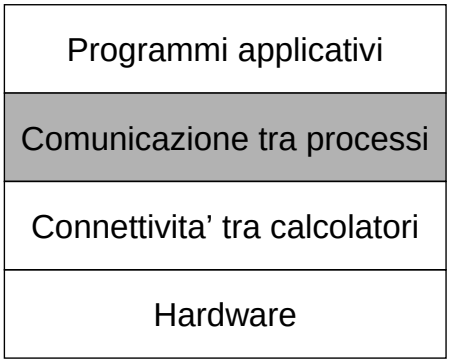
Livello di Trasporto

Introduzione
Problemi e requisiti
Livello di trasporto in Internet
UDP - User Datagram Protocol
TCP - Transmission Control Protocol
Meccanismo di ritrasmissione
Controllo del flusso
Three-way handshake
Controllo di congestione

Prof. Filippo Lanubile

Introduzione

- Obiettivo del livello di trasporto
  - Offrire un canale di comunicazione tra una coppia di processi (comunicazione point-to-point)



Problemi e requisiti

- Problemi posti dal livello inferiore (rete)
  - perdita di messaggi
  - riordino di messaggi
  - inoltro di copie multiple dello stesso messaggio
  - limitazione sulla dimensione dei messaggi
  - inoltro di messaggi con un ritardo arbitrariamente lungo
- Requisiti richiesti dal livello superiore (applicazioni)
  - garantire l'inoltro del messaggio
  - rispettare l'ordine di invio dei messaggi
  - inoltrare solo una copia di ogni messaggio
  - supportare messaggi di lunghezza arbitraria
  - consentire la sincronizzazione tra chi invia e chi riceve
  - consentire a chi riceve di controllare il flusso dei messaggi del mittente
  - supportare la presenza di processi multipli sullo stesso host

Livello di trasporto in Internet

- Il protocollo di rete IP fornisce un servizio senza connessione (a datagramma) non affidabile tra gli host

Protocolli di trasporto

- User Datagram Protocol (UDP)
  - servizio senza connessione (datagramma),
  - non affidabile (scoperta errori opzionale e nessun recupero),
  - multiplazione/demultiplazione: indirizzamento dei processi
- Transmission Control Protocol (TCP)
  - servizio orientato alla connessione (flusso di bytes)
  - decomposizione delle sequenze di byte in messaggi
  - affidabile (scoperta e recupero degli errori)
  - controllo del flusso
  - multiplazione/demultiplazione: indirizzamento dei processi

Indirizzamento dei processi

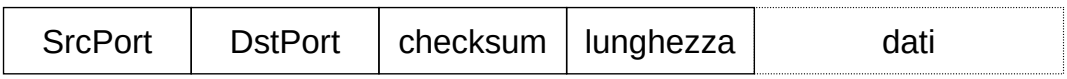
- Ogni processo comunicante è associato a un numero di porta a 16-bit, locale e unico rispetto all'host
- Necessita' di individuare un processo con un indirizzo di rete globalmente unico (IP address e port number)
- Tutti messaggi contengono due coppie di indirizzi:
  - Mittente: (IP address, port number)
  - Destinatario: (IP address, port number)
- Il sistema operativo dell'host destinatario consegna al processo un messaggio in arrivo sulla base del port number

Numeri di porta

- 1 - 255: riservati a servizi standard
  - 21: ftp
  - 23: telnet
  - 25: SMTP
  - 80:http daemon
- 1 - 1023: disponibili solo a utenti privilegiati
- 1024 - 4999: per processi automaticamente assegnati
- 5000 - : solo per processi utente

UDP - User Datagram Protocol

- Fornisce un servizio punto a punto senza connessione
- Spedizione di datagram (includono l'indirizzo del mittente e del destinatario) con lunghezza max definita dal datagram IP (<= 64 KB)



16 bit 16 bit 16 bit 16 bit

- Indirizzi dei processi basati sul numero di porta (16 bit)
- Controllo degli errori opzionale basato su checksum (se non usato e' settato a zero)
- Non fornisce conferma di ricezione dei messaggi
- Nessun controllo del flusso sul mittente
- Usato da SNMP (Simple Network Management Protocol), e da applicazioni basate su semplici servizi richiesta-risposta (RPC)

