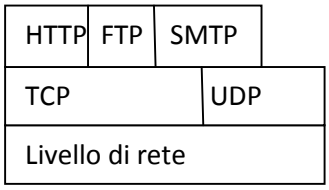
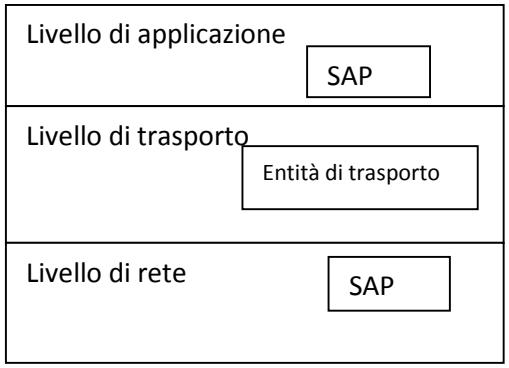
Strato di trasporto

Lo strato di trasporto si colloca al livello 4 dello strato ISO-OSI e svolge il compito di mettere in comunicazione diversi processi software.

La comunicazione tra applicazioni avviene con scambi di messaggi che vengono segmentati e trasformati in TPDU, Transport Protocol Data Unit





SAP= Service Access Point, interfaccia logica tra due entità una di livello N-1 e l'altra di livello N

Il livello di trasporto è caratterizzato da:

- Servizi che possono essere
  - Affidabili se eseguono le operazioni nel perfetto ordine
  - Non affidabili se garantiscono solo l'indirizzamento
- Protocolli che si distinguono in:
  - UDP User Datagram Protocol – protocollo asincrono che non richiede trasmissione di conferma di ricezione l
  - TCP Trasmission Control Protocol – protocollo sincrono dove è richiesto il messaggio di accettazione dei dati

I protocollo di trasporto sono implementati nei più diffusi sistemi operativie forniscono ai programmatori le funzioni base dette primitive:

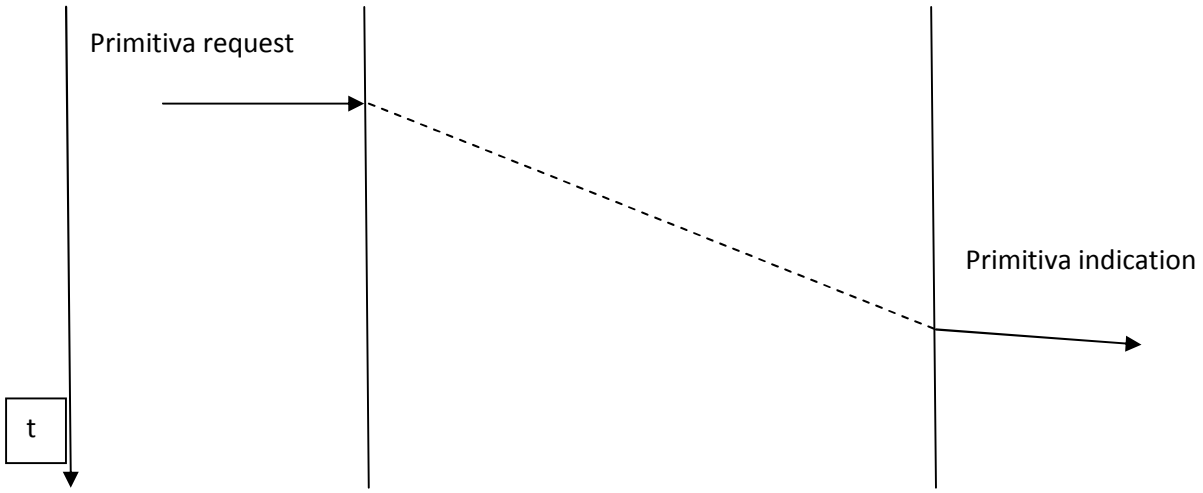
- LISTEN si mette in attesa di richiesta di connessione
- SEND DATA per trasmettere un contatto

- RECEIVE DATA per ricevere un contenuto
- T-CONNECT per aprire una connessione
- T-DISCONNECT per chiudere una connessione

