



Lecture 9

S4 - Core Distributed Middleware Programming in JEE

presentation

DAD – Distributed Applications Development

Cristian Toma

D.I.C.E/D.E.I.C – Department of Economic Informatics & Cybernetics

www.dice.ase.ro



Cristian Toma – Business Card



Cristian Toma

IT&C Security Master

Dorobantilor Ave., No. 15-17
010572 Bucharest - Romania

<http://ism.ase.ro>
cristian.toma@ie.ase.ro
T +40 21 319 19 00 - 310
F +40 21 319 19 00



Agenda for Lecture 7





DAD Section 4 - CORBA Intro, CORBA Architecture & API, g-RPC

CORBA – Common ORB Architecture

CORBA & g-RPC are NOT Required for Exam

1. CORBA Overview

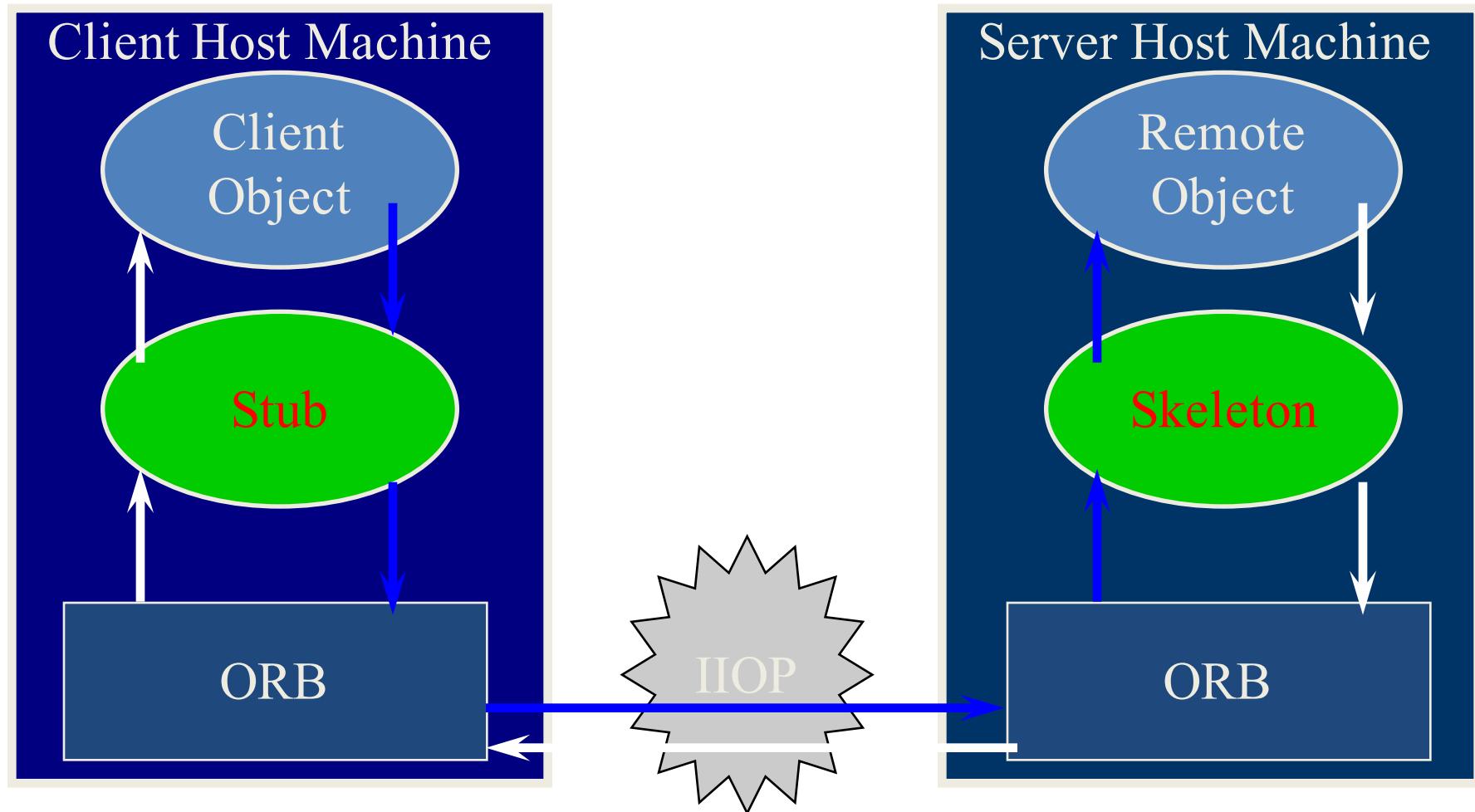
CORBA – Common Object Request Broker Architecture
Java impl. Is deprecated starting with Java 9+

OMG created CORBA. CORBA loves JAVA.

1. CORBA Basic Architecture
2. CORBA Basic Flow
3. ORB – Object Request Broker
4. GIOP vs. IIOP, IOR
5. IDL is CORBA “language independency”
6. CORBA Services
7. Products – SUN Java IDL

Lecture 7

Part I – CORBA – Basic Architecture

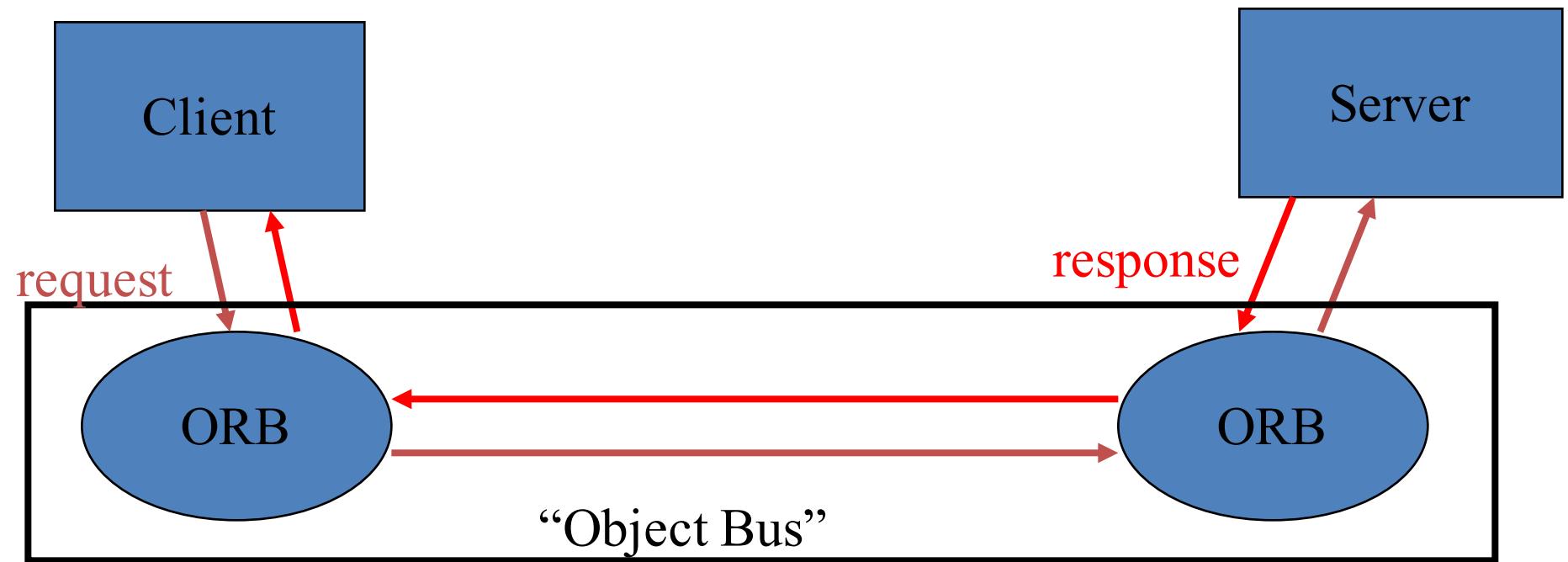


Stubs and Skeletons

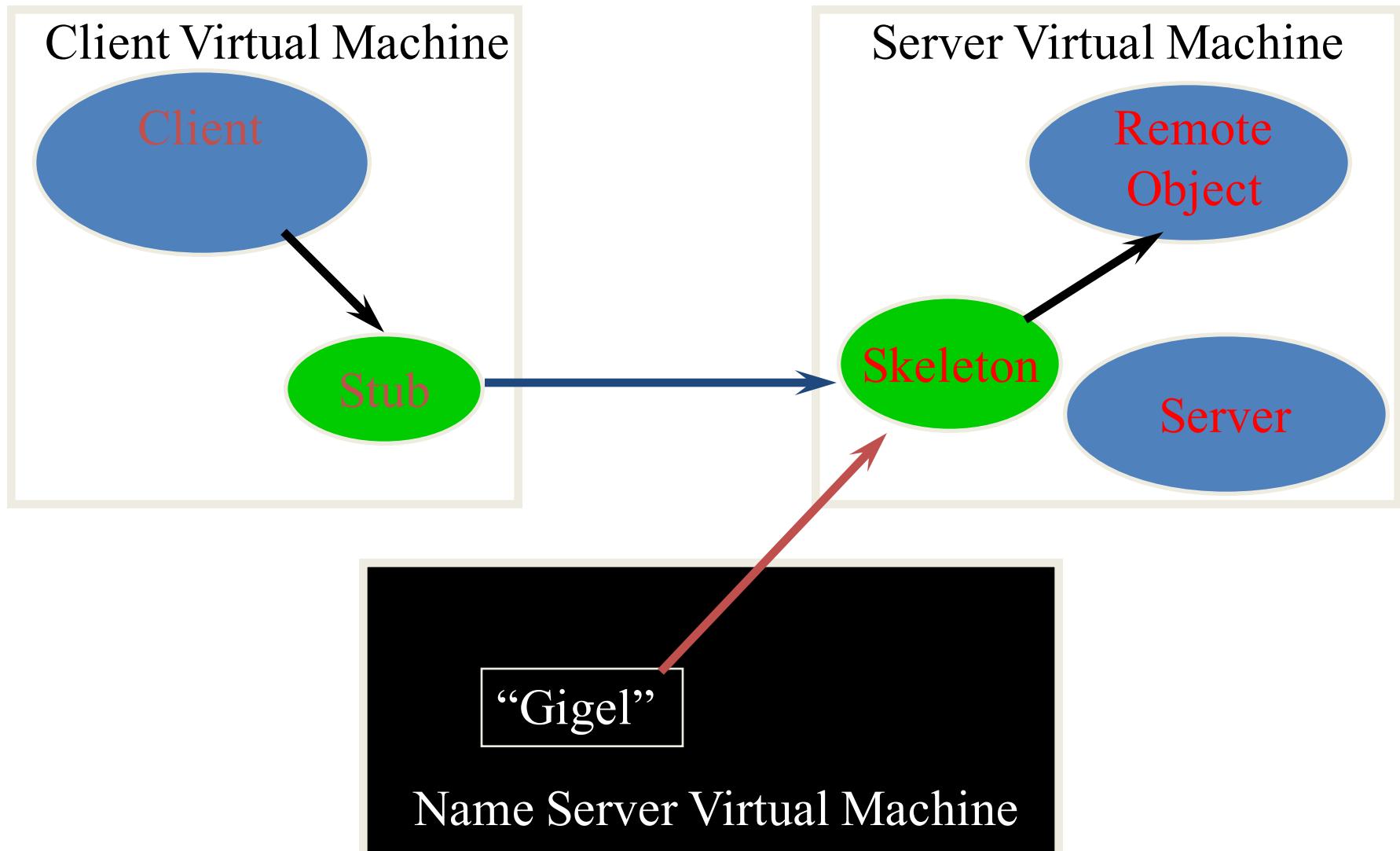
- Stub
 - lives on client
 - pretends to be remote object
- Skeleton
 - lives on server
 - receives requests from stub
 - talks to true remote object
 - delivers response to stub

