





Sistemi Distribuiti

Corso di Laurea in Ingegneria

Prof. Paolo Nesi
Parte: 2 – Modelli, Middleware, e Remote Call
Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796363

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet
<http://www.disit.dsi.unifi.it/>
nesi@dsi.unifi.it paolo.nesi@unifi.it
<http://www.dsi.unifi.it/~nesi>, <http://www.axmedis.org>




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 1

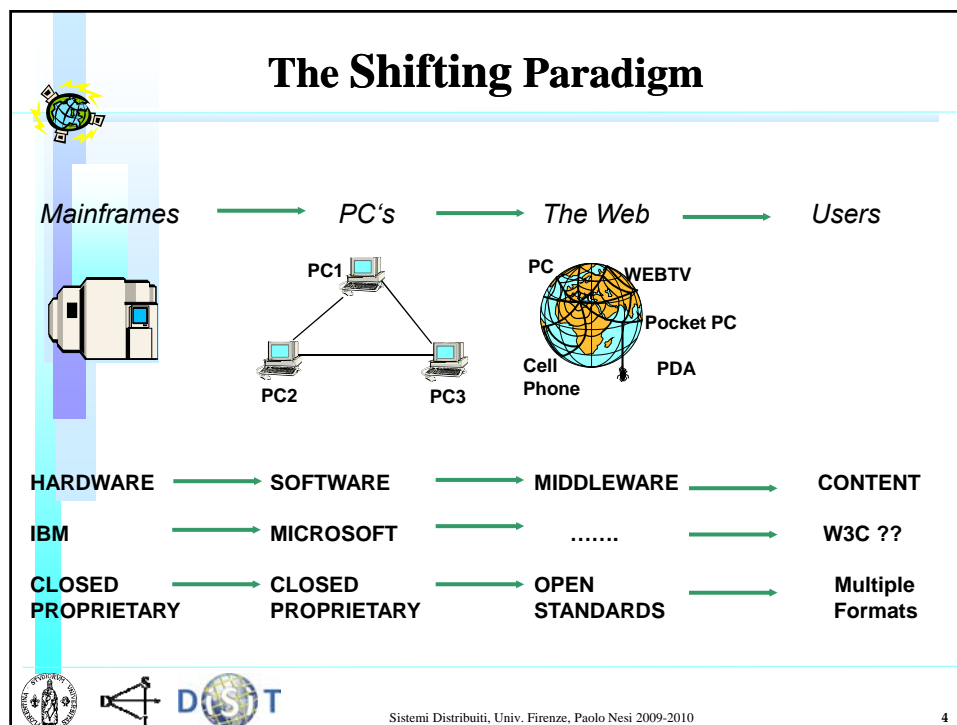
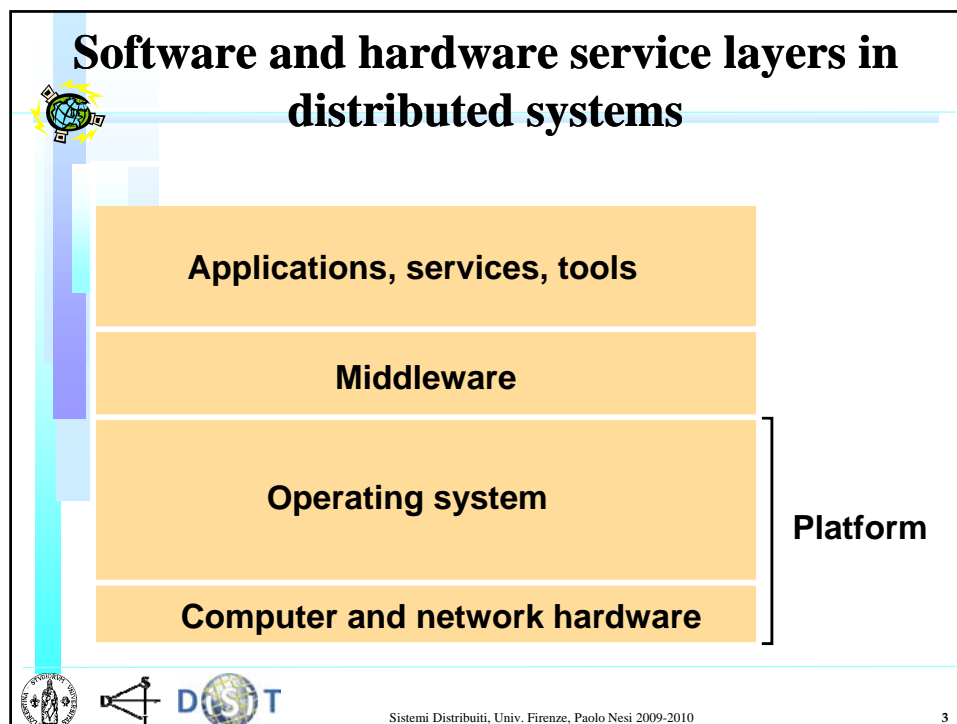


Modelli ed Architetture

- Evoluzione delle architetture
- Client Server, Comunicazione fra processi
- Proxy, peer process, WEB applets, Thin clients
- Modelli di Sistemi Mobili
- Problemi di progettazione di Sistemi Distribuiti
- Modelli di Interazione sincroni ed asincroni
- Modelli di Sincronizzazione di eventi
- Modelli di Sicurezza e distribuzione contenuti




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 2



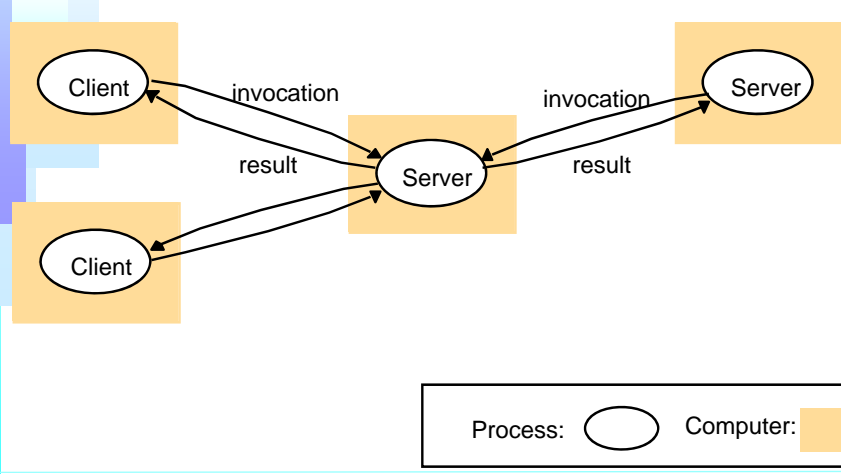
Modelli e Architetture

- **Nelle prossime slide sono proposti**
 - ✦ i principali pattern/configurazioni e concetti dietro alle architetture distribuite
 - ✦ I principali problemi derivati dalle architetture e dei sistemi distribuiti in genere
- **Gli elementi di base sono**
 - ✦ **Server Processes**
 - Provide Services/Answers to the Clients
 - ✦ **Client Processes**
 - Produce Requests to the Servers
 - ✦ **Peer Processes**
 - Play the role of both server and client
 - Provide Services and Produce Requests
 - Sharing data and controls, Collaborating to processes