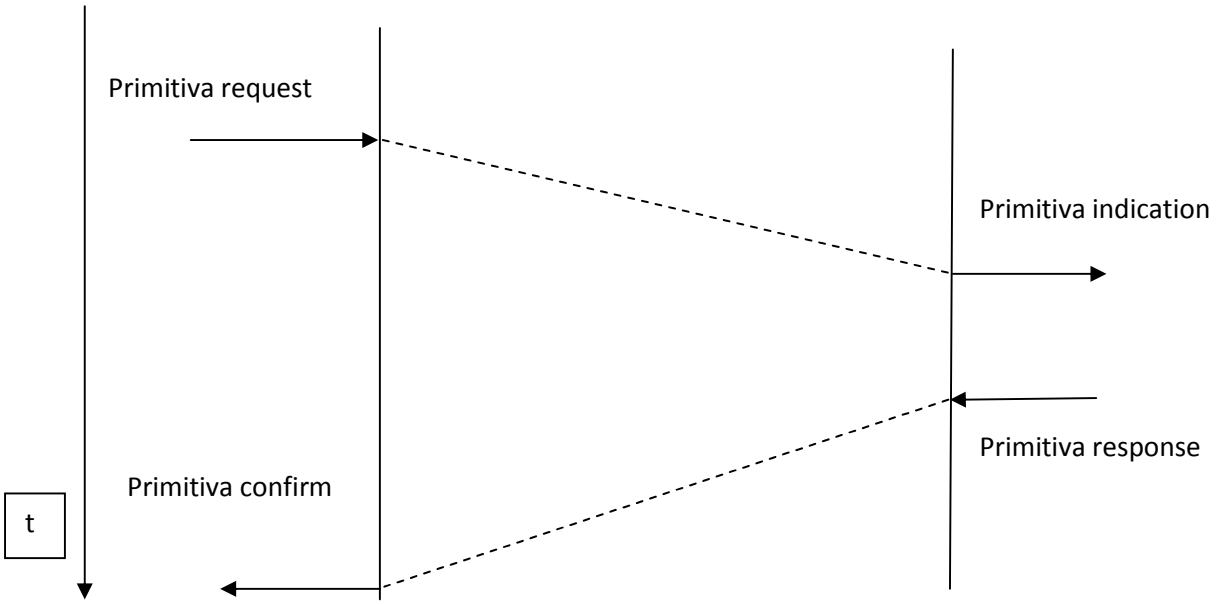
Per ogni primitiva ci sono i seguenti metodi:

Metodo primitiva	**Descrizione**
request()	Si chiede al servizio di compiere un'azione
indication()	Il servizio segnala un evento
response()	Si chiede al servizio di rispondere all'evento
confirm()	Il servizio segnala l'arrivo di una conferma

Connessione asincrona o connection-less



Connessione sincrona



L’indirizzamento di trasporto

Per poter risolvere il problema della trasmissione dei dati tra applicazioni diverse sui medesimi host, il protocollo di trasporto utilizza il meccanismo delle porte.

Una porta è un valore numerico di due Byte che identifica un canale. Esse possono assumere valore da 0 a 65535. L’utilizzo di porte differenti permette più comunicazioni sulla stessa rete. Si viene allora a delineare il concetto di socket

Il socket è un indirizzo numerico formato dall’indirizzo IP e dal numero che individua la porta:

IP locale:porta locale

Una connessione tra due computer viene identificata dalle coppie:

A. Indirizzo IP mittente: porta mittente
B. Indirizzo IP destinatario: porta destinatario

I valori delle porte sono scritti nell’header

I numeri delle porte però non hanno tutti lo stesso significato

- 0-1023 sono porte per applicazioni particolari
- 1024 – 49151 sono porte riservate
- 49152 -65535 sono numeri liberi

Esempi di porte

21/tcp FTP

22/tcp SSH secure shell

25/tcp SMTP

42 WINS Windows Internet Naming Service

53 DNS

80/tcp http

110/tcp POP3 Post Office Protocol, v3

QoS

Il livello di trasporto si occupa anche della qualità del servizio Quality of Service

I parametri indicatori della qualità di servizio sono:

- ritardo massimo nell'attivazione della connessione
- numero di byte trasferiti nell'unità di tempo
- velocità di consegna
- probabilità di fallimento della connessione
- probabilità che la connessione non venga stabilita entro il massimo tempo di ritardo
- tasso di errore
- protezione contro le intercettazioni dati
- priorità della connessione

UDP

È stato concepito per tutte quelle applicazioni per le quali non è necessaria una completa gestione delle connessioni.