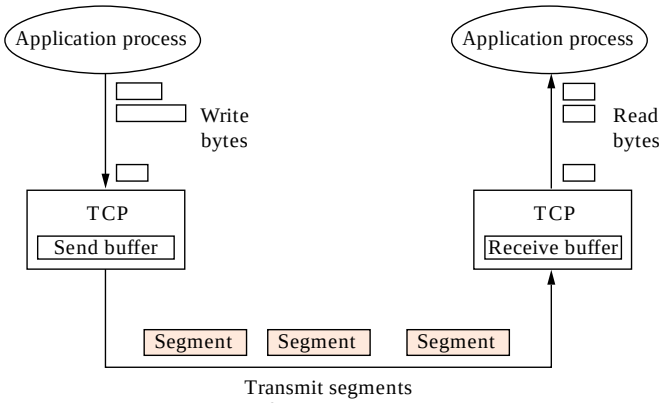
TCP - Transmission Control Protocol

Caratteristiche fondamentali

- Orientato alla connessione
  - un mittente richiede una connessione al destinatario e poi usa la connessione per il trasferimento dei dati
- Comunicazione punto a punto
  - due punti finali che non si interessano di cosa sta nel mezzo
- Trasferimento affidabile
  - garanzia che i dati arrivano senza errori, nel giusto ordine e senza duplicazioni
- Comunicazione full duplex
  - scambio di dati in entrambi le direzioni simultaneamente

TCP - Transmission Control Protocol

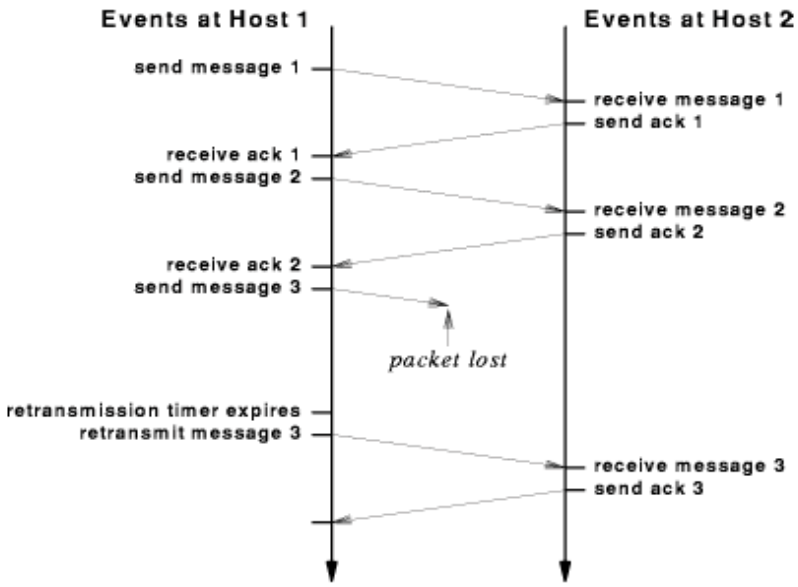
- Interfaccia a stream
  - sequenza continua di bytes (octets): nessuna garanzia che i dati arrivano negli stessi blocchi con cui sono stati spediti
- Segmentazione del flusso di byte
  - il flusso di bytes e' suddiviso in segmenti di lunghezza accettabile per il livello sottostante (IP accetta massimo 64 KB) e poi riassembla i segmenti nella sequenza corretta