Similar with RMI 1.1 in concept

Partea I – CORBA – Basic Architecture

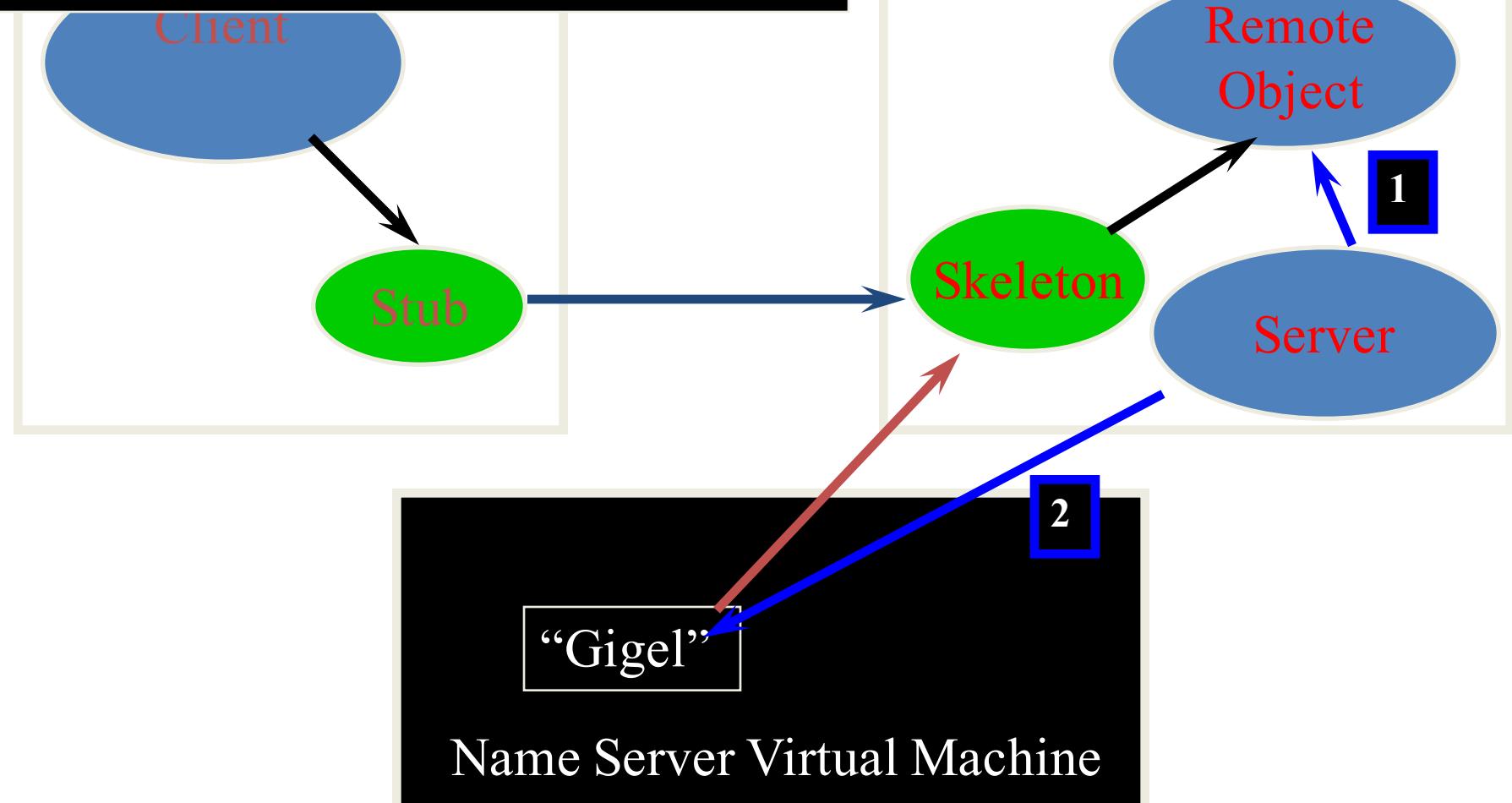


Partea I – CORBA Basic Flow without ORB is like RMI

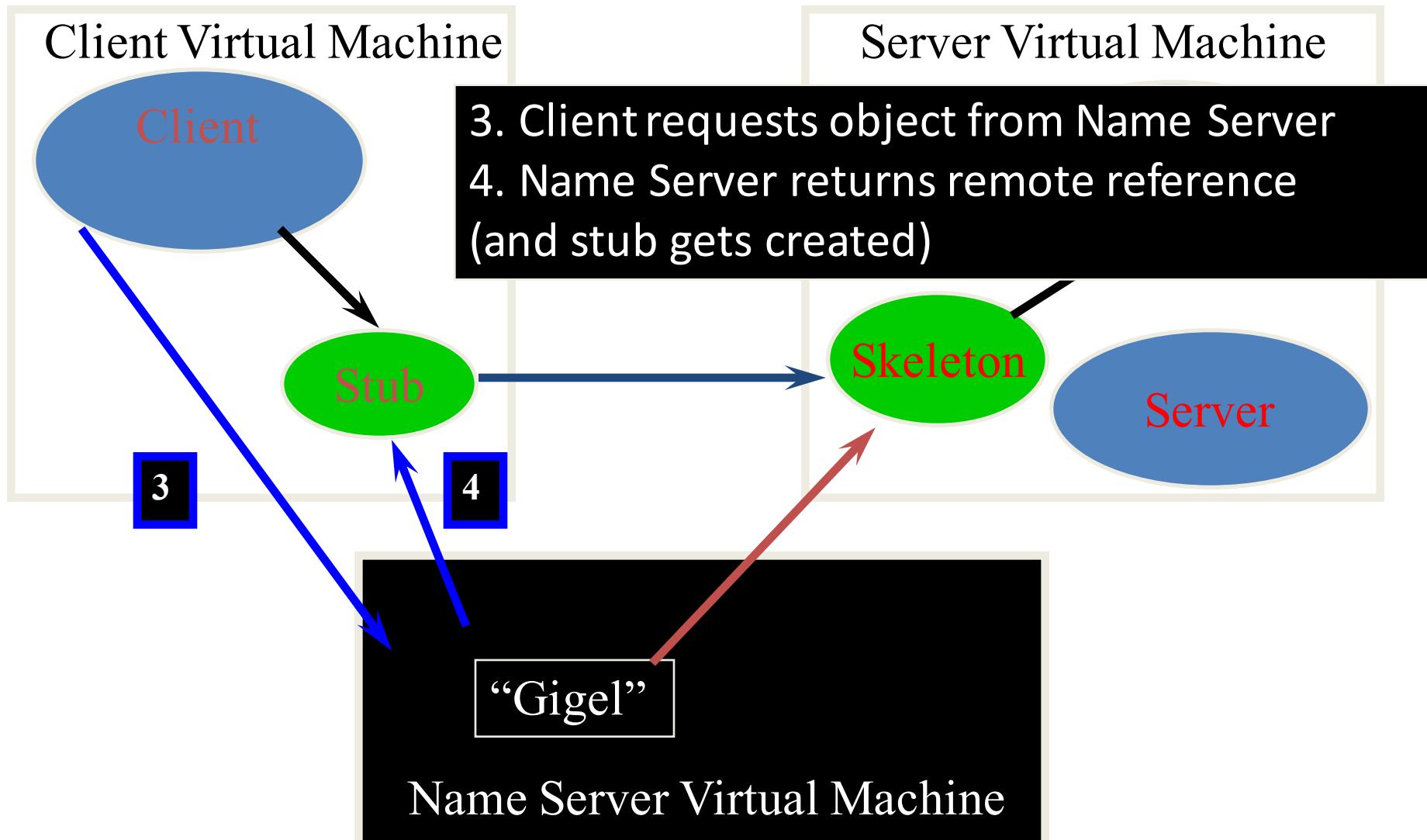


Partea I – CORBA Basic Flow without ORB is like RMI

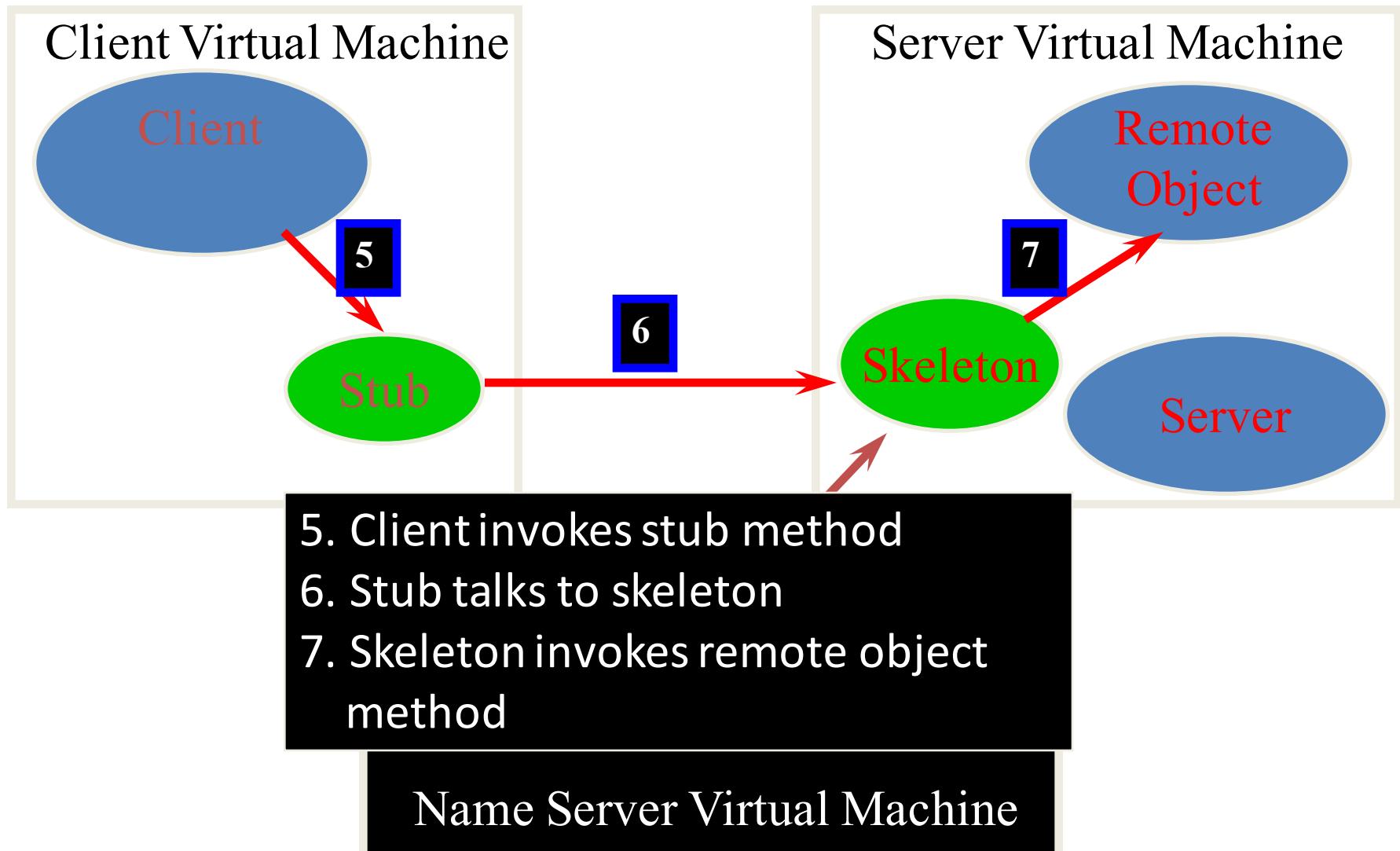
1. Server Creates Remote Object
2. Server Registers Remote Object



Partea I – CORBA Basic Flow without ORB is like RMI



Partea I – CORBA Basic Flow without ORB is like RMI



Partea I – CORBA – ORB Conceptual

- OMG does not specify exactly where the ORB is in CORBA.
- Depending on which CORBA implementation (product) is used, the ORB may be:
 - A set of run-time libraries
 - A server machine/process
 - Part of the operating system (Spring)
 - Part of a web browser (Netscape)

Partea I – CORBA – ORB in Practice

FEATURES:

- Object Request Broker - “Object Bus”
- Handles all communication among objects
- Each host (machine) has its own ORB
- ORBs know how to talk to each other
- ORB also provides basic services to client

RESPONSABILITIES:

- Find the object implementation for the request
- Prepare the object implementation to receive the request
- Communicate the data making up the request
- Retrieve results of request

Note:

- There's an ORB on the server too, and ORB receives request
- *ORB is good if Stub and Skeleton are written in different programming language*

Partea I – CORBA – ORB Features

- Method invocations
 - Static and Dynamic
 - Remote objects or CORBA services
- High-level language bindings
 - Use your favorite language; ORB translates
- Self-describing
 - Provides metadata for all objects and services

Partea I – CORBA – ORB Features

- Local or remote
 - Same API wherever target object lives
- Preserves context
 - Distributed security and transactions
- Coexistence with legacy code
 - Just provide a wrapper object

What is an ORB really?

- Not a separate process
- Library code that executes in-process
- Listens to TCP ports for connections
 - One port per local object
- Opens TCP sockets to other objects
 - N ports per remote machine

Partea I – CORBA – GIOP and IIOP

- The OMG agreed protocol for ORB interoperability is called the General Inter-ORB Protocol (GIOP).
- GIOP defines the logical data representation and message formats for communication.
- The OMG defines a realization of GIOP that uses TCP/IP as the transport layer. This specialization is called the Internet Inter-ORB Protocol (IIOP).

Interoperability

Message type	Originator	Description
Request	Client	Contains an invocation request
Reply	Server	Contains the response to an invocation
LocateRequest	Client	Contains a request for the exact location of an object
LocateReply	Server	Contains location information on an object
CancelRequest	Client	Indicates client no longer expects a reply
CloseConnection	Both	Indication that connection will be closed
MessageError	Both	Contains information on an error
Fragment	Both	Part (fragment) of a larger message

Message types in GIOP

- General Inter-ORB Protocol (GIOP) assumes a transport protocol that is reliable, connection oriented and support byte stream notion
- Internet Inter-ORB Protocol (IIOP) is GIOP built on TCP

GIOP Messages:

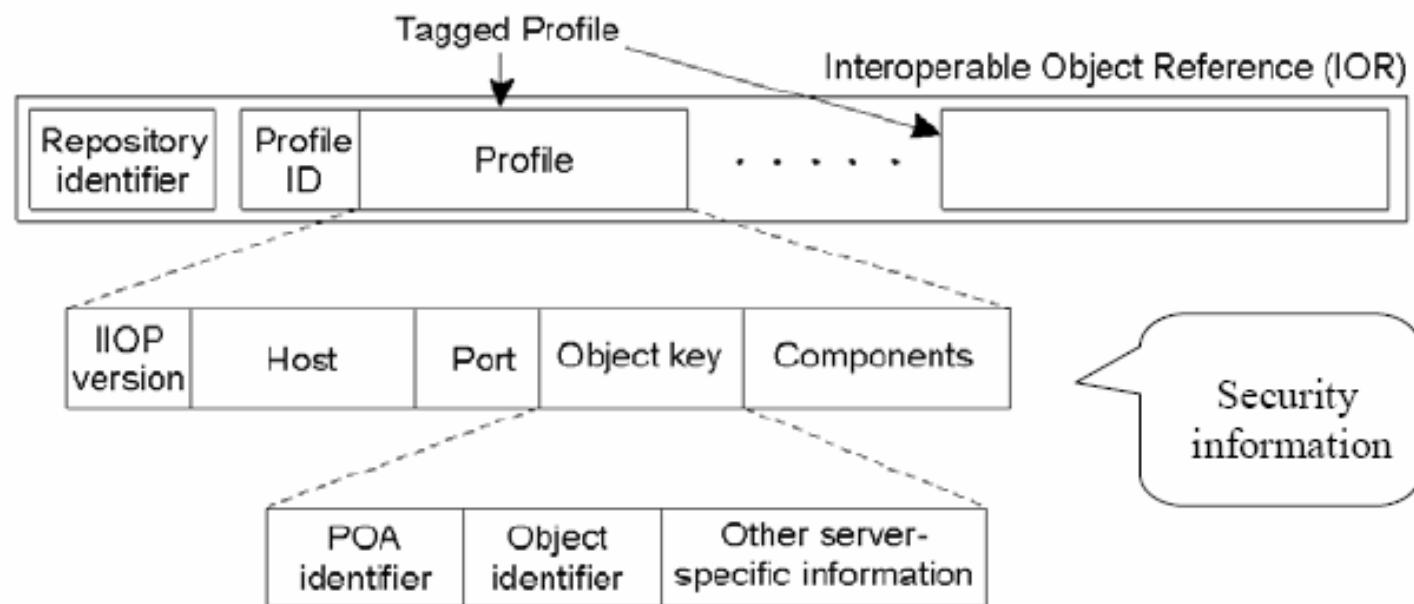
- *Request message* contains a complete marshaled invocation request (object reference, name of the method, input parameters)
 - Each request has request ID
- *Reply message* contains a marshaled return values and output parameters
 - Reply message has a corresponding request ID of the request message

Partea I – CORBA – ORB

- An IOR (Interoperable Object Reference) is managed internally by the interoperating ORBs.
- An IOR may include:
 - ORB's internal object reference
 - Internet host address
 - Port number
- It is not necessary for an application programmer to know the structure of an IOR.

Partea I – CORBA – ORB

```
IOR:0000000000000001049444c3a466f7274756e653a312e300000  
0000010000000000000005e00010000000000186d6179666c792  
e73642e6d6f6e6173682e6564752e617500070a000000000036  
3a5c6d6179666c792e73642e6d6f6e6173682e6564752e61753  
a666f7274756e653a466f7274756e65313a3a49523a466f7274  
756e65
```



Pseudo-objects

- The ORB is a pseudo-object
- It works just like a remote object, only it's local

The Basic Object Adapter (BOA)

- Another pseudo-object
- Helps register objects with the ORB
- Functions
 - Maintain Implementation Repository
 - Generate and interpret object references
 - Activate and deactivate implementation objects
 - Invoke methods via skeletons

Why do you need both an ORB and a BOA?

- Allows vendors to optimize or enhance functionality
 - register many objects *en masse*
 - cache object state elsewhere
- E.g. Object database

Using the BOA

- Slightly different procedure for initializing objects
- Hides name service from you
 - Ask the BOA to register the object
 - Ask the Helper object to bind the object
- Once the object is created, interface is identical
 - Just call methods using normal Java syntax

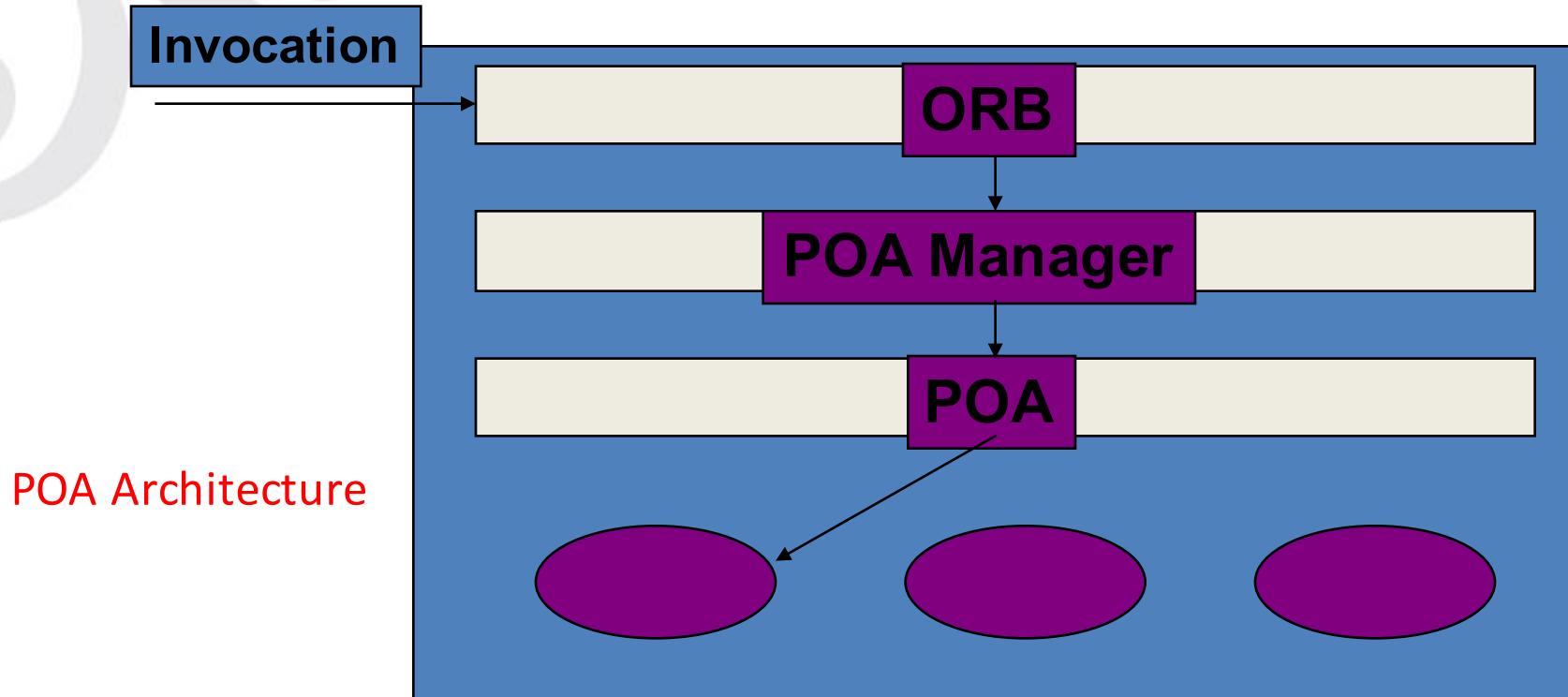
Object Adapters

Portable Object Adapter – POA

The POA:

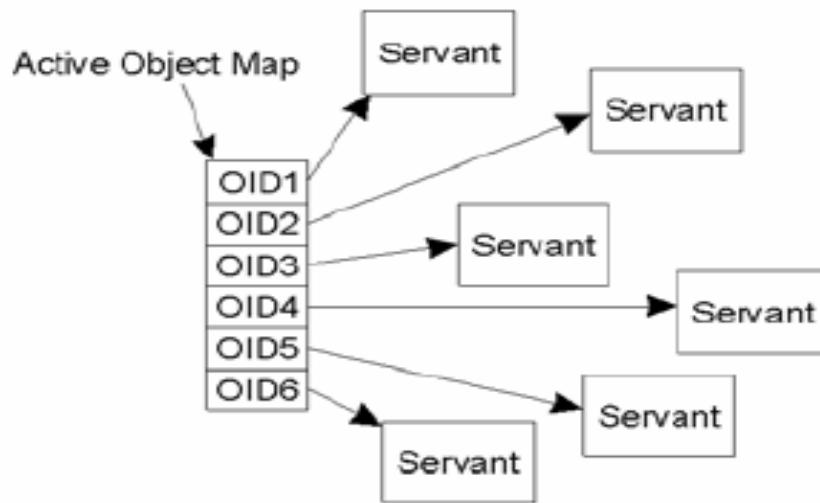
- Allows the ORB and Objects to communicate
- Provides many services:
 - Dispatches client calls to server objects
 - Handles incoming client calls
 - Handles registration of servers
 - Instantiates objects at runtime and creates and manages object references
- *POA is a BOA – Basic Object Adapter*

