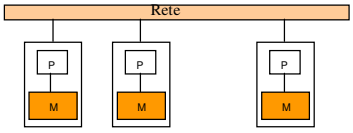
Comunicazione tra Processi

Comunicazioni in un Sistema Distribuito

- Un sistema software distribuito è realizzato tramite un insieme di processi che comunicano, si sincronizzano, cooperano.
- Il meccanismo di comunicazione di basso livello in un sistema distribuito è lo **scambio di messaggi**.
- Su di esso possono essere costruiti meccanismi di comunicazione più semplici:
  - Remote procedure call,
  - Active messages,
  - Publish/subscribe,
  - Streams

Linguaggi a Memoria Distribuita

- Questi linguaggi riflettono il modello dei calcolatori a memoria distribuita composti da un insieme di elementi di elaborazione connessi da una rete (multicomputers, clusters, LAN).



- In questo modello un programma parallelo consiste da un insieme di processi in esecuzione su più processori che cooperano tramite lo scambio di messaggi (message passing).

- Due principali aspetti in questo tipo di programmazione sono la creazione/attivazione dei processi concorrenti ed la loro cooperazione.

Linguaggi a Memoria Distribuita

- Alcuni forniscono delle primitive per la creazione esplicita dei processi durante l'esecuzione del programma (*creazione dinamica*):

  `fork/join`, **`new`** e **`create`**.

- In altri il numero dei processi è definito a tempo di compilazione (*creazione statica*):

  `par`, **`parbegin`**, **`cobegin/coend`**.

MPI

- MPI (*Message Passing Interface*) è una libreria standard per lo sviluppo di programmi paralleli e distribuiti attraverso primitive di scambio messaggi.
- MPI è disponibile per macchine massicciamente parallele, reti di workstation eterogenee, PC, etc. → applicazioni portabili
- Un programma parallelo in MPI è strutturato come una collezione di processi concorrenti che eseguono programmi scritti in un linguaggio sequenziale con chiamate ad una libreria (MPI) per realizzare lo scambio di messaggi.

In MPI-1 non c'è creazione di processi !

MPI

- La libreria MPI contiene funzioni per supportare la comunicazione punto-a-punto fra coppie di processi, come ad esempio

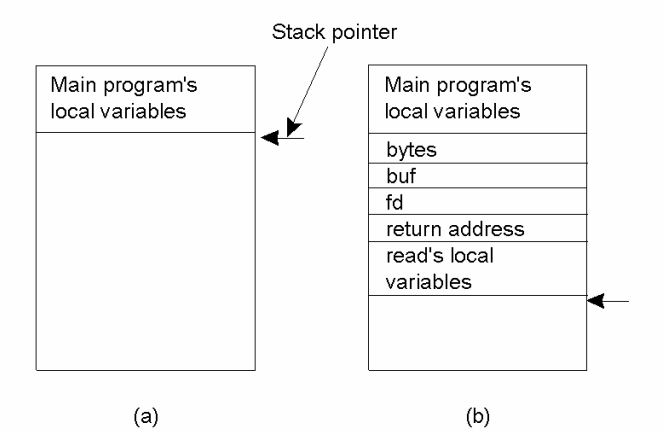
```
MPI_Send(mess, strlen(mess)+1, type, 1, tag, MPI_COM);
MPI_Recv(mess, leng, type,0, tag, MPI_COM, &status);
```

- Le funzioni per comunicazioni collettive all'interno di gruppi di processi come:

```
MPI_Bcast (inbuf, incnt, intype, root, comm);
MPI_Gather (outbuf, outcnt, outype, inbuf, incnt,..);
```

- MPI offre un modello di programmazione di basso livello, tuttavia anch'esso è molto usato molto a causa della sua portabilità.

