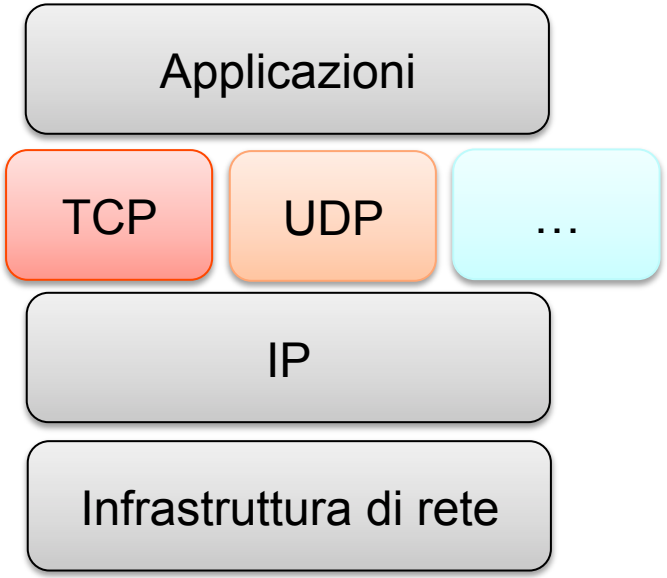
Strato di trasporto in Internet

Prof. Franco Callegati
DEIS Università di Bologna
http://deisnet.deis.unibo.it

Trasporto in Internet

- Architettura protocollare tradizionale di Internet:
  - Tipologia di servizio: dati
  - Due protocolli di trasporto:
    - TCP: connection oriented
    - UDP: connectionless



- Nuovi servizi multimediali:
  - Emergono problematiche di trasporto real time
  - Vengono definiti nuovi protocolli di trasporto
    - RTP, RTCP

Funzioni dello strato di trasporto

- Consente la Multiplazione:
  - Permette a più processi applicativi di utilizzare le sue funzioni di comunicazione in contemporanea
- Utilizza il numero di porta per distinguere flussi dati di applicazioni diverse
- Controlla il comportamento del canale di comunicazione end-to-end
  - L'obiettivo è quello di garantire la qualità del trasferimento dati a livello di trasporto richiesto dall'applicazione

