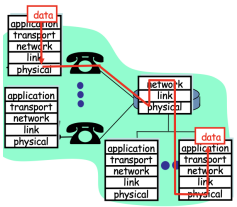
Socket

Monday, 20 March 2023 11:09

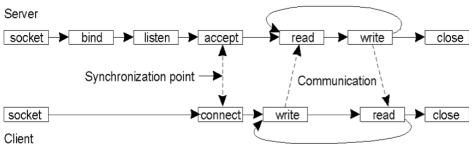
Per comprendere le socket, prima ci dobbiamo ricordare ISO/OSI



- E' un qualcosa a layer
- Modelli:
  - Client/Server
  - Peer-to-Peer
- I programmi sono eseguiti dai processi
  - Programmi = Sequenza di istruzioni eseguibili
  - Processi = Entità gestite dal sistema operativo che è una memoria in RAM + registri + canali di comunicazioni
    E questi canali sono ingresso/uscita, che interno=canale, esterno=porta
- TPC
  - Connection oriented
  - Affidabile
  - Controllo flusso = rallenta se l'altro non ce la fa
  - Controllo congestione = rallenta quando rete è sovraccaricata
  - NO garanzia banda o ritardi
  - Politiche:
    - Scompone ed invia come UDP
    - Ogni segmento numerato
    - Utilizza variabili e buffer per trasferimento bidirezionale
    - Prevede client/server per connessione
    - Non prevede client/server per la comunicazione
- UDP
  - Non affidabile
  - No connection less e tutte le cose belle scritte sopra
  - Però è veloce, facile, ottimo per applicazioni con tolleranza perdite
  - Politiche:
    - Scompone il flusso in byte e li invia uno a volta

Tutto questo per dire che, i socket sono particolari canali tra processi senza memoria condivisa

- Bisogna conoscere host e porta
- Sono delle API per accedere a TCP e UDP
  Aka due processi comunicano inviando/leggendo dati dalle socket
- Problemi:
  - Ciclo di vita: come si termina/attiva?
    Ex. Manuale/Middleware
  - Come si identifica accesso al server?
    - Chiede
    - DNS
    - DHCP
    - Hard coded nel codice
  - Comunicazioni: quali sono i dati inviati
  - Chi fa cosa in client/server

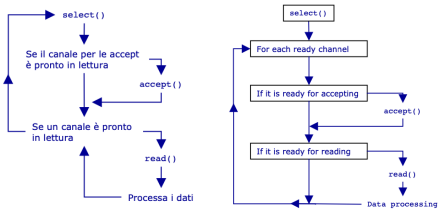


- Il client crea una socket, facendo binding automatico, e poi fa una connect al server
- Il server, dopo essersi creato:
  - Crea socket
  - Fa il bind alla porta
  - Inizia ad ascoltare

  Ascolta per possibile connect, ed appena sente una connect crea una nuova socket alla porta well knwon desiderata
  Che servirà per la comunicazione per il client.
  Perché? Senò come cavolo ascolta nuovi client?
- Inizia una comunicazione non più client server siccome, qui tutti e due possono sia scrivere che leggere dall'altro senza l'attesa di uno.

Il server può essere di diverse tipologie a seconda di come come crea i thread per la listen:

- Iterativa, non la creiamo e soddisfiamo 1 sola richiesta alla volta
- Corcorrente:
  - Processo singolo
    Noi qui rendiamo i nostri input da bloccanti, a non bloccanti
    Quando accettiamo un client, noi facciamo tutto ciò che dobbiamo fare fino a che non veniamo bloccati da esempio, dobbiamo ascoltare se ci dice qualcosa. Qui, anziché rimanere bloccati ad aspettare noi rendiamo questo input non bloccanti, attraverso la select(), e ritorniamo a fare la listen.
    Se nessuno è in listen, ritorniamo a controllare se il nostro client ci ha scritto qualcosa



  - Multi processo
    Noi qui usiamo il costrutto fork() per creare un nuovo processo

Che è lo stesso del padre, sad questo è quasi impossibile fare su java
Puoi anche ignorare tutto ciò che c'è scritto sul pdf riguardante questo

- Multi thread
  Questo ci interessa: noi creiamo 1 thread per ogni nuovo client, and that's it
  Fine