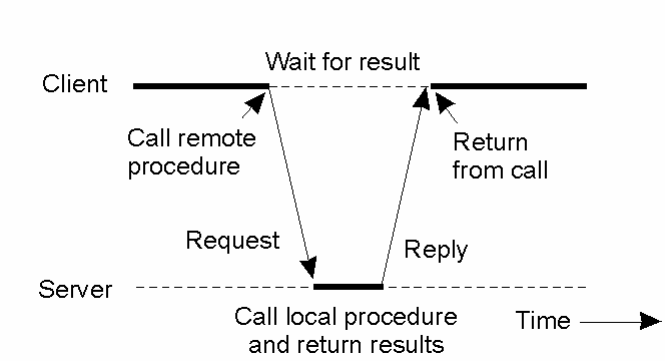
Chiamata di Procedura Convenzionale



Passaggio di parametri in una chiamata di procedura (read):

a) lo stack prima della chiamata
b) lo stack mentre la chiamata della procedura è attiva

Stubs per Client e Server

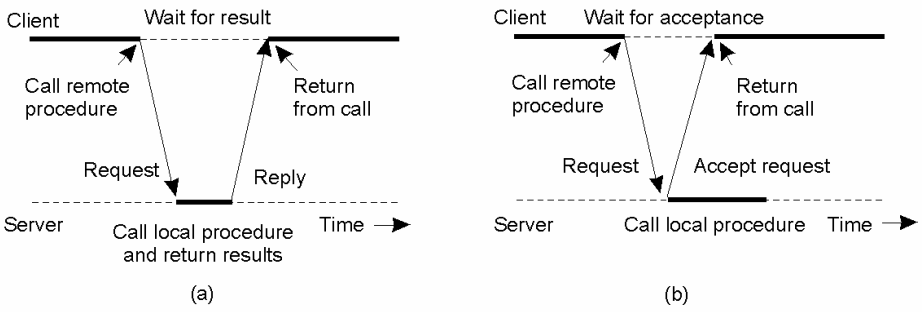


Schema di una RPC tra un programma cliente and programma server.

Passi di una Remote Procedure Call

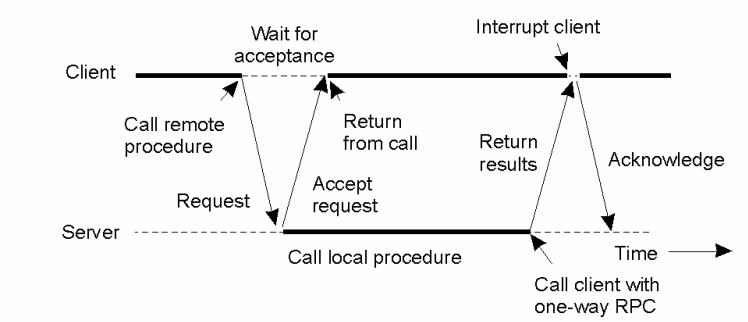
1. La chiamata di procedura del Client chiama un client stub
2. Il Client stub costruisce un messaggio e chiama il SO locale
3. Il SO locale invia un messaggio al SO remoto
4. Il SO remoto passa il messaggio al server stub
5. Il server stub preleva i parametri e invoca il Server
6. Il Server effettua le operazioni, ritorna il risultato allo stub
7. Il server stub mette il risultato in un messaggio, chiama il SO del server
8. SO del server invia il messaggio al SO del client
9. Il SO del client passa il messaggio allo stub del client
10. Lo stub preleva il risultato e lo ritorna al Client

RPC Asincrona (1)



a) L'interazione tra client e server in una RPC tradizionale
b) L'interazione usando una RPC asincrona

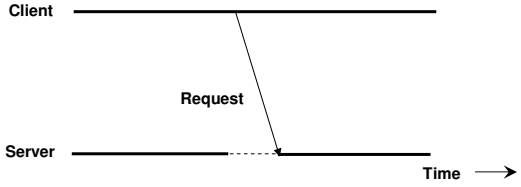
RPC Asincrona (2)



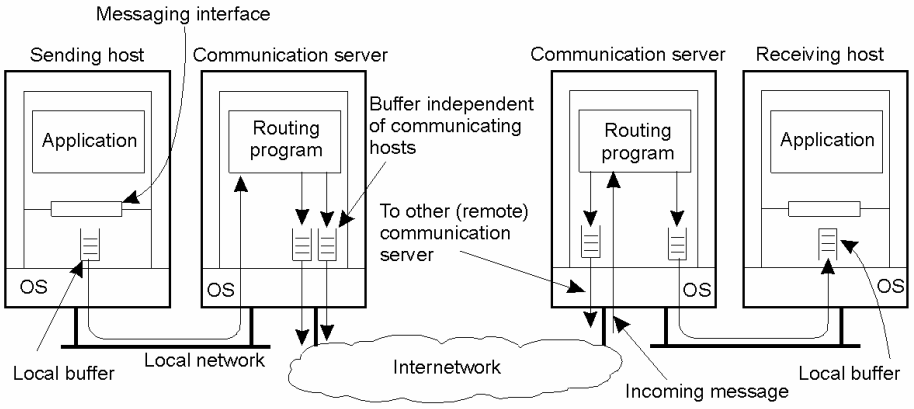
Un client e un server interagiscono con RPC asincrone

RPC Asincrona (3)

- RPC asincrone one-way
  - Il client continua dopo avere effettuato una chiamata di procedura remota
  - Simile ad una **send** senza risposta
  - Affidabilità non assicurata: il cliente non sa se la richiesta verrà servita.