TCP - Transmission Control Protocol

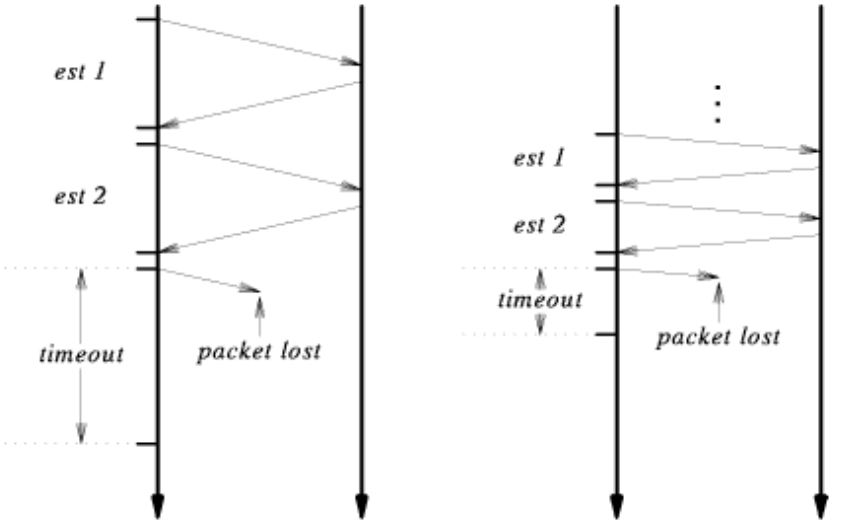
- Creazione affidabile della connessione
  - i dati di una vecchia connessione non si confondono con le nuove connessioni
- Distruzione dolce della connessione
  - i dati spediti prima di chiudere una connessione non sono perduti
- Controllo di flusso
  - garanzia che il mittente non eccede la capacita' di ricezione del destinatario
- Controllo di congestione
  - previene che un eccesso di dati venga immesso nella rete

Ritrasmissione di pacchetti



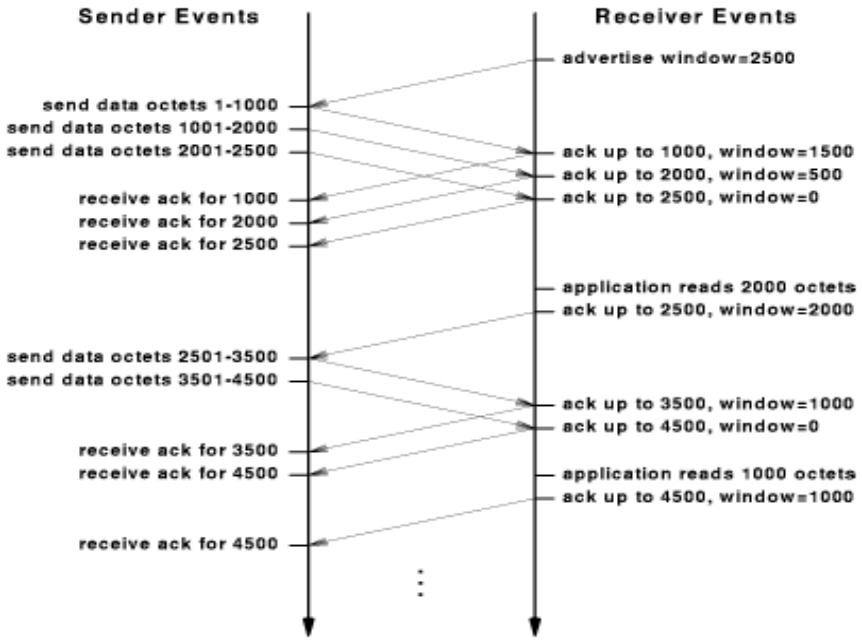
- acknowlegment dal receiver per verifica di pacchetti persi
- quanto bisogna aspettare?

Ritrasmissione adattativa



- Gestione dinamica dei timer di ritrasmissione
- Strategia: Aspettare quanto basta per considerare un pacchetto perso ma non piu' del necessario
- Soluzione: algoritmi basati sul round trip time

Controllo del flusso mediante finestra scorrevole

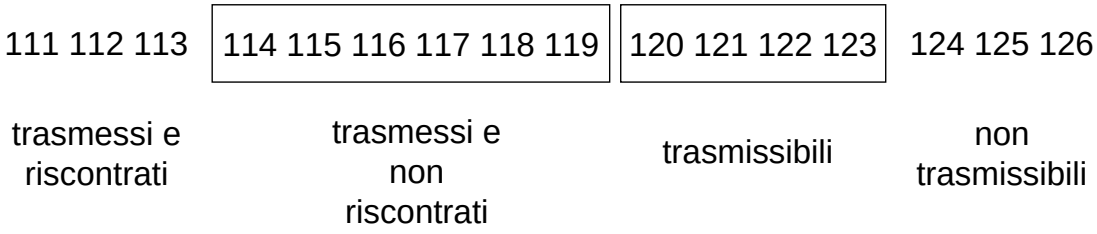


Window advertisement: Il receiver specifica insieme all'ack il numero di byte liberi nel buffer