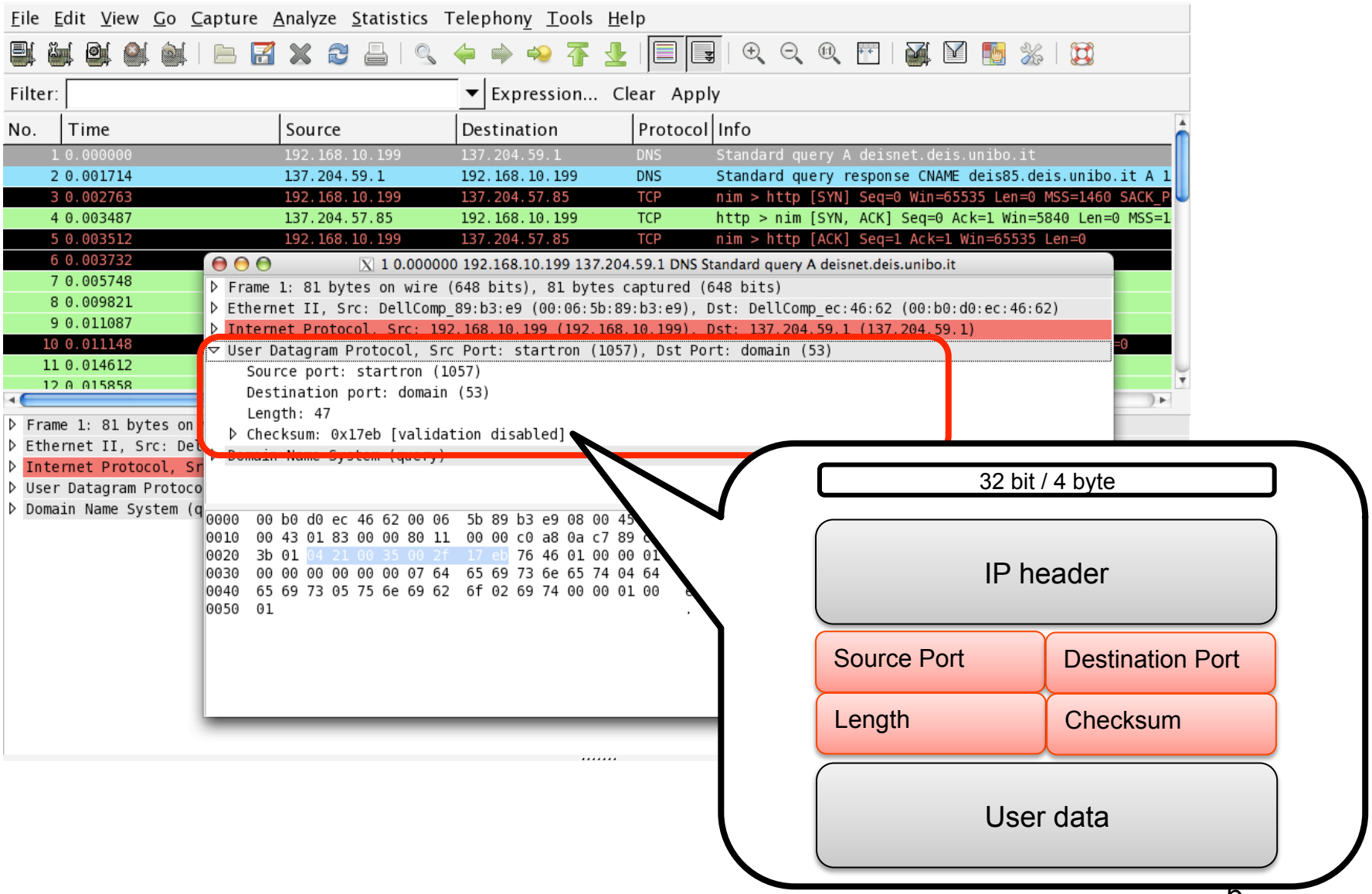
Controllo del canale

- Controllo dell'errore
  - Errori di trasmissione
  - Perdita di unità informative
- Controllo di sequenza
  - Errato ordine di consegna a causa di perdite
  - A causa di tempi di propagazione variabili
- Controllo di flusso e di congestione
  - Il trasmettitore invia i dati
    - La sua velocità deve essere compatibile con quella del canale e con quella del ricevitore

User Datagram Protocol (UDP)

- Protocollo “connectionless”
  - Non esiste il concetto di “connessione”
  - Ogni messaggio è indipendente da tutti gli altri
- Pensato per
  - Invio di blocchi dati di limitate dimensioni
  - Comunicazione fra applicazioni che non richiede un controllo della qualità del trasporto
    - Esempi: e-mail, DNS, …

