




## Comunicazione in distribuito



**Anno accademico 2018/19**  
**Sistemi Concorrenti e Distribuiti**  
**Tullio Vardanega, [tullio.vardanega@math.unipd.it](mailto:tullio.vardanega@math.unipd.it)**

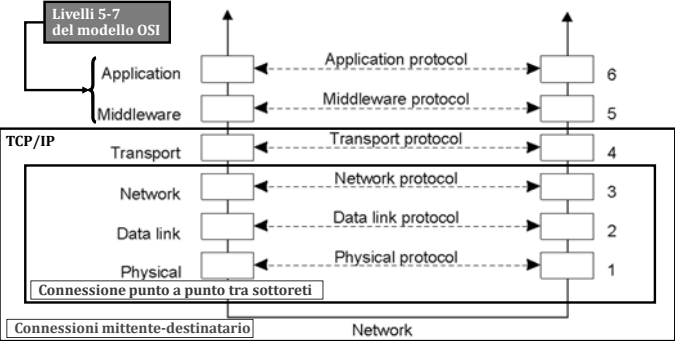
Laurea Magistrale in Informatica, Università di Padova

**1/38**



Sistemi distribuiti: comunicazione


Visione a livelli – 1



The diagram illustrates the 7 layers of the OSI model and their mapping to the TCP/IP model. On the left, layers 5-7 are grouped as 'Livelli 5-7 del modello OSI'. On the right, layers 6 and 5 are grouped as 'Middleware'. The TCP/IP model is shown as a box containing layers 4, 3, 2, and 1. A 'Network' box encompasses layers 3, 2, and 1. A 'Connessioni punto a punto tra sottoreti' box encompasses layers 3, 2, and 1. A 'Connessioni mittente-destinatario' box encompasses layers 4, 3, 2, and 1. Dashed arrows indicate protocol associations: Application protocol (6), Middleware protocol (5), Transport protocol (4), Network protocol (3), Data link protocol (2), and Physical protocol (1).

Laurea Magistrale in Informatica, Università di Padova


**3/38**



Sistemi distribuiti: comunicazione

Evoluzione di modelli


- ❑ **Remote Procedure Call (RPC)**
  - Trasparente rispetto allo scambio messaggi che realizza l'interazione cliente-servente a livello applicazione
- ❑ **Remote (Object) Method Invocation (RMI)**
  - Interazione a livello applicazione come sopra ma attraverso oggetti remoti
- ❑ **Scambio messaggi a livello middleware**
  - Con paradigmi espliciti a livello applicazione (*event-based*, *Rx*)
  - Interazione a livello HTTP (*pull*, REST) e superiore (*push*, WebSocket)
- ❑ **Streaming o comunicazioni a flusso continuo**
  - Flusso di dati che richiedono continuità temporale
  - Ma di queste non tratteremo



(POSTPONED)

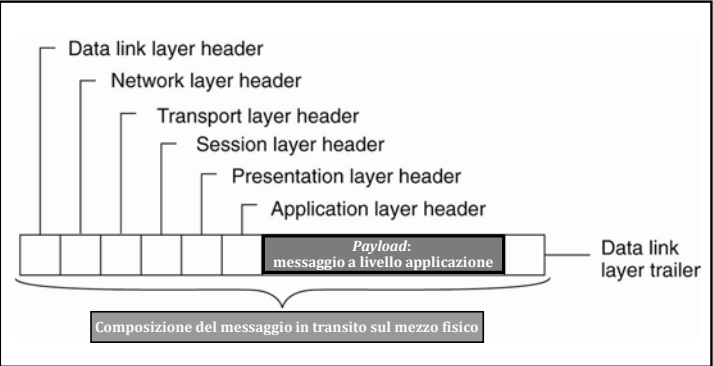
Laurea Magistrale in Informatica, Università di Padova

**2/38**



Sistemi distribuiti: comunicazione

Visione a livelli – 2



The diagram shows a message structure for transmission over a physical medium. It consists of a sequence of headers: Data link layer header, Network layer header, Transport layer header, Session layer header, Presentation layer header, and Application layer header. The payload is a 'messaggio a livello applicazione'. The message ends with a 'Data link layer trailer'. A bracket below the entire structure is labeled 'Composizione del messaggio in transito sul mezzo fisico'.

Laurea Magistrale in Informatica, Università di Padova

**4/38**

Sistemi distribuiti: comunicazione

### Visione per analogie

The diagram illustrates the evolution of communication paradigms through three levels of abstraction:

- Level 1:** Programmazione non strutturata ↔ Scambio messaggi via socket
- Level 2:** Programmazione strutturata ↔ RPC
- Level 3:** Programmazione a oggetti ↔ RMI

Arrows indicate that each level builds upon the previous one, with a final arrow pointing to a box labeled "Paradigmi più avanzati".

