

# Processi, Threads e Agenti

# Processi in Sistemi Distribuiti

- Un **sistema software distribuito** è composto da un insieme di processi in esecuzione su più nodi del sistema.
- Un **algoritmo distribuito** può essere definito come un insieme  $\{P_1, P_2, \dots, P_n\}$  dove  $P_i$  è un processo.
- I processi possono comunicare, sincronizzarsi e cooperare con le stesse modalità sia se sono in esecuzione su nodi remoti, sia se eseguono sullo stesso nodo.

# Processi e Thread

- Il processo è una entità autonoma ed attiva che svolge un compito specifico.
- Il concetto di Thread ha permesso di ottimizzare l'esecuzione dei programmi, tramite una ottimizzazione delle fasi di allocazione ed esecuzione (condivisione dello stato e dello spazio degli indirizzi).
- Un singolo thread completo costituisce un processo.
- La condivisione di risorse rende differenti i thread e i processi.
- I thread permettono l'esplicitazione parallelismo (*multithreading*) in un processo. Questo è vero anche su PC multi-core.

# Multi-Threading

- Alcune fasi indipendenti in un processo possono essere eseguite tramite un insieme di threads (server multi-threaded).
- Il multi-threading può essere usato per ridurre la sincronizzazione e l'impatto delle operazioni di I/O bloccanti.
- In un server multi-thread, si possono attivare tanti thread per quante richieste sono da servire (un thread per client).
- Si può usare un pool di thread per limitare i costi della creazione di threads.

# Migrazione del Codice

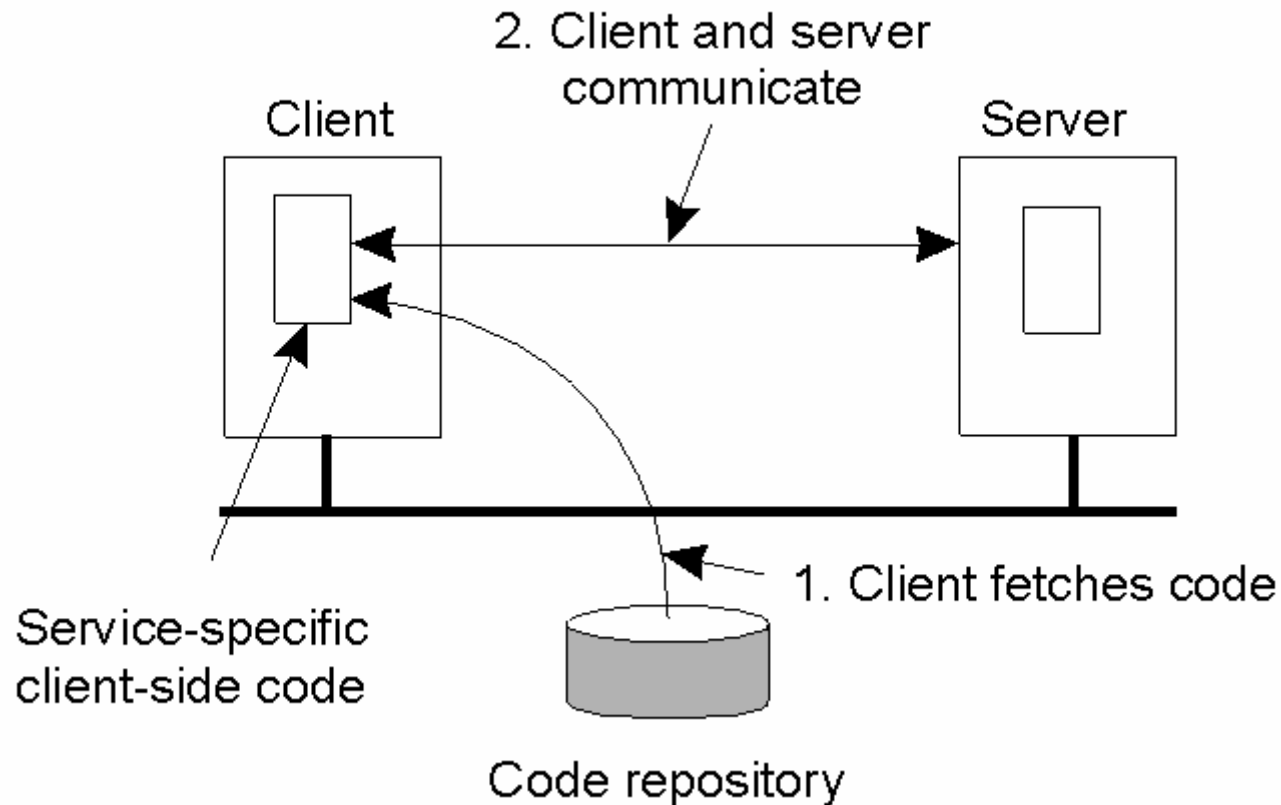
- La migrazione del Codice nei sistemi distribuiti :