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 5

Clients invoke individual servers Client Server Model





```
graph LR; subgraph C1 [Computer]; direction TB; C1_C[Client]; end; subgraph C2 [Computer]; direction TB; C2_C[Client]; end; subgraph S [Computer]; direction TB; S_S[Server]; end; C1_C -- invocation --> S_S; S_S -- result --> C1_C; C2_C -- invocation --> S_S; S_S -- result --> C2_C;
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 6

Comunicazioni fra processi

- Due operazioni: *send* and *receive*
 - ♣ un processo invia un comando o un dato ad un altro processo
 - ♣ il destinatario riceve il comando e lo processa,
 - ➔ al limite da conferma a chi gli ha inviato il messaggio/comando,
 - ➔ la conferma viene chiamata
 - Acknowledgement, ACK

- Comunicazione puo' essere
 - ♣ Sincrona, Synchronous
 - ♣ Asincrona, Asynchronous

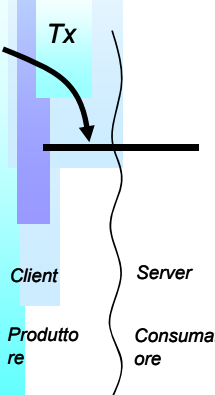



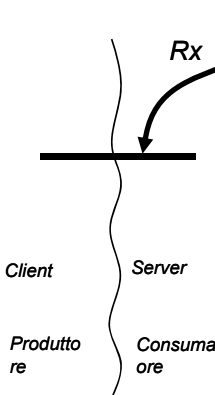
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

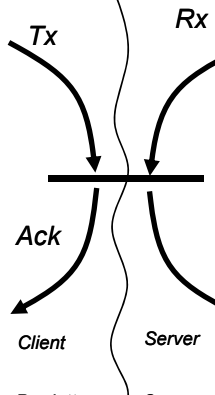
7



Comunicazione Sincrona

- Send and receive are blocking operations







Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

8

Comunicazione Sincrona

- Send and receive are blocking operations

Comando Tx Rx
 Esecuzione comando
 durata definita
 Ack

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 9

Comunicazione Asincrona

- Send is non blocking
- Receiving is blocking (for the first msg), and non blocking
- A queue is associated with each message destination
 - ♣ Send => message added to remote queue
 - ♣ Receive => message removed from local queue

Tx Rx' Rx
 Ack
 Client Produttore Server Consumatore


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 10

History of Distributed Object Models

Communication Protocol Models:

- ♣ Message passing/queueing (DCE, Distrib. Comp. Environment)
- ♣ Request/response (RPC, remote procedure call)

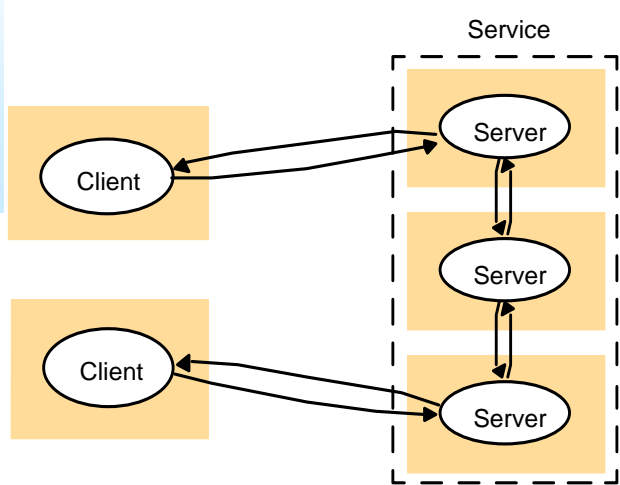
- 1980: model based on network layer
 - ♣ NFS (network file system)
 - ♣ DCE (Distrib. Comp. Environment)
 - ♣ RPC (remote procedure call)
- 1990: object-oriented RPC, to link objects




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 11

A service provided by multiple servers

Clients of each other



The diagram illustrates a distributed service architecture. On the left, two separate boxes represent clients, each containing an oval labeled 'Client'. On the right, a dashed box labeled 'Service' contains three ovals labeled 'Server' stacked vertically. Bidirectional arrows connect each 'Client' to the top 'Server' and the bottom 'Server' of the 'Service' box. Additionally, bidirectional arrows connect the top 'Server' to the middle 'Server', and the middle 'Server' to the bottom 'Server', indicating that the servers themselves are clients of each other.



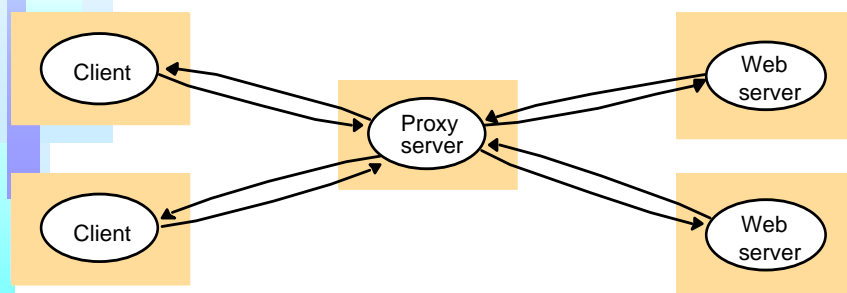
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 12

Cluster

- Questo pattern viene usato quando si realizzando cluster di server che virtualizzano servizi
- L'insieme di server si presenta verso l'esterno come nun unico server anche se di fatto il servizio al suo interno viene decomposto in sottoservizi.
- Il c luster puo' avere svariate soluzioni, fra queste:
 - ♣ Bilanciamento del carico
 - ♣ Duplicazione dei database
 - ♣ Server in fail over, copia calda
 - ♣ Etc.



Web proxy server

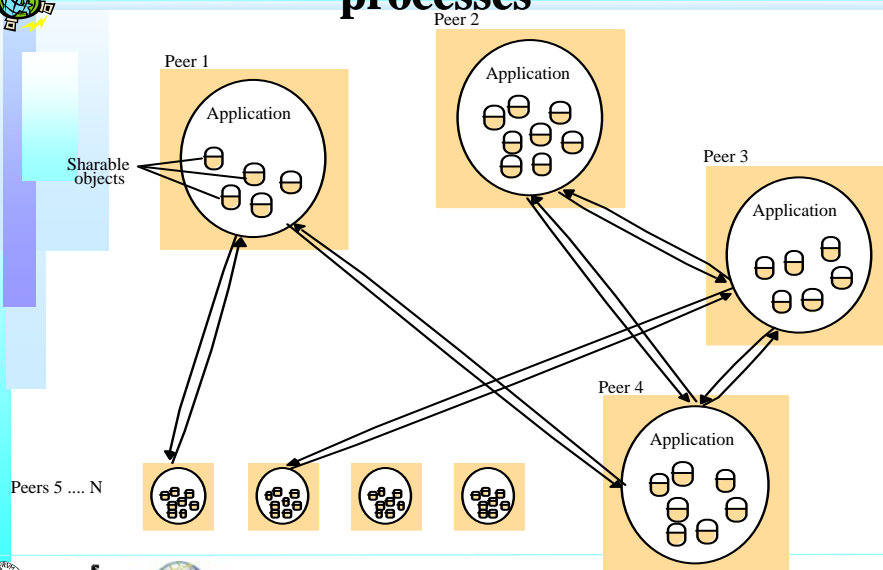


Proxy

- Il concetto di Proxy modella un patter secondo il quale un server o agente intermedio viene utilizzato per accedere a servizi remoti anche molto complessi ed in numero elevato.
- Esempi sono i proxy per accesso HTTP
 - ♣ Proxy del browser,
- I server proxy possono avere
 - ♣ Una memoria cache dei risultati delle richieste
 - ♣ Mascherano la complessita' del sistema reale
 - Semplificano la gestione del server chiamante che non deve conoscere la complessita' del sistema cluster server
 - ♣ non solo http protocol ma anche servizi complessi



A distributed application based on peer processes



Web applets

a) client request results in the downloading of applet code

b) client interacts with the applet

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

17

Migration and cloning

Migration
Muovo


cloning
copio

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

18

Mobile Agents

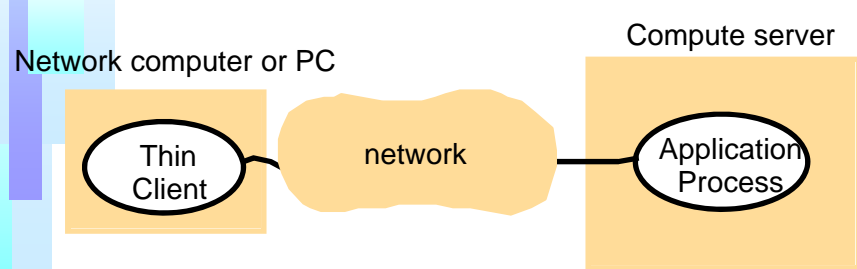
- **Un Agente e' running program**
 - ♣ puo' includere data e codice
- **Un Agente puo' muoversi/migrare da un computer ad un altro nella rete per compiere un certo compito, per esempio:**