Persistenza e Sincronia nella Comunicazione (1)



Organizzazione generale di un sistema di comunicazione in cui i nodi sono connessi in rete.

Persistenza e Sincronia nella Comunicazione (2)

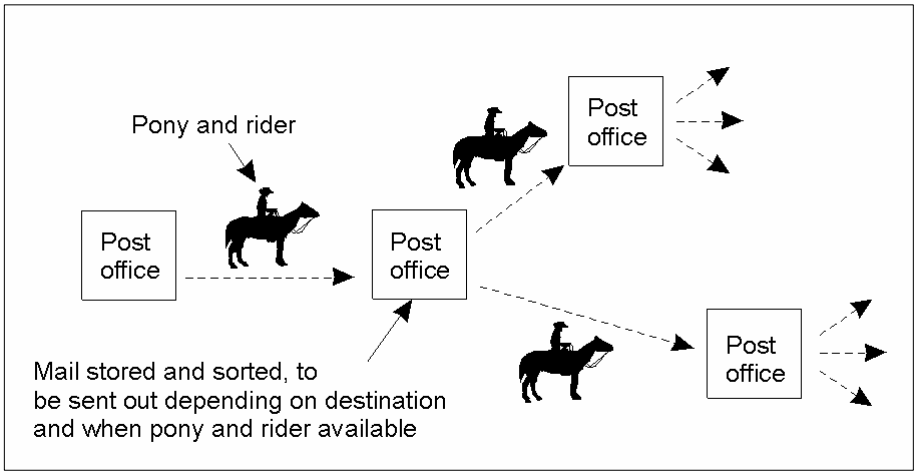
- Una comunicazione si dice **persistente** se un messaggio che è stato inviato rimane memorizzato nel sistema di comunicazione finchè non verrà consegnato al destinatario.
- Il destinatario non deve essere necessariamente attivo contemporaneamente al mittente. (Esempi: message-queuing sys)
- Una comunicazione si dice **transiente** se il messaggio viene inviato solo se il mittente e il destinatario sono attivi contemporaneamente.
- Se il communication server non puo' inviare il messaggio, questo viene eliminato. (Esempi: socket, MPI)

Persistenza e Sincronia nella Comunicazione (3)

- Le comunicazioni possono essere anche
  - **Sincrone**: il mittente si blocca fino a che il destinatario riceve il messaggio

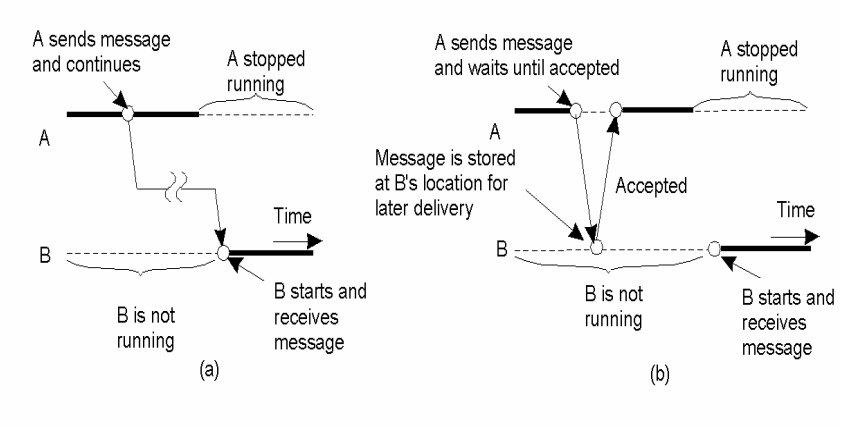
    o
  - **Asincrone**: il mittente continua senza attendere che il destinatario riceva il messaggio.
- Queste si possono combinare con le comunicazioni persistenti e transienti