Non è richiesto un messaggio di accettazione, acnowledgement e nemmeno di un messaggio di handshaking dove i dispositivi definiscono i protocolli e le velocità di trasmissione.

Le applicazioni sono:

telefonia voip, protocolli di instradamento RIP, risoluzione dei nomi DNS, amministrazione di rete SNMP, file server remoti NFS, network time protocol

Un datagram UDP è fatto nel seguente modo:

IP Header (20 Bytes)	UDP header (8 Bytes)	UDP data

Un socket è un è una coppia di parametri <indirizzo IP: Porta >

Es. Il server ha indirizzo 130.130.12.17 e punto di accesso 309. Il server manda in esecuzione l'applicazione e si mette in attesa fino a che il client non invia un datagram

Il server ha socket <130.130.12.17:309>

Quando il server riceve un messaggio:

- legge il numero di porta del mittente
- estrae il messaggio contenuto nel segmento
- invia il messaggio al socket specificato
- es: sia un destinatario A con socket <9.12.0.54:300> e destinatario B con socket <137.200.70.14:3010>

Il servizio di trasferimento affidabile

Lo strato di trasporto attua meccanismi che permettono di eliminare i problemi presenti agli strati di trasporto

Un servizio di trasferimento si dice affidabile se:

- tutti i messaggi sono consegnati a destinazione e giungono privi di errori
- ciascun messaggio è consegnato una e una sola volta
- i messaggi sono consegnati nello stesso ordine in cui sono consegnati

L trasmissione deve essere:

- priva di errori
- senza perdita di dati
- senza duplicazioni nella consegna dei segmenti

meccanismi impiegati

I meccanismi impiegati per realizzare un trasferimento affidabile sono i seguenti:

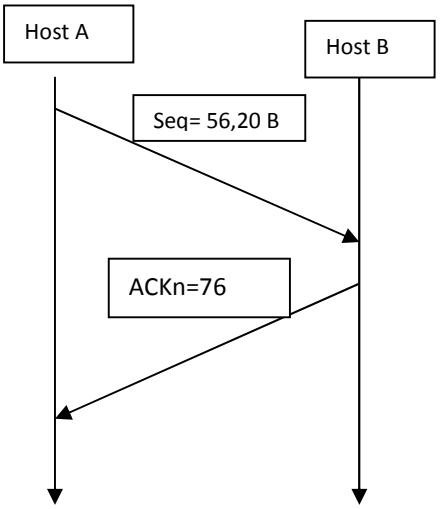
- numerazione dei segmenti trasmessi e trasmissione dei segnali di riscontro con numero di sequenza
- impiego di temporizzazione
- impiego di finestre in trasmissione e ricezione

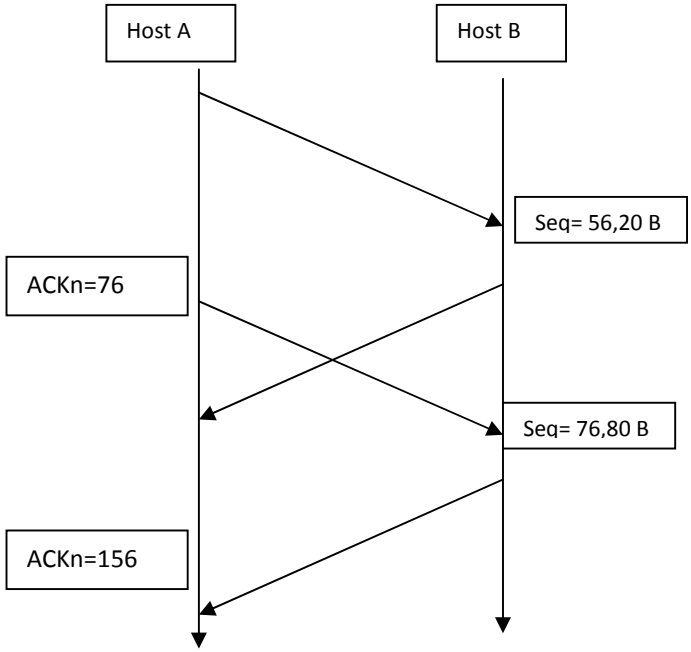
Numerazione dei segmenti trasmessi

- I dati sono una sequenza di bit e, sono una sequenza ordinata
- I dati vengono suddivisi in pacchetti formati da un certo numero di byte
- Ogni pacchetto ha un numero d'ordine che corrisponde al numero d'ordine del primo bit in esso contenuto ed è detto SN, sequence number

