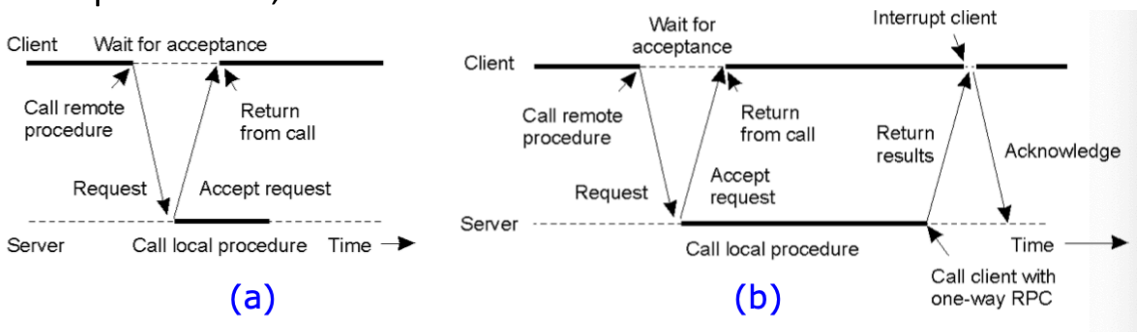
Remote Procedure call

Tuesday, 2 May 2023 10:32

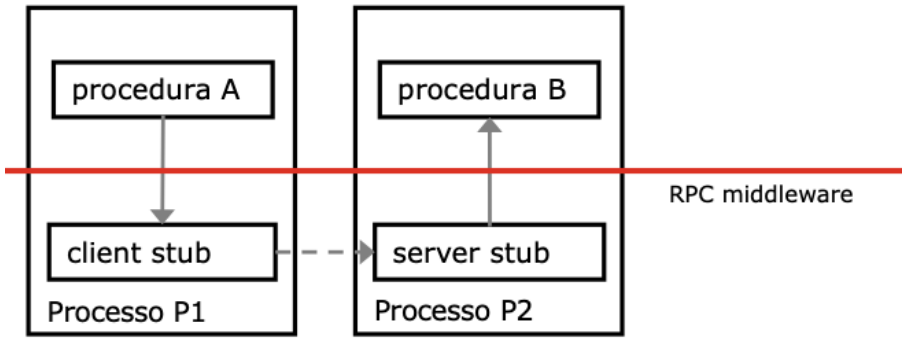
- Sono una estensione del protocollo di chiamata procedurale
  E quindi ereditano i seguenti vantaggi:
  - Semantica nota
  - Facile implementazione

  Con i seguenti nuovi problemi:
  - Sono procedure bloccanti, quindi si può avviare solamente con asincronizzazione
    Ed in questo caso, esistono 2 casistiche di bloccanti:



    A. Il client invia ed aspetta risposta del server, ed il server restituisce una risposta prima di eseguire la richiesta
    B. Come prima però, sta volta il server chiama il client anche dopo aver finito di eseguire la sua richiesta

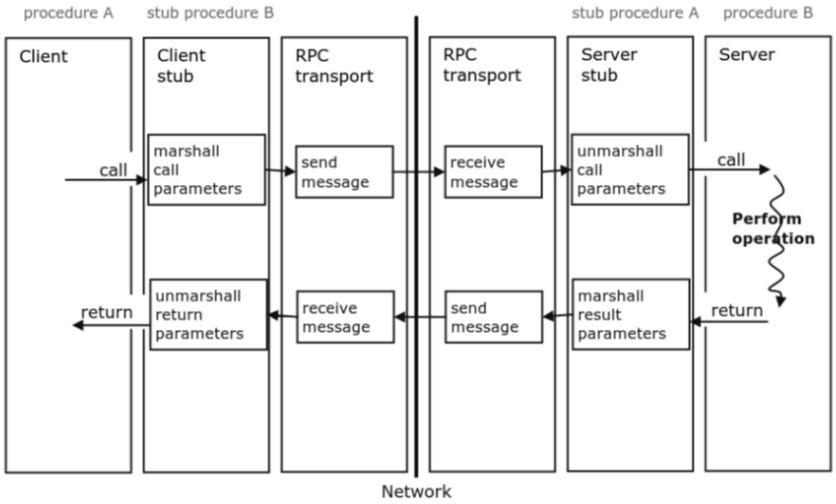
  E tutto questo per potere simulare ad il client ed il server di avere delle variabili locali, aka permettere il passaggio di variabili.
  E qui viene definito lo **Stub**, aka ciò che permette lo scambio di informazioni



  Piccola chiarifica tra puntatore e referenza:
  - Un puntatore è una variabile che tiene un indirizzo di memoria
  - Una referenza è tipo, un puntatore immutabile. Quando un puntatore vuole sapere oppure cambiare il valore dell'indirizzo, deve diventare una referenza, possiamo immaginare il puntatore come "indirizzo di

casa" mentre la referenza come "sono sotto casa tua"

- Come funziona il passaggio dei parametri
  1) Il client richiama la procedura remota
  2) Lo stub costruisce il messaggio
  3) Il messaggio viene inviato, attraverso la rete, al server (Marshalling, replica stack)
  4) Il sistema del server gestisce i messaggio e lo trasferisce allo stub del server
  5) Lo stub spacchetta il messaggio
  6) Lo stub gestisce localmente la richiesta (unmarshalling, puntatori)