Il messaggio UDP



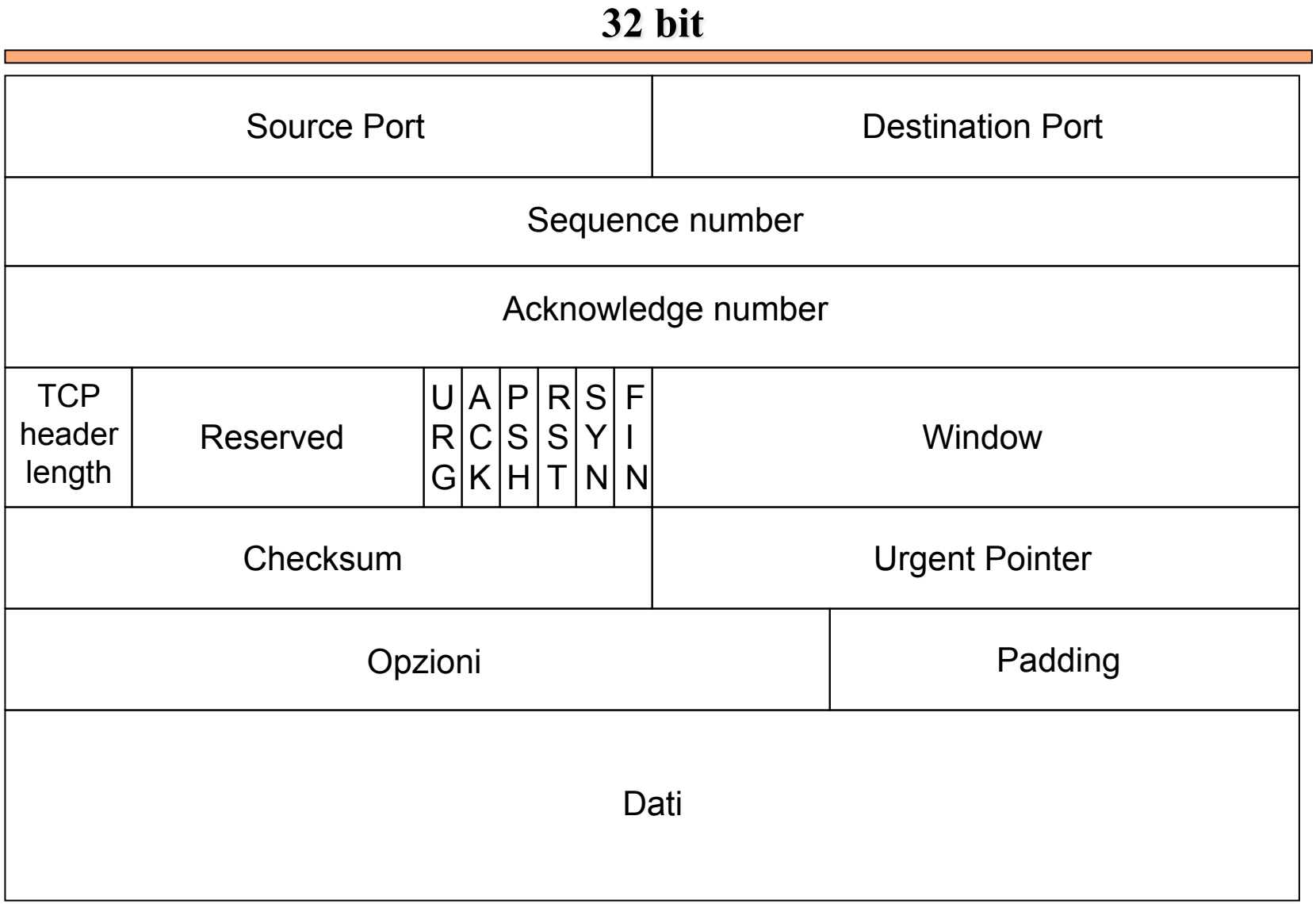
Transmission Control Protocol - RFC 793

- Obiettivi:
  - Controllo della comunicazione *end-to-end* e full-duplex fra processi applicativi
  - Garanzia di affidabilità del trasporto
- Assume che:
  - lo strato di rete fornisca solamente un semplice ed inaffidabile servizio di trasferimento dei pacchetti di tipo connectionless (quello che fa l'IP)

Il segmento TCP

- TCP incapsula i dati delle applicazioni in pacchetti detti “segmenti”
- Il segmento TCP prevede
  - Un header standard di 20 byte
  - Un header variabile per negoziare delle opzioni
  - Un payload di dimensione variabile contenente i dati di applicazione
- Il segmento TCP ha una dimensione massima detta Maximum Segment Size (MSS)
  - MSS corrisponde alla massima dimensione del blocco dati di applicazione che può essere contenuto nel segmento

Formato del segmento TCP (1)



Formato del segmento TCP (2)

- **Source (Destination) port**: numero della porta sorgente (destinazione)
- **Sequence number**: numero di sequenza del primo byte del pacchetto; se è presente il bit SYN questo è il numero di sequenza iniziale su cui sincronizzarsi
- **Acknowledge number**: se il bit ACK è a 1 allora questo numero contiene il numero di sequenza del blocco di dati che il ricevitore si aspetta di ricevere
- **TCP Header Length (4 bit)**: numero di parole di 32 bit dell'intestazione TCP; indica dove iniziano i dati
- **Reserved**: sei bit riservati per uso futuro

Formato del segmento TCP (3)

- **Control bit**: sono 6 bit di controllo

  - **URG** posto a 1 se si deve considerare il campo Urgent Pointer

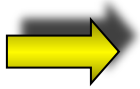
  - **ACK** posto a 1 se si deve considerare il campo Acknowledge

  - **PSH** posto a 1 serve per la funzione di push

  - **RST** posto a 1 per resettare la connessione

  - **SYN** posto a 1 per sincronizzare i numeri di sequenza

  - **FIN** posto a 1 per indicare la fine dei dati



Formato del segmento TCP (4)