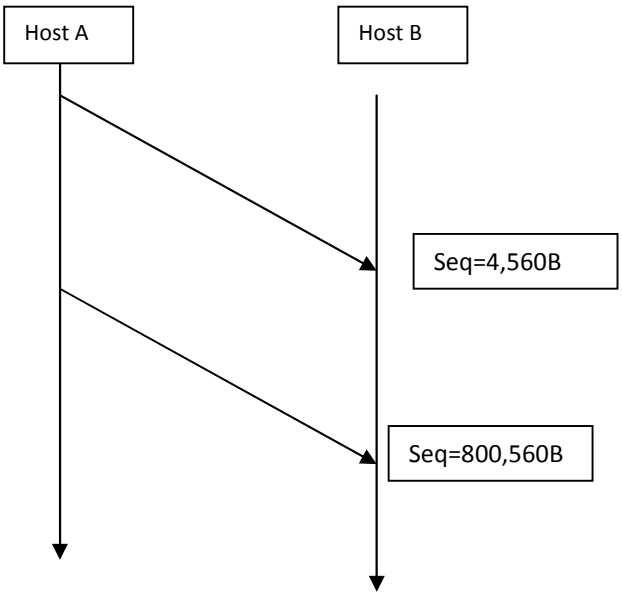
- Il pacchetto viene inizializzato da un numero detto ISN=0 Initial Sequence Number
- Il primo byte vrà sequence number SN=ISN+1
- Il pacchetto successivo avrà SN=SN+k, dove k è il numero di byte del pacchetto