Il sender usa la dimensione della finestra per determinare l'ammontare dei dati che possono essere spediti senza attendere l'ack

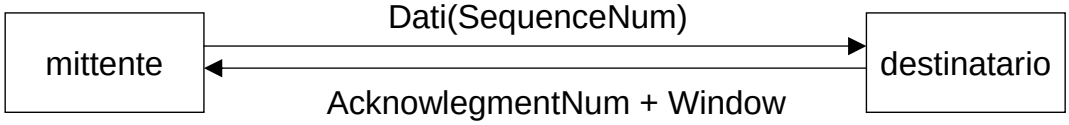
Gestione delle finestre in TCP

- L'algoritmo di finestra scorrevole e' usato da TCP per
  - garantire l'inoltro affidabile e ordinato dei dati
  - realizzare il controllo di flusso
- Caratteristiche
  - ogni byte del flusso complessivo ha un numero di sequenza
  - i riscontri (ack) sono cumulativi
- La dimensione della finestra determina l'ammontare di dati che possono essere trasmessi senza attendere il riscontro (ack)

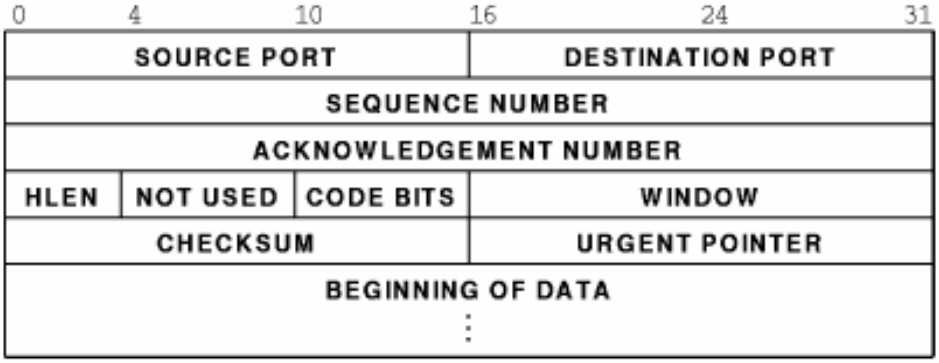


Formato del segmento TCP

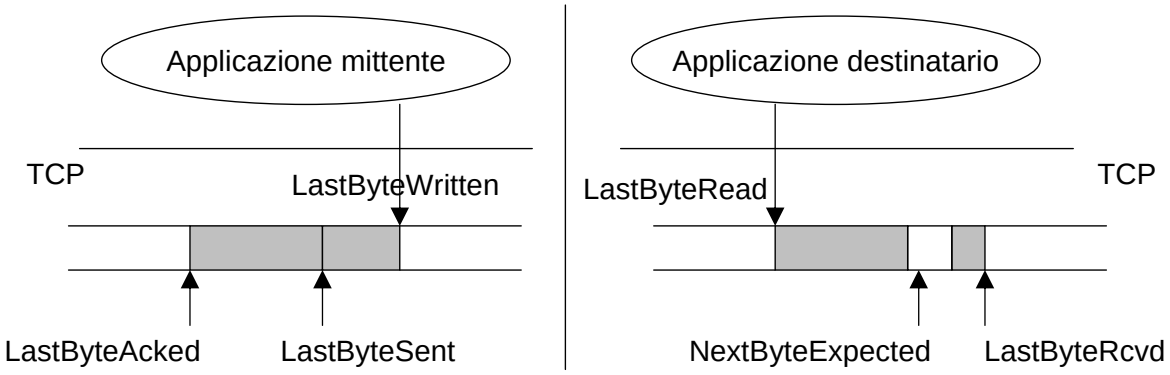
- Il segmento contiene una porzione del flusso di bytes complessivo