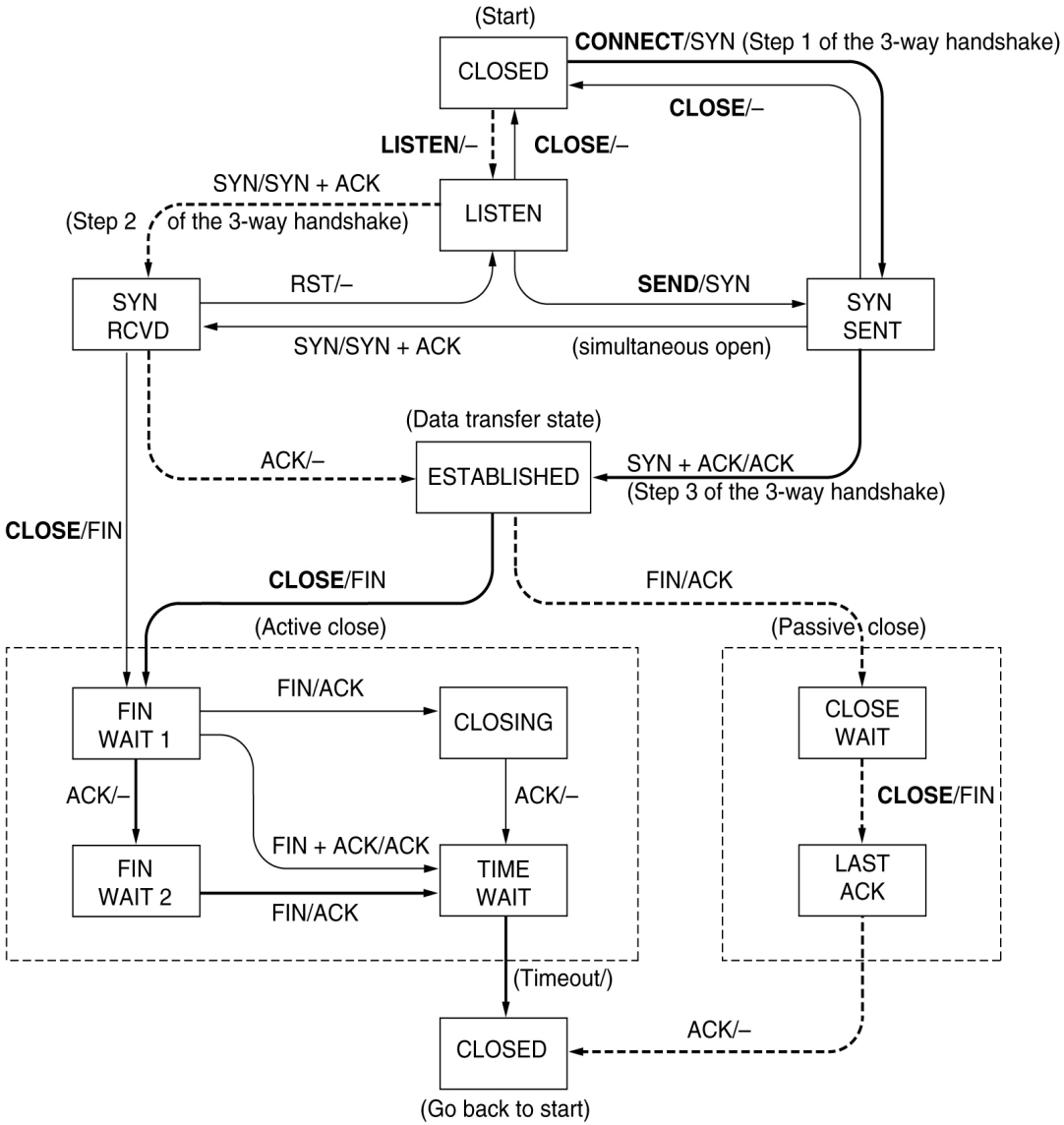
- **Window**: finestra del ricevitore, cioè il numero di byte che il ricevitore è disposto a ricevere, partendo dal numero di sequenza di quello contenuto nel campo acknowledge
- **Checksum**: ridondanza per la rilevazione degli errori
- **Urgent Pointer**: contiene puntatore a dati urgenti eventualmente presenti nel pacchetto (es. per abortire programma remoto in esecuzione), ha senso se il bit URG è posto ad 1
- **Options**: contiene opzioni per la connessione
- **Padding**: bit aggiuntivi per fare in modo che l'intestazione sia multipla di 32 bit

Checksum

- Viene calcolato utilizzando l'Internet Checksum su:
  - Pseudo-header (psedo-intestazione)
    - Indirizzi IP sorgente e destinatario
    - Protocol
    - Lunghezza in byte del segmento TCP (payload IP)
      - Non viene trasmessa ma viene calcolata in trasmissione e in ricezione
  - Intestazione TCP
    - Con campo checksum posto a 0
  - Dati del segmento TCP
    - Se il numero di byte è dispari viene aggiunto un byte di padding con tutti i bit a 0
    - Il padding non viene trasmesso