process migration

- *statica*
- *dinamica*

Per migliorare le performance o per obiettivi applicativi.

# Motivi per la Migrazione del Codice



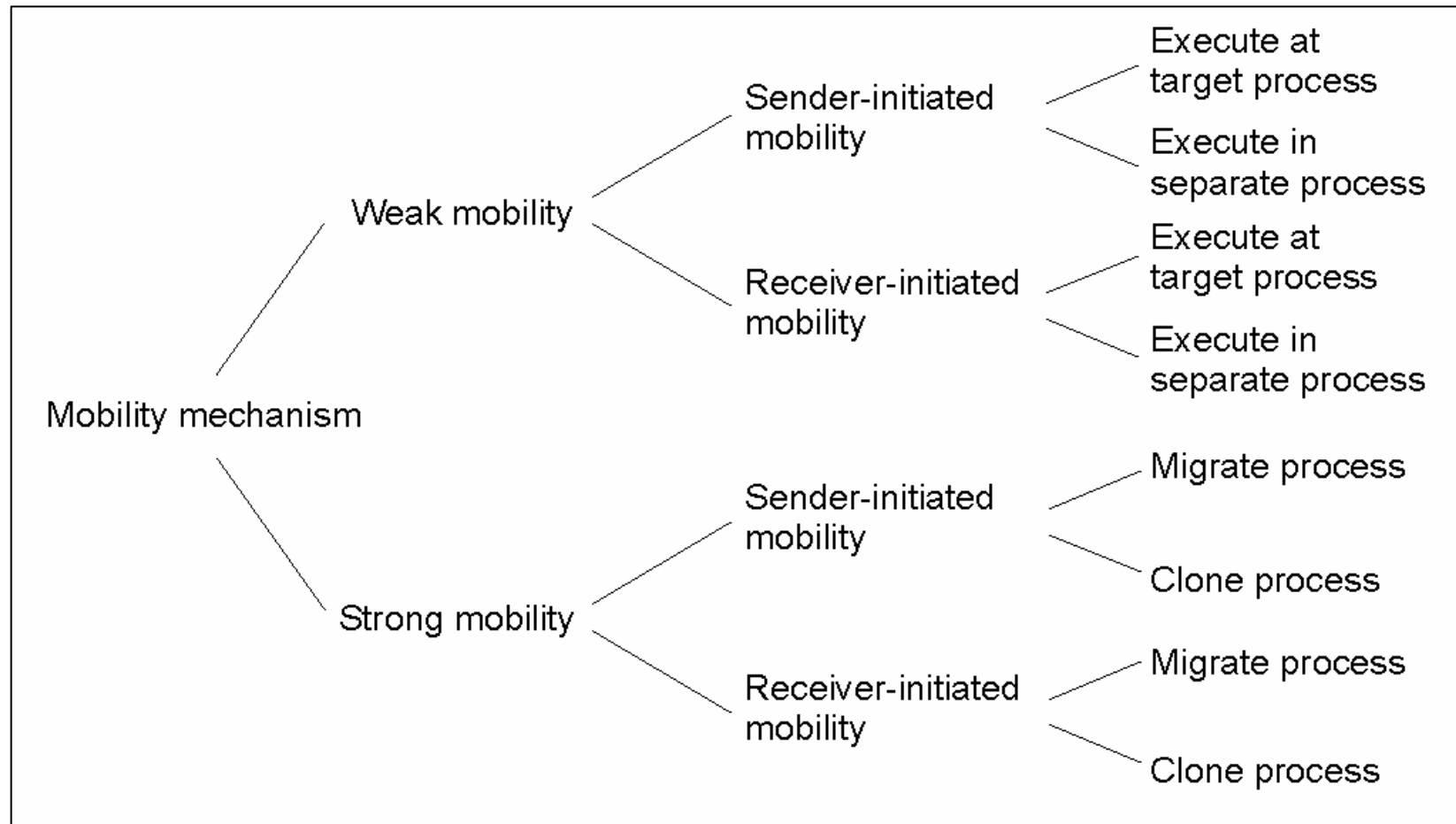
La configurazione dinamica di un cliente per comunicare con un server.

