finestre necessarie per inviare l'oggetto O*
    - *W = numero di segmenti presenti in una finestra*

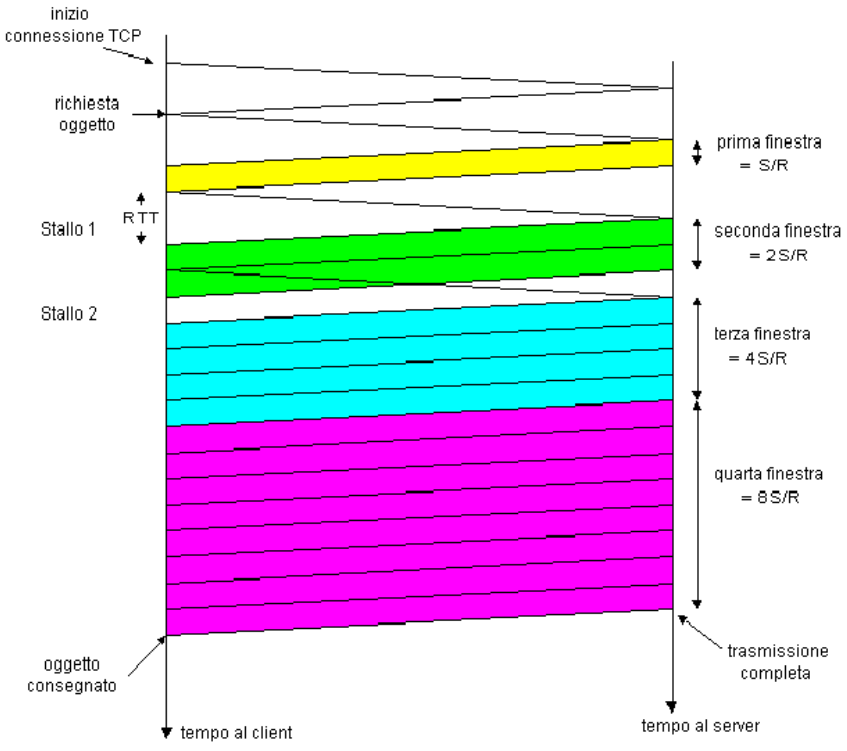
- **Finestra di congestione dinamica**

  - **Slow Start**

$$Latenza = 2RTT + \frac{O}{R} + P \times \left(RTT + \frac{S}{R} \right) - (2^P - 1) \times \left(\frac{S}{R} \right)$$

  - ***P** è il numero di volte che TCP si blocca lato mittente per aspettare che arrivi un ACK e riapra la finestra:*
    - $P = min\{Q, K - 1\}$
  - ***Q** è il numero di volte che il mittente si arresterebbe se l'oggetto fosse di dimensioni infinite. Di solito vale 0.*
    - $Q = 1 + \log_2 \left(RTT \times \frac{R}{S} + 1 \right)$
  - ***K** è il numero di finestre necessario per coprire l'oggetto*
    - $K = \log_2 \left(\frac{O}{S} + 1 \right)$
  - ***S/R** è il tempo necessario per trasmettere un segmento*
  - ***RTT + S/R** è il tempo necessario per ricevere un ACK*

- *O/S = 15 segmenti*
- *K = 4 finestre*
- *Q = 2*
- *P = min{Q, K-1} = 2*



2.8 Fairness

- K = n connessioni TCP
- R = banda collegamento tra le connessioni

Un meccanismo di controllo di congestione è **fair (equo)** se la velocità trasmissiva media di ciascuna connessione è approssimativamente R/K

Le applicazioni che utilizzando connessioni in parallelo ottengono una porzione di banda maggiore sui collegamenti congestionati

2.9 Esercizi

- *Quali tra i seguenti servizi sono forniti sia da TCP che da UDP:*
  - **Controllo di errore ed inoltro dei dati all'applicazione appropriata**
- *I meccanismi di Go-Back-N e di Ritrasmissione Selettiva (SR) hanno in comune:*
  - **La presenza di un buffer con finestra scorrevole su lato sender**
- *Qual è la massima dimensione L di un file - in byte - per la quale i numeri di sequenza TCP non si ripetono mai?*
  - **Il campo sequence number di un segmento TCP ha 32 bit**
  - **2^32 bit = 4 GB**
- *Supponiamo che ci sia un link con banda 900 kbps che supporta 10 applicazioni client/server su TCP in contemporanea che trasferiscono file di grandi dimensioni.Secondo il meccanismo di fairness di TCP, ciascuna connessione in media ottiene 1/10 della banda disponibile, cioè 90 kbps. Se si aggiunge una 11esima applicazione che usa 20 connessioni TCP parallele (ad esempio HTTP parallelo), per trasferire file di grandi dimensioni, quanto sarà la frazione di banda in media occupata da questa nuova applicazione?*
  - **I processi che utilizzano connessioni TCP parallele ottengono più banda**
  - **30 connessioni**
  - **10/30 occupate da 10 app**
  - **20/30 occupate dall'app aggiuntiva**
  - **Banda app aggiuntiva = 900 kbps * 20 / 30 = 600 kbps**