La macchina a stati finiti del TCP

- Linee tratteggiate
  - Azioni tipiche di un server
- Linee nere
  - Azioni tipiche di un client
- Linee chiare
  - Eventi inusuali
- Transizioni
  - Causa/effetto

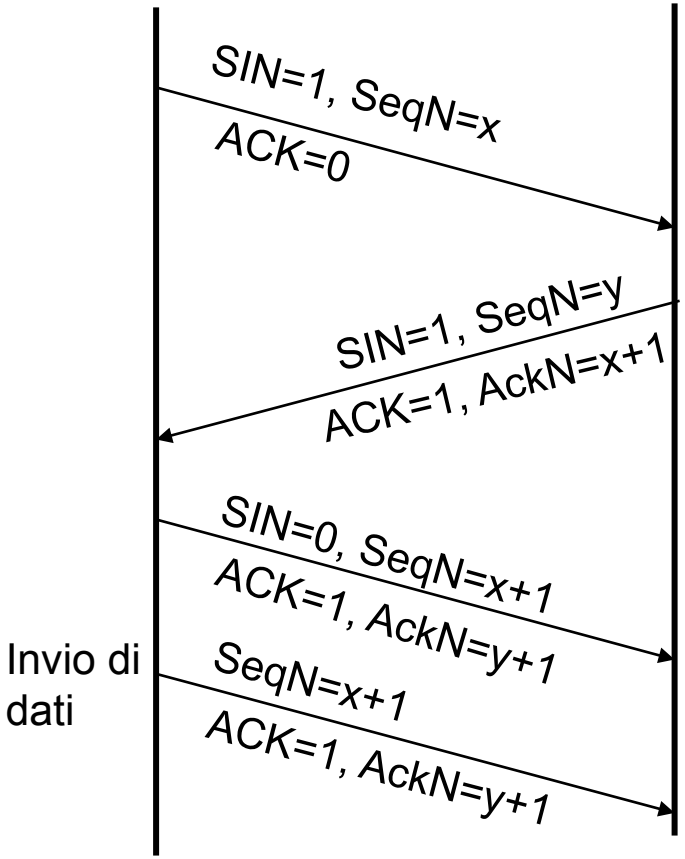


Da A.S. Tanenbaum, “Reti di Calcolatori”

Dialogo su rete inaffidabile

- Se il mezzo di comunicazione è inaffidabile risulta sostanzialmente impossibile avere uno scambio di informazioni con conferma certa
  - Problema logico delle 3 armate
  - A invia un messaggio e B lo conferma
  - Se A non riceve la conferma non può sapere se B abbia ricevuto il messaggio o meno
    - Perdita del messaggio o della conferma?
  - Il ragionamento si può proseguire sulla conferma della conferma ecc.
- È necessario decidere dove fermarsi per raggiungere un determinato grado di affidabilità