 - ♣ Recuperare dei dati
 - ♣ Definire la struttura delle comunicazioni di rete
 - ♣ Fare dei conti
 - ♣ Etc.
- **Un Agente si puo' comportare in base ad un**
 - ♣ Approccio definito, basato su obiettivi prefissati:
 - ricerca dati, per esempio un crawler,
 - i worm sono degli agenti, etc.
 - ♣ Approccio dinamico in base a funzionali di costo che cerca di minimizzare, o strategie di vario tipo, ...




19

Thin clients and compute servers

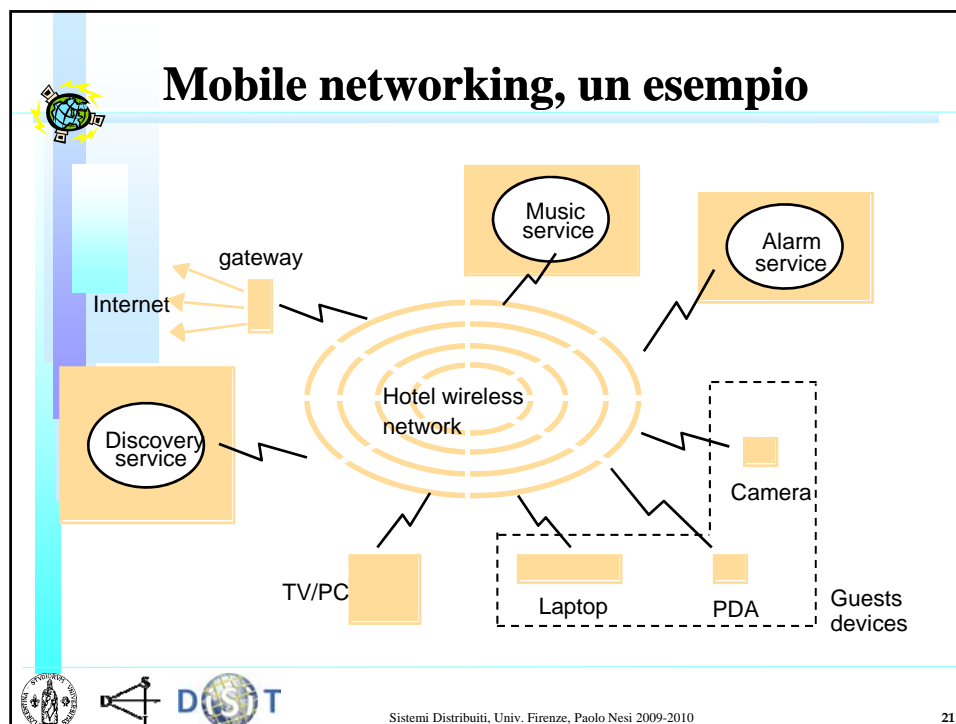
X11, Citrix, Window Terminal Server



- VNC
- Radmin
- PCanyware
- Remote Desktop




20



- ## Problemi progettuali dei Sistemi Distribuiti
- **Prestazioni, performance**
 - ♣ Responsiveness: reattività
 - ♣ Throughput: velocità di esecuzione, banda
 - Time critical applications: e.g., streaming, WEB caching, proxy, etc.
 - **Quality of Service, QOS, qualità del servizio**
 - ♣ Reliability, affidabilità
 - ♣ Security, sicurezza delle informazioni
 - ♣ Performance, prestazioni
 - ♣ Adaptability, adattabilità a nuove funzioni e modelli
 - **Dependability:**
 - ♣ Correctness+security+fault tolerance
 - ♣ Correttezza+sicurezza+tolleranza ai guasti
- Logos for the University of Florence and DISIT are at the bottom left. Text at the bottom center reads 'Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010'. The page number '22' is at the bottom right.

Modelli di Interazione per Sistemi Distribuiti

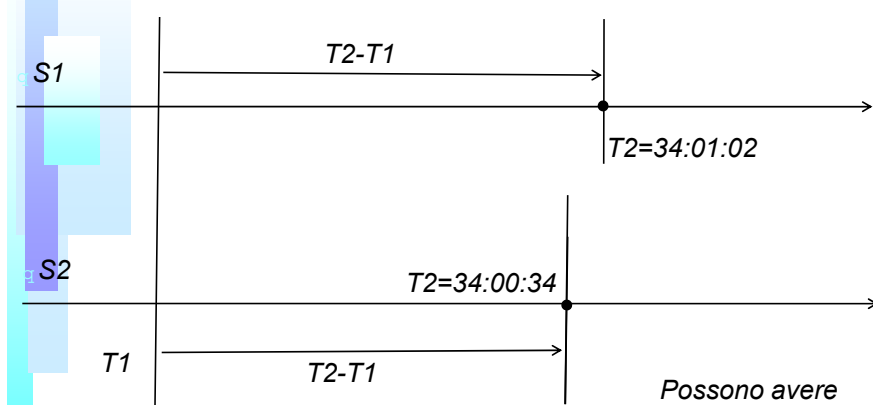
- **Sincroni**
 - ♣ i tempi di comunicazione sono vincolati da vincoli temporali
 - Temporal Constraint, TC,
 - ♣ Ogni messaggio ha un TC
 - un TC=[Tmin, Tmax]
 - Fai XX non prima di 3s ma entro 5s
 - ♣ Ogni processo ha un clock con una deriva nota (errore di sincronizzazione, drift, deriva)
- **Asincroni**
 - ♣ The execution time cannot be predicted,
 - da un ms a un anno ?? . La comunicazione deve poter avvenire anche fra sistemi che hanno diverse velocità
 - il tempo di trasmissione da 0 a anni !!...
 - il Drift è arbitrario!



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010


23

Differenze di tempo Assoluto



Due computer partono con time identico ma in evoluzione libera si trovano ad averlo diverso....time drift

Possono avere
 ≠Start
 ≠Stop
 ≠ Duration





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

24

Ordinamento di Eventi

- **Affidabilità**
 - ♣ garantire: ordinamento di eventi
 - ♣ garantire: tempi di consegna
 - ♣ produrre: protocolli robusti
- **Eventi ordinati**
 - ♣ Time stamp, time information associated with the event and information....
 - ♣ Implica avere un clock comune fra i vari sistemi inclusa la presenza di eventuali fusi orari
- **Problemi di Consistenza temporale**
 - ♣ causalita', convergenza, etc.

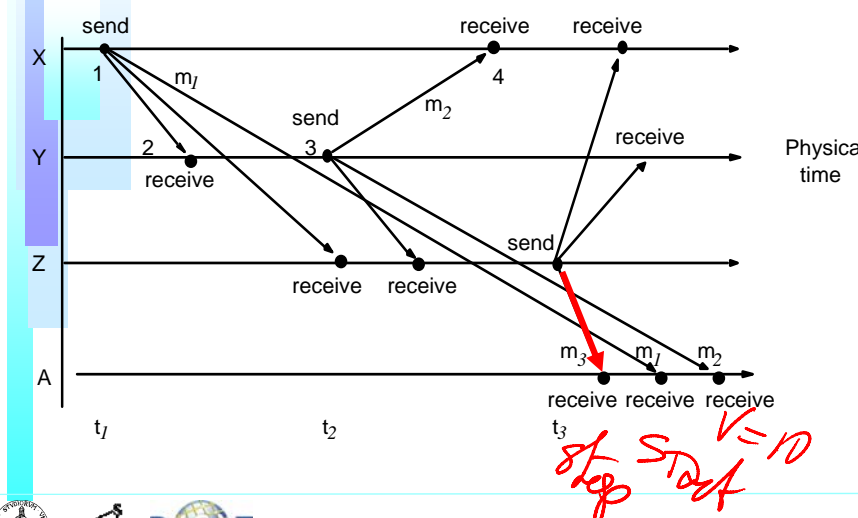





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

25

Problems of time ordering of events







Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

26

Network performance

	<i>Example</i>	<i>Range</i>	<i>Bandwidth (Mbps)</i>	<i>Latency (ms)</i>
<i>Wired:</i>				
LAN	Ethernet	1-2 kms	10-1000	1-10
WAN	IP routing	worldwide	0.010-600	100-500
MAN	ATM	250 kms	1-150	10
Internetwork	Internet	worldwide	0.5-600	100-500
<i>Wireless:</i>				
WPAN	Bluetooth (802.15.1)	10 - 30m	0.5-2	5-20
WLAN	WiFi (IEEE 802.11)	0.15-1.5 km	2-54	5-20
WMAN	WiMAX (802.16)	550 km	1.5-20	5-20
WWAN	GSM, 3G phone nets	worldwide	0.01-02	100-500






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 27

Ethernet ranges and speeds

	<i>10Base5</i>	<i>10BaseT</i>	<i>100BaseT</i>	<i>1000BaseT</i>
Data rate	10 Mbps	10 Mbps	100 Mbps	1000 Mbps
<i>Max. segment lengths:</i>				
Twisted wire (UTP)	100 m	100 m	100 m	25 m
Coaxial cable (STP)	500 m	500 m	500 m	25 m
Multi-mode fibre	2000 m	2000 m	500 m	500 m
Mono-mode fibre	25000 m	25000 m	20000 m	2000 m


645 *646*

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 29

Failure Model

- Nei sistemi distribuiti la comunicazione fra processi puo' ovviamente fallire
 - ♣ Vi sono vari modelli di fallimento e varie cause
- I modelli sono principalmente:
 - ♣ **Omission Failures:**
 - il processo non effettua quello che ci sia aspetta da lui
 - Fail to start, fail to stop, etc..
 - Fail to send, fail to acknowledgment, etc.
 - ♣ **Bizantine Failure:**
 - fallimento arbitrario, set wrong value, return wrong values, etc.
 - ♣ **Timing Failure:**
 - too late, too early, too durate, not expected at that time, etc.




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

30

Omission and arbitrary failures

<i>Class of failure</i>	<i>Affects</i>	<i>Description</i>