Partea I – CORBA – ORB

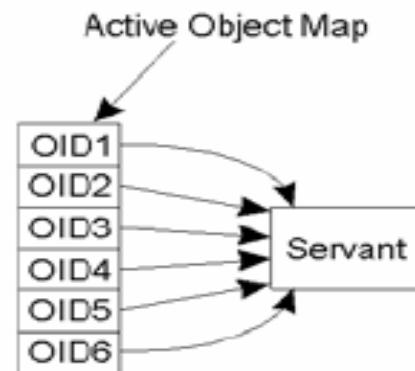


- A typical example of the relationship between POA and servants

Different Approaches for POA



(a)



(b)

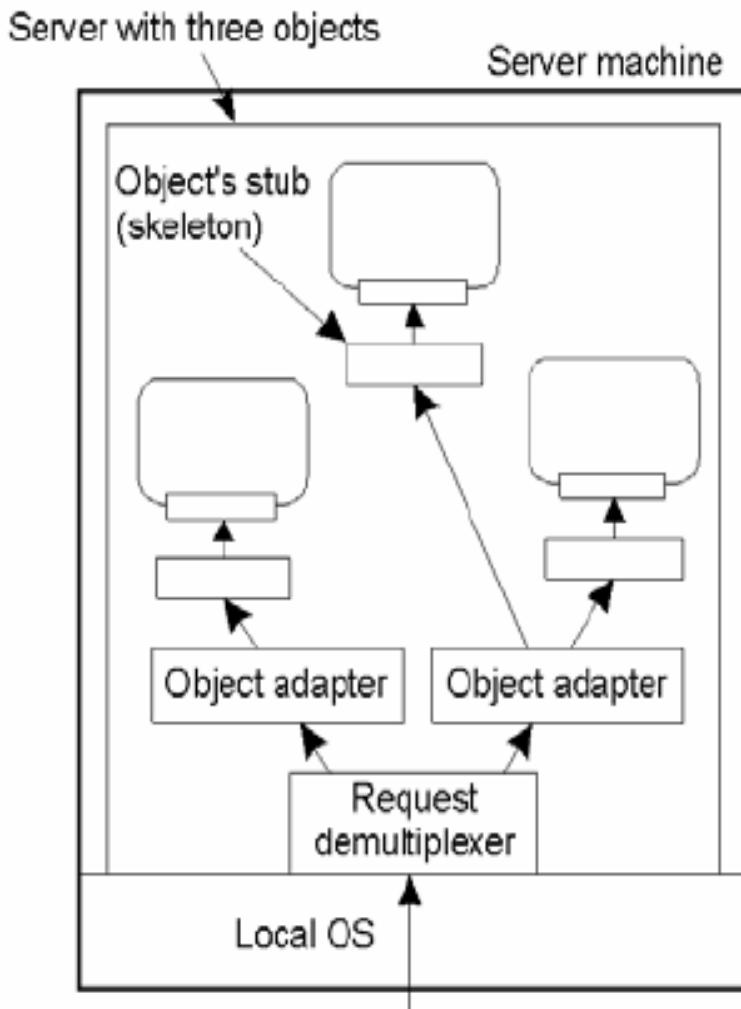
Mapping of CORBA object identifiers to servants of the same class

- a) The POA supports multiple servants.
- b) The POA supports a single servant.

Object adaptor for implementing a specific activation policy

Partea I – C

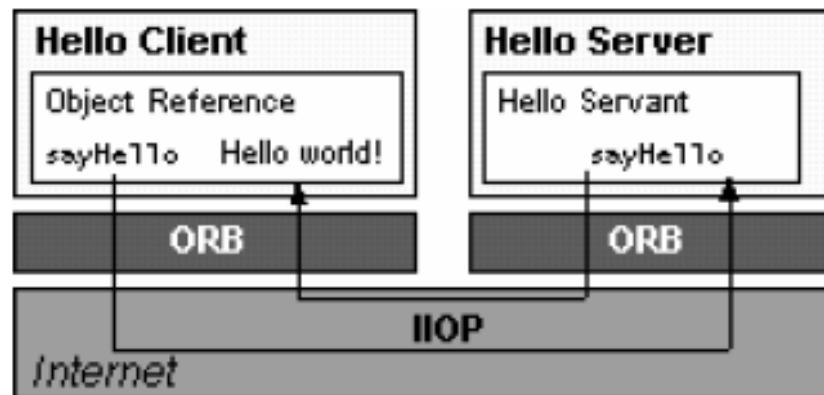
- An object adaptor may be responsible for one or more objects
- Object adaptor is not aware of interfaces implemented on the object it controls
- Object skeleton provides `invoke()` function for the adaptor to call and pass the message the adaptor receives
- Skeleton does marshall and unmarshall



Partea I – CORBA use IDL for independency

- Interface Definition Language
- Defines protocol to access objects
- Like a contract
- Well-specified
- Language-independent

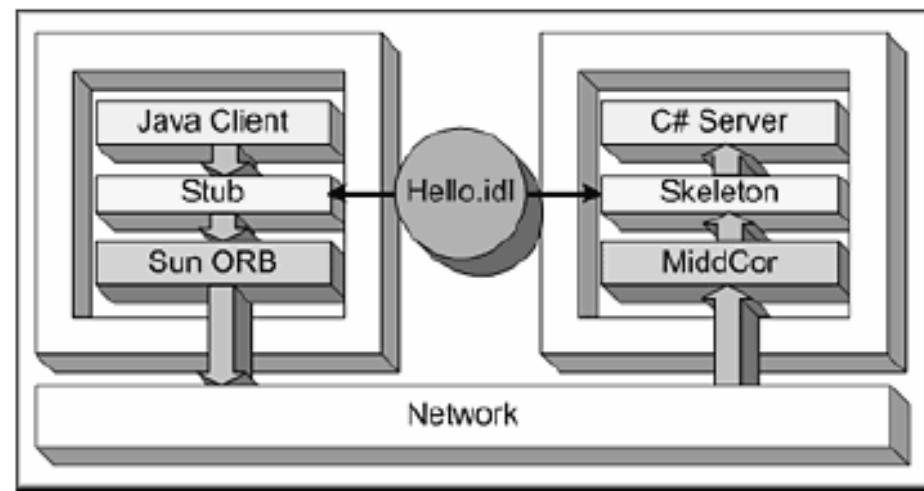
Partea I – CORBA use IDL for independency



From: <http://java.sun.com/docs/books/tutorial/idl/hello/index.html>

Hello.idl

```
interface Hello {  
    string sayHello();  
};
```



From: http://www.molecular.com/news/recent_articles/051904_javaworld.aspx

Partea I – CORBA use IDL for independency

```
//IDL Sample:  
module Calc {  
    interface Adder {  
        long add(in long x, in long y);  
    }  
}
```

- Defines an class called Adder which generates objects with a method called “add”

Partea I – CORBA use IDL for independency

IDL vs. Java vs. C++ concepts

IDL

module

interface

operation

attribute

Java

package

interface

method

pair of methods

C++

namespace

abstract

class

member

function

pair of

functions

Partea I – CORBA use IDL for independency

IDL Modules

- **Map to Java packages**
- **Unfortunately, it has the root level name of the module**
- **Clutters up your package hierarchy**

IDL Interfaces

- **Map to Java interfaces**

IDL Operations

- **Map to Java methods**

IDL Attributes

- Map to pair of functions – like C# do
- IDL
 - `string name;`
- Java
 - `public void name(String val);`
 - `public String name();`

CORBA Services

- APIs for low-level, common tasks
- Life Cycle Service
 - creating, copying, moving, removing objects
- Naming Service
 - Register objects with a name
 - Look up objects by name

Partea I – CORBA Services

- Concurrency Control Service
 - Obtain and release exclusive locks
- Transaction Service
 - Two-phase commit coordination
 - Supports nested transactions
- Persistence Service
 - Storing objects in a variety of databases
 - RDBMS, OODBMS, file systems
- Security Service
 - Authentication, ACLs, encryption, etc.
- Event Service
 - Uncoupled notifications

Partea I – CORBA Services

- Relationship
- Externalization
- Query
- Licensing
- Properties
- Time
- Trader
- Collection
- ... and so on...
- See what means about CORBA will be never being implemented?

Remember!

- CORBA is a standard by OMG, not an implementation.
- There are many implementations in Java and C/C++:
 - SUN JDK – Java 2 ORB
 - VisiBroker for Java or for C++
 - Orbacus
 - Orbix
 - Visigenic(freely available),
- Depending on the particular CORBA implementation, nonstandardized aspects may be different.

Partea I – SUN Java IDL

- Should be named “Java CORBA”
 - More than just IDL
 - Full (?) implementation of CORBA in 100% Java
- **SUN Java IDL has 3 Parts – NOW, DEPRECATED:**
 - ORB
 - Naming Service – COS – CORBA Object Service:
tnameserv.exe (Non-persistent) & orbd.exe
(Persistent)
 - idltojava & javatoidl compiler – now: idlj.exe
- Ships starting with JDK 1.2

The Java ORB

- 100% Java
- Generic
- Allows Java IDL applications to run either as stand-alone Java applications, or as applets within Java-enabled browsers
- Uses IIOP

The compiler: Transparent API

- JavaIDL turns IDL into direct method calls
- Easy to program
- Clients have no knowledge of implementation
- Highly portable

The compiler idlj: IDL to Java Mapping

- Defined by OMG and implemented here by SUN
- Translates IDL concepts into Java language constructs
- Everything is accessible by writing normal-looking Java code

The compiler idlj: IDL to Java Type Mapping

<u>IDL Type</u>	<u>Java Type</u>
boolean	boolean
char / wchar	char
octet	byte
short / unsigned short	short
long / unsigned long	int
long long / unsigned long long	long
float	float
double	double
string / wstring	String



The compiler: idlj or idltojava

- Development tool provided by SUN
- Automatically generates Java stubs, skeletons, helpers, holders, ... from IDL
- Generates stubs for specific remote interfaces

Stubs – Client Side

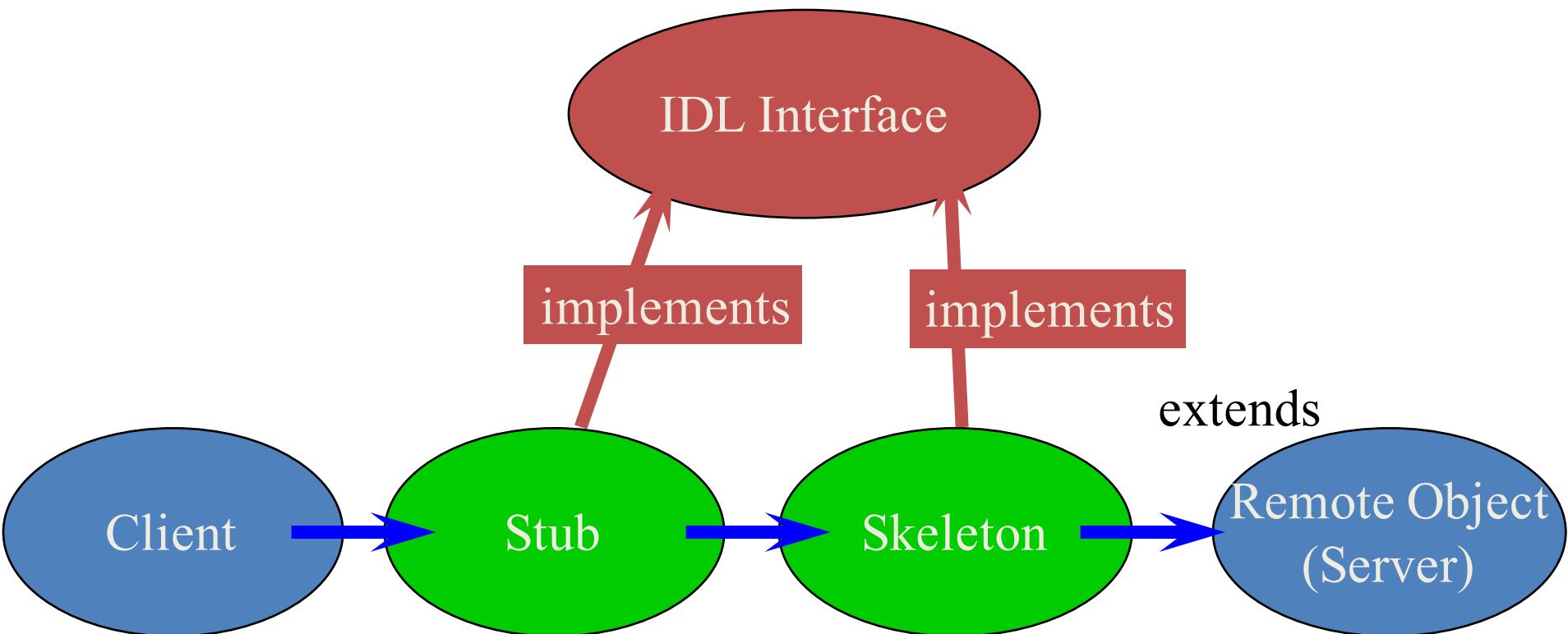
- Java objects call stub methods
- Stubs communicate with CORBA objects
 - and vice versa
- Transparent integration

Skeletons – Server Side

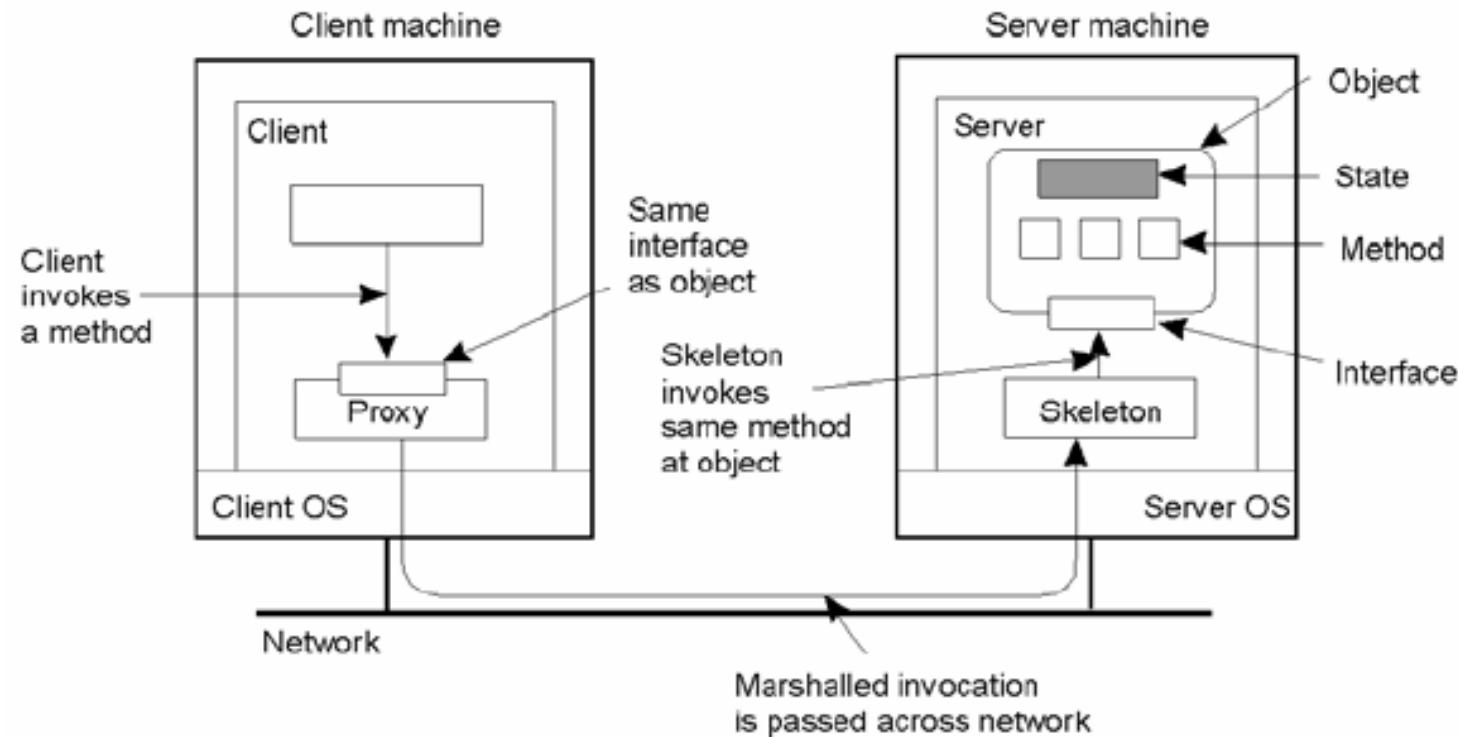
- ORB passes request to skeleton (like a stub)
- Skeleton calls local implementation