Es di trasmissione andata a buon fine



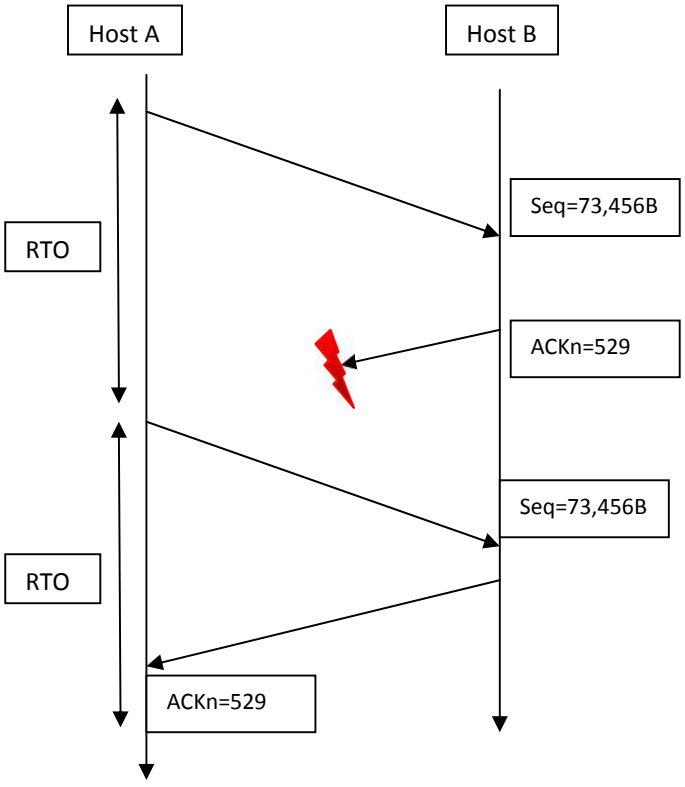


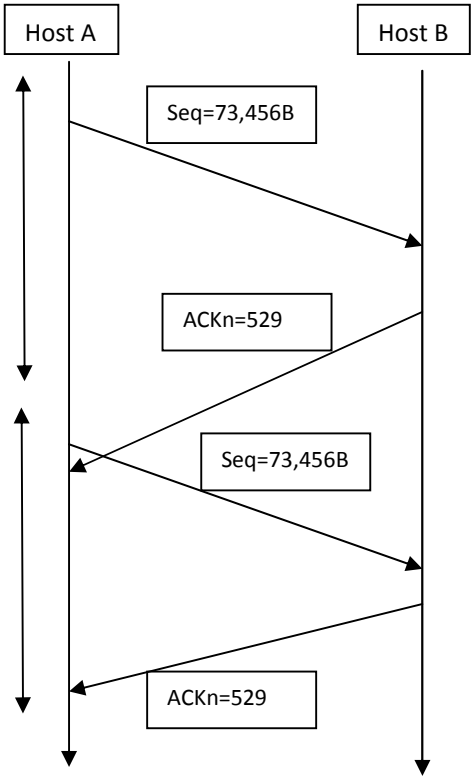
Es di ricezione di dati mancanti



Temporizzazione della trasmissione

Per ciascun segmento inviato, TCP avvia un timer detto timer di ritrasmissione RTO Retrasmission Time Out indicato come time out; indica dopo quanto tempo deve considerarsi perso l'ultimo segmento trasmesso e quindi, bisogna ritrasmetterlo se nessun ACK viene pervenuta.





Gli esempi precedenti evidenziano che nel primo caso, il segnale ACK non giunge all'host mittente e, il segnale viene inviato di nuovo quando scade il tempo RTO; nel secondo caso, il segnale ACK giunge al mittente ma non nel tempo utile RTO e quindi, i dati vengono inviati di nuovo.

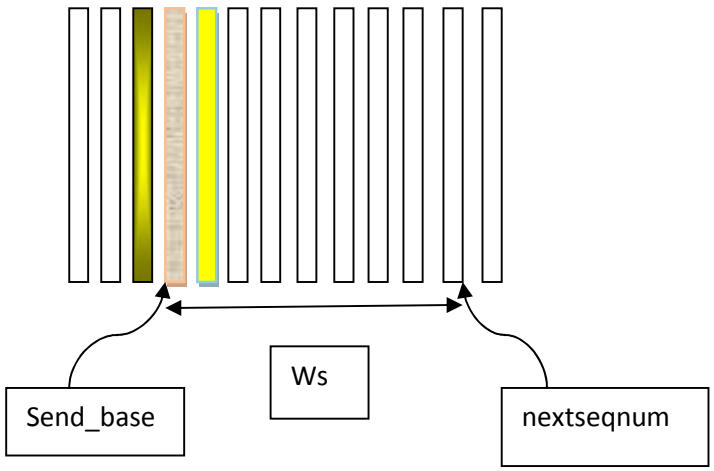
Il time di Keepalive è un ulteriore temporizzazione. Inizia il conteggio alla ricezione di ogni pacchetto e dichiara scaduta la connessione se c'è un tempo di inattività

Finestra di trasmissione e ricezione

Per effettuare la gestione dei segnali inviati e ricevuti il terminale mittente mette a disposizione una finestra di trasmissione.

La finestra di trasmissione viene gestita come una struttura a coda utilizzando due variabili: sendbase e nextseqnum

- Sendbase= il numero d'ordine del byte più vecchio tra quelli trasmessi ma di cui non si conosce l'esito. È l'estremo inferiore della finestra
- Nextseqnum= il numero d'ordine del prossimo byte che deve essere ancora trasmesso



Ws(byte)=larghezza della finestra

Protocollo TCP

La connessione avviene tramite handshaking cioè tramite lo scambio di messaggi di controllo. Questo tipo di connessione si chiama punto punto perché si crea un collegamento diretto tra client e server.

La prima fase della connessione è quello dell'invio di un pacchetto dati tra due computer per stabilire i criteri di connessione. Successivamente, avviene il trasferimento dati bidirezionale.

Ogni pacchetto viaggia attraverso un canale che si crea ogni volta tra client e server. Il server può creare diverse connessioni contemporaneamente ma ognuna, è identificata da un unico socket .

Il suono prodotto dal modem quando si ha l'inizio della connessione si chiama handashaking

Successivamente avviene la comunicazione vera e propria per un flusso di dati affidabile full duplex

Il segmento dati TCP è simile al segmento dati UDP non più di 64 KB

IP header (20B)	TCP header (20B)	Data

Apertura della connessione

La procedura per instaurare una connessione è detta handshaking a tre vie ed avviene secondo la seguente modalità

1. Il server manda in esecuzione l'applicazione e rimane in ascolto passivo (passive open)
2. Se il client vuole comunicare con il server manda l'applicativo specifico con l'indirizzo IP del server ed il numero di porta. Viene quindi inviata una richiesta attiva (active open)
3. Il client TCP genera un numero casuale di sequenza Seq (es Seq=55000) ed invia un messaggio di sincronizzazione SYNchronize, flag SYN=1
4. Alla ricezione del segmento SYN=1 il server genera un numero di sequenza seq=8000 e risponde con un segnale SYN=1 e ACK=1, ACKn=55001.
5. Alla ricezione del SYN/ACK il client risponde con un ack di conferma ACKn=8001 e Seq=55001
6. Inizia lo scambio vero dei dati

Chiusura della connessione

La chiusura della connessione è più delicata dell'apertura. Se la connessione non viene fatta contemporaneamente, o da entrambe le parti, uno dei due può inviare ancora dati che vengono persi.