Essenziale per interconnessione scalabile  
 Per uso strutturato deve essere reso trasparente all'applicazione

Laurea Magistrale in Informatica, Università di Padova
5/38

Sistemi distribuiti: comunicazione

### RPC – 1

- ❑ Consentire a un processo residente su un nodo di invocare **localmente** una procedura remota, cioè residente su un altro nodo
- ❑ Chiamante sospeso durante la chiamata
  - I parametri *in viaggio* da chiamante a chiamato
  - I parametri *out viaggio* da chiamato a chiamante
- ❑ Chiamante (processo) e chiamato (procedura) **non sono consapevoli dello scambio di messaggi sottostante**
  - Un caso evidente di trasparenza

Laurea Magistrale in Informatica, Università di Padova
7/38

Sistemi distribuiti: comunicazione

### La negazione dell'astrazione

Chi definisce sintassi e semantica della comunicazione?  
 Chi ne garantisce la corretta interpretazione?

The diagram shows the sequence of operations for a Server and a Client:

- Server:** socket → bind → listen → accept → read → write → close
- Client:** socket → connect → write → read → close

A "Synchronization point" is marked at the `accept` operation on the server side. "Communication" is shown as a dashed line between the `read` of the server and the `write` of the client, and vice versa.

Tanenbaum & Van Steen, *Distributed Systems: Principles and Paradigms*, 2e, (c) 2007 Prentice-Hall, Inc.

Laurea Magistrale in Informatica, Università di Padova
6/38

Sistemi distribuiti: comunicazione

### RPC – 2

- ❑ **Chiamata di procedura locale**

Stack del processo (prima)

Variabili locali dell'unità principale del programma (**main**)

1ª posizione libera

Area libera

**Read(fd,buf,nbytes)**

Stack del processo (dopo)

Variabili locali dell'unità principale del programma (**main**)

nbytes

buf

fd

Indirizzo di ritorno

Variabili locali di **Read**

1ª posizione libera

Il linguaggio C pone i parametri sullo *stack* in ordine inverso

Ogni linguaggio ha le sue proprie convenzioni di chiamata (p.es., **cdecl**)

Laurea Magistrale in Informatica, Università di Padova
8/38



Sistemi distribuiti: comunicazione

**RPC – 3**

- ❑ **I parametri di procedura locale possono essere inviati per valore (*call-by-value*) o per riferimento (*call-by-reference*)**
  - **Quelli inviati per valore sono copiati sullo *stack* del chiamato**
    - Le modifiche apportate dal chiamato non hanno effetto sul chiamante
  - **Quelli passati per riferimento puntano alla memoria del chiamante**
    - Ogni modifica fatta dal chiamante ha effetto sul chiamato
- ❑ **La variante (*in out*), *call-by-value-result*, produce effetto sul chiamante solo al ritorno**
  - **Producendo un bel risparmio di operazioni**

Laurea Magistrale in Informatica, Università di Padova

**9/38**




Sistemi distribuiti: comunicazione

**RPC – 5**

- ❑ **Nello spazio del chiamante le procedure remote sono descritte da una procedura fittizia (*client stub*) invocabile con le convenzioni locali**
  - **Essa svolge le azioni necessarie per effettuare la chiamata remota e riceverne il ritorno**
    - Inoltro chiamata e attesa ritorno
  - **Tali azioni avvengono tramite scambio messaggi in modo trasparente all'applicazione**

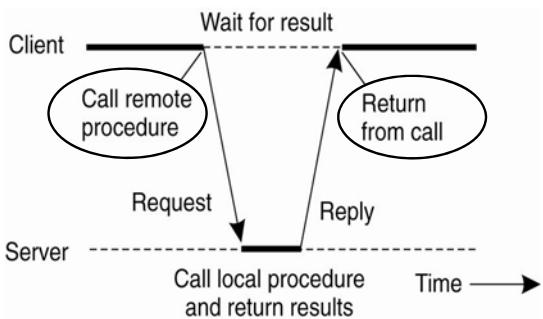
Laurea Magistrale in Informatica, Università di Padova

**11/38**



Sistemi distribuiti: comunicazione

**RPC – 4**



```

sequenceDiagram
    participant Client
    participant Server
    Note over Client: Wait for result
    Client->>Server: Request
    Note over Server: Call local procedure and return results
    Server-->>Client: Reply
    Note over Client: Return from call
    
```

Tanenbaum & Van Steen, *Distributed Systems: Principles and Paradigms*, 2e, (c) 2007 Prentice-Hall, Inc.