CORBA Server & Client

Remote Interfaces and Stubs



Partea I – CORBA vs RPC/RMI

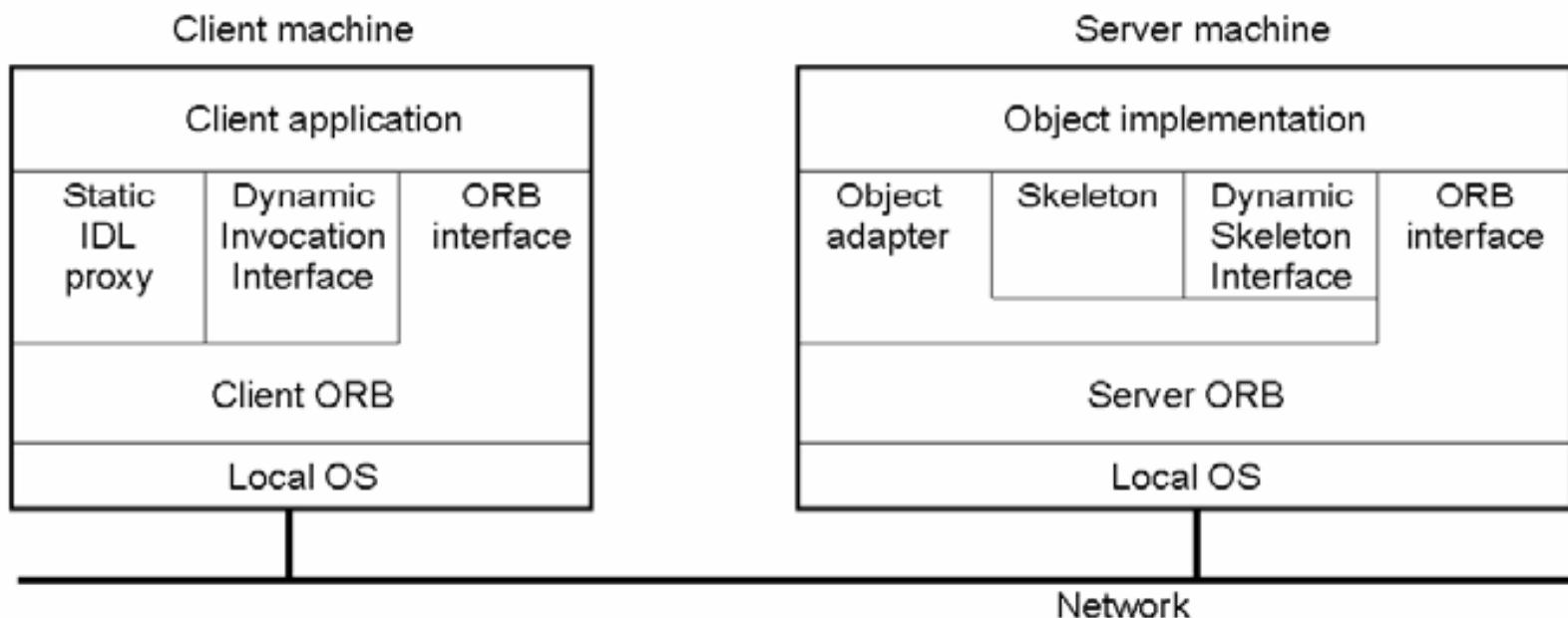


- Proxy is equivalent to client stub in RPC/RMI; it provides the same object interface as the server object
- Proxy marshalls method invocations into messages and unmarshall the reply messages
- Skeleton is like a server stub in RPC/RMI

Partea I – CORBA Advanced Architecture

Object Model

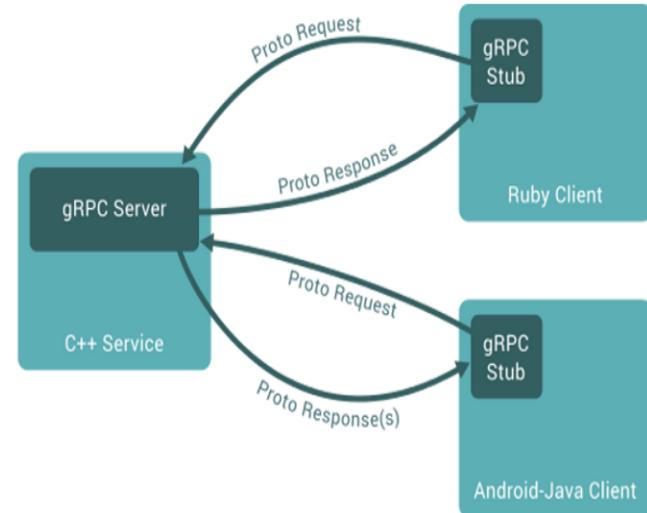
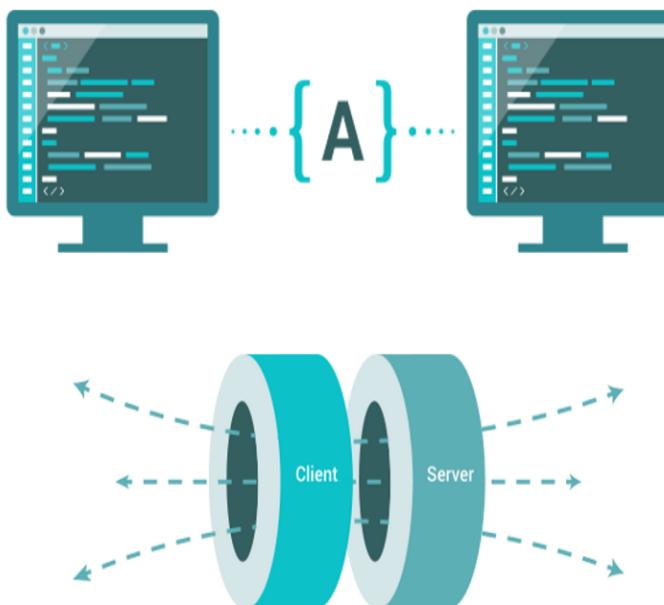
- ORB provides few services through ORB interface
- Operations to marshall and unmarshall object references
- Getting object reference to an object implementing a specific CORBA service



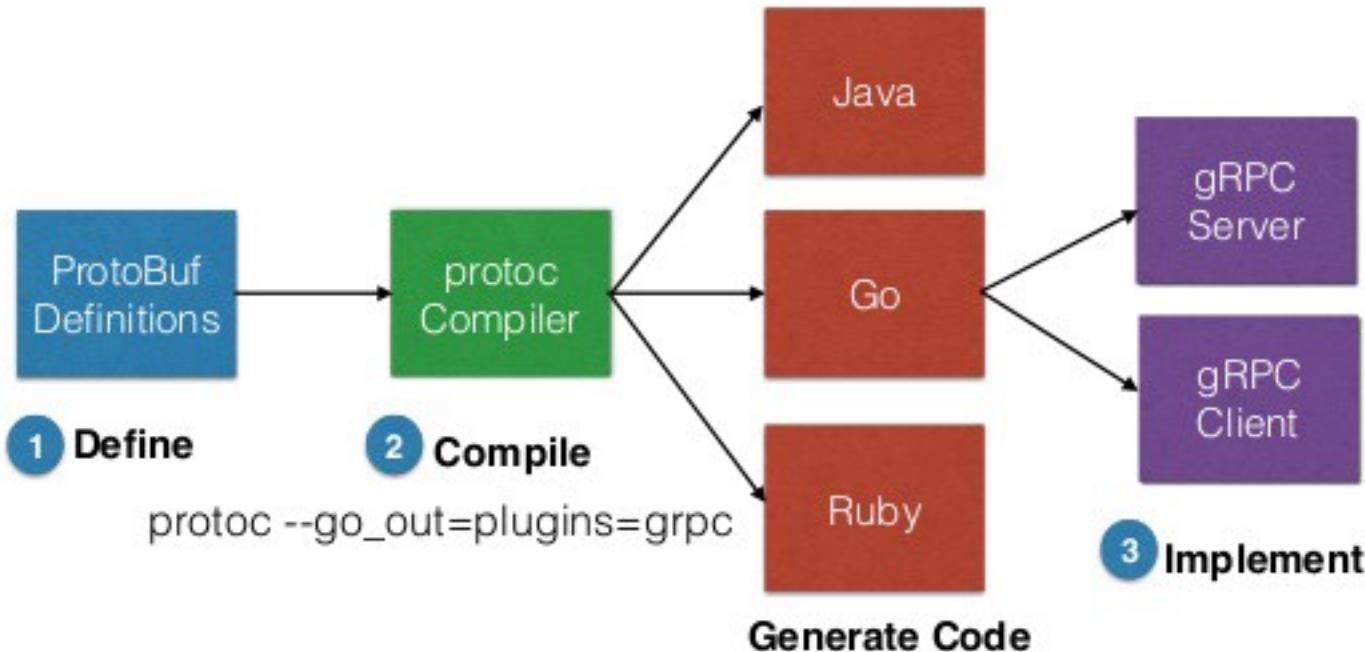
Part I – Will be CORBA replaced by gRPC?



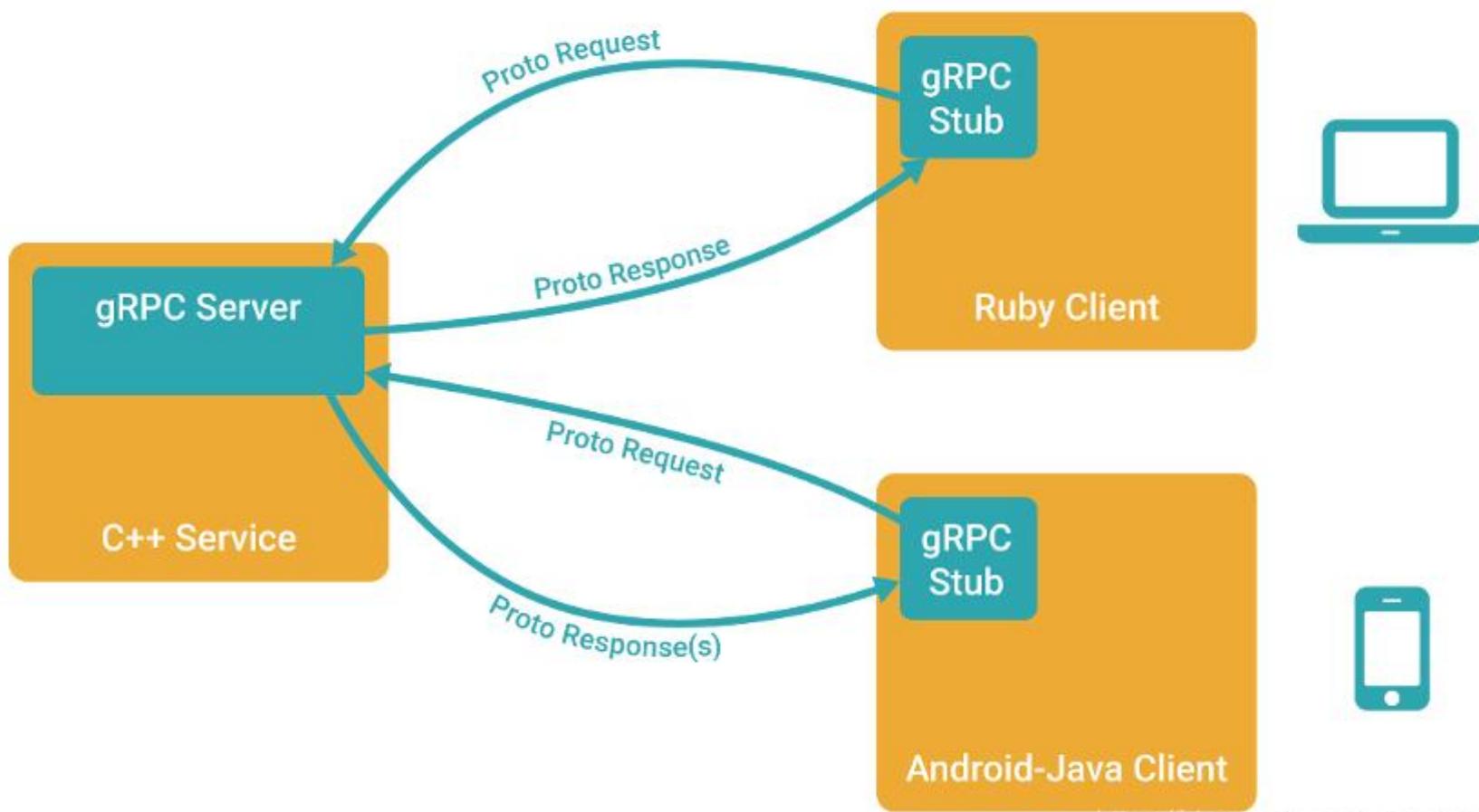
A high performance, open-source universal RPC framework



gRPC Workflow

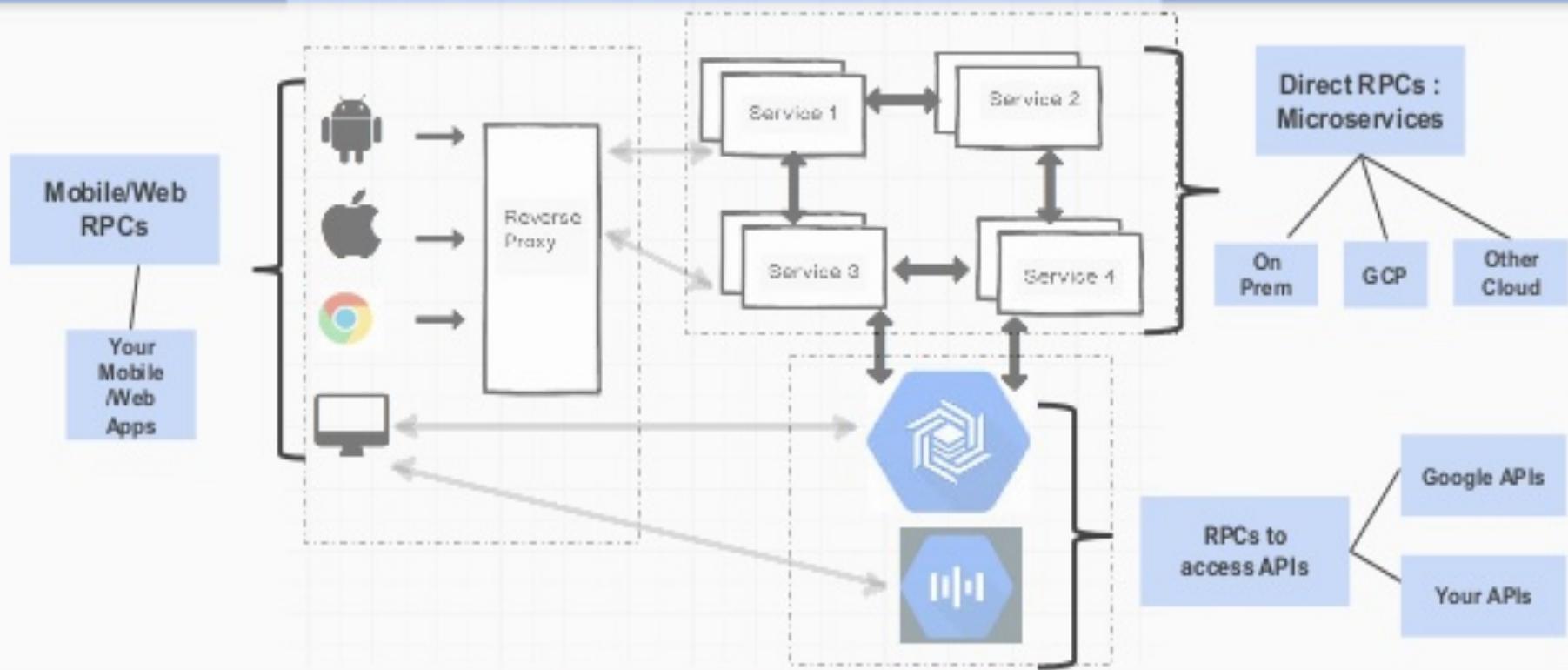


Part I – Will be CORBA replaced by gRPC?



Part I – Will be CORBA replaced by gRPC?

How is gRPC Used?



Section Conclusion

Fact: **DAD core is based on RPC concept**

In few **samples** it is simple to remember: Java RMI Architecture with JRMP protocol analysis in real time plus the core actions for distributed computing and systems:

Picture processing within a RMI Cluster.