- Il cliente prima carica il software necessario (1),
- Quindi invoca il server (2).

# Migrazione del Codice

- **Weak mobility (mobilità debole):** si sposta il *code segment* e i dati di inizializzazione.  
(es. Java applet)
- **Strong mobility (mobilità forte):** si sposta il *code segment*, *execution segment* e i dati di inizializzazione.

# Modelli di Migrazione del Codice



Alternative per la migrazione codice



# Migrazione e Risorse Locali

## Collegamento Risorsa-macchina

	Unattached	Fastened	Fixed
Collegamento processo- risorsa	<b>By identifier</b> <i>MV (or GR)</i>	<i>GR (or MV)</i>	<i>GR</i>
	<b>By value</b> <i>CP ( or MV, GR)</i>	<i>GR (or CP)</i>	<i>GR</i>
	<b>By type</b> <i>RB (or GR, CP)</i>	<i>RB (or GR, CP)</i>	<i>RB (or GR)</i>

- Azioni da prendere rispetto ai riferimenti alle risorse locali quando si ha migrazione di codice verso un'altra macchina:

*MV* (move)

*GR* (global referenced)

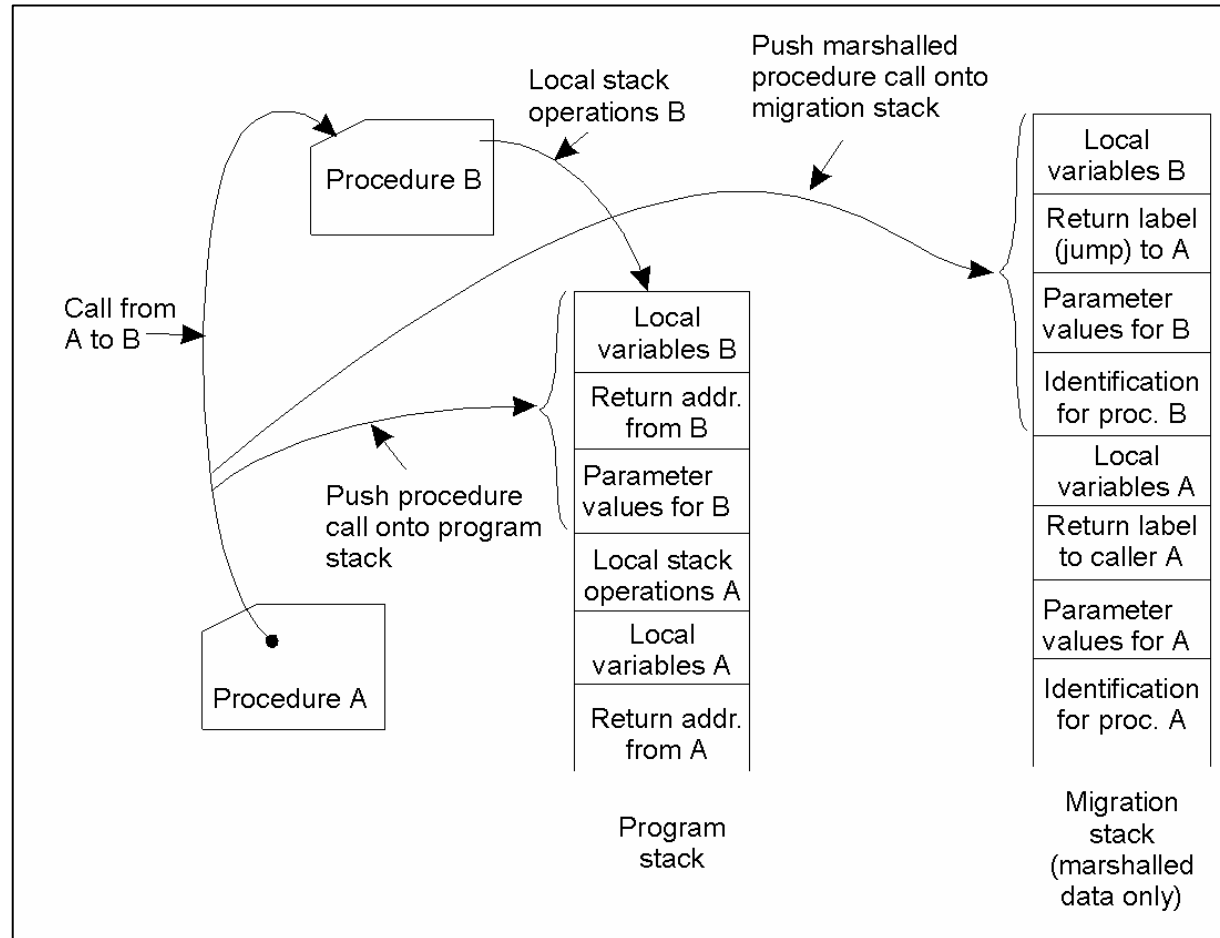
*CP* (copy)