- *Consideriamo il caso di due terminali, uno al CERN di Ginevra e l'altro al Fermilab di Chicago collegati tra loro da un link con ampiezza di banda R=1Gbps. Il tempo di RTT tra di essi vale 30 ms. Supponiamo che i dati siano scambiati in pacchetti della dimensione L=1.000 byte. Quanto deve essere grande la finestra affinché l'utilizzazione del canale sia continua? Il numero di pacchetti minimo (si trascurino i meccanismi TCP di controllo flusso e congestione) è di:*
  - **R = 1 Gbps = 1.000.000.000 bit/s**
  - **RTT = 30 ms = 0,03 s**
  - **L = 1.000 byte = 8.000 bit**

$$Dimensione\ Minima\ Finestra = 1 + \frac{R}{MSS} \times RTT$$

$$Dimensione\ Minima\ Finestra = 1 + \frac{1.000.000.000\ \frac{bit}{s}}{8.000\ bit} \times 0{,}03\ s = 3751\ pacchetti$$

- *Scaricate un file di grandi dimensioni via HTTP alla velocità media di T = 50kBps da un sito per cui RTT=90 ms.La dimensione dei singoli pacchetti trasferiti è di L=500 byte. Trascurando gli eventi di timeout si hanno solo ACK ripetuti tre volte per cui si passa con incremento lineare da finestre della dimensione di nL/2 a finestre della dimensione di nL e poi si ripete per l'arrivo di 3 ACK duplicati. Il valore di n , numero di pacchetti massimo nella finestra è quindi:*
  - **T = 50 kBps = 50.000 Bps = 400.000 bit/s**
  - **RTT = 90 ms = 0,09 s**
  - **L = 500 byte = 4.000 bit**

$$T = \frac{3}{4}\frac{nL}{RTT}$$

$$n = \frac{4RTT \times T}{3L} = \frac{4 \times 0{,}09\ s \times 400.000\ bit/s}{3 \times 4.000\ bit} = 12\ pacchetti$$

- *È aperta una connessione Telnet. Il client invia un pacchetto con carico utile di 20 caratteri con numero di sequenza 51 e numero di riscontro 60.Quale saranno rispettivamente i corrispondenti numeri di sequenza e di riscontro del successivo pacchetto di echo back da server a client?*
  - **seq = client_ACK, ACK = client_seq + payload (20 char * 1 byte = 20)**
  - **seq = 60, ACK = 71**

- *Si definisce latenza come il tempo trascorso da quando il client inizia la connessione TCP a quando riceve l'oggetto richiesto nella sua interezza. Spedite un oggetto della dimensione di O=100 Kbyte. Se la MSS e' S=500 byte ed RTT=100 ms, supponendo che il protocollo di trasporto usi finestre statiche di dimensione W (numero intero di segmenti), determinare il più piccolo valore di W per cui la latenza e' la minima possibile (nonché il valore della latenza stessa) per una ampiezza di banda di R=1 Mbps. Ignorate gli eventuali overhead di intestazione introdotti dagli strati di trasporto, rete e link ed escludete che ci possano essere eventi di perdita dati.*

  - **O = 100 Kbyte = 100.000 byte = 800.000 bit**
  - **S = 500 byte = 4.000 bit**
  - **RTT = 100 ms = 0,1 s**

$$Latenza = 2 \times RTT + \frac{O}{R}$$

$$Latenza = 2 \times 0{,}1\ s + \frac{800.000\ bit}{1.000.000\ \frac{bit}{s}} = 1\ s$$

$$W \geq \frac{RTT \times R}{S} + 1 = 0{,}1\ s = 26$$

- *In una connessione ad alta velocità i dati vengono trasferiti in segmenti di MSS pari a 1500 byte, su un tratto in cui in media RTT=100ms. Per raggiungere un throughput di 10 Gbps, quale deve essere la frequenza di smarrimento(numero di pacchetti smarriti su numero di pacchetti inviati)?*

  - **MSS = 1500 byte = 12.000 bit**
  - **RTT = 100 ms = 0,1 s**
  - **T = 10 Gbps = 10.000.000.000 bit/s**