DAD Section 4 – Core Middleware Technologies for Distributed Computing / Distributed App Development

Java WS – Web Services



2. Java WS Overview

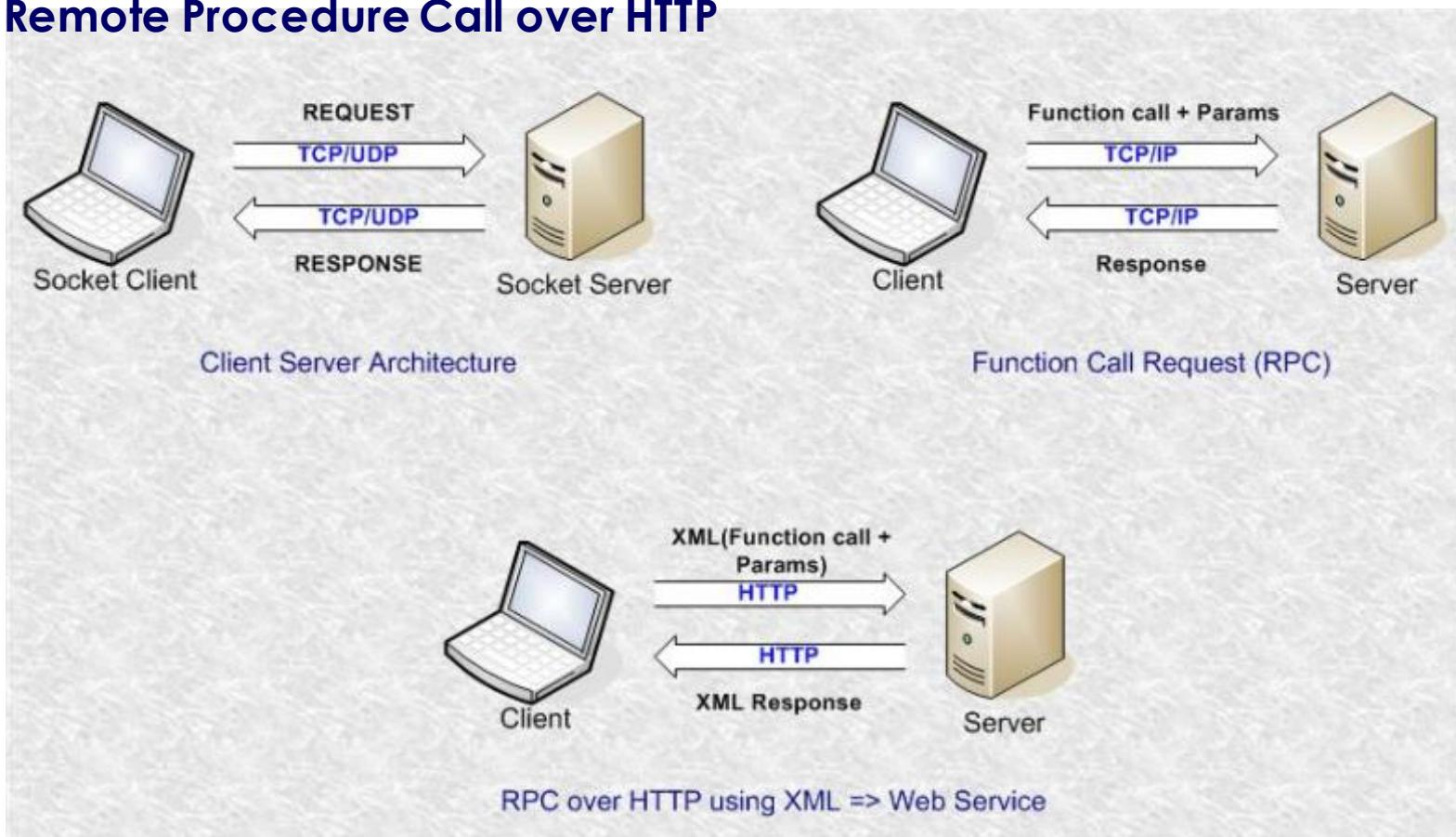
Java WS – Web Services

1. Web Service Overview
2. XML-RPC
3. Web Service WSDL
4. Web Service SOAP Req & Resp

Curs 9

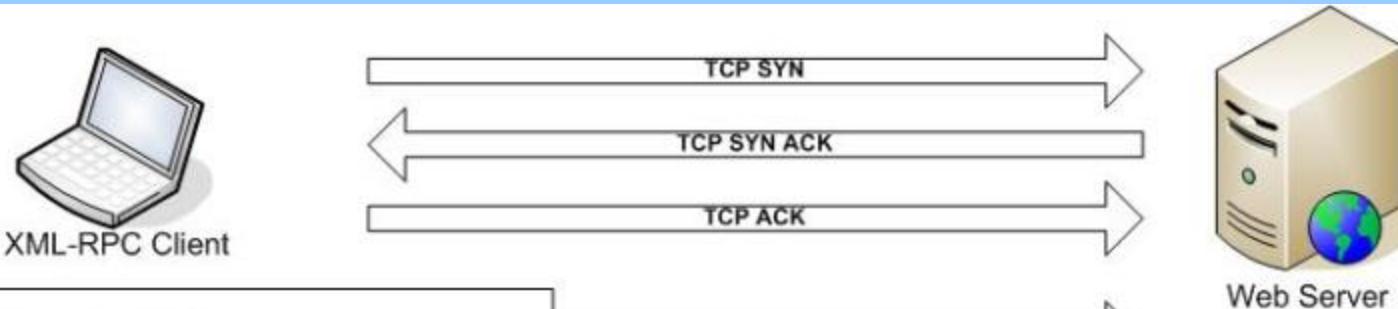
Partea II – Web Services Overview

- "a software system designed to support interoperable Machine to Machine interaction over a network." (W3C)
- Remote Procedure Call over HTTP



Lecture 7

Part II – Web Services – XML-RPC



```
POST /xmlrpc1/xmlrpc HTTP/1.1
Content-Type: text/xml
User-Agent: Apache XML RPC 3.0 (Sun HTTP Transport)
Content-Length: 199
Cache-Control: no-cache
Pragma: no-cache
Host: 192.168.1.100:8080
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
  <methodName>Calculator.add</methodName>
  <params>
    <param><value><i4>2</i4></value></param>
    <param><value><i4>3</i4></value></param>
  </params>
</methodCall>
```

HTTP Command over TCP PUSH,ACK →

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml
Content-Length: 189
Date: Thu, 19 Jul 2007 09:15:36 GMT

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse xmlns:ex="http://ws.apache.org/
xmlrpc/namespaces/extensions">
  <params>
    <param><value><i4>5</i4></value></param>
  </params>
</methodResponse>
```

← HTTP Response over TCP PUSH,ACK

Part II – Web Services – XML-RPC

- remote procedure calls using HTTP for transport and XML as encoding
- XML-RPC message is an HTTP-POST request
- very simple XML - doesn't use XML namespaces or attributes
- works only with HTTP

```
POST /XMLRPCServer/xmlrpc HTTP/1.1
Content-Type: text/xml
User-Agent: Apache XML RPC 3.0 (Jakarta Commons httpclient Transport)
Host: 192.168.1.103:8080
Content-Length: 199

<?xml version="1.0" encoding="UTF-8"?><methodCall><methodName>calculator.add</methodName><params><param><value><i4>2</i4></value></param><param><value><i4>3</i4></value></param></params></methodCall>HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.2.2.GA (build: SVNTAG=JBoss_4_2_2_GA
date=200710221139)/Tomcat-5.5
Content-Type: text/xml
Content-Length: 128
Date: Mon, 21 Jan 2008 21:46:57 GMT

<?xml version="1.0" encoding="UTF-8"?><methodResponse><params><param><value><i4>5</i4></value></param></params></methodResponse>
```

Part II – Web Services – Products & WSDL

Web Services Products:

- **Apache AXIS2 for Java – course**
- **SUN Java-WS – integrated in NetBeans 6.0.1 – seminar**

WSDL Concept:

- **WSDL – Web Service Definition Language**
- **WSDL = XML document that describes a web service**
- **WSDL – Specifies the location of the service and the operations (or methods) it exposes**

Part II – Web Services – WSDL

A WSDL describes a web service using the following elements:

- **<binding>** contains communication protocols used by the web service
- **<portType>** defines the operations that can be performed by a web service, and the messages that are involved
- **<message>** defines the in/out messages that are used by the operations ⇔ methods
- **<types>** defines data types used by the messages

A WSDL document can also contain extension elements and a **<service>** element that makes it possible to group together the definitions of several web services .



WSDL format

```
<definitions>
  <types>
    <!--definition of types-->
  </types>

  <message>
    <!--definition of a message-->
  </message>

  <portType>
    <!--definition of a port-->
  </portType>

  <binding>
    <!--definition of a binding-->
  </binding>

</definitions>
```

Part II – Web Services – SOAP

- **SOAP – Simple Object Access Protocol**
- **simple XML based protocol to let applications exchange information over HTTP**
- **defines a format for sending messages**
- **allows you to get around firewalls (firewalls and proxy servers normally block RPC traffic)**
- **platform and language independent**
- **transport Protocols: TCP, HTTP, SMTP, and MQ – Message Queues**

Part II – Web Services – SOAP

A SOAP message is an ordinary XML containing the following elements:

- **<Envelope>** which identifies that the XML document is a SOAP message
- **<Header>** which contains application specific information about the SOAP message (such as authentication, payment, etc).
- **<Body>** contains the actual SOAP message
- **<Fault>** contains eventual error messages



SOAP message format

```
<?xml version="1.0"?>

<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-
  envelope" soap:encodingStyle="http://
  www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    ...
  </soap:Header>