*RB* (rebind)

# Migrazione in Sistemi Eterogenei

- E' più complesso.
- Richiede portabilità del codice.
- E' usato un approccio basato sul modello di macchina virtuale.
- La Weak mobility è più semplice.
- Nella strong mobility è necessario gestire l'execution segment.
- E' usato un migration stack.

# Migrazione in Sistemi Eterogenei



Il funzionamento basato sul migration stack per supportare la migrazione di un execution segment in un ambiente eterogeneo.

# Agenti Software in Sistemi Distribuiti

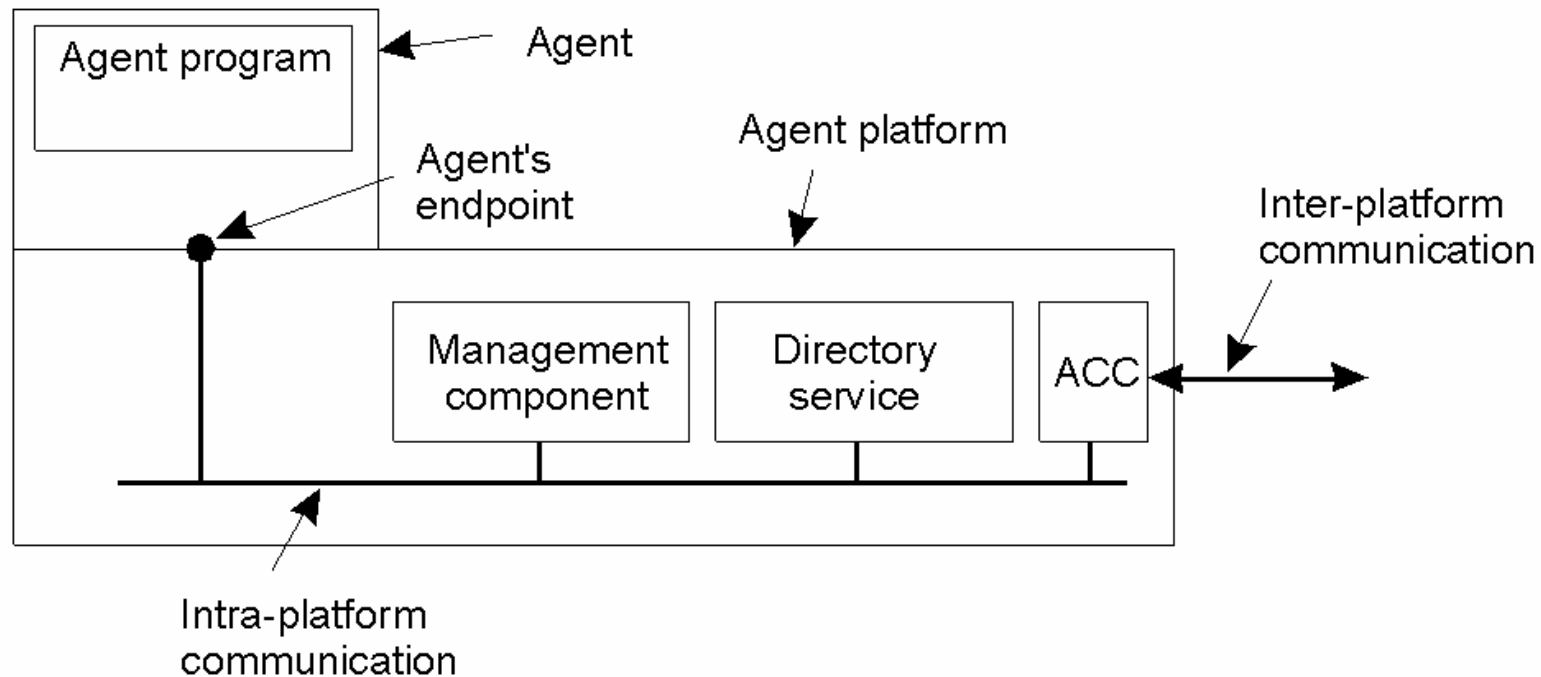
- **Software agent:** un processo *autonomo* capace di reagire e iniziare dei cambiamenti in *collaborazione* con utenti e altri agenti.
- Un agente software può prendere l'iniziativa.
- Diversi tipi di agenti:
  - Mobile agents,
  - Interface agents,
  - Information agents.

# Agenti Software in Sistemi Distribuiti

Proprietà	Comune a tutti gli agenti?	Descrizione
<b>Autonomous</b>	<b>Si</b>	Può agire di propria iniziativa
<b>Reactive</b>	<b>Si</b>	Risponde alle modifiche dell'ambiente
<b>Proactive</b>	<b>Si</b>	Inizia azioni che modificano l'ambiente esterno