Esistono due modi per chiudere la connessione: con handshaking a tre vie e con handshaking a quattro vie. La prima avviene se il client e il server decidono di chiudere contemporaneamente la connessione; la seconda invece, si ha quando solo il client di chiudere la conessione.

Nel caso di chiusura con handshanking a tre vie, la modalità è analoga a quella di apertura della connessione. L'unica differenza è che al posto del flag SYN c'è il flag FIN.

Con handshake a quattro vie la modalità di chiusura avviene secondo la seguente sequenza:

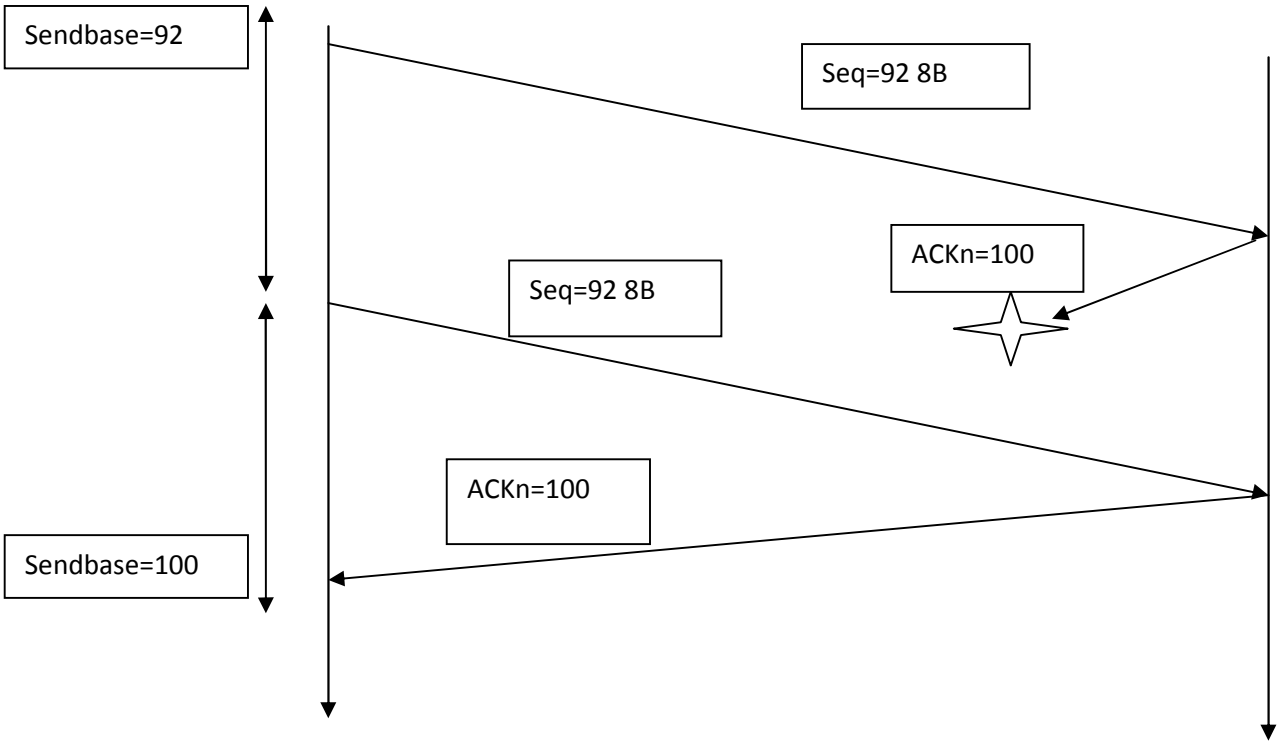
1. Il client inizia la procedura di chiusura della connessione ed invia il flag FIN=1 con un numero casuale da esso generato es Seq=600
2. Il server risponde con il flag ACK=1 e ACKn=601
3. Se il server decide di chiudere la connessione invia un flag FIN=1 con un numero casuale da esso generato es Seq=770
4. Questo messaggio di FIN viene confermato dal client con risposta ACK=1 e ACKn=771