  <soap:Body>
    ...
    <soap:Fault>
      ...
    </soap:Fault>
  </soap:Body>

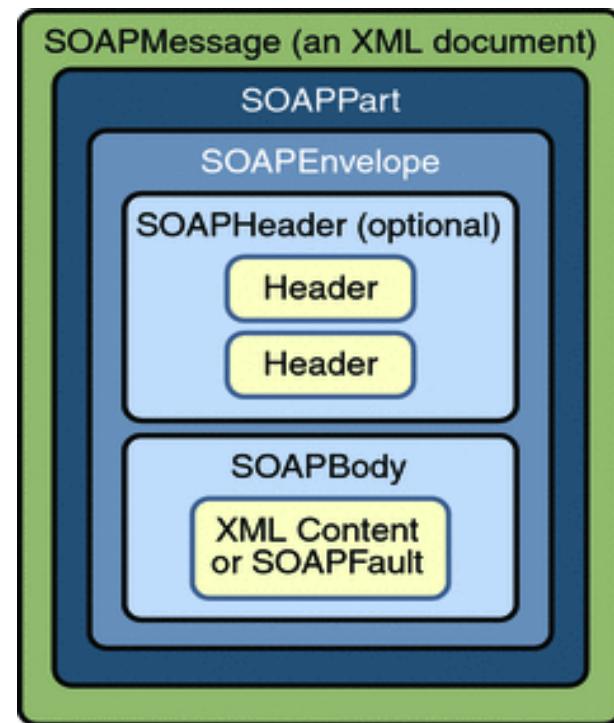
</soap:Envelope>
```

Part II – Web Services – SOAP with Attachments

- The **SOAP messaging protocol allows you to send MIME attachments via SOAP messages. WSDL provides a description of these attachments.**
- **SOAP with Attachments API for Java (SAAJ) allows you to do XML messaging from the Java platform**
- **The SAAJ API conforms to the Simple Object Access Protocol (SOAP) 1.1 and 1.2 specifications and the SOAP with Attachments specification.**

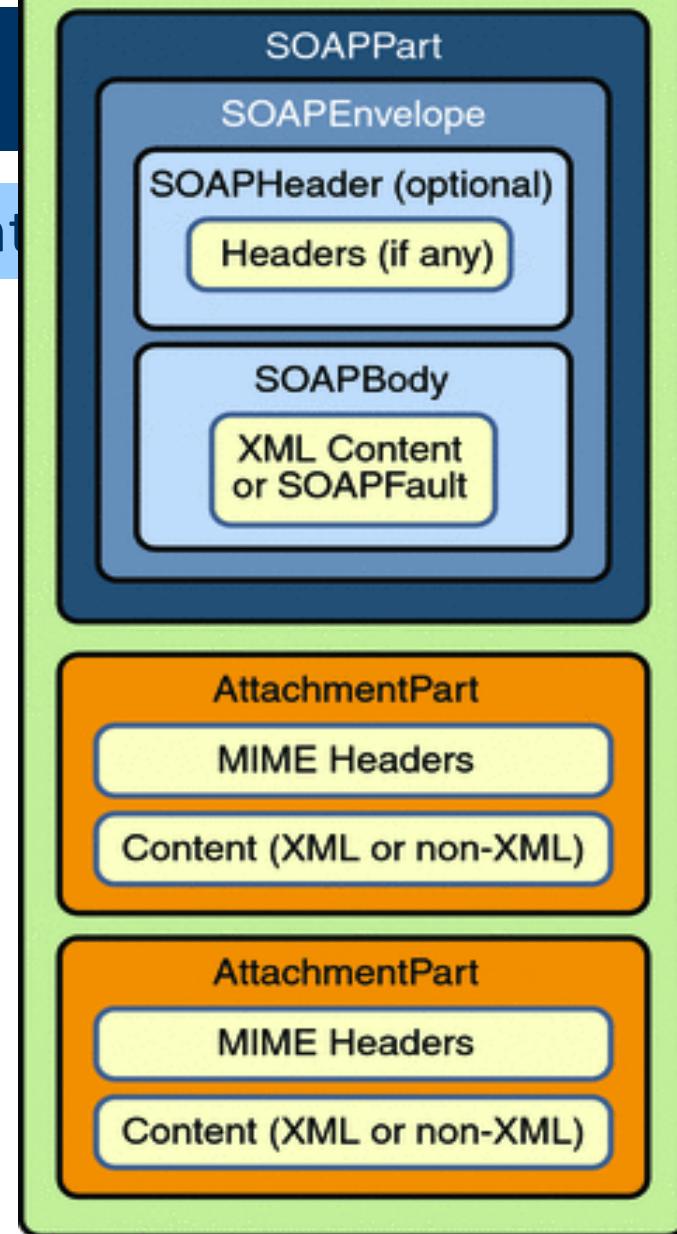
Part II – Web Services – SOAP with Attachments

- The SAAJ API provides the **SOAPMessage** class to represent a SOAP message.
- SAAJ API also provides the **SOAPPart** class that is the container for the SOAP-specific portion of a **SOAPMessage** object. All messages are required to have a **SOAPPart**.
- A **SOAPPart** object is a MIME part and has the MIME headers Content-Id, Content-Location, and Content-Type



SOAP with no attachments

SOAPMessage (an XML document)



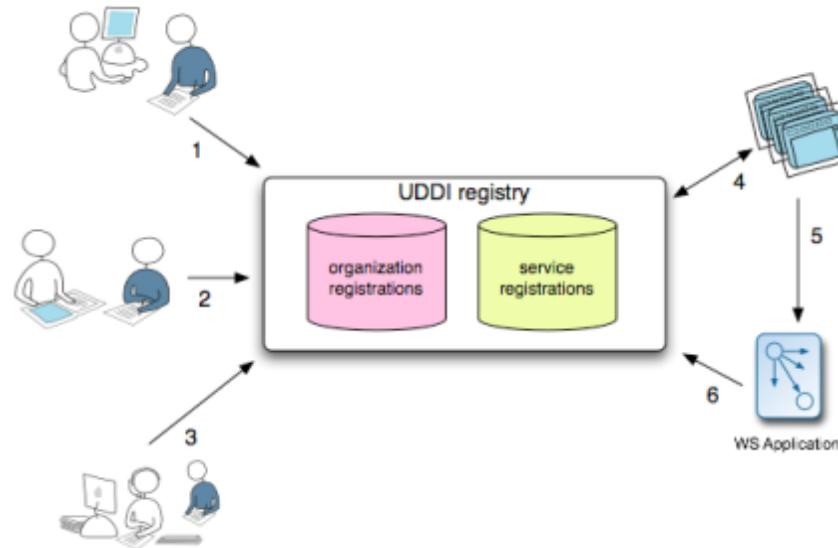
Part II – Web Services – SOAP with Attachment

- A SOAP message may include one or more attachment parts in addition to the SOAP part.
- The SAAJ API provides the **AttachmentPart** class to represent an attachment part of a SOAP message.
- An attachment part can contain any kind of content.

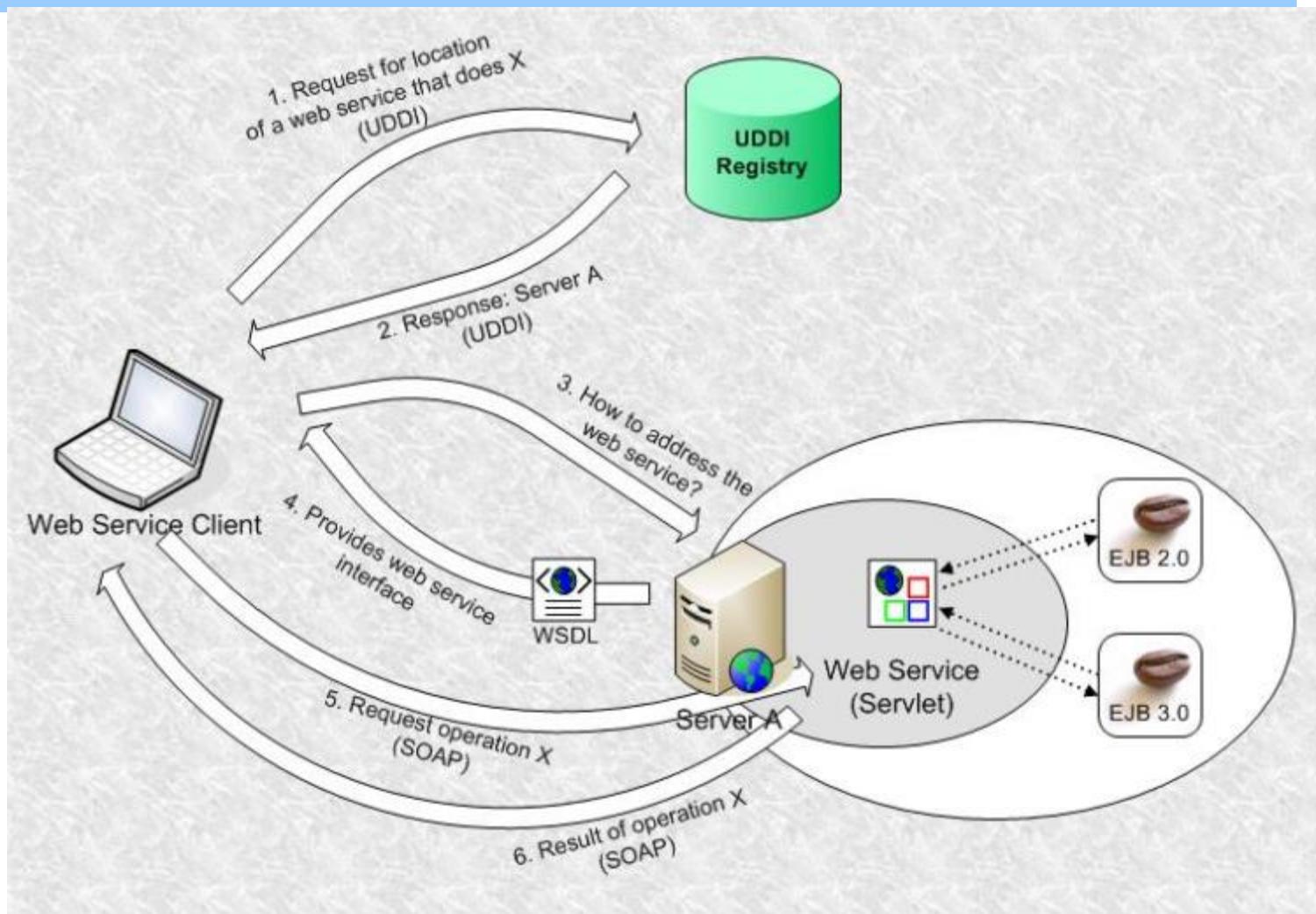
SOAP with two attachments

Part II – Web Services - UDDI

- an **XML-based registry for businesses worldwide to list themselves on the Internet.**
- used in a similar manner to JNDI – to register and locate web services
- is a directory for storing information about web services
- is a directory of web service interfaces described by WSDL



Part II – Web Services – A Java Architecture



Section Conclusions

Java WS – Web Services DEMO

REST-ful services and Micro-Services replace the SOA-Web-Services

Will be Internet 3.0 deployed for technologies which provide data & process migration?

Java WS DEMO
for easy sharing



Distributed Application Development – Data vs. Process Migration

Communicate & Exchange Ideas





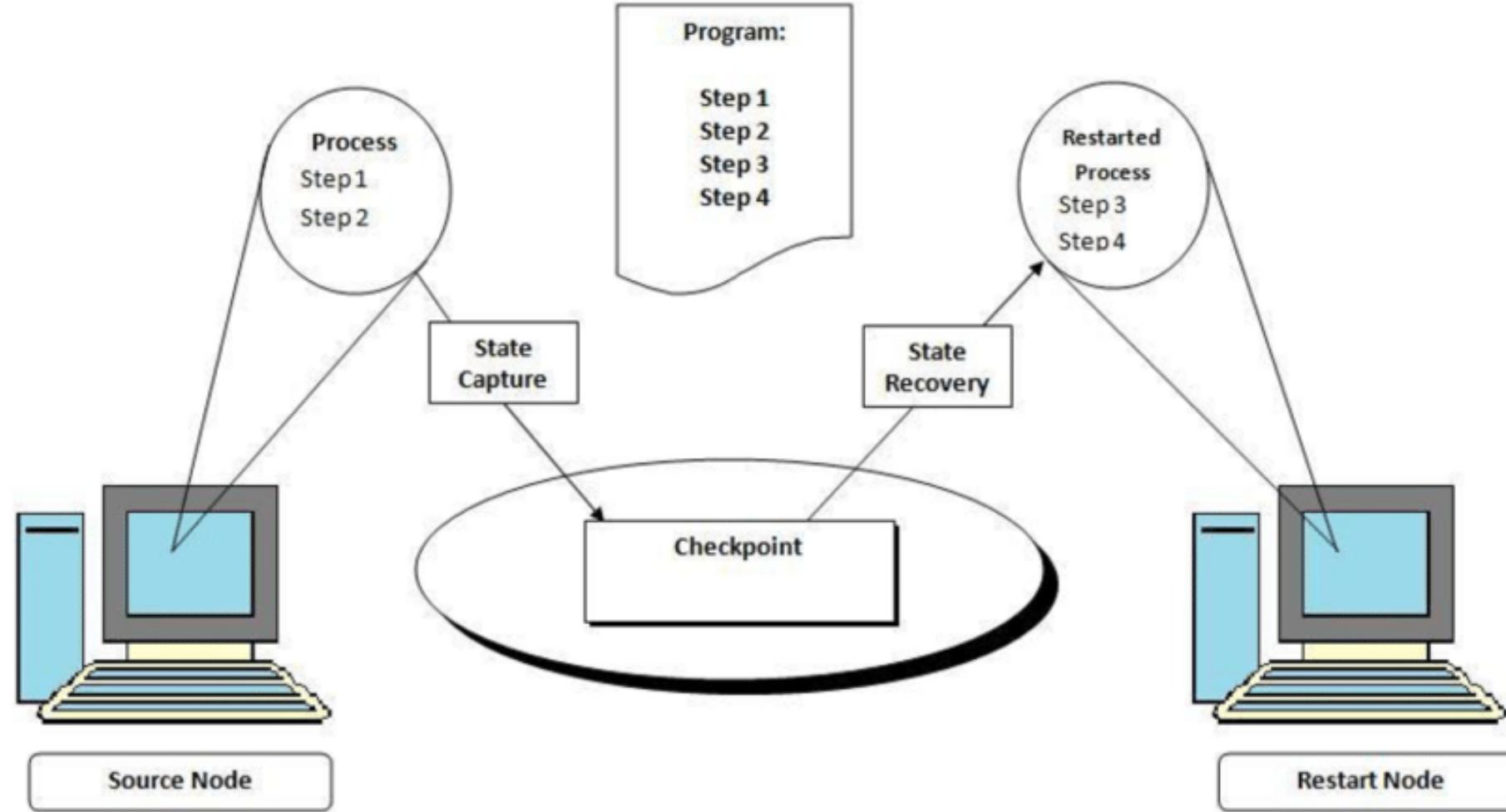
Questions & Answers!

But wait...

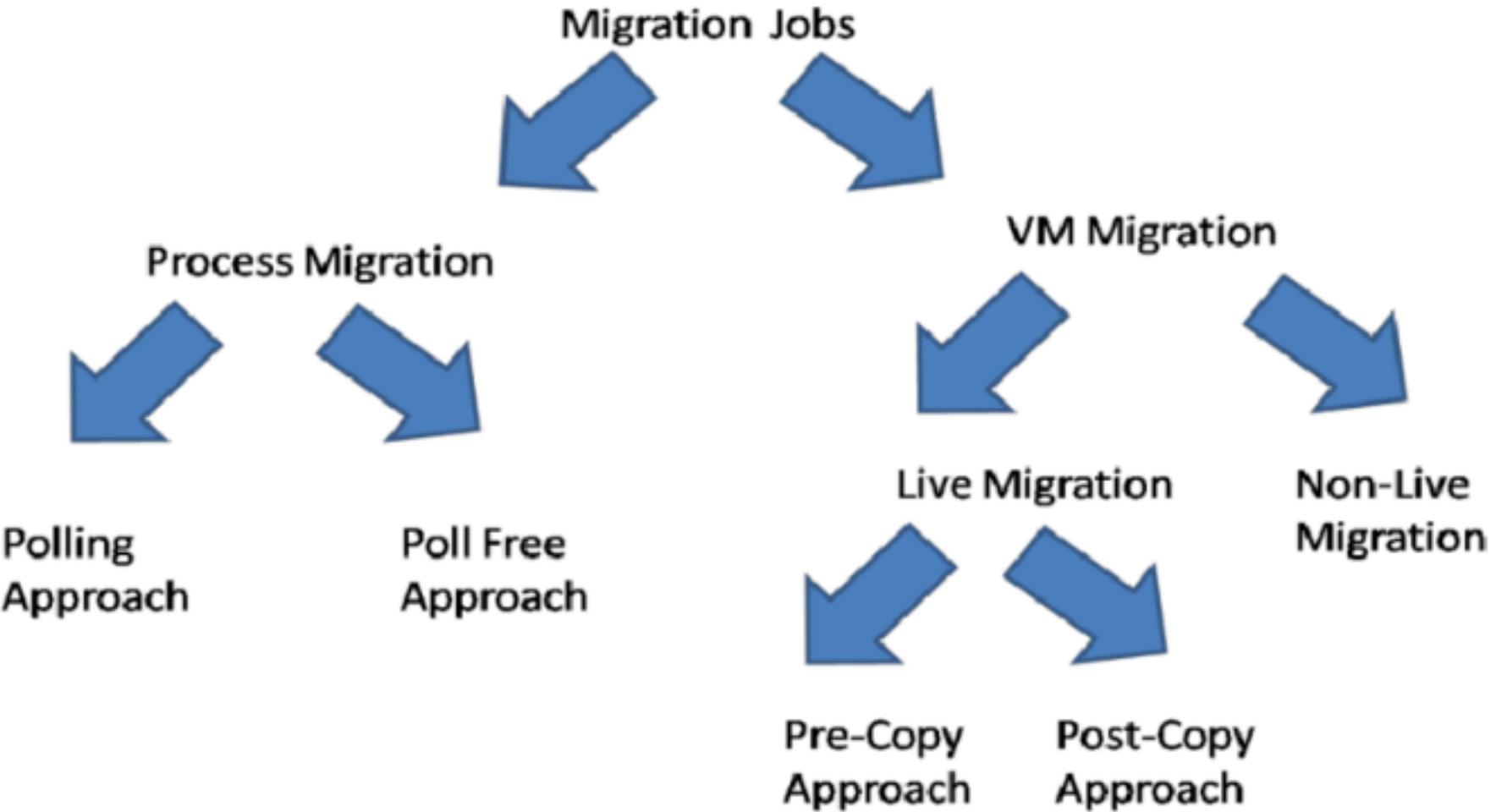
There's More! – Data vs. Process Migration



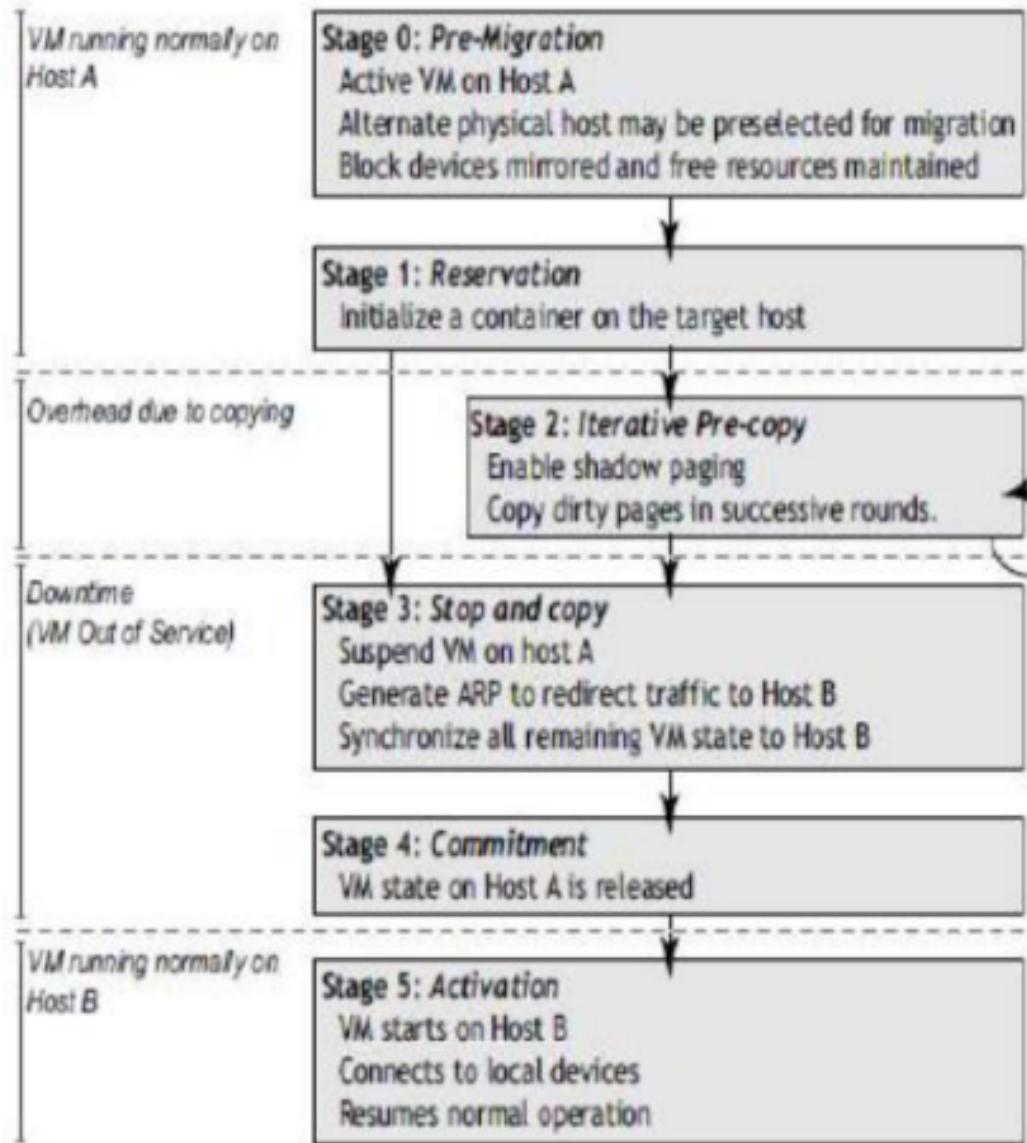
Part III – Process Migration



Part III – Process Migration - Migration Jobs in Cloud Computing



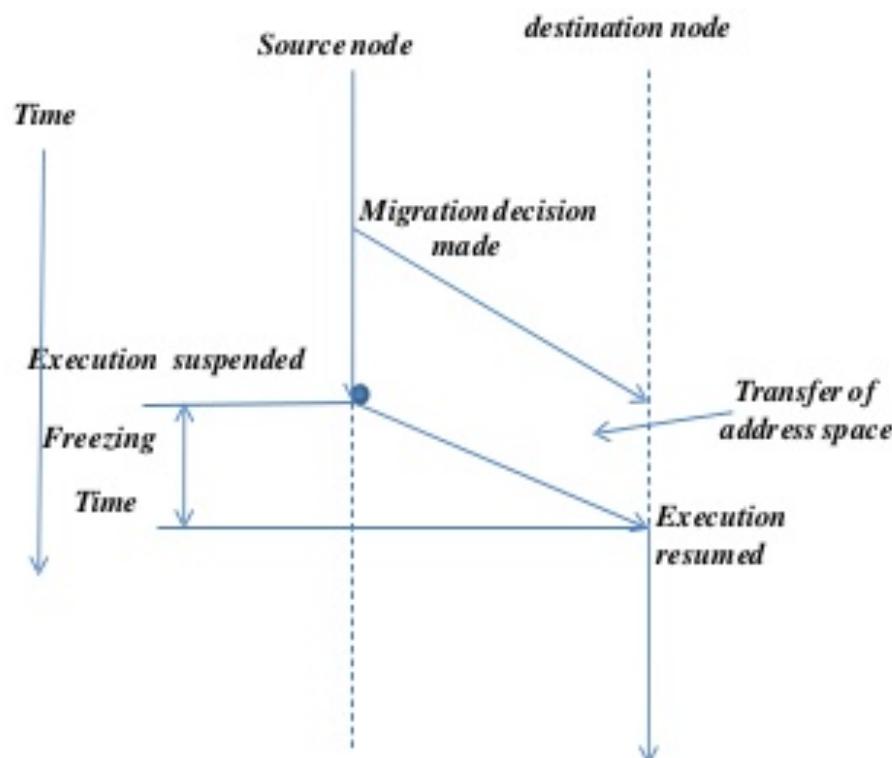
Part III – Process Migration - Pre-Copy Algorithm VMs in Cloud Computing



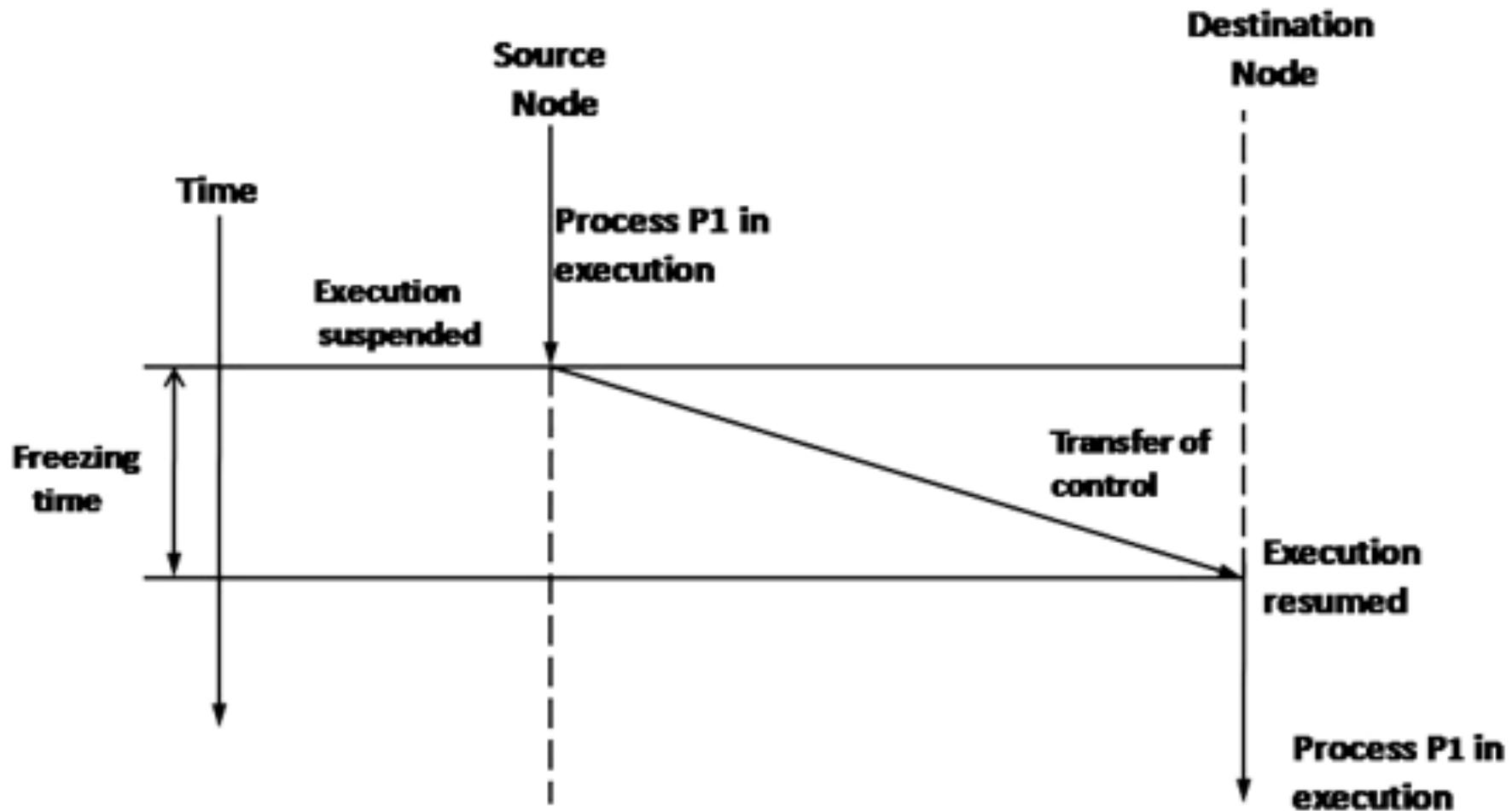
Process Migration Mechanisms

❑ Pretransferring (precopying)

- ❑ Address space is transferred while the process is still running on the source node
- ❑ Initial transfer of the complete address space followed by repeated transfers of the pages modified during previous transfer
- ❑ Remaining modified pages are retransferred after the process is frozen for transferring its state information
 - ❑ freezing time is reduced
 - ❑ Pretransfer operation is executed at a higher priority than all other programs on the source node
 - ❑ Total time of migration is increased due to the possibility of redundant page transfers