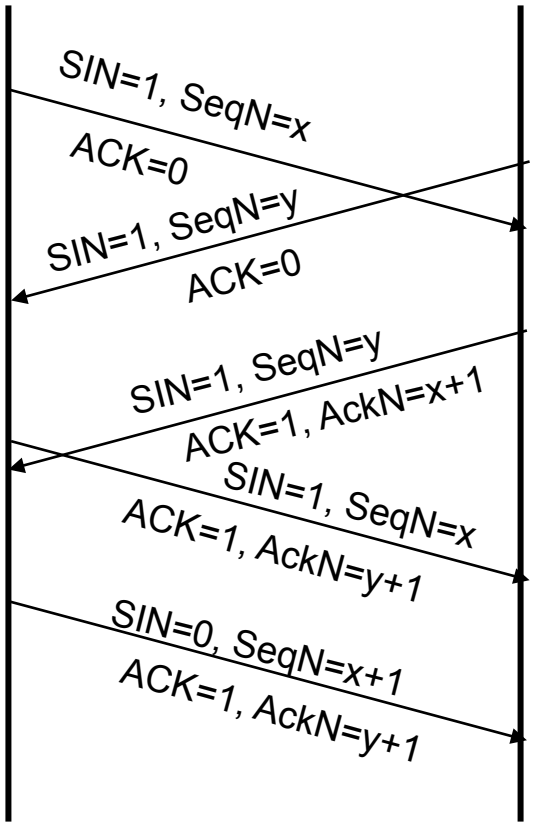
Apertura della connessione TCP

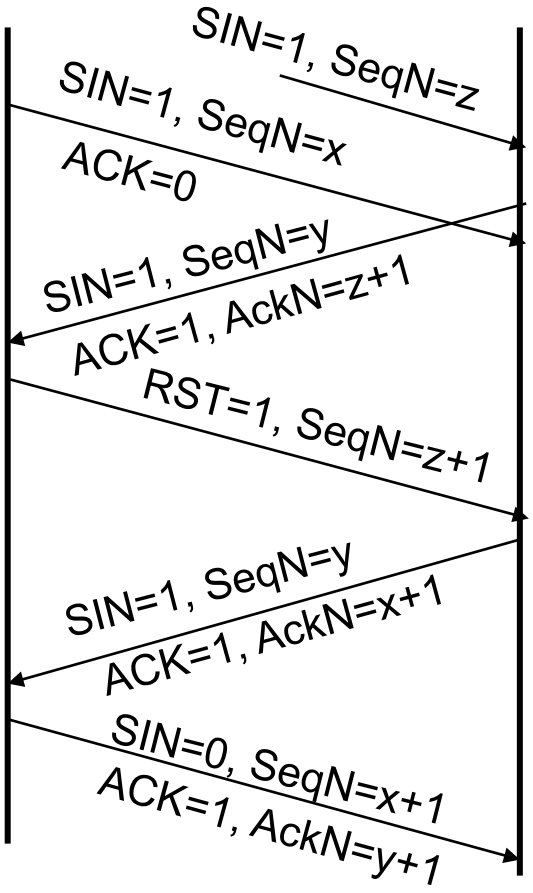


- L'apertura della connessione è critica
  - I segmenti di segnalazione possono essere persi, duplicati e ritardati
- Three ways handshake
  - Si è dimostrato molto robusto alla prova dei fatti
- Utilizza sinergicamente i bit di flag e quelli di numerazione
  - Si noti che il primo pacchetto dati ha numero di sequenza uguale all'ACK precedente (ACK non occupa spazio di numerazione)

Caratteristiche del TWH

- Il three-ways handshake
  - resiste alla instaurazione contemporanea di due connessioni
  - ignora pacchetti di apertura ritardatari





Chiusura della connessione TCP

- Soft release
  - Il TCP cerca di realizzare la chiusura ordinata della connessione, garantendo che non vadano persi dati
- Anche questo problema non può essere risolto in modo rigoroso su una rete inaffidabile
- TCP sceglie di realizzare la chiusura con modalità "simplex"
  - Le due direzioni vengono rilasciate in modo indipendente
  - Il TCP che intende terminare la trasmissione emette un segmento con FIN=1
    - Quando questa entità riceve l'Ack la direzione si considera chiusa
    - Se dopo un certo tempo non arriva l'Ack il mittente del FIN rilascia comunque la connessione
  - L'altra direzione può continuare a trasmettere dati finché non decide di chiudere

Esempio di chiusura normale

- TCP A decide di chiudere
  - Inizia la procedura inviando un segmento con FIN=1
- TCP B rileva la richiesta di chiusura
  - Procede anche lui all'invio di un segmento con FIN=1

```
TCP A                                                TCP B
 1.  ESTABLISHED                                          ESTABLISHED
 2.  (Close)
     FIN-WAIT-1  --> <SEQ=100><ACK=300><CTL=FIN,ACK>  --> CLOSE-WAIT
 3.  FIN-WAIT-2  <-- <SEQ=300><ACK=101><CTL=ACK>      <-- CLOSE-WAIT
 4.                                                       (Close)
     TIME-WAIT   <-- <SEQ=300><ACK=101><CTL=FIN,ACK>  <-- LAST-ACK
 5.  TIME-WAIT   --> <SEQ=101><ACK=301><CTL=ACK>      --> CLOSED
 6.  (2 MSL)
     CLOSED
```

Affidabilità del collegamento

- Il TCP cerca di garantire l'affidabilità del canale end-to-end utilizzando
  - numerazione sequenziale dei dati
    - unità di riferimento il byte
  - conferma della ricezione di ogni byte da parte del ricevitore (acknowledge)
  - ritrasmissione dei dati di cui non viene confermata la ricezione
- Tutto questo viene implementato con un protocollo di tipo Automatic Repeat Request

Apertura della connessione: www.ietf.org