- Sequence number: numero di sequenza del primo byte nel segmento TCP
- Acknowlegment number: prossimo byte aspettato (= LastByteAcked+1)
- Window:: numero di byte liberi nel buffer del destinatario
- Code bits. SYN, FIN, RESET, PUSH, URG, ACK



Algoritmo di finestra scorrevole rivisitato: inoltro affidabile e ordinato



Mittente

- LastByteAcked <= LastByteSent
- LastByteSent <= LastByteWritten
- I bytes tra LastByteAcked e LastByteWritten sono nel buffer

Destinatario

- LastByteRead < NextByteExpected
- NextByteExpected <= LastByteRcvd +1
  (= se i dati sono arrivati in ordine altrimenti punta all'inizio del primo intervallo vuoto)
- I bytes tra LastByteRead e LastByteRcvd sono nel buffer

Algoritmo di finestra scorrevole rivisitato: controllo di flusso

Mittente

- Dimensione buffer: MaxSendBuffer
- LastByteSent - LastByteAcked <= AdvertisedWindow
- EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)
  spedisci se EffectiveWindow > 0
- LastByteWritten - LastByteAcked <= MaxSendBuffer
  blocca applicazione mittente se (LastByteWritten - LastByteAcked) + *y* > MaxSendBuffer
- spedisci segmenti sonda (un byte di dati) se AdvertisedWIndow=0

Destinatario

- Dimensione buffer: MaxRcvBuffer
- LastByteRcvd - LastByteRead <= MaxRcvBuffer
- AdvertisedWindow = MaxRcvBuffer - (LastByteRcvd - LastByteRead)

Creazione e distruzione della connessione

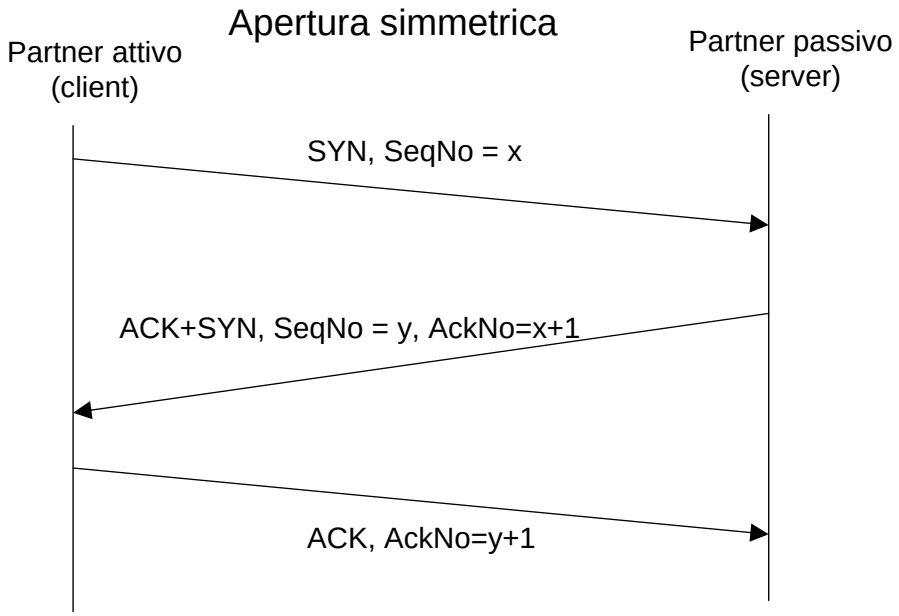
Requisiti

- Creazione affidabile della connessione
  - i dati di una vecchia connessione non si devono confondere con le nuove connessioni
- Distruzione dolce della connessione
  - i dati spediti prima di chiudere una connessione non devono essere persi

Soluzione: three-way handshake

- Coinvolge lo scambio di tre messaggi tra client e server
  - apertura della connessione
    - lo scambio permette un accordo su alcuni parametri: per TCP i sequence number iniziali dei rispettivi flussi di byte
    - Uso del flag (code bit) SYN
  - chiusura della connessione
    - uso del flag (code bit) FIN