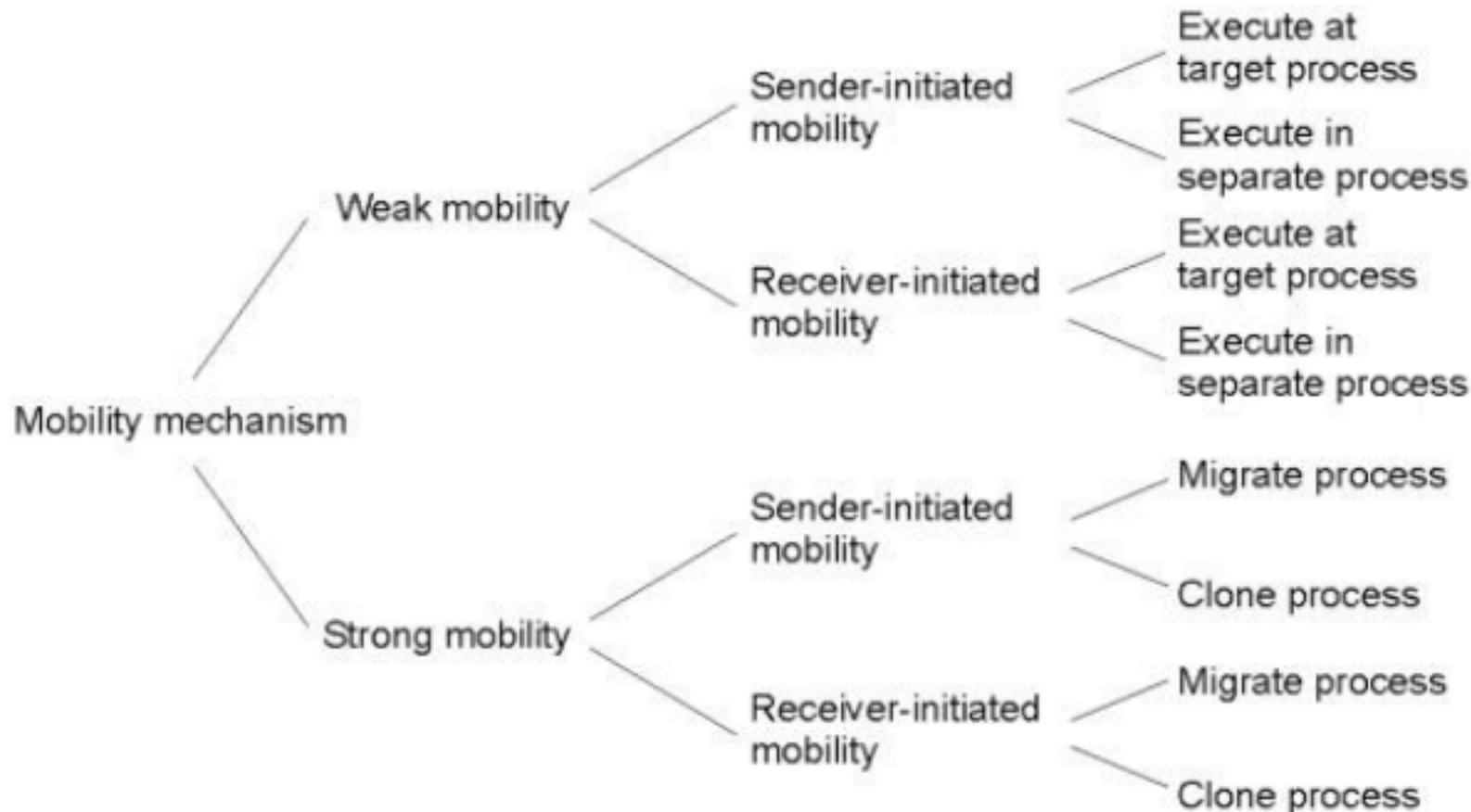


Part III – Process Migration



Part III – Process Migration

Models for Code Migration



Part III – Process Migration

Process Migration

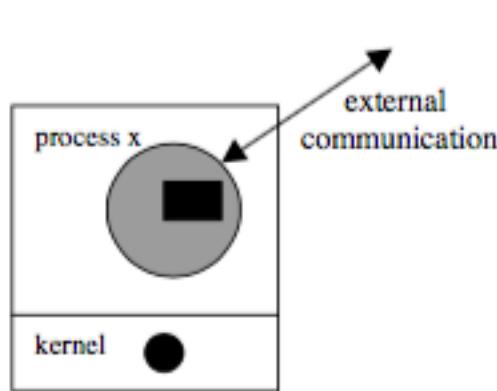
Motivations:

- load sharing/balancing - migrate jobs to less utilized machines
- resource locality - migrate jobs to the machine where needed resources exist
- resource sharing - make distinctive resources available to all jobs
- fault tolerance - migrate jobs away from failing machines
- system administration - migrate jobs temporarily for administrative reasons
- mobility/ubiquitous computing - migrate jobs to follow the user

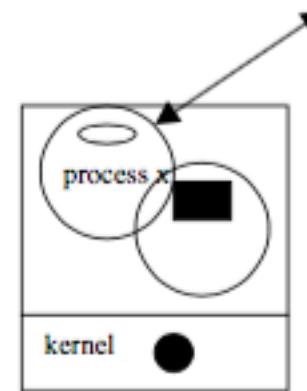
Architectures:

- MPP - massively parallel processors/clusters
- NOW - networks of workstations

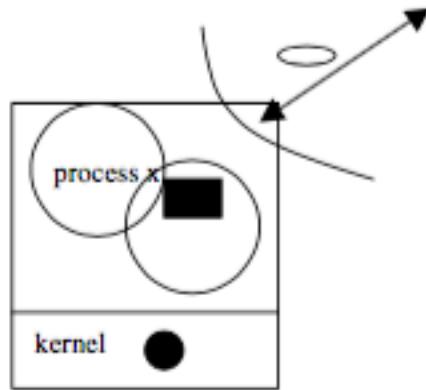
Part III – Process Migration



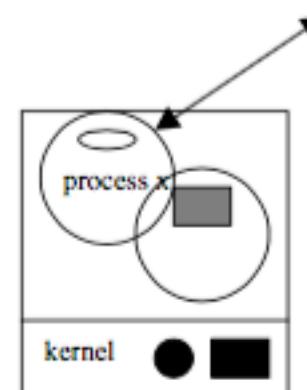
1. Select process to migrate



3. Redirect communication

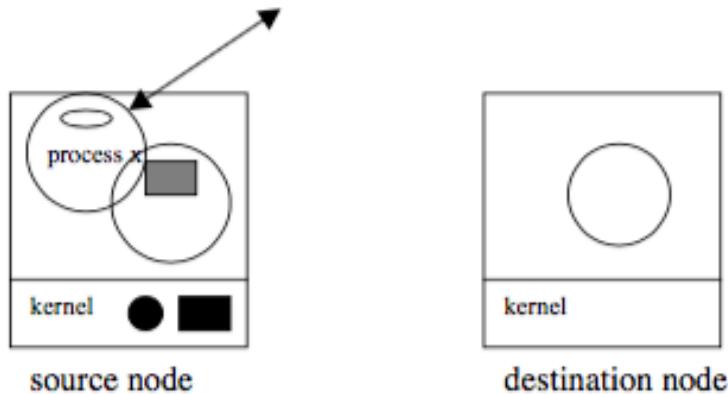


2. Detach process from source node

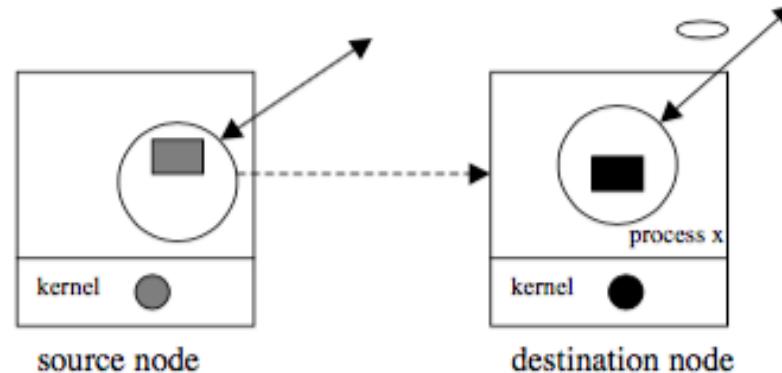


4. Extract state

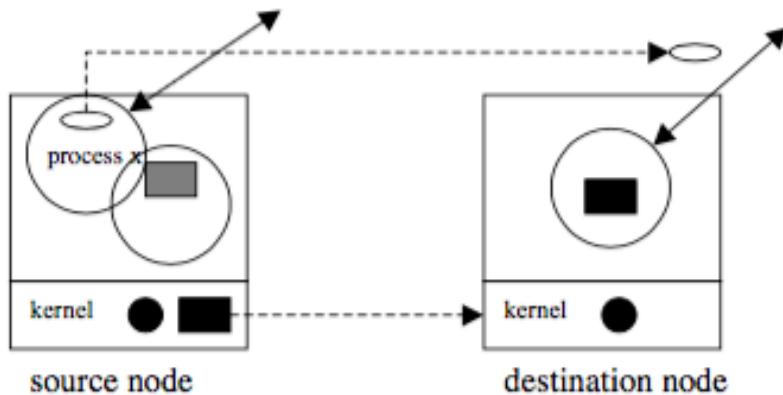
Part III – Process Migration



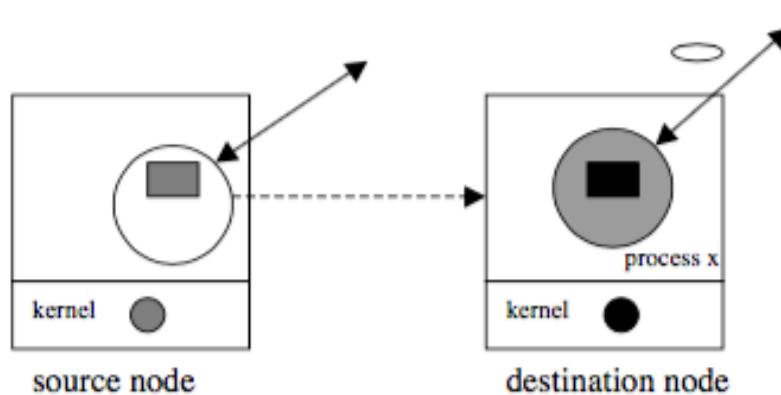
5. Create new process at destination node



7. Create forwarding reference



6. Transfer state

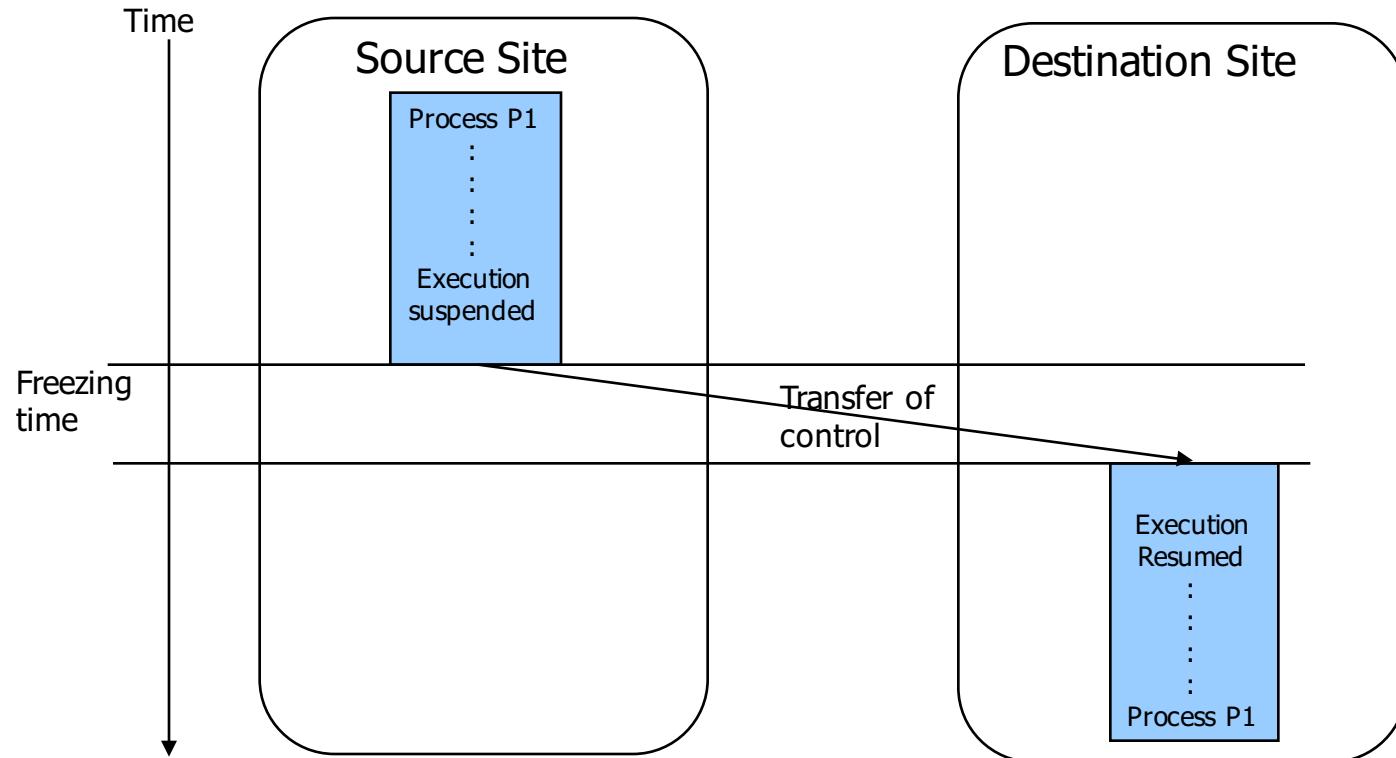


8. Resume migrated process

Part III – Process Migration

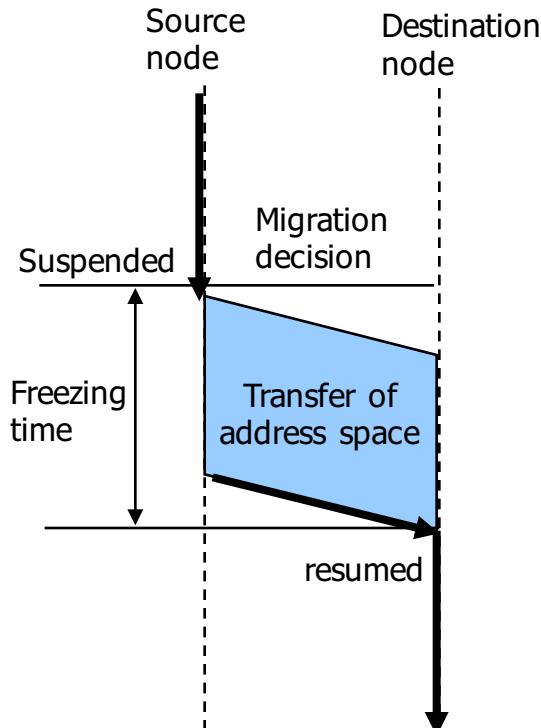
	MOSIX	V	Sprite	Condor
Category	UNIX-like	Message-passing/Microkernel	UNIX-like	User-space
Goals	<ul style="list-style-type: none"> • Dynamic Process Migration • Single System Image • Autonomy • Dynamic Configuration • Scalability 	<ul style="list-style-type: none"> • Network Transparency • Minimal Interference • No Residual Dependencies 	<ul style="list-style-type: none"> • Autonomy • Location Transparency • Using Idle Cycle • Simplicity 	<ul style="list-style-type: none"> • Maximize Utilization • No Modification to Kernel
Extensibility	Fair	Fair	Fair	Very Good
Transparency	Full	Full	Full	Limited
1. (Open Files)	(Yes)	(Yes)	(Yes)	No signals, timers, memory-mapped files and shared libraries
2. (Fork)	(Yes)	(Yes)	(Yes)	
3. (Communication Channel)	(Yes)	(Yes)	(Yes)	
Transfer Strategy	Dirty Pages	Precopy	Flushing	Entire Address Space
Freeze Time	Moderate	Very Low	Moderate	High
Residual Dependency	No	No	No	No
Load information	Distributed	Unknown	Centralized	Combination of both
Performance	Good	Good	Good	Poor

Process migration



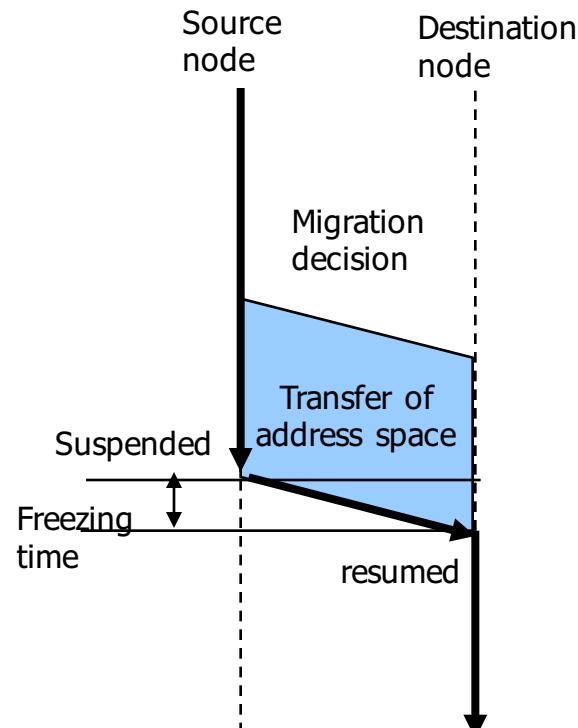
Link redirecting

Total Freezing



Merits: easy implementation
Demerits: long delay time

Pre transferring



Merits: freezing time reduce
Demerits: total time extended

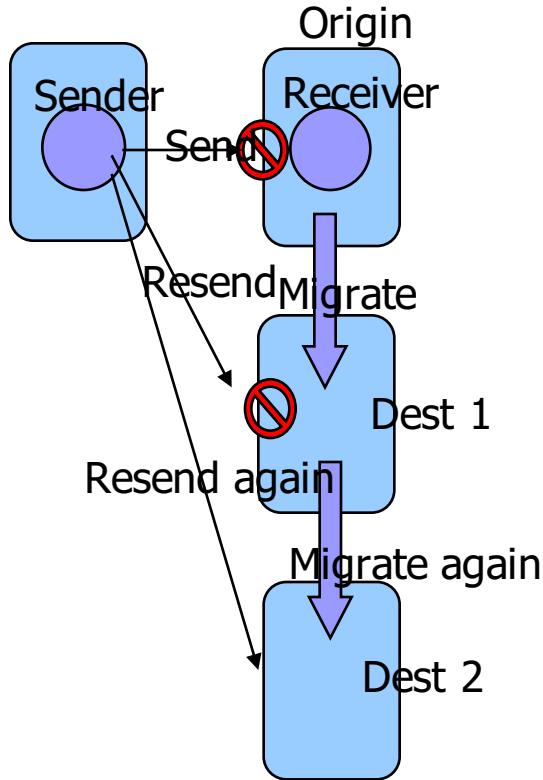
Message forwarding

Three types of messages:

- 1. Received when the process execution is stopped on the source node and has not restarted on the destination node.
- 2. Received on the source node after the execution started on destination node.
- 3. Sent to the migrant process after it started execution on destination node.

Message forwarding

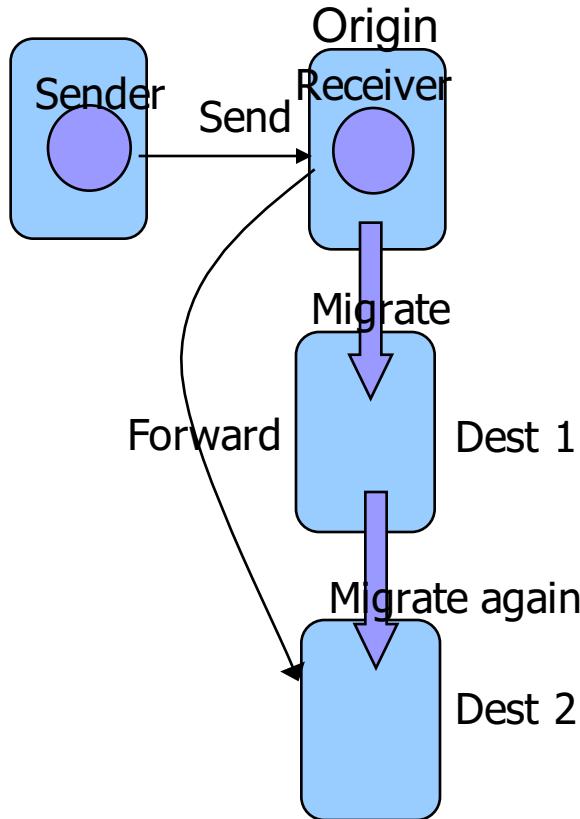
Resending messages



- Messages of type 1 and 2 are either dropped or negatively acknowledged.
- The sender is notified and it needs to locate the migrant process