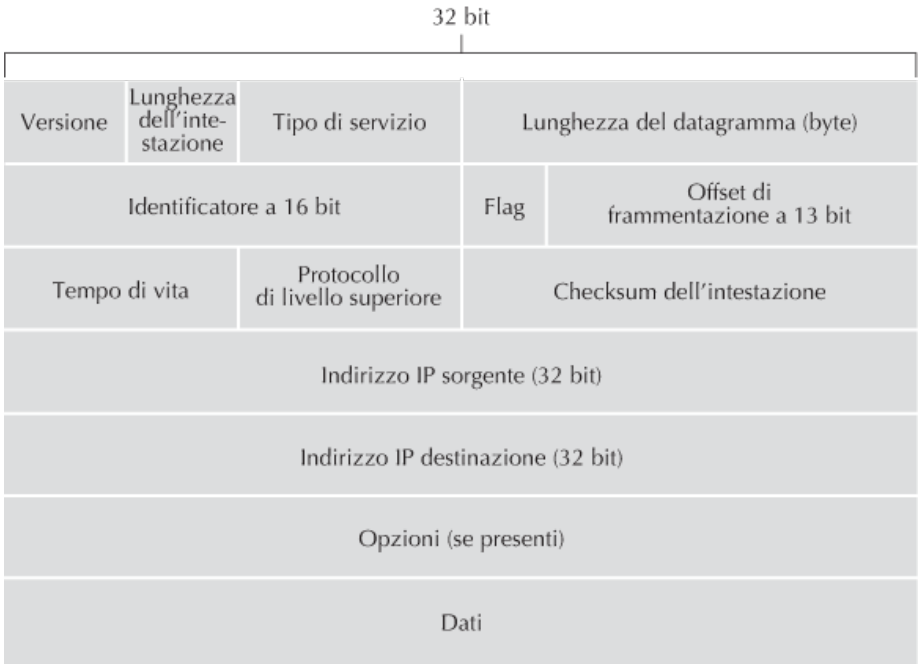
$$T = \frac{1{,}22MSS}{RTT\sqrt{L}}$$

$$L = \left(\frac{1{,}22MSS}{RTT \times T}\right)^2 = \left(\frac{14640\ bit}{0{,}1\ s \times 10.000.000.000\frac{bit}{s}}\right)^2 = 2{,}1433 \times 10^{-10}$$

3. Livello di rete

- **Inoltro** (forwarding)
  - Quando un router riceve un pacchetto, lo deve trasferire sull'appropriato collegamento di uscita
- **Instradamento** (routing)
  - **Algoritmi di instradamento**
  - **Tabelle di forwarding**

3.1 Pacchetto IPv4



3.2 Frammentazione datagrammi IPv4

- **MTU *(Maximum transmission unit)***
  - Massima quantità di dati che un frame a livello di collegamento può trasformare

3.3 DHCP

- **DHCP discover**
  - Client manda in broadcast (255.255.255.255) pacchetto UDP
- **DHCP offer**
  - Server manda in broadcast IP proposto al client, subnet mask, e **lease time**
- **DHCP request**
  - Client sceglie tra le proposte e risponde con una request
- **DHCP confirm**
  - Server risponde con DHCP ACK

3.4 NAT

- Traslazione indirizzo IP pubblico in indirizzi IP rete privata
- Tabella di traduzione

3.5 Algoritmo di instradamento

- Centralizzato
  - **Algoritmi LS (Link-State)**
    - **Dijkstra**
- Decentralizzato
  - **Algoritmi DV (Distance Vector)**
    - **Bellman-Ford**