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a <i>send</i> , but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

31

Timing failures

<i>Class of Failure</i>	<i>Affects</i>	<i>Description</i>
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

32

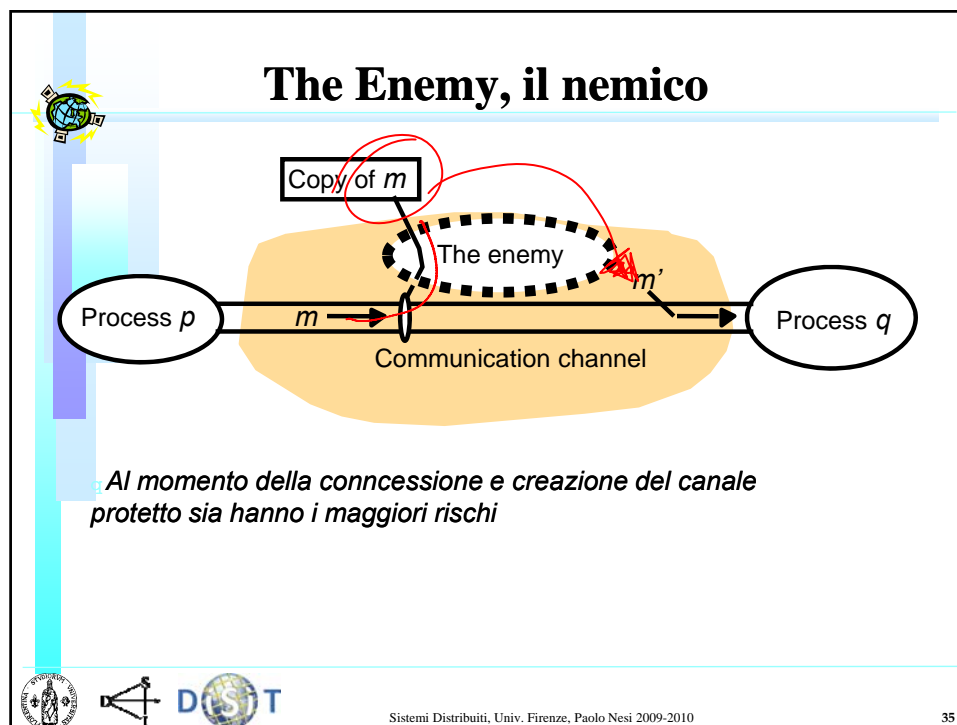
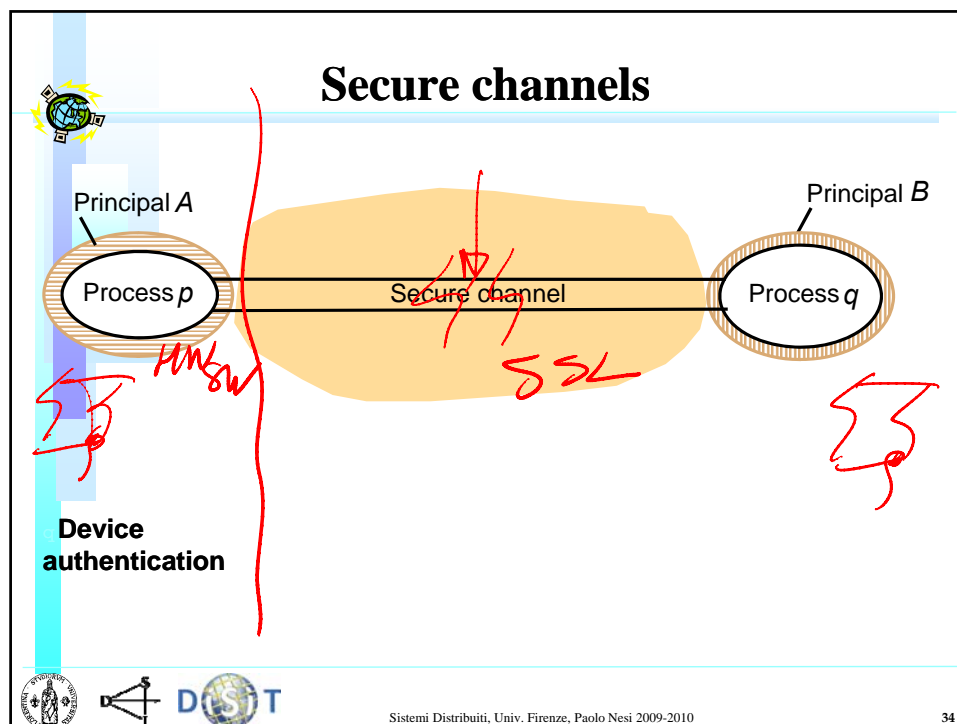
Security Model, controllo accesso

- **Authentication** of users
 - ♣ User profiling
 - ♣ Production of certificates per la gestione delle chiavi
- **Certification** delle integrita' di elementi come
 - ♣ devices (HW+SD), SW tools, etc.
 - ♣ data e content
 - ♣ channel
- **Protezione di canale**
 - ♣ Per esempio: SSL, HTTPS, SFTP
 - ♣ Certificati per la connessione


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

33



Security Model con diritti (rights)

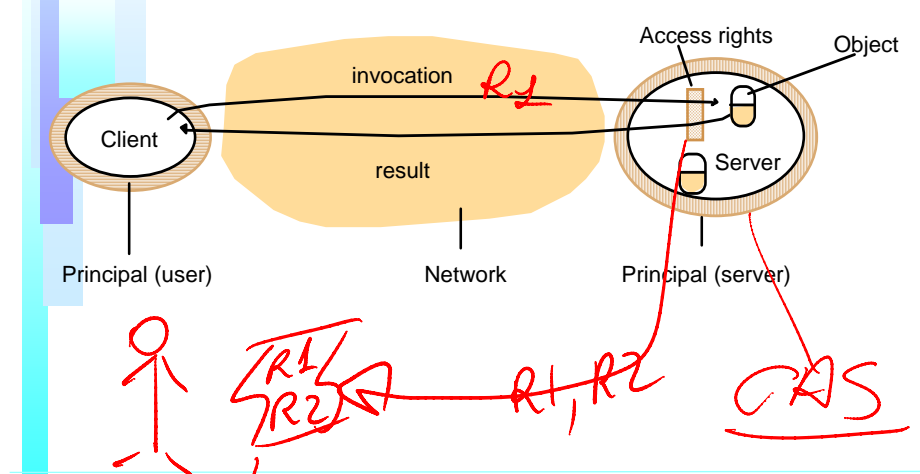
- **Protezione** dei contenuti, degli oggetti:
 - ♣ Objects/components: software tools
 - ♣ Data: digital media, audiovisual, etc.
- **Controllo e prevenzione**
 - ♣ Controllo accesso all'oggetto:
 - Conditional Access Systems, CAS
 - ♣ Controllo dei diritti per l'uso dei servizi degli oggetti
 - Digital Rights Management
- **Sfruttamento controllato** di ogni loro caratteristica/ feature/ servizio
 - ♣ Conto corrente: leggo, muovo soldi, cambio parametri, etc.
 - ♣ Processo: start, stop, uso F1, uso F2, etc.
 - ♣ Contenuto digitale: play, print, copy, etc.




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

36

Objects and principals



The diagram illustrates the interaction between a **Client** (Principal user) and a **Server** (Principal server) over a **Network**. The Client sends an **invocation** (with handwritten R_1) to the Server, and the Server returns a **result**. The Server contains **Access rights** and an **Object**. Handwritten notes include R_1, R_2 and CAS.





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

37




Controllo e supervisione dei diritti

- In seguito vi sarà una parte nel corso in cui queste questioni saranno prese in considerazione in modo molto più dettagliato:
 - ♣ Controllo dei diritti
 - ♣ IPR: Intellectual Property Rights
 - ♣ DRM: Digital Rights Management
 - ♣ Distribuzione dei contenuti digitali
 - ♣ Distribuzione dei componenti software
 - ♣ DRM vari come: MS DRM, Apple iTunes, MPEG-21, OMA, ...
 - ♣ Etc.






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 38

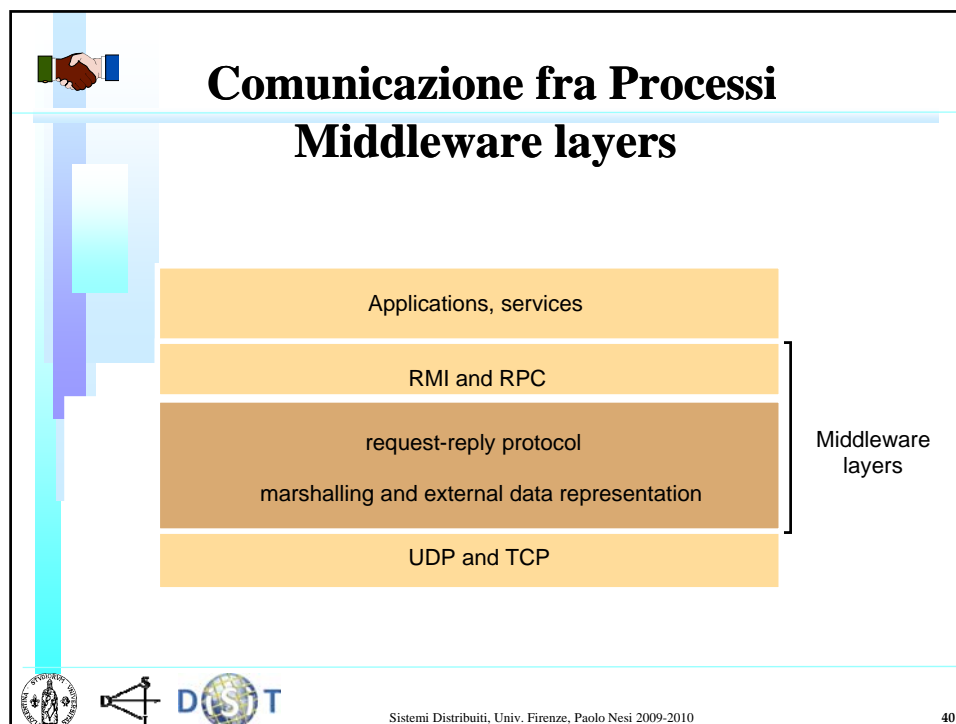


Middleware

- Comunicazione fra processi, Livelli OSI
- Perché il Middleware
- Sockets and ports
- UDP e TCP
- Example: TCP communication
- RPC e RMI
- Data representation and coding for transmission
- MIME, Multipurpose Internet Mail Extensions



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 39

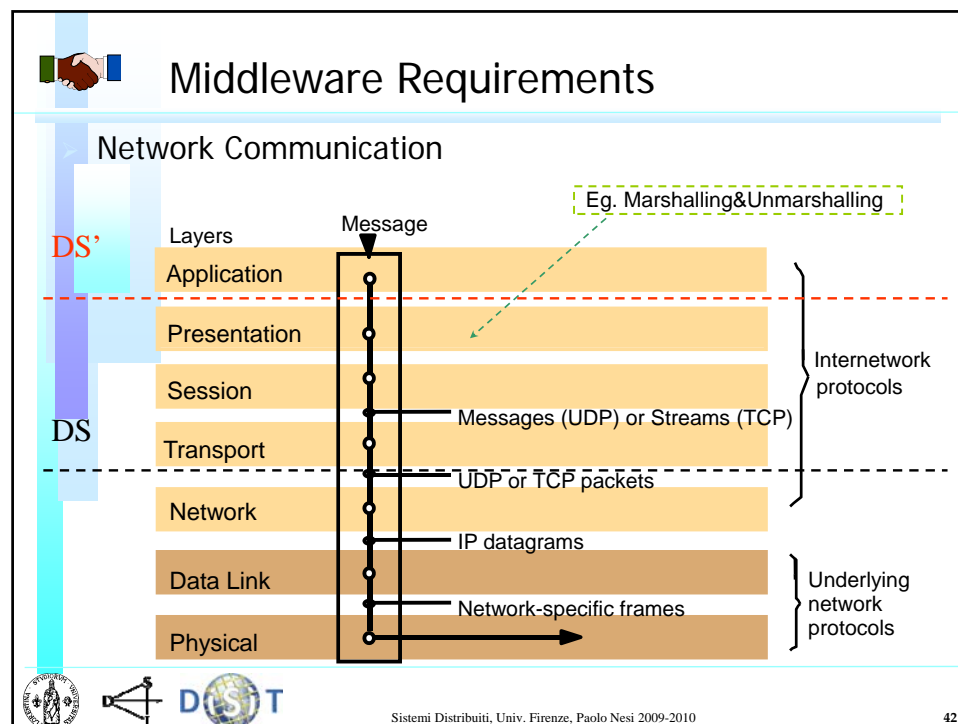


OSI protocol summary

Layer	Description	Examples
Application	Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service.	HTTP, FTP, SMTP, CORBA IIOP
Presentation	Protocols at this level transmit data in a network representation that is independent of the representations used in individual computers, which may differ. Encryption is also performed in this layer, if required.	Secure Sockets (SSL), CORBA Data Rep.
Session	At this level reliability and adaptation are performed, such as detection of failures and automatic recovery.	
Transport	This is the lowest level at which messages (rather than packets) are handled. Messages are addressed to communication ports attached to processes. Protocols in this layer may be connection-oriented or connectionless.	TCP, UDP
Network	Transfers data packets between computers in a specific network. In a WAN or an internetwork this involves the generation of a route passing through routers. In a single LAN no routing is required.	IP, ATM virtual circuits
Data link	Responsible for transmission of packets between nodes that are directly connected by a physical link. In a WAN transmission is between pairs of routers or between routers and hosts. In a LAN it is between any pair of hosts.	Ethernet MAC, ATM cell transfer, PPP
Physical	The circuits and hardware that drive the network. It transmits sequences of binary data by analogue signalling, using amplitude or frequency modulation of electrical signals (on cable circuits), light signals (on fibre optic circuits) or other electromagnetic signals (on radio and microwave circuits).	Ethernet base-band signalling, ISDN

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

41



UDP or TCP

- **UDP messages**
 - ♣ Basata su messaggi fra processi
 - ♣ I singoli pacchetti vengono chiamati Datagram