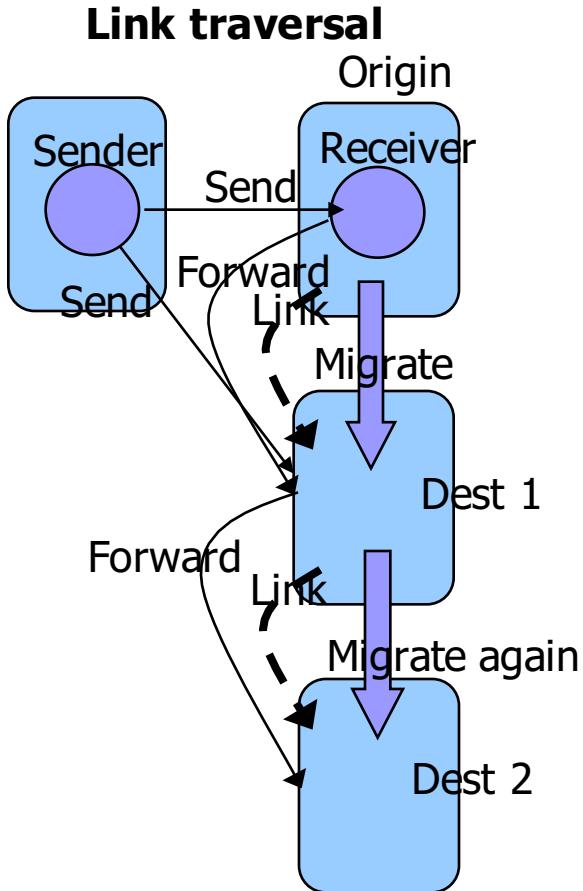
Message forwarding

Ask origin site



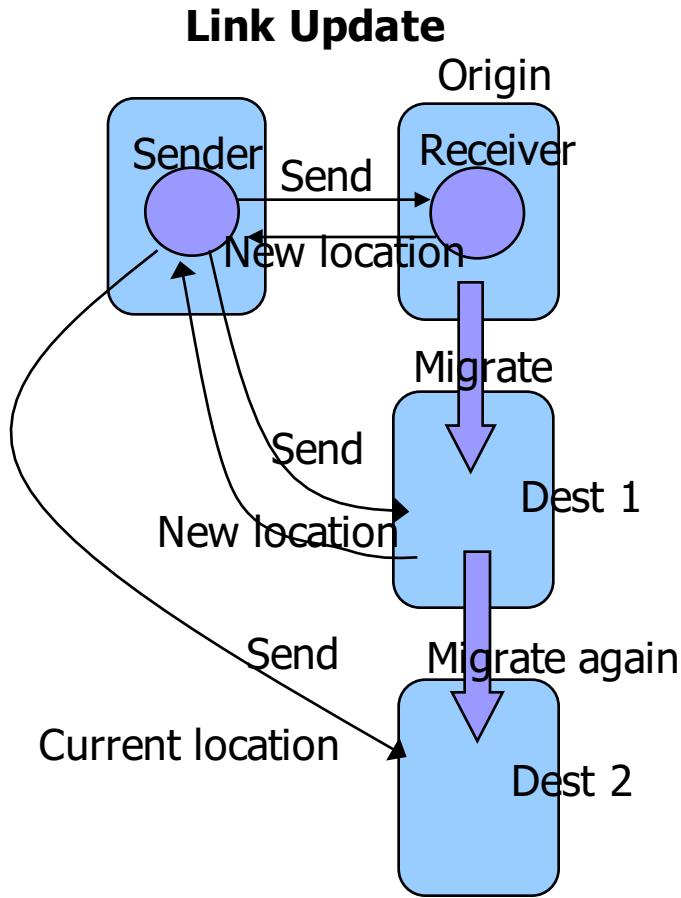
- Origin node keeps the info on the current location of the process created there.
- All messages are sent to origin which forwards them to migrant process.

Message forwarding



- Messages of type 1 are queued and sent to destination node as part of migration procedure.
- Link is left on source node to redirect messages of type 2 and 3.
- Link contains the system-wide unique id of a process and its last known location.

Message forwarding



- During the transfer, the source node sends the notification (link update) of the transfer to all the nodes to which the process communicates.
- Messages of type 1 and 2 are forwarded by the source node
- Messages of type 3 are sent directly to the destination node

Mobile agents

- A mobile agent is a process that can transport its state from one environment to another, with its data intact, and be capable of performing appropriately in the new environment. Mobile agents decide when and where to move.
- Common applications include:
 - Resource availability, discovery, monitoring
 - Information retrieval
 - Network management
 - Dynamic software deployment

Mobile agents

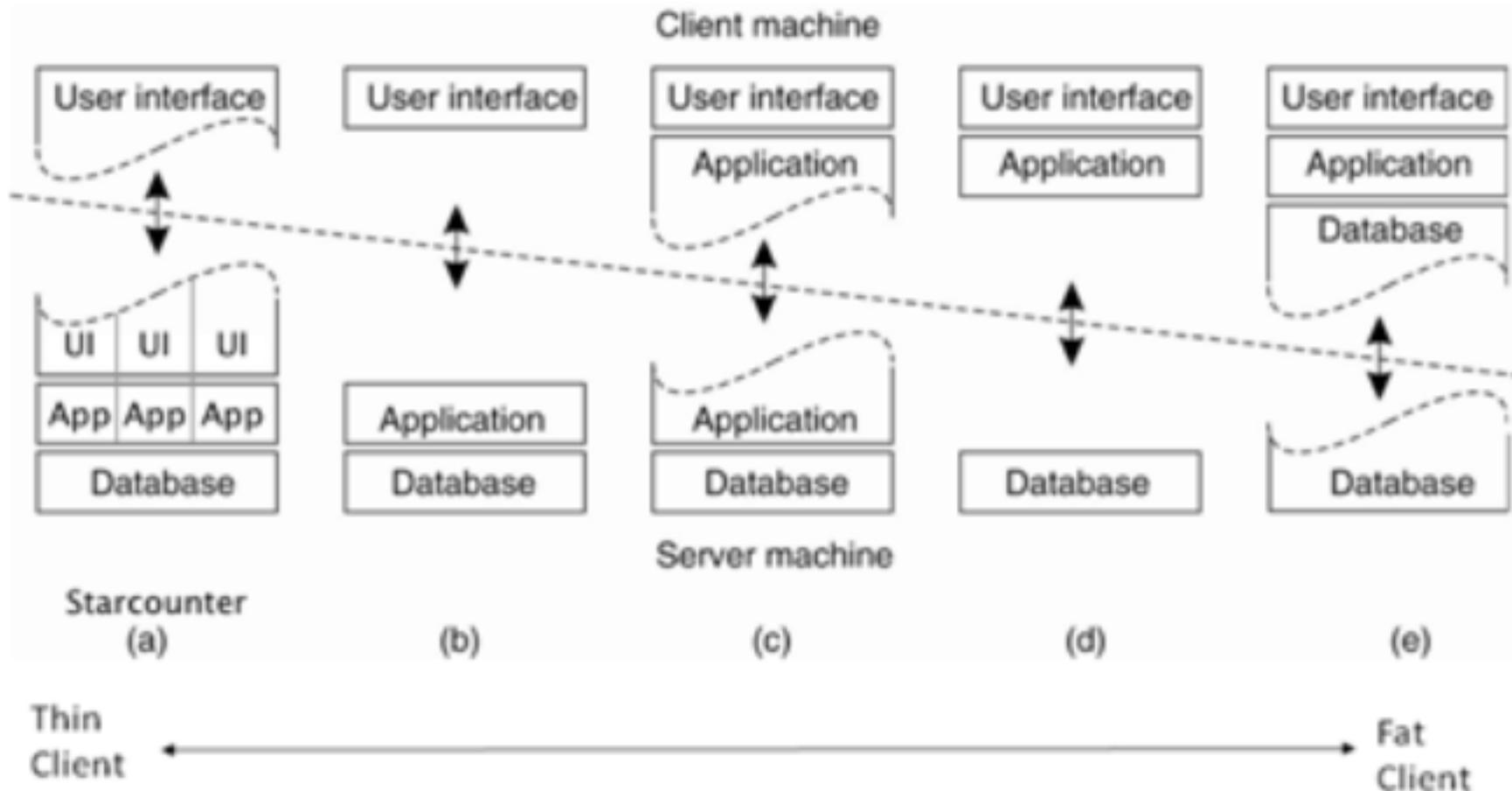
- Mobile agents are agents that can physically travel across a network, and perform tasks on machines that provide agent hosting capability.
- This allows processes to migrate from computer to computer, for processes to split into multiple instances that execute on different machines, and to return to their point of origin.
- Unlike remote procedure calls, where a process invokes procedures of a remote host, process migration allows executable code to travel and interact with databases, file systems, information services and other agents.
- Mobile agents can decide when and where to move next.
- When a mobile agent decides to move, it saves its own state and transports this saved state to next host and resume execution from the saved state.

Process Migration vs. Mobile Agents

	Process migration	Mobile agents
Navigational Autonomy	Migration decision is made by system.	Agents decide where and where to go
Code Execution	Programs are fully compiled and executed in native mode.	Most agents are coded in Java and are interpreted by their execution engine.
Strong/Weak Migration	Execution is resumed where it has been suspended.	Java-based agents resume their execution from the top of a given method.
I/O State	Long-term I/Os are forwarded to processes migrated to the destination.	Agents relinquish I/O connections every time they depart for their next destination.

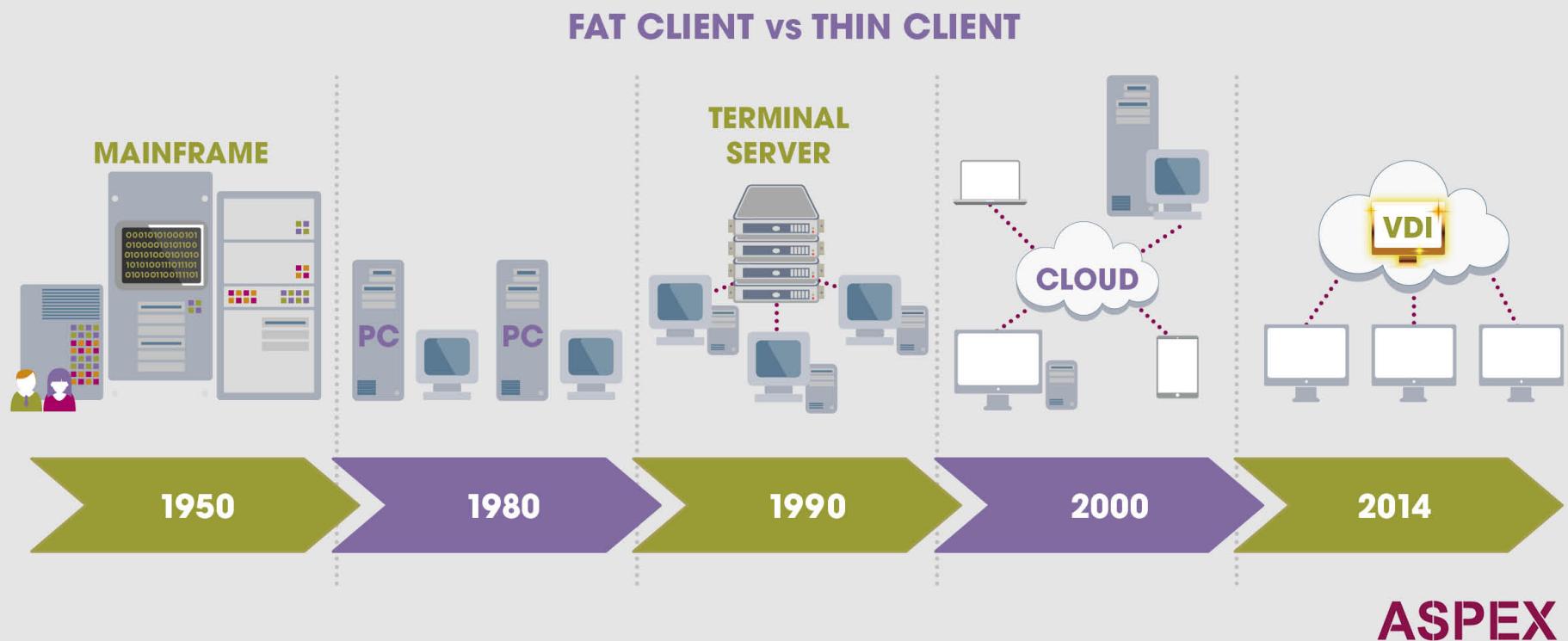
Distributed Systems

Thin vs. Thick/Fat Client



Distributed Systems

Thin vs. Thick/Fat Client



Part III – Process Migration

Are the following instances of data and/or process migrations?

- **WASM – Web Assembly vs. Java Applets**
- **Web Front-end ECMAScript vs. OS process migration**
- **RPC vs. RMI – “only data migration”**
- **RESTful vs. WS- Web Services – “only data migration”**
- **g-RPC vs. CORBA – “only data migration”**
- **Cloud VMs & Docker/Kubernets Containers “hot swap” migration vs. Fog Computing**



Thanks!



DAD – Distributed Application Development
End of Lecture 9