- **Statico**
- **Dinamico**

- **Sensibile al carico**
- **Insensibile al carico**
  - Internet

3.6 ICMP (Internet control message protocol)

- Pacchetto UDP
- Traceroute / Ping

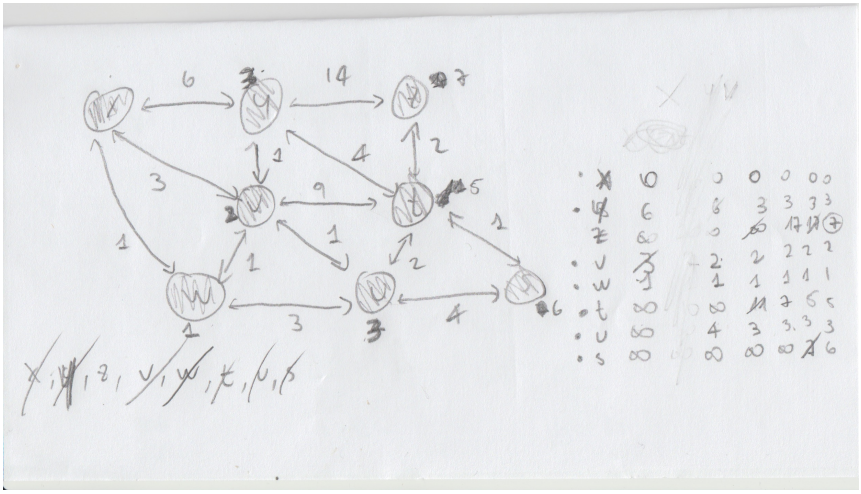
Esercizi

- *In un algoritmo di routing che utilizza una metodologia distance-vector*
  - **È necessario conoscere quali connessioni ci sono con i router vicini**
- *Durante una connessione TCP un pacchetto IPv4 in viaggio dall'host A all'host B viene frammentato in nove parti. Una delle parti, diciamo la settima non arriva a destinazione all'host B. Che cosa succede?*
  - **Dopo la ricezione di un ACK duplicato tre volte del pacchetto precedente o allo scadere del timeout, A rimanda l'intero pacchetto**
- *Se ci sono sei router tra due host, ignorando la frammentazione quante tabelle di routing saranno utilizzate da un pacchetto che viaggia tra i due host?*
  - **7**
- Se invio un datagramma di 3000 byte in un link con MTU di 500 byte quanti frammenti sono generati? Il datagramma originale ha il numero 305 nel campo "identification number". Assumete un overhead di intestazione tipico.
  - **3000 byte / 500 byte = 6 frammenti**
  - **6 frammenti di dati + 1 intestazione = 7 frammenti**
- *Se un applicativo genera una sequenza di 40 byte ogni 20ms e ciascuna sequenza viene incapsulata in un pacchetto TCP e poi in un pacchetto IP quale percentuale del datagram ottenuto sarà almeno di overhead rispetto alla sequenza generata dall'applicativo?*
  - **Overhead TCP + IP = 40 byte**
  - **Payload = 40 byte**
  - **Totale = 80 byte**
  - **Overhead/Totale * 100 = 50%**
- *Una organizzazione vuole mantenere tutti i suoi client dietro un router NAT su rete privata. Il router NAT ha una sola uscita su IP pubblica fornita dal ISP dell'organizzazione. Gli indirizzi privati sono del tipo 10.0.0.0/x. Quanti bit al minimo caratterizzano la rete privata?*
  - **Utilizzando solo il primo byte degli indirizzi di rete avremo 1 byte dedicato alla rete (8 bit) e 3 byte per gli host (24 bit)**

- *Una organizzazione ha 1600 host a cui deve assegnare un indirizzo IP. Richiede al suo ISP un blocco di indirizzi della forma a.b.c.d/x in cui x indica il numero di bit che denotano la parte di rete dell'indirizzo. Quale percentuale di indirizzi rimangono liberi nel blocco assegnato?*
  - **Per gestire 1600 host prendo la potenza di 2 immediatamente superiore**
  - **2^{11} = 2048**
  - **1600/2048 * 100 = 78% —> Percentuale utilizzata dagli host**
  - **100% - 78% = 22% —> Percentuale di posti liberi nella sottorete**
- *Arrivate in una organizzazione e vi danno il seguente configurazione per la interfaccia di rete del vostro calcolatore:IP: 200.23.10.40 NetMask: 255.255.255.240 Gateway: 200.23.10.33Quante schede di rete, inclusa la vostra, possono essere configurate su questa LAN, esclusa l'interfaccia del router?*
  - **Indirizzi per ogni rete = 255 - 240 + 1 = 16**
  - **Indirizzi disponibili per gli host = Indirizzi per ogni rete - Indirizzo di rete - Indirizzo di broadcast = 16 - 1 - 1 = 14**
  - **Indirizzi disponibili per gli host, escluso router = Indirizzi disponibili per gli host - Indirizzo router = 14 - 1 = 13**
- *Una organizzazione gestisce un indirizzo 149.76.1.0/24. Deve suddividerlo tra le sue 6 sedi. Imposta una maschera a 27 bit che in notazione decimale risulta: 255.255.255.224. Ad un host viene assegnato l'indirizzo 149.76.1.84: quali sono gli indirizzi riservati - indirizzo di rete ed indirizzo di broadcast - per quella sottorete?*
  - **Indirizzi per ogni rete = 2^{32} - 2^{27} oppure 255 - 224 + 1 = 32**
  - **.0 —> .31**
  - **.32 —> .63**
  - **.64 —> .95 —> .84 si ritrova in questa sottorete**
- *Supponiamo che i datagrammi che si scambiano due host A e B siano di dimensione massima di 1540 byte (header inclusi). Assumendo intestazioni tipiche a livello di rete e trasporto, quanti datagrammi sono necessari per inviare un file MP3 da 3,001MB su una connessione TCP?*
  - **Dati trasportabili per ogni frammento = 1540 byte - 40 byte (overhead TCP + IP) = 1500 byte**