Laurea Magistrale in Informatica, Università di Padova

**10/38**




Sistemi distribuiti: comunicazione

**RPC – 6**

- ❑ **Nello spazio del chiamato l'arrivo del messaggio attiva una procedura fittizia detta *server stub***
  - **Essa trasforma il messaggio in chiamata locale alla procedura invocata, ne raccoglie l'esito e lo inoltra al chiamante come messaggio**
- ❑ **In questo modo chiamante e chiamato hanno reciproca trasparenza di locazione**

Laurea Magistrale in Informatica, Università di Padova

**12/38**



Sistemi distribuiti: comunicazione

**RPC – 7**

- ❑ Il *client stub* trasforma la chiamata in una sequenza di messaggi da inviare sulla rete
  - *Parameter marshaling*
  - Relativamente agevole con parametri passati per valore per i quali occorre solo assicurare trasparenza di accesso
    - Secondo le convenzioni di chiamante e chiamato
  - Molto più difficile con parametri passati per riferimento
- ❑ Il *server stub* esegue una trasformazione analoga e opposta
  - *Parameter un-marshaling*

Laurea Magistrale in Informatica, Università di Padova

**13/38**




Sistemi distribuiti: comunicazione

**RPC – 9**

- ❑ Tre aspetti chiave caratterizzano lo specifico protocollo RPC
  - Il formato dei messaggi scambiati tra gli *stub*
  - La rappresentazione dei dati attesa da chiamante e chiamato
    - *Encoding*
  - La modalità di comunicazione su rete
    - P.es.: TCP, UDP, ...

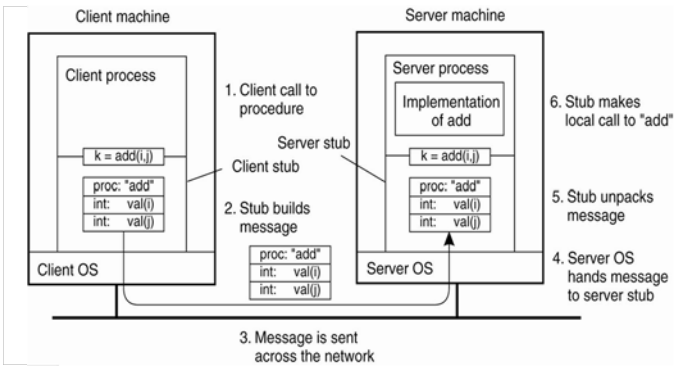
Laurea Magistrale in Informatica, Università di Padova

**15/38**



Sistemi distribuiti: comunicazione

**RPC – 8**



Laurea Magistrale in Informatica, Università di Padova

**14/38**




Sistemi distribuiti: comunicazione

**RPC – 10**

- ❑ Il servente registra il suo nodo (e porta) di residenza presso una anagrafe pubblica
- ❑ Con quell'informazione, lo *stub* del cliente innesca una connessione di rete (TCP) per inoltrare la chiamata e acquisirne l'esito
- ❑ L'azione di lato cliente è detta *binding*
- ❑ Per risparmiare porte TCP, in ascolto sul nodo del servente può trovarsi un *daemon*

Laurea Magistrale in Informatica, Università di Padova

**16/38**



**Sistemi distribuiti: comunicazione**  
**RPC – 11**

- ❑ **RPC è sincrona**
  - Può essere asincrona solo se priva di parametri *out*
- ❑ **L'eventualità di errori trasmissivi produce una tassonomia di semantiche**
  - *At-least-once*
  - *At-most-once*
  - *Exactly-once*
- ❑ **Determinata dal protocollo *request-reply* in uso tra gli *stub***

Laurea Magistrale in Informatica, Università di Padova17/38



**Sistemi distribuiti: comunicazione**  
**Semantica della comunicazione – 2**

- ❑ **Semantica *best effort***
  - Nessun meccanismo in uso
  - Il cliente non può sapere quante volte la sua richiesta sia stata eseguita
- ❑ **Semantica *at least once***
  - Il lato cliente usa RR1
  - Il lato servente niente
  - Se la risposta arriva, il cliente non sa quante volte sia stata calcolata dal servente