 - ♣ Si identifica una Socket port
- **TCP Streams**
 - ♣ Si definisce una canale stream bidirezionale
 - ♣ I messaggi non hanno limitazioni di dimensioni, sono decomposti
 - ♣ Modello produttore-consumatore
 - ♣ Il consumatore deve aspettare
 - ♣ Si identifica due Socket port

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 43

Sockets and ports

Internet address = 138.37.94.248:XX Internet address = 138.37.88.249

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 47

Why use Middleware?

Middleware in Distributed System Construction

Think...

- Using **Middleware** to build **DS** is comparable to that of using **DBMS** when building **IS**.

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 51

RPC and RMI

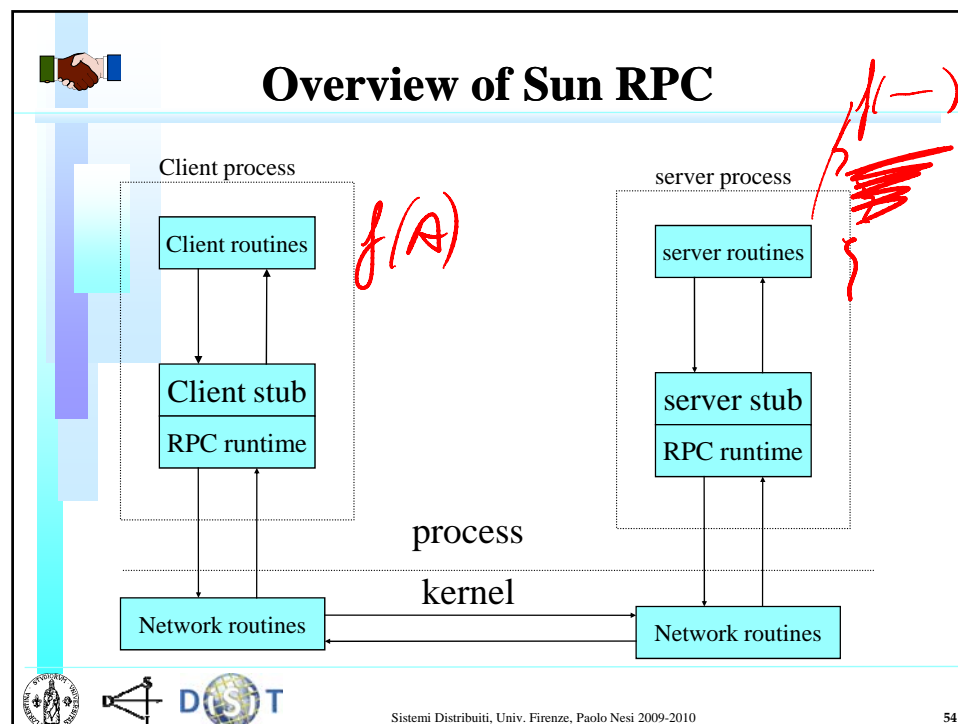
- Remote procedure call, RPC
 - ♣ Procedural Models
 - ♣ Introdotto da Sun agli inizi degli anni 80
- Remote Method Invocation, RMI
 - ♣ Object Oriented Context
 - ♣ Corba or Java RMI
- Necessario garantire
 - ♣ Affidabilità
 - ♣ Ordinamento dei messaggi

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010
52

Elements of RPC

- Client process
- Server process
- Gli Stub:
 - ♣ **Client**: packages the data input to the remote procedures and un-packages the data output from the remote procedures
 - ♣ **Server**: un-packages the data received from client, and packages the return value before sending to client
- "RPC Runtime
 - ♣ library routines
- Network communication:
 - ♣ Via data marshalling and un-marshalling
 - ♣ Via RPC language


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010
53



XDR: eXternal Data Representation



- Standard for eXternal Data Representation
 - ♣ Sun RPC servers transmit data in this representation,
 - ♣ it can be in XML, or ASCII.. etc..
- **Marshalling:**
 - ♣ Any internal data structure is packaged into XDR before sending, this is done by the client stub
- **Un-marshalling:**
 - ♣ Any received external data structure in XDR is unpackaged into local representation by the server stub

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 55




Data Representation and coding for Communication

- **Micro processori, CPU, calcolatori hanno modelli diversi per la rappresentazione dei dati nella memoria fisica: per esempio riguardo all'ordinamento dei nibble:**
 - ♣ Intel – Motorola $\leftarrow \rightarrow$ Little endian --- big endian
 - ♣ Rappresentazione in virgola mobile
 - ♣ Numero dei bit per rappresentare un integer
 - ♣ Etc.
- **I meccanismi di marshalling e unmarshalling**
 - ♣ Prendono i dati e li trasformano avanti e indietro per essere inviati in uno o più pacchetti
 - ♣ Questo lavoro viene effettuato dal layer middleware
 - ♣ Portare tutto a livello ASCII
 - ♣ Necessitano di linguaggi standard per la formalizzazione dei dati

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010



56



CORBA CDR (Common Data Representation) for complex types

<i>Type</i>	<i>Representation</i>
<i>sequence</i>	length (unsigned long) followed by elements in order
<i>string</i>	length (unsigned long) followed by characters in order (can also have wide characters)
<i>array</i>	array elements in order (no length specified because it is fixed)
<i>struct</i>	in the order of declaration of the components
<i>enumerated</i>	unsigned long (the values are specified by the order declared)
<i>union</i>	type tag followed by the selected member

- 15 different basic types: short, long, float, etc...
- Plus the above components
- Common Object Request Broker Architecture
- Marshalling performed into the CORBA Support



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

57

CORBA CDR message

<i>index in sequence of bytes</i>	← 4 bytes →	<i>notes on representation</i>
0-3	5	<i>length of string</i>
4-7	"Smit"	<i>'Smith'</i>
8-11	"h "	
12-15	6	<i>length of string</i>
16-19	"Lond"	<i>'London'</i>
20-23	"on "	
24-27	1934	<i>unsigned long</i>

The flattened form represents a *Person* struct with value: {'Smith', 'London', 1934}



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

58

Indication of Java serialized form


<i>Serialized values</i>				<i>Explanation</i>
Person	8-byte version number		h0	<i>class name, version number</i>
3	int year	java.lang.String name:	java.lang.String place:	<i>number, type and name of instance variables</i>
1934	5 Smith	6 London	h1	<i>values of instance variables</i>

The true **serialized** form contains additional type markers; h0 and h1 are handles

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

59





Recently: Person structure