Three-way handshake per l'apertura di una connessione



Three-way handshake modificato per la chiusura di una connessione

Chiusura asimmetrica

- quando un partner ha finito di trasmettere, chiude la sua metà connessione

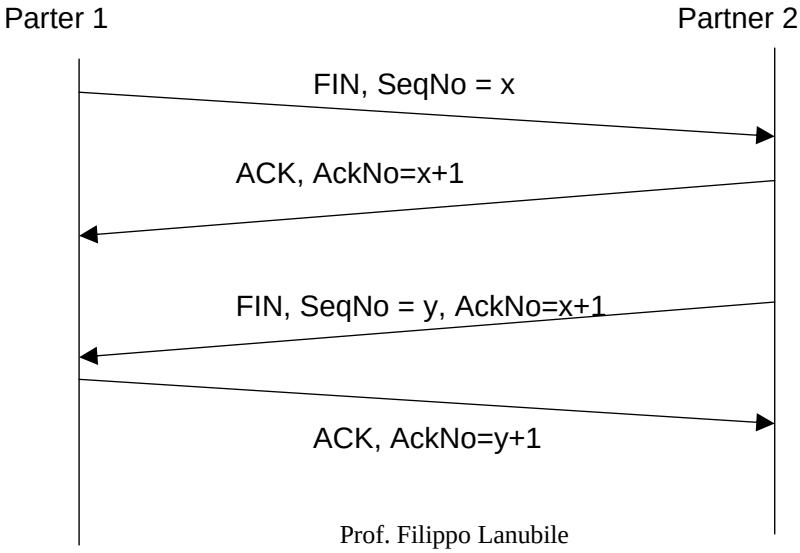
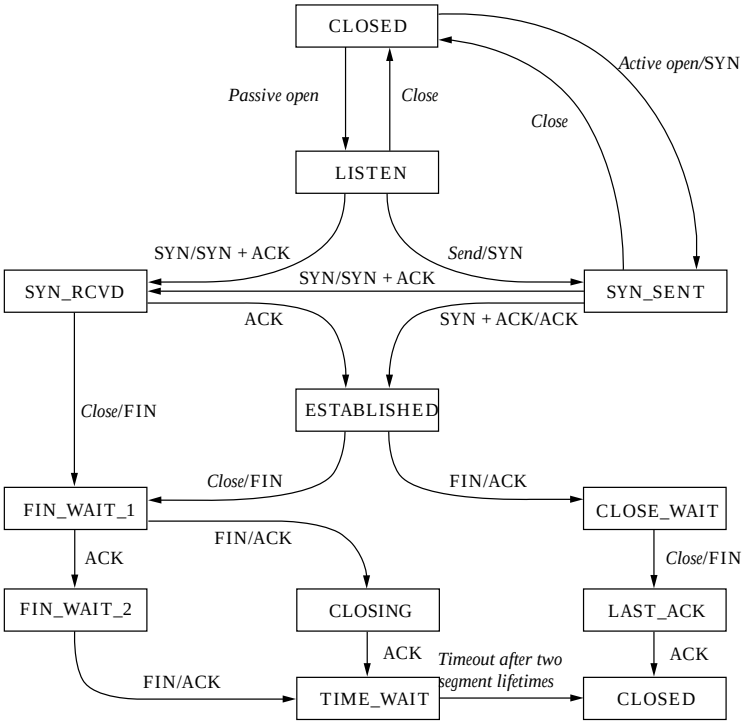


Diagramma di transizione degli stati



Controllo di congestione

- Un traffico eccessivo può causare la perdita di messaggi
  - i protocolli di trasporto rispondono con ritrasmissione
  - Un'eccessiva ritrasmissione causa una maggiore congestione fino al collasso della rete
- TCP interpreta la perdita di messaggi come spia della congestione e previene il collasso
  - il mittente ritarda la trasmissione dei messaggi e la raddoppia fino a che si e' raggiunta la meta' dell'advertising window e non ci sono altre perdite