Laurea Magistrale in Informatica, Università di Padova19/38



**Sistemi distribuiti: comunicazione**  
**Semantica della comunicazione – 1**

- ❑ **Il protocollo *request-reply* di RPC utilizza 3 meccanismi basati sull'attesa di conferma**
  - **Lato cliente: *Request Retry – RR1***
    - Il cliente prova fino a ottenere risposta o certezza di irraggiungibilità
  - **Lato servente: *Duplicate Filter – DF***
    - Il servente scarta eventuali duplicati provenienti dallo stesso cliente
  - **Lato servente: *Result Retransmit – RR2***
    - Il servente conserva le risposte per poterle ritrasmettere senza ricalcolo
    - Fondamentale per calcolo non idempotente


Laurea Magistrale in Informatica, Università di Padova18/38



**Sistemi distribuiti: comunicazione**  
**Semantica della comunicazione – 3**

- ❑ **Semantica *at most once***
  - Il lato cliente usa RR1
  - Il lato servente usa DF e RR2
  - Se la risposta arriva, è stata calcolata una sola volta
  - La risposta non arriva solo a servente guasto
- ❑ **Semantica *exactly once***
  - Ha bisogno di meccanismi aggiuntivi per tollerare guasti nel servente
    - P.es. replicazione trasparente

Laurea Magistrale in Informatica, Università di Padova20/38




Sistemi distribuiti: comunicazione

**RMI – 1**

- ❑ **Il paradigma RPC può essere facilmente esteso al modello a oggetti distribuiti**
- ❑ **Soluzioni storiche**
  - **CORBA: Common Object Request Broker Architecture**
    - OMG – con enfasi sulla interoperabilità
  - **DCOM: Distributed Component Object Model, poi diventato .NET**
    - Microsoft – con enfasi sull'omogeneità
  - **J2EE: Java Platform, Enterprise Edition**
    - Sun Microsystems, poi acquisita da Oracle – con enfasi sull'omogeneità

Laurea Magistrale in Informatica, Università di Padova

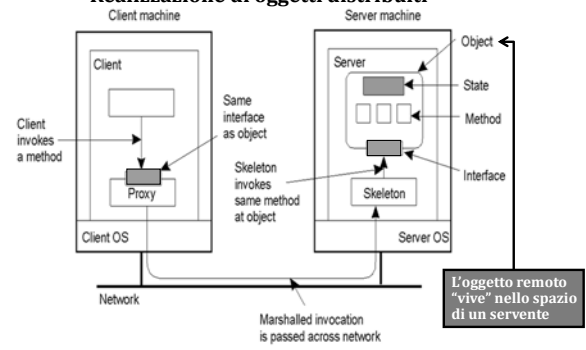
**21/38**



Sistemi distribuiti: comunicazione

**RMI – 3**

**Realizzazione di oggetti distribuiti**



Laurea Magistrale in Informatica, Università di Padova

**23/38**



Sistemi distribuiti: comunicazione

**RMI – 2**

- ❑ **La separazione logica tra interfaccia e oggetto facilita la distribuzione**
  - L'interfaccia di oggetto può essere pubblicato remotamente senza coinvolgere l'istanza corrispondente
  - Al *binding* di un cliente con un oggetto remoto, una copia dell'interfaccia del "servente" (*proxy*) viene caricata nello spazio del cliente
    - Il ruolo del *proxy* è analogo a quello del *client stub* in ambiente RPC
  - La richiesta in arrivo all'oggetto remoto viene trattata da un "agente" (*skeleton*) del cliente localmente al servente
    - Il ruolo dello *skeleton* è analogo a quello del *server stub* in ambiente RPC

Laurea Magistrale in Informatica, Università di Padova

**22/38**



Sistemi distribuiti: comunicazione

**RMI – 4**

- ❑ **Vi sono oggetti di tipo *compile-time***
  - Con realizzazione completamente determinata dal linguaggio di programmazione
    - Ambiente e protocollo d'uso noti e uniformi ma non *interoperable*
- ❑ **E oggetti di tipo *run-time***
  - Quando si vuole far apparire come oggetto ciò che non lo è in realtà
    - L'entità concreta (la sua interfaccia) viene incapsulata in un *object wrapper* che appare all'esterno come un normale oggetto remoto
    - In questo modo si ottiene *interoperability*

Laurea Magistrale in Informatica, Università di Padova

**24/38**

Sistemi distribuiti: comunicazione

**RMI – 5**

❑ **Vi sono oggetti persistenti**

- **Che continuano a esistere anche al di fuori dello spazio di indirizzamento del servernte**
  - Lo stato persistente dell'oggetto distribuito viene salvato in memoria secondaria e da lì ripristinato dai processi servernte delegati a farlo

❑ **E oggetti transitori**

- **Che cessano di esistere con la terminazione del servernte che li contiene**

❑ **Modelli RMI diversi fanno scelte diverse**

Laurea Magistrale in Informatica, Università di Padova

**25/38**

Sistemi distribuiti: comunicazione

**RMI – 8**

Laurea Magistrale in Informatica, Università di Padova

**27/38**

Sistemi distribuiti: comunicazione

**RMI – 6**

❑ **RMI offre maggiore trasparenza di RPC**

- **I riferimenti a oggetti distribuiti hanno scope globale possono essere liberamente scambiati a livello sistema**
- **Un riferimento poco scalabile usa un analogo del daemon RPC per interconnettere cliente e servernte dell'oggetto**
  - <indirizzo di rete del daemon>, identificatore del servernte
  - Cosa è questo identificatore?