TCP:Problemi di connessione e congestione

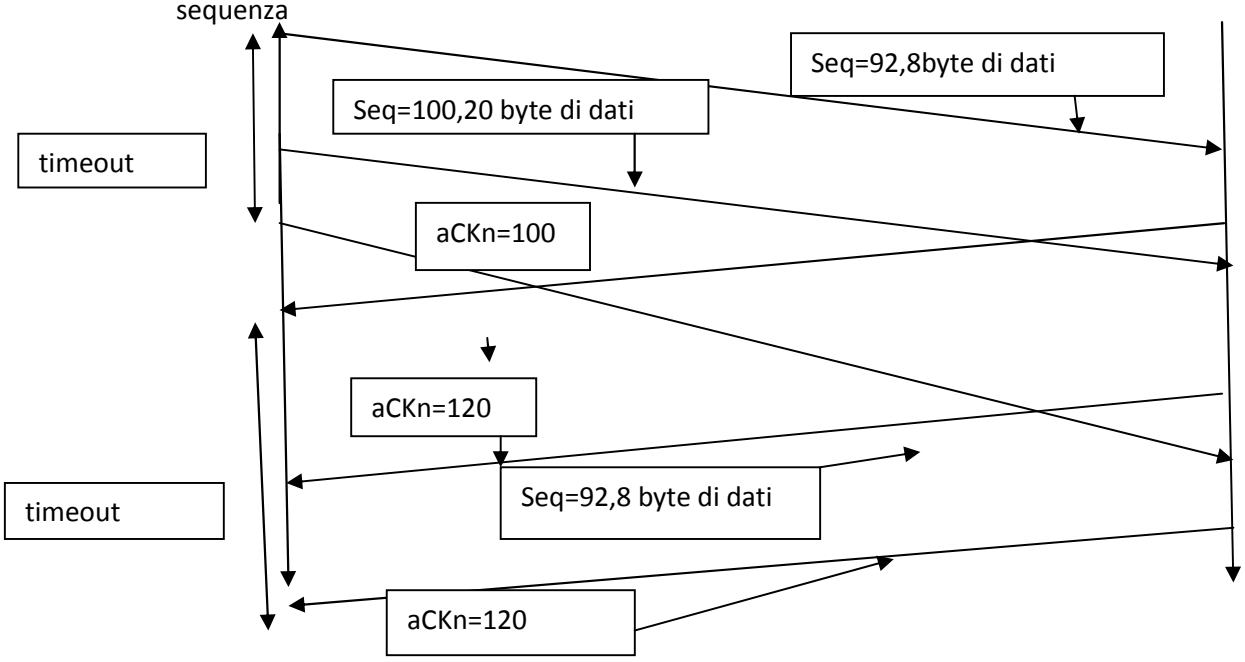
Durante la fase della connessione, a causa dei ritardi interni nell'invio dell'ack, la subnet può perdere o duplicare i pacchetti e possono quindi arrivare dati duplicati o non nella giusta sequenza oppure che vengono aperte più connessioni nello stesso canale.

Problemi durante la connessione

1. Nel caso in cui il mittente invii un solo messaggio nel timeout e non riceve un ACK positivo, non può rispondere con la successiva trasmissione. Il mittente invia di nuovo i dati anche se sono stati già ricevuti dal destinatario.



2. Il mittente invia una sequenza di dati nello stesso timeout prima di attendere la conferma. Nel caso in cui la conferma arrivi fuori dal timeout, il mittente deve rinviare tutti i dati perché immagina che siano stati tutti persi. Il destinatario riconosce il problema ed invia solo l'ACK relativo all'ultima sequenza