<b>Communicative</b>	<b>Si</b>	Può scambiare informazione con utenti e agenti
Continuous	No	Ha un tempo di vita relativamente lungo
Mobile	No	Può migrare da un processore all'altro
Adaptive	No	E' capace di apprendere

Alcune proprietà importanti per distinguere gli agenti

# Tecnologia degli Agenti



Il modello generale per una piattaforma ad agenti (da [fipa98-mgt]).

# Agent Communication Languages (1)

Scopo del messaggio	Descrizione	Contenuto del messaggio
<i>INFORM</i>	Informa che una data proposition è vera	Proposition
<i>QUERY-IF</i>	Chiede se una data proposition è vera	Proposition
<i>QUERY-REF</i>	Cerca un dato oggetto	Expression
<i>CFP</i>	Chiede una proposal	Proposal specifics
<i>PROPOSE</i>	Fornisce una proposal	Proposal
<i>ACCEPT-PROPOSAL</i>	Afferma che una data proposal è accettata	Proposal ID
<i>REJECT-PROPOSAL</i>	Afferma che una data proposal è rifiutata	Proposal ID
<i>REQUEST</i>	Richiede che una data azione sia eseguita	Action specification
<i>SUBSCRIBE</i>	Sottoscrive per una sorgente informativa	Reference to source

Esempi di tipi di messaggi nel FIPA ACL [fipa98-acl], che descrivono lo scopo e il contenuto di un messaggio.