- **Dati da trasportare = 3.001.000 byte**
- **Frammenti da generare = Dati da trasportare / Dati trasportabili per ogni frammento = 3.001.000 byte / 1500 byte = 2.000,6 frammenti necessari**
- **2.001 frammenti generati**

- *Considerate l'insieme di 8 nodi e 14 link con le seguenti caratteristiche: c(x,y)=6; c(x,v)=3; c(x,w)=1; c(y,z)=14; c(y,v)=1; c(y,t)=4; c(v,w)=1; c(v;u)=1; c(v;t)=9; c(w;u)=3; c(u,t)=2; c(u,s)=4; c(s,t)=1, c(t,z)=2. Determinare il numero di passi necessari al calcolo del percorso di minima distanza da x a z, nonché il costo di questo percorso ed il nodo immediatamente precedente a z. Per passo si intende l'insieme di operazioni eseguite in ogni ciclo, in particolare: aggiunta all'insieme dei nodi esaminati di un nodo con percorso di costo minimo, aggiornamento dei percorsi di costo minimo.*



Guarda Youtube: Djikstra in 3 minuti e Bellman-Ford in 5 minuti

Formulario

Nome	Formula	Note
Ritardo Trasmissivo	$\frac{L}{R}$	L = Lunghezza Pacchetto R = Banda collegamento
Ritardo End-to-End	$N\left(\frac{L}{R}\right)$	N = Numero di collegamenti tra sorgente e destinazione
Ritardo di Propagazione	$\frac{d}{v}$	d = Lunghezza del collegamento v = Velocità trasmissiva nel collegamento
Ritardo Complessivo	$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$	
Intensità di traffico	$\frac{L \times a}{R}$	a = Tasso di arrivo pacchetti [pacchetti/s] $\frac{L \times a}{R}$ ~ 0 —> **Ritardo piccolo** $\frac{L \times a}{R}$ ~ 1 —> **Ritardo grande** $\frac{L \times a}{R} > 1$ —> **Ritardo infinito**
Dimensione minima finestra di congestione (per non avere stalli)	$W \geq \frac{RTT \times R}{MSS} + 1$ $\frac{W \times MSS}{R} \geq \frac{MSS}{R} + RTT$	
Efficienza Rete	$\frac{1}{1 + 5 \times \frac{Tprop}{Ttrans}}$	Tprop = d/v Ttrans = L/R
Throughput TCP	$T = \frac{3}{4}\frac{W}{RTT}$ $T = \frac{3}{4}\frac{nL}{RTT}$	T = Throughput TCP W = Larghezza finestra quando si verifica una perdita n = Numero di segmenti massimo nella finestra L = Lunghezza segmento
Throughput TCP con perdita di pacchetti	$T = \frac{1{,}22MSS}{RTT\sqrt{L}}$	MSS = Maximum Segment Size L = Tasso di perdita dei pacchetti
Latenza con buffer infiniti	$2RTT + \frac{O}{R}$	O = Lunghezza oggetto da trasmettere [byte]

Nome	Formula	Note
Latenza con buffer finiti	$2RTT + \frac{O}{R} + (K-1) \times \left(\frac{S}{R} + RTT - \frac{W \times R}{R} \right)$	K = numero di finestre necessarie per inviare O W = numero di segmenti presenti in una finestra
Latenza con buffer finiti e finestra dinamica	$2RTT + \frac{O}{R} + P \times \left(RTT + \frac{S}{R} \right) - (2^P - 1) \times \left(\frac{S}{R} \right)$	***P*** *è il numero di volte che TCP si blocca lato mittente per aspettare che arrivi un ACK e riapra la finestra:* $P = min\{Q, K-1\}$ ***Q*** *è il numero di volte che il mittente si arresterebbe se l'oggetto fosse di dimensioni infinite. Di solito vale 0.* $Q = 1 + \log_2 \left(RTT \times \frac{R}{S} + 1 \right)$ ***K*** *è il numero di finestre necessario per coprire l'oggetto* $K = \log_2 \left(\frac{O}{S} + 1 \right)$ ***S/R*** *è il tempo necessario per trasmettere un segmento* ***RTT + S/R*** *è il tempo necessario per ricevere un ACK*