```

<person id="123456789">
    <name>Smith</name>
    <place>London</place>
    <year>1934</year>
    <!-- a comment -->
</person >
    
```

Per delle slide sull'XML si consulti le pagine web del corso di sistemi distribuiti .

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 60







Illustration of the use of a namespace in the *Person* structure

```

<person pers:id="123456789"
  xmlns:pers = "http://www.cdk4.net/person">
  <pers:name> Smith </pers:name>
  <pers:place> London </pers:place >
  <pers:year> 1934 </pers:year>
</person>
    
```



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 61



An XML schema for the Person structure


```

<xsd:schema xmlns:xsd = URL of XML schema definitions >
  <xsd:element name= "person" type = "personType" />
  <xsd:complexType name= "personType">
    <xsd:sequence>
      <xsd:element name = "name" type="xs:string"/>
      <xsd:element name = "place" type="xs:string"/>
      <xsd:element name = "year" type="xs:positiveInteger"/>
    </xsd:sequence>
    <xsd:attribute name= "id" type = "xs:positiveInteger"/>
  </xsd:complexType>
</xsd:schema>
    
```



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

62



MIME type for file types

- **Multipurpose Internet Mail Extension**
 - ♣ Use for sending binary and files in the emails
 - ♣ Classificazione delle estensioni
- **MIME Type**
 - ♣ Struttura: Type/subtype
 - ♣ E.g.: text/doc, text/html, image/gif, image/jpeg
 - ♣ il sistema operativo presenta API per la loro gestione
 - ♣ Ad ogni tipo vi sono associate delle funzionalità e dei programmi eseguibili, opzioni di Explorer
 - ➔ Le operazioni che il SO puo' chiedere a tali programmi per quei tipi di dati: Open, Print, copy, etc.
 - ♣ Informazioni/tabelle contenute nel Registry


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

63




Call Remote

- Invocazioni Remote
- Interfacce, IDL
- Remote Procedure Call
- CORBA IDL
- Modello ad oggetti di sistemi distribuiti
- Oggetti remoti ed interfacce
- Comunicazione fra oggetti, RMI

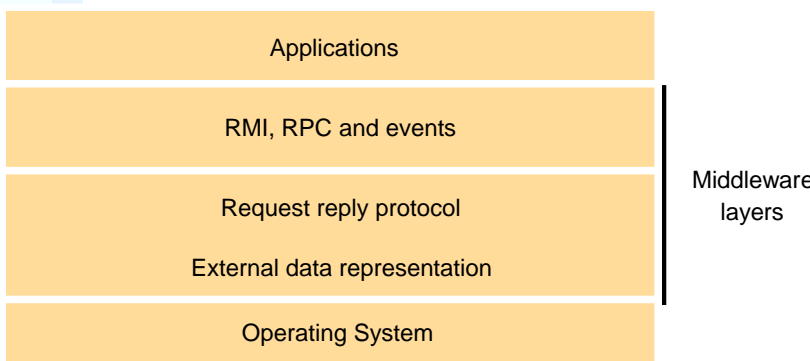


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

64



Oggetti Distribuiti e Invocazioni Remote



Applications




RMI, RPC and events

Request reply protocol

External data representation

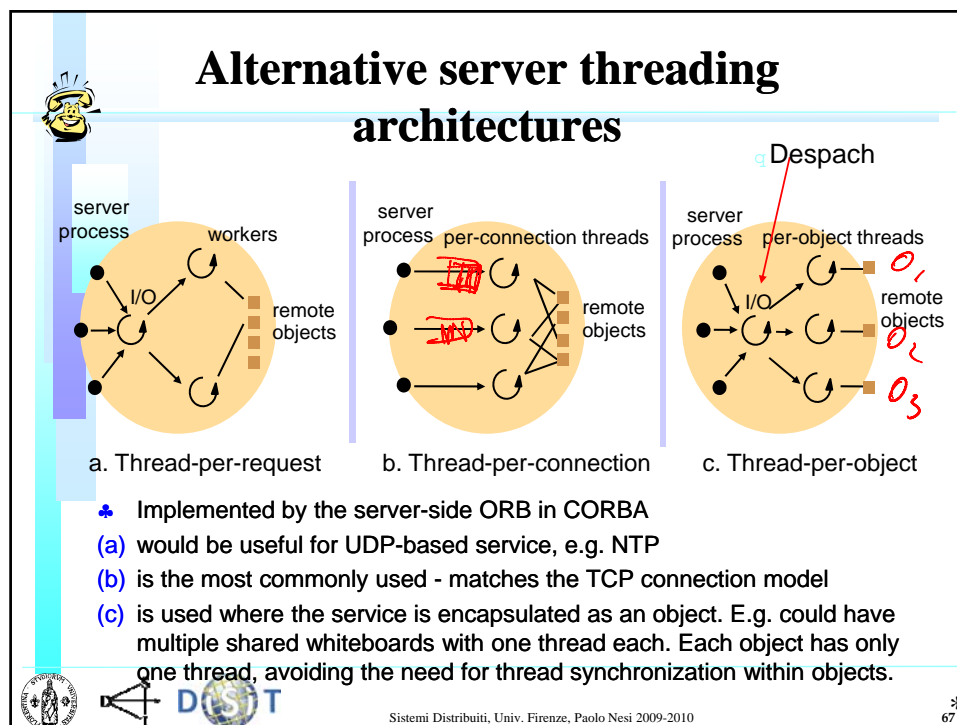
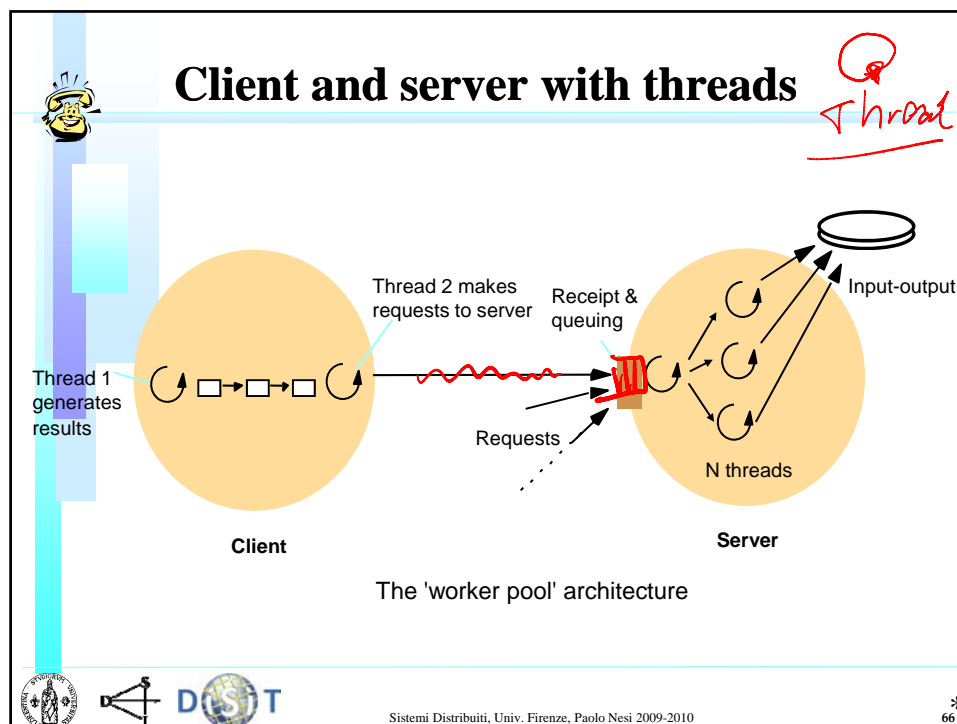
Operating System

Middleware layers



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

65



Java thread constructor and management methods



Methods of objects that inherit from class Thread

- **Thread(ThreadGroup group, Runnable target, String name)**
 - Creates a new thread in the *SUSPENDED* state, which will belong to *group* and be identified as *name*; the thread will execute the *run()* method of *target*.
- **setPriority(int newPriority), getPriority()**
 - Set and return the thread's priority.
- **run()**
 - A thread executes the *run()* method of its target object, if it has one, and otherwise its own *run()* method (*Thread* implements *Runnable*).
- **start()**
 - Change the state of the thread from *SUSPENDED* to *RUNNABLE*.
- **sleep(int millisecs)**
 - Cause the thread to enter the *SUSPENDED* state for the specified time.
- **yield()**
 - Enter the *READY* state and invoke the scheduler.
- **destroy()**
 - Destroy the thread.



The Middleware



- **Middleware is a software layer that provides**
 - ♣ a programming model above the basic building blocks of processes and message passing
 - ♣ location transparency
 - ♣ independence from the details of communication protocols
 - ♣ independence on the underlying transport protocols
 - ♣ Thus independent on the low level coding of types
- **Gestione statica e dinamica di**
 - ♣ Remote Object Reference
 - ♣ Remote Objects
 - ♣ Remote Interfaces



Remote and local method invocations

- each process contains objects, some of which can receive remote invocations, others only local invocations
- those that can receive remote invocations are called *remote objects*
- objects need to know the *remote object reference* of an object in another process in order to invoke its methods. *How do they get it?*
- the *remote interface* specifies which methods can be invoked remotely

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 70

A remote object and its remote interface

Client

remote interface

m1, m2, m3

Server

remoteobject

Data

implementation of methods



m4, m5, m6

Remote object reference

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 71

Descrizione delle Interfacce

- RMI: Remote Method Invocation
- RCP: Remote Procedure Call
- L'interfaccia di comunicazione deve essere pubblica:
 - ♣ Nome dei metodi, tipo, ordine e numero di parametri della RPC/RMI, in-out parameter
- La descrizione dell'interfaccia puo' essere utile per la generazione automatica del contesto di Marshalling
- Esistono linguaggi per la descrizione delle interfacce,
 - ♣ Per esempio: IDL, Interface Description Language
- Lo stesso oggetto puo' avere diverse interfacce, una pubblica, una per l'interno...
- Oggetti che presentano RMI vengono detti oggetti distribuiti





Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

72

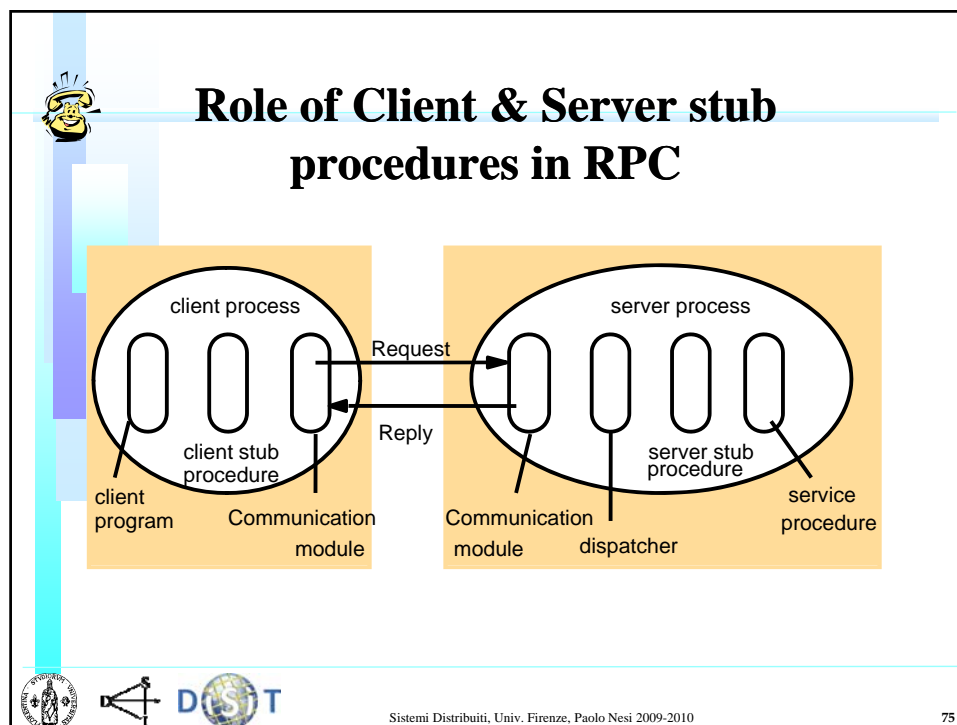
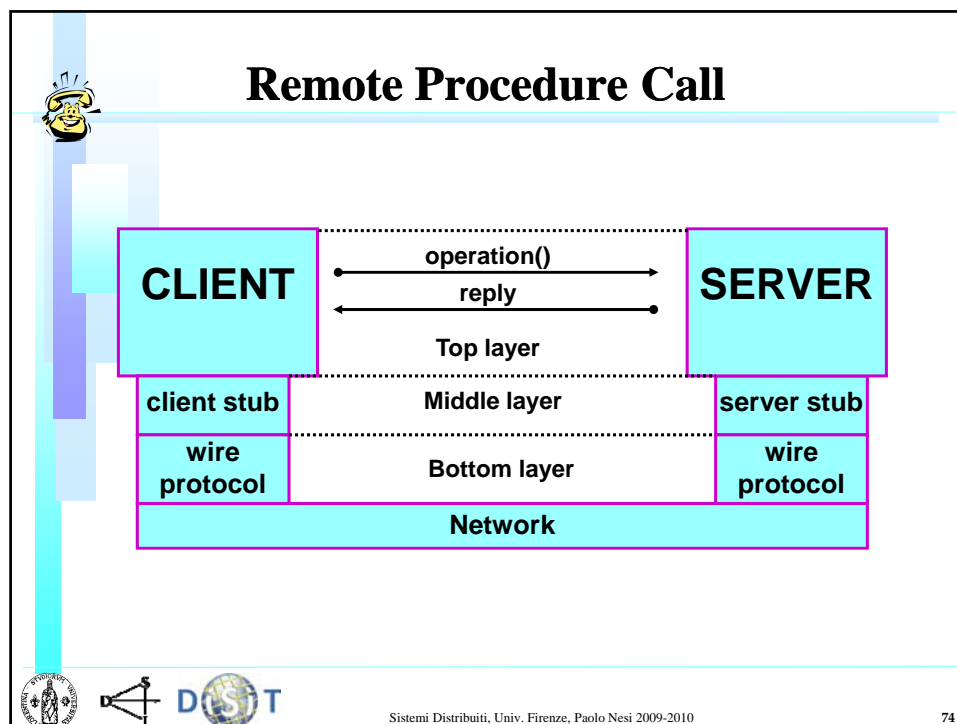
IDL, Interface Description/Definition Language

- Sun XDR has IDL for RPC
 - ♣ Obsolete
- DCOM (Distributed Common Object Model)
 - ♣ IDL based on DCE (Distributed Computing Environment) for RPC
 - ♣ Microsoft COM and OLE, alla base degli ActiveX.....
- CORBA IDL, RMI
- Java RMI, RMI
- .NET, etc...



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

73



CORBA IDL example



- Interface Definition Language
- Data hiding, utilizzato in OO
- L'interfaccia è il solo modo di accedere all'informazione
- Formalizzazione della chiamata con parametri di In/out
- L'interfaccia trova la sua implementazione nella classe

```

// In file Person.idl
struct Person {
    string name;
    string place;
    long year;
};

interface PersonList {
    readonly attribute string listname;
    void addPerson(in Person p) ;
    void getPerson(in string name, out Person p); OBJECT-ID
    long getyear();
    