# Agent Communication Languages (2)

<b>Campo</b>	<b>Valore</b>
Purpose	INFORM
Sender	max@http://fanclub-beatrix.royalty-spotters.nl:7239
Receiver	elke@iiop://royalty-watcher.uk:5623
Language	Prolog
Ontology	genealogy
Content	female(beatrix),parent(beatrix,juliana,bernhard)

Un semplice esempio di una messaggio nel FIPA ACL scambiato tra due agenti che fanno uso di regole Prolog per esprimere informazioni genealogiche.



# DEFINIZIONE DI AGENTE MOBILE

## Che cosa è un agente mobile?

- Gli agenti mobili sono programmi interoperabili, altamente compatibili che possono spostarsi su nodi di elaborazione differenti e vengono eseguiti in ambienti poco sensibili ai guasti, orientati agli oggetti e con memoria sicura.

## DEFINIZIONE:

**Agente mobile:** entità software autocontenuta di stato e di comportamento, in grado di migrare in rete ed interagire con le risorse nei nodi visitati, scoprendo i servizi offerti.

# CAMPI APPLICATIVI

- **Motori di Ricerca distribuiti**

Utilizzare agenti mobili per accogliere informazioni che verranno scaricate in un database centralizzato.

- **Amministrazione remota di sistemi distribuiti**

Agenti mobili potrebbero circolare in una intranet e segnare eventuali anomalie delle risorse (stampanti senza carta, interconnessioni non funzionanti).

- **Messaggi Interattivi**

Spedire messaggi di posta elettronica che interagiscono direttamente con l'utente, implementare wizard per guidare l'utente nello svolgimento di un'operazione.

- **Applicazioni client/server**

Utilizzare client autonomi.

# CAMPI APPLICATIVI

- **Piazzista Virtuale**

I rappresentanti di prodotti potrebbero utilizzare un agente per contattare i loro fornitori, estendendo il campo d'azione a tutto il mondo oppure fornendo un servizio disponibile 24 ore su 24 per l'acquisto di articoli.

- **Debugging di applicazioni**

Applicazioni in versione beta potrebbero essere distribuite sotto forma di agenti mobili, in modo che quando si verifica un errore l'applicazione venga clonata e inviata al programmatore che potrà provvedere al debugging.

- **Ubiquitous computing**

Le applicazioni installate da un utente potranno seguirlo nel caso la sua login venga trasferita su un'altra macchina.

# SICUREZZA

- **L'esecuzione di un agente mobile implica la gestione di diversi aspetti di sicurezza.**
- **SecurityManager**  
Controllare e limitare gli accessi al file system locale.
- **Digital signatures**  
Autenticare la provenienza di un oggetto.

# AGLET: Agenti Mobili in Java

- E' una tecnologia ad agenti ideata e creata dalla IBM of Japan, completamente scritta in Java.
- Quando un aglet viene inviato, clonato o disattivato, il suo stato viene salvato con il meccanismo di serializzazione di Java. Quindi se nel codice vengono inseriti tipi che non supportano questo meccanismo, questi devono essere dichiarati transient, altrimenti verrà lanciata un'eccezione.
- Il meccanismo di serializzazione non è attualmente capace di salvare lo stack d'esecuzione dell'oggetto, infatti prima di eseguire funzioni che implicino tale operazione, nel pacchetto di bytes verrà indicato il punto di entrata del nuovo thread.

# AGLET: ARCHITETTURA

- **Aglet API**  
Insieme di comandi che permettono di utilizzare l'aglet
- **Aglets Runtime Layer**  
Implementazione delle API
- **Agent Transport and Communication Interface**  
Effettua la traduzione dei comandi con il livello di trasporto
- **Transport Layer**  
Provvede alla migrazione degli oggetti nella rete

# MIGRAZIONE DEGLI AGLET

1. Sospensione dei threads creati dall'aglet.
2. Serializzazione dello stato dell'aglet usando il meccanismo *Java Serialization*.
3. Serializzazione del bytecode della classe.
4. Trasferimento dello stream di dati a destinazione e creazione di una copia dell'aglet.
5. Inizializzazione dello stato dell'aglet.
6. Assegnazione di un nuovo thread d'esecuzione.