Programmazione del socket

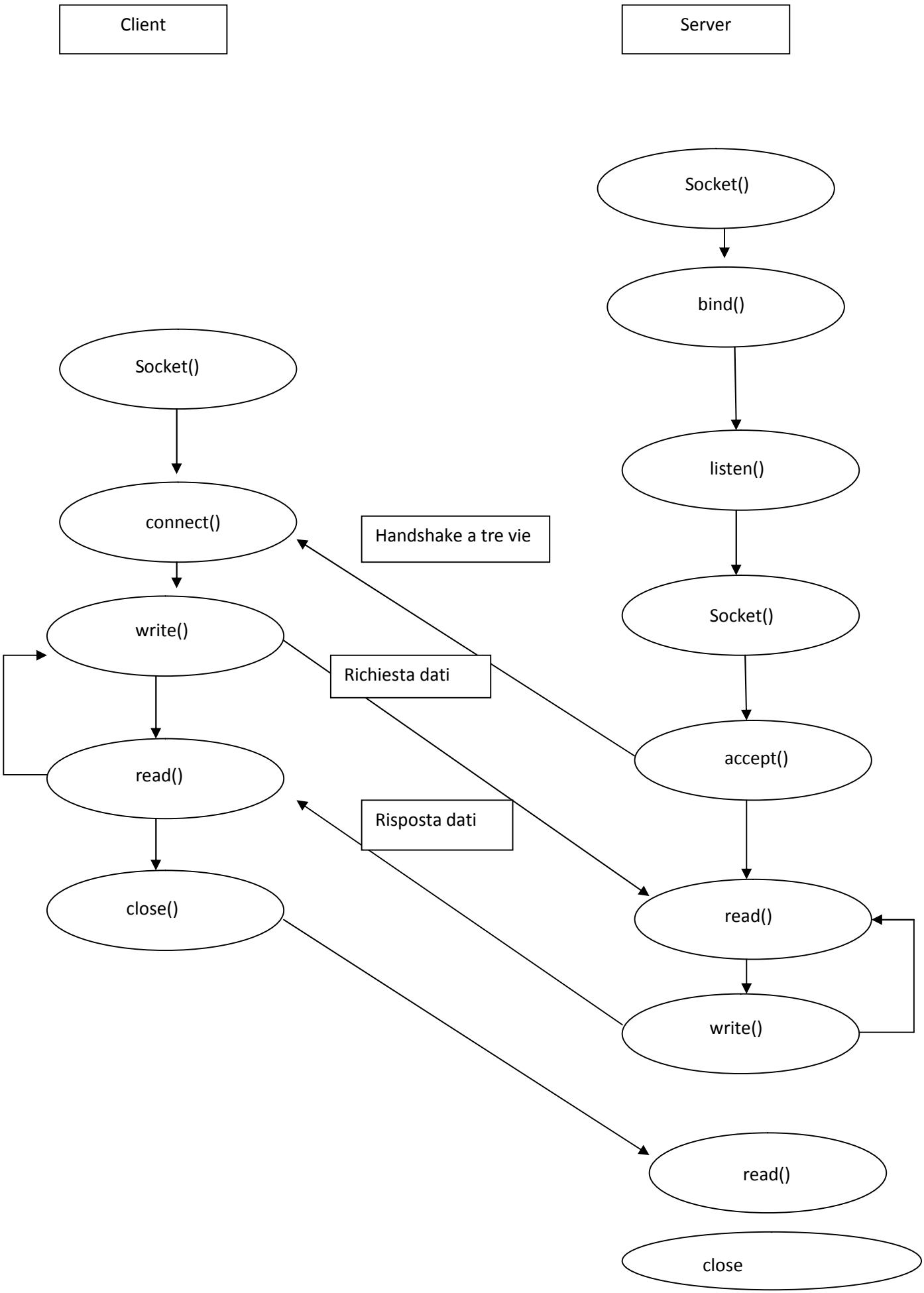
Pseudo codice

Lato server

```
main(){
  socket(indirizzo_IP,protocollo_TCP);
  bind(descrittore_socket);  // associazione al socket del suo descrittore con il
//numero di porta
  listen();
  while(1){
   while(accept());
  if(!fork()){
       do{
            read(buff);
          }
      write("fatto");
      close();
     }
    close();
   }
```

Client

```
main(){
  socket(indirizzo_IP,Uso di TCP);
 port_server=5517;
hp=gethostbyname(“server”);
connect();
do{leggi_stringa();
write();
}while(“quit”);
read();
close();
}
```


Client
Server
Socket()
bind()
listen()
Socket()
accept()
read()
write()
read()
close
Socket()
connect()
write()
read()
close()
Handshake a tre vie
Richiesta dati
Risposta dati