Persistenza e Sincronia nella Comunicazione (4)



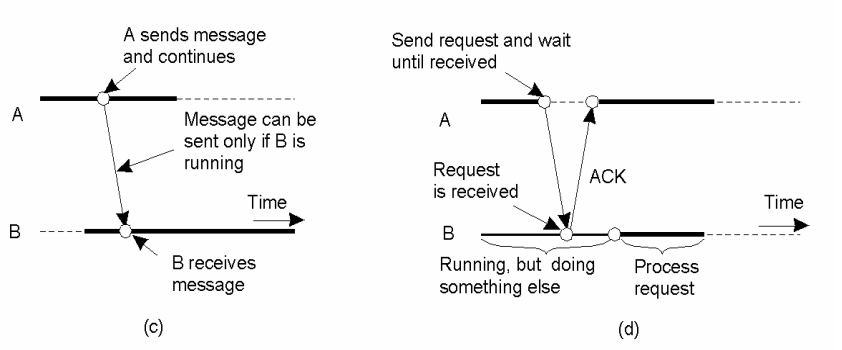
Comunicazione persistente di lettere nell'epoca del Pony Express.

Persistenza e Sincronia nella Comunicazione (5)



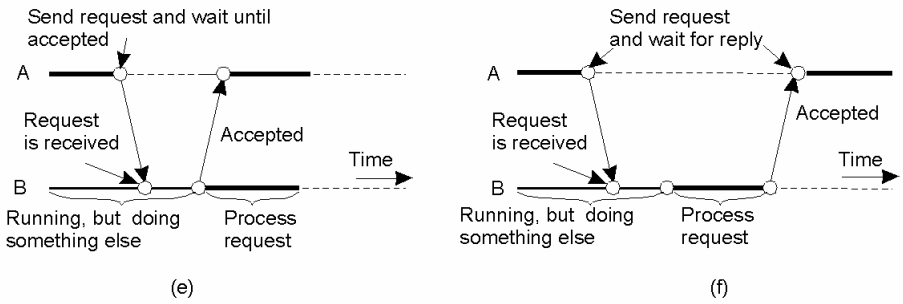
a) Comunicazione persistente asincrona
b) Comunicazione persistente sincrona

Persistenza e Sincronia nella Comunicazione (6)



c) Comunicazione transiente asincrona
d) Comunicazione transiente sincrona Receipt-based

Persistenza e Sincronia nella Comunicazione (7)

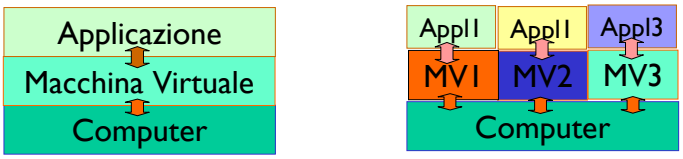


e) Comunicazione transiente sincrona Delivery-based

f) Comunicazione transiente sincrona Response-based

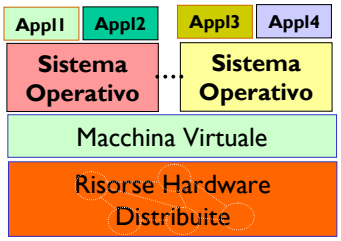
Macchine Virtuali

- Il concetto di Macchina Virtuale è stato definito molto tempo fa (anni '60) in un contesto centralizzato.
- Una Macchina Virtuale permette di rappresentare le risorse hardware diversamente dai loro limiti fisici. Un singolo computer puo' essere rappresentato e usato come differenti ambienti di elaborazione.



Macchine Virtuali in Sistemi Distribuiti

- In un sistema distribuito tramite le Macchine Virtuali si possono virtualizzare risorse remote per comporre sistemi distribuiti virtuali che integrano risorse hardware (RAM, CPU, …) presenti in siti remoti.
- Un livello di Macchina Virtuale nasconde la diversità delle piattaforme ed offre un ambiente omogeneo e "confezionato" sulle esigenze degli utenti e delle loro applicazioni



Macchine Virtuali in Sistemi Distribuiti

- Secondo questo approccio si può "costruire" dinamicamente un sistema distribuito composto da diverse CPU, memorie, dischi, reti, device, etc.
- L'approccio basato su una macchina virtuale permette di sviluppare applicazioni distribuite che usino molte macchine in maniera trasparente. (ES: JVM ?).