❑ **Modalità di riferimento**

- **Explicit binding**
  - Il cliente si rivolge a una anagrafe che restituisce un puntatore localmente utilizzabile al proxy dell'oggetto servernte (**Java RMI**)
- **Implicit binding**
  - Il linguaggio risolve direttamente il riferimento in modo invisibile al cliente

Laurea Magistrale in Informatica, Università di Padova

**26/38**

Sistemi distribuiti: comunicazione

**Scambio messaggi: sincronizzazione**

Laurea Magistrale in Informatica, Università di Padova

**28/38**

**Sistemi distribuiti: comunicazione**  
**Scambio messaggi: persistenza**

**Laurea Magistrale in Informatica, Università di Padova** **29/38**

**Sistemi distribuiti: comunicazione**  
**Varianti scambio messaggi – 2**

**Laurea Magistrale in Informatica, Università di Padova** **31/38**

**Sistemi distribuiti: comunicazione**  
**Varianti scambio messaggi – 1**

**Laurea Magistrale in Informatica, Università di Padova** **30/38**

**Sistemi distribuiti: comunicazione**  
**Varianti scambio messaggi – 3**

**Laurea Magistrale in Informatica, Università di Padova** **32/38**



Sistemi distribuiti: comunicazione

**Scambio messaggi: realizzazione – 1**

❑ **Middleware orientato a messaggi**

- **Applicazioni distribuite comunicano tramite inserzione di messaggi in specifiche code**
  - Modello a code di messaggi
  - Eccellente supporto a comunicazioni persistenti e asincrone
  - Senza garanzia che il destinatario prelevi il messaggio dalla sua coda
- **Di immediata realizzazione tramite**
  - `Put` non bloccante (asincrona → come trattare il caso di coda piena?)
  - `Get` bloccante (sincrona rispetto alla presenza di messaggi in coda)
    - Un meccanismo di *callback* separa la coda dall'attivazione del destinatario
  - Un PO realizza la coda con metodo `Put` di tipo `P` e metodo `Get` di tipo `B`
  - Coda *proxy* @ mittente e coda *skeleton* @ destinatario

Laurea Magistrale in Informatica, Università di Padova

**33/38**

Sistemi distribuiti: comunicazione

**Scambio messaggi: realizzazione – 3**

❑ **Il *middleware* realizza una rete logica sovrapposta a quella fisica con topologia propria e distinta**

- **Overlay network con proprio servizio di routing**
  - Quei *router* conoscono la topologia della rete logica e inoltrano i messaggi del mittente alla coda del destinatario
- **Topologie complesse e variabili richiedono gestione dinamica delle corrispondenze < coda destinatario - indirizzo di rete >**
  - In totale analogia con quanto avviene nel modello IP

Laurea Magistrale in Informatica, Università di Padova

**35/38**

Sistemi distribuiti: comunicazione

**Scambio messaggi: realizzazione – 2**

Laurea Magistrale in Informatica, Università di Padova

**34/38**

Sistemi distribuiti: comunicazione

**Scambio messaggi: realizzazione – 4**

Laurea Magistrale in Informatica, Università di Padova

**36/38**



## Scambio messaggi: realizzazione – 5

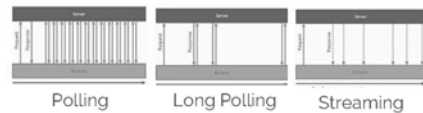
- ❑ Un *broker* fornisce trasparenza di accesso a messaggi il cui formato aderisce a standard di trasporto diversi nel suo percorso
  - Analogamente ai *gateway* della rete Internet
- ❑ La natura del *middleware* è di essere adattivo e non intrusivo rispetto all'ambiente ospite



## A volte ritornano ...

### Il protocollo HTTP

- Architettura client-server
- Richieste del client
  - GET
  - POST
  - PUT
  - DELETE
- Simulazione notifiche push



### WebSocket

- Canale full-duplex
- Latenza ridotta
- Ridotto overhead