```

- Parametri per valore nelle due direzioni
- puntatori non hanno senso, i processi sono diversi, la memoria è diversa
- Reference: in alcuni casi esistono reference assoluti del sistema distribuito che si possono usare come OBJECT-ID






Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

77

Object Model

- **Object References**
 - ♣ Java and C++, possono essere passati
- **Interfaces**
 - ♣ Insieme di metodi con la loro signature (tipo e ordine dei parametri incluso quello di ritorno)
 - ♣ una classe puo' implementare diverse interfacce
 - ♣ L'interfaccia è considerata anche un tipo
- **Actions**
 - ♣ Per cambiare lo stato
 - ♣ Per attivare altre azioni, chiamare altri metodi
- **Exceptions**
 - ♣ Throw-catches
- **Garbage Collection**
 - ♣ Singolo o distribuito

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010


78

Distributed Objects

- Comunicazione tramite RMI
- Instanziazione di oggetti in locazioni diverse
- Migrazione degli oggetti con il loro codice (classe)
- Realizzazione di soluzioni fault-tolerant

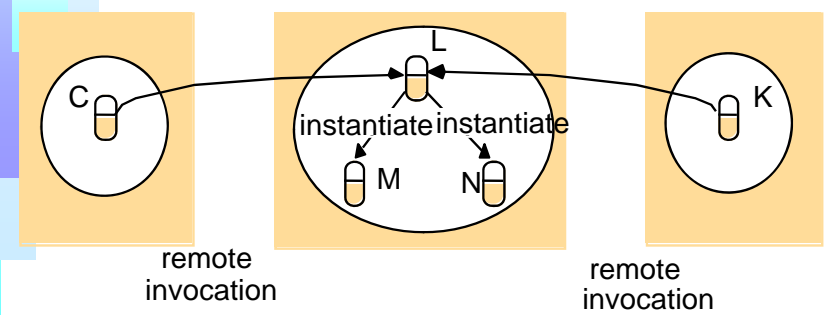
- Remote object reference, OBJID, OBJ unique ID
- Remote Interface

- Accesso ai descrittori dell'interfaccia degli oggetti
 - ✦ per definire in modo dinamico come accedere e quali accedere




79

Instantiation of remote objects



remote invocation remote invocation



80

Object and Component Middleware

- Remote Method Invocation (RMI)
 - ✦ Main Idea: Working with an object on a remote machine is *made to look like calling a procedure on the remote site*, i.e.
 - the application/client sends a message to the remote object
 - the remote object receives message, does processing and sends back message with results - the server side;