- Qui viene creato RMI
  - E' un middleware che permette l'invocazione di metodi su macchine distinte
  - Si basa sulla portabilità del bytecode
    - Aka noi possiamo convertire intere .class , che ci permette di trasferire non solo parti di memoria ma anche funzioni
    - Utilizza sia interfacce statiche, aka interfacce note in compilazione, ma anche invocazioni dinamiche, cioè noi inviamo informazioni aggiuntive dell'oggetto
- In un mondo oggetti si scambiano oggetti, ed avvengono in 2 modi:
  - Reference
    - Individuare un oggetto dentro ad una virtual machine java, e c'è bisogno delle seguenti informazioni:
      - Macchina su cui sta
      - La porta del processo
      - L'oggetto stesso
    - Condiviso
    - Avviene tramite serializzazione
      - Serializzare vuol dire rappresentare stato di un oggetto come stream di byte

- Questo stream di byte si può sia inviare via rete che nel file system
- La descrizione della classe viene passata a parte, qui noi abbiamo solo i valori

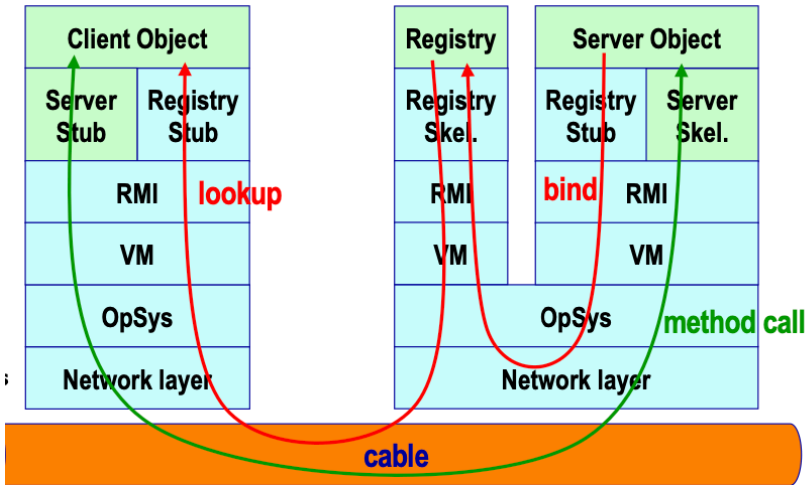
```
FileOutputStream f = new FileOutputStream("tmp")
ObjectOutput  s  =  new  ObjectOutputStream(f);
s.writeObject("Today");
s.writeObject(new Date());
s.flush();

 Deserialize a string and date from a file.
FileInputStream in = new FileInputStream("tmp");
ObjectInputStream s = new ObjectInputStream(in);
String today = (String)s.readObject();
Date date = (Date)s.readObject();
```

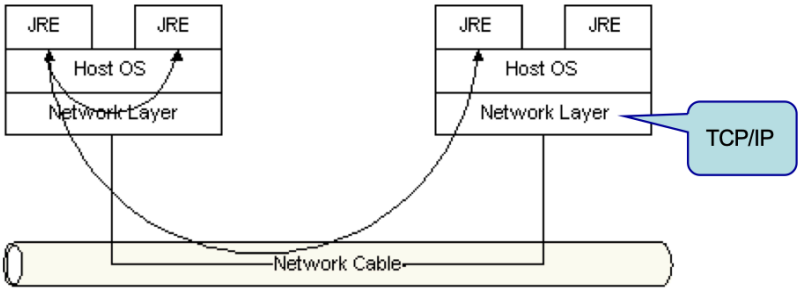
Per serializzare una classe dobbiamo implementare le funzioni writeObject, readObject nella nostra classe

- Il riferimento all'oggetto si passa per valore
- Per accedere ad un oggetto remoto
  - Quando un server crea un file condivisibilmente remoto, il server registra (bind) il riferimento con un nome al rmiregistry
  - Il client fa una lookup al nome che ritorna una riferenza
  - Ora il client può fare normali invocazioni al server
    - Si utilizza la classe naming

Architettura:



Schema semplificato:



- Problemi RMI:

        - Problemi RMI:
            - Naming: identificazione
              Host name + object che desideriamo
            - Access poiint
              Prima prendiamo la riferenza dal RMI registri, e poi possiamo lavorare in maniera locale
            - Communication 1, data format
              Java object
            - Communication 2, semantica
              Tipi di java (classi e metodi)

          Qual'è il livello di trasparenza?
            - Basso no siccome non diamo i byte, ma diamo bytecode, quindi abbiamo anche i tipi
            - Direi medio siccome l'ip macchina non è assoluta e condivisa per tutti quanti al mondo
    - Copio l'intero oggetto (passaggio per valore)
        - Avviene nelle copie locali
        - Copiamo la classe, quindi con stato ecc
            - Gli oggetti non remoti possono essere passati per valori solamente se serializzabili

    E per trasferire un oggetto abbiamo bisogno:
    - Classe, aka la struttura
    - Dati, quindi i suoi dati effettivi
- Se voglio trasferire una coppie di una classe in java:
    - Guardo se ho la classe studente, se non ce l'ho la chiedo al richiedente
    - I tipi primitivi vengono passati per valore
    - Per gli oggetti non remoti, aka li abbiamo in locali, passati per valore se sono serializzabili