 - the client receives message and uses/prints result
 - ✦ RMI Communication

Client

doOperation

⋮

(wait) *sleep*

⋮

(continuation)

Request
message

←

Reply
message

Server

getRequest

select object

execute method

sendReply

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

81

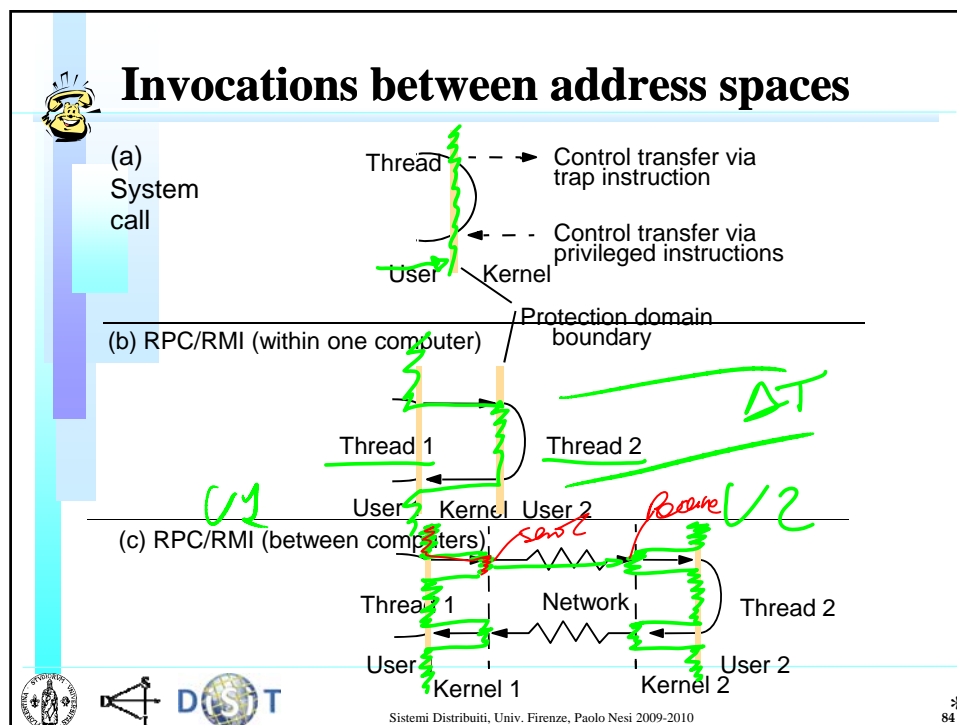
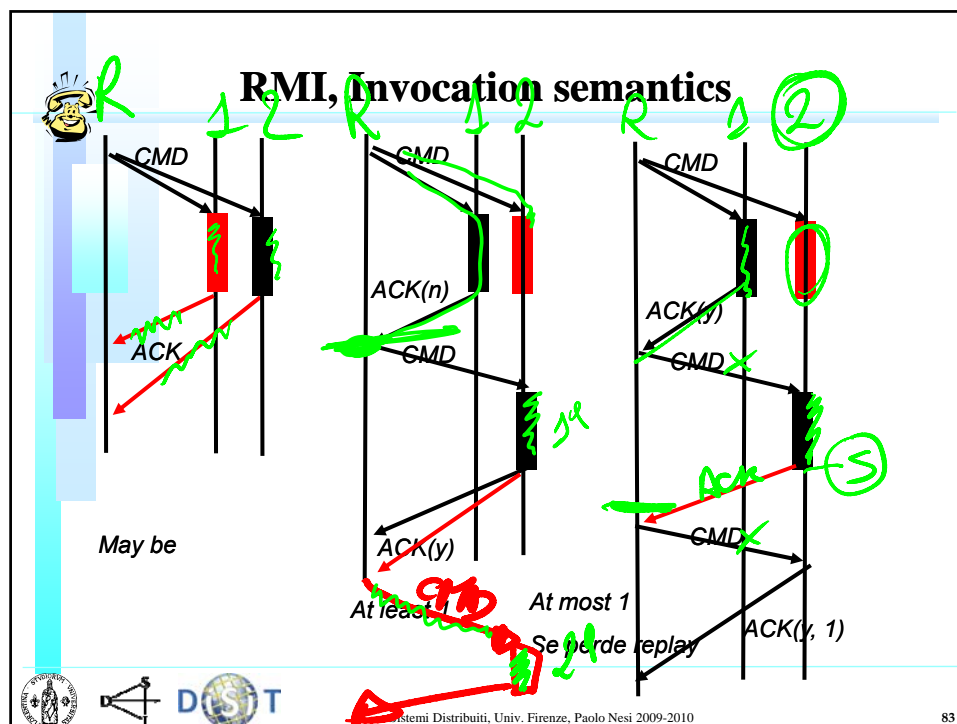
RMI, Invocation semantics

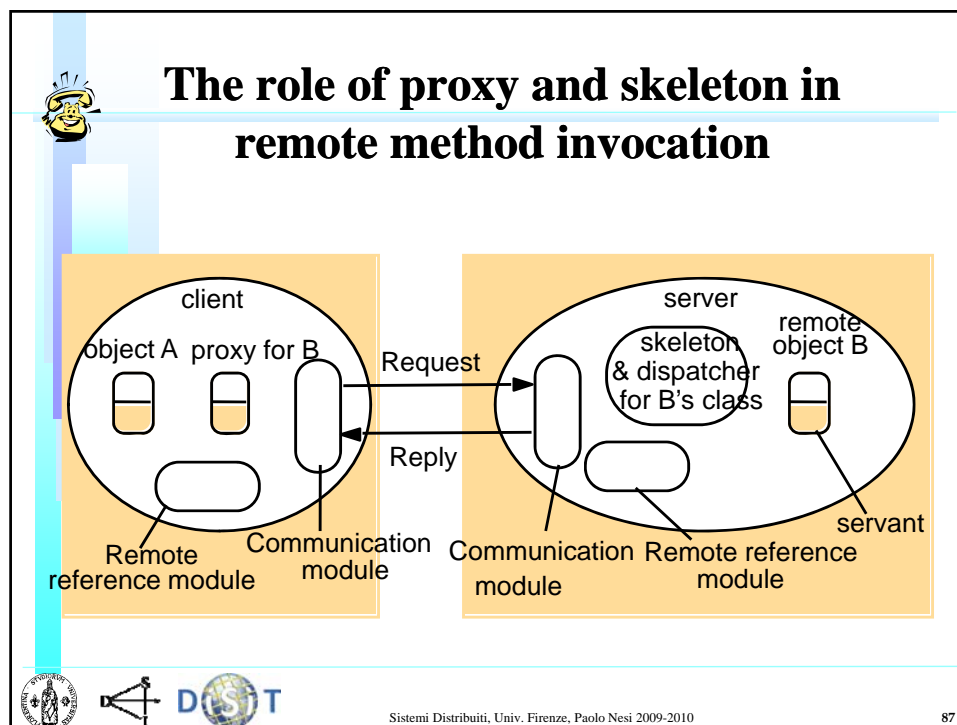
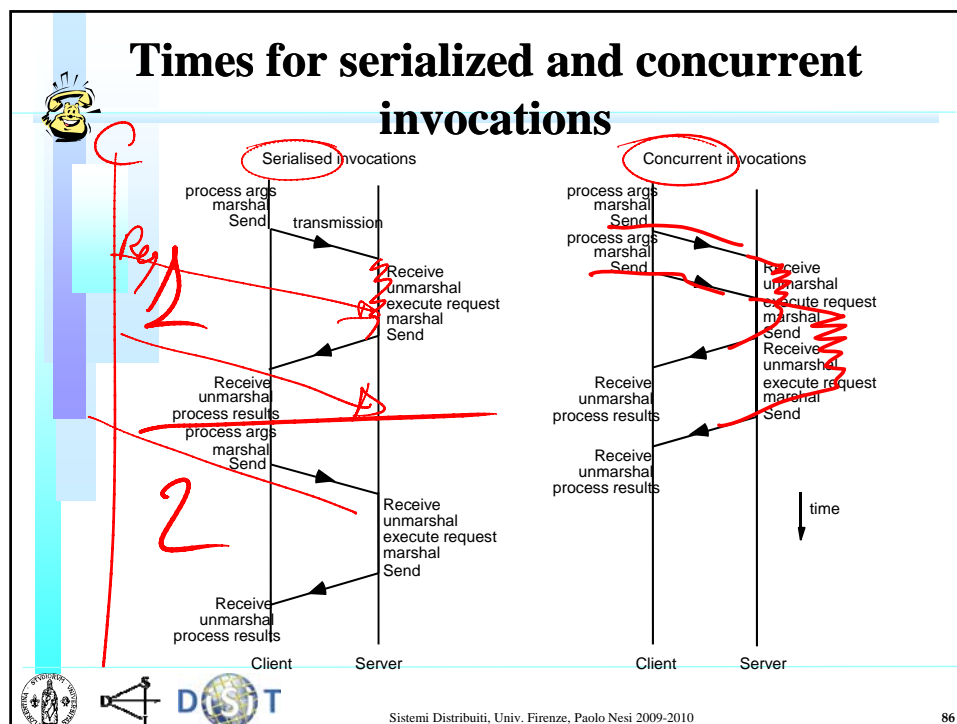
<i>Fault tolerance measures</i>			<i>Invocation semantics</i>
<i>Retransmit request message</i>	<i>Duplicate filtering</i>	<i>Re-execute procedure or retransmit reply</i>	
No	Not applicable	Not applicable	<i>Maybe</i>
Yes	No	Re-execute procedure	<i>At-least-once</i>
Yes	Yes	Retransmit reply	<i>At-most-once</i>

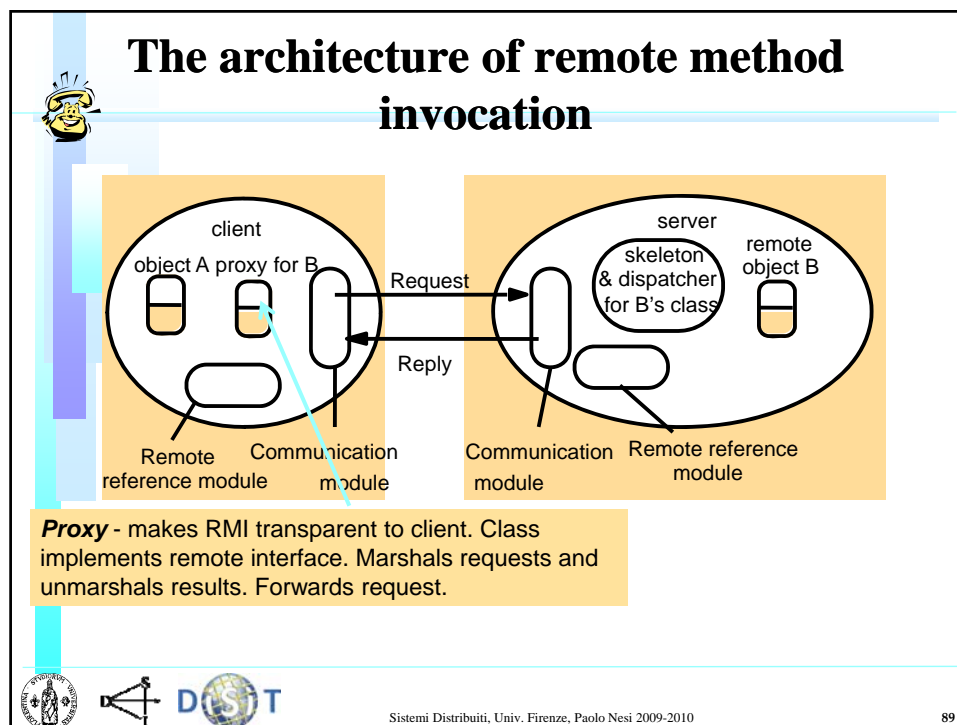
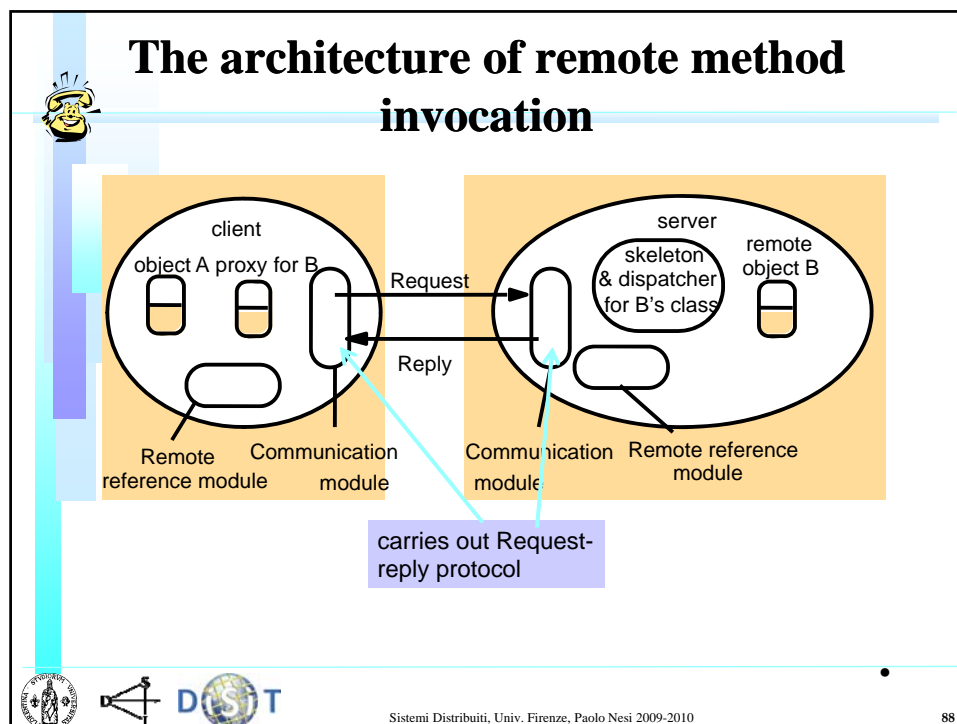
• *Invocation Semantics:*
Esattamente una esecuzione solo un modello per local call

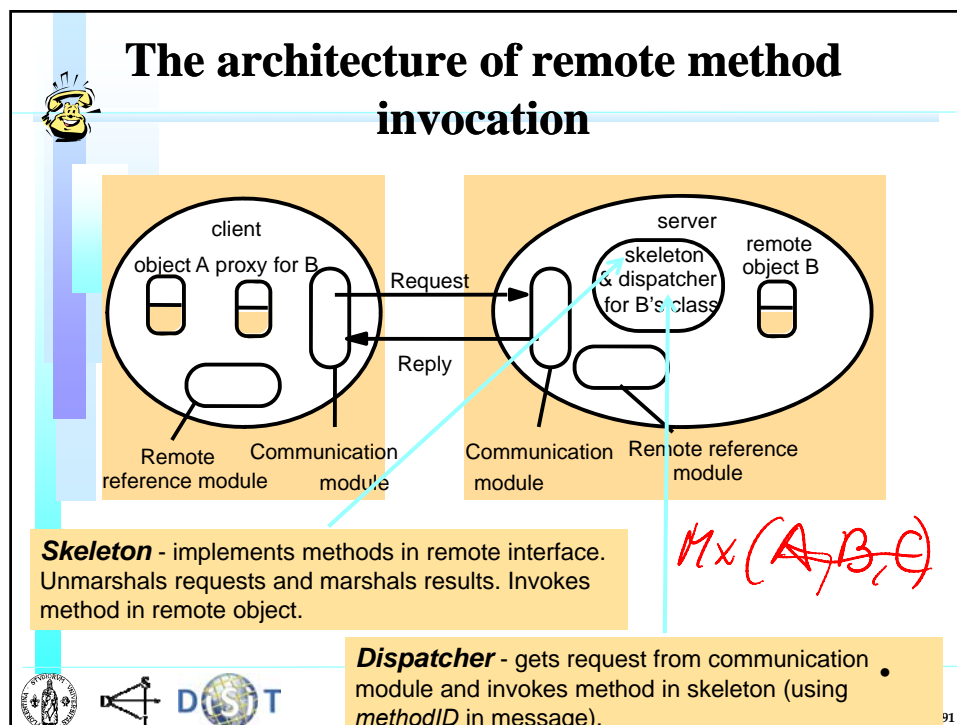
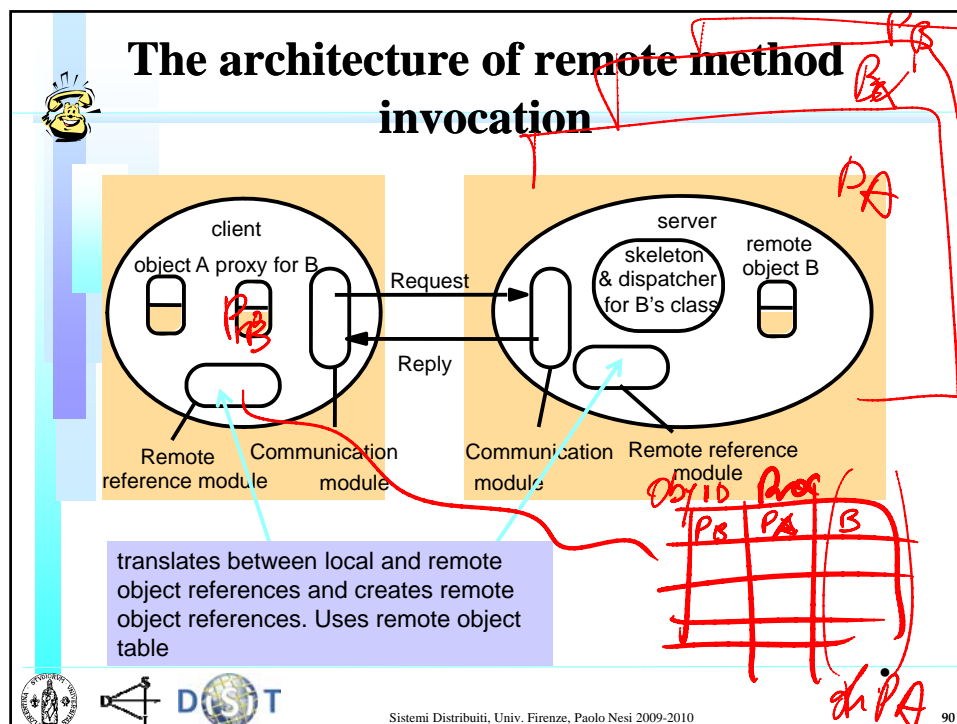
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

82











History of Distributed Object Models

- Communication Protocol Models:
 - ♣ Message passing/queuing (DCE, Distrib. Comp. Environment),
 - ♣ Request/response, (RPC, Remote Procedure Code)
 - ♣ Virtual processes: PVM (Parallel Virtual Machine), etc.
- 1980 model based on network layer (NFS, DCE, RPC)
- 1990 object-oriented RPC, to link objects
- 1993 COM (Comp. Object Model), (from OLE, Active X)
- 1996 Java
- 1997 Mary Kirtland's articles in Microsoft System Journal first sketch (COM+)
- 1997 Sun vs Microsoft over Java licensing
- 1999 Java 1.2
- 2000 Microsoft announces .net, C#

