Principi di progettazione di sistemi distribuiti

- ✓ Rappresentazione esterna dei dati
- ✓ Marshalling e unmarshalling
- ✓ Esemplificazione con Corba e Java

Rappresentazione esterna dei dati

- **Problema**: un oggetto deve essere convertito dalla sua rappresentazione in memoria in una sequenza di byte da trasmettere in rete, *formato esterno di rappresentazione*(*)
  - una struttura dati ricca di puntatori deve essere "appiattita"
  - piattaforme diverse possono rappresentare i tipi in modo diverso
    - ✓floating point
    - ✓char (ASCII, Unicode)
    - ✓integer (big endian, little endian)

(*) il problema si verifica anche nel caso della memorizzazione dei dati

Endiannness

- **Endianness (byte order, byte sex)**
  - Quando un dato può essere rappresentato su due o più byte, non c'è un modo univoco di ordinare tali byte in memoria o in uno stream, cosicchè l'ordine diventa oggetto di convenzione
- **Esempio: intero 0x4A3B2C1D**
  - big-endian (msb first): 4A 3B 2C 1D
  - little-endian (lsb first): 1D 2C 3B 4A
  - le due modalità di rappresentazione sono equivalenti

Big-endian vs little-endian

- **Pure big-endian**
  - Sun SPARC, Motorola 68000, PowerPC 970, IBM System/360
- **Bi-endian**(*) (big-endian by default)
  - MIPS running IRIX, PA-RISC, most POWER and PowerPC systems
- **Bi-Endian**(*) (little-endian by default)
  - MIPS running Ultrix, most DEC Alpha, IA-64 running Linux
- **Pure little-endian**
  - Intel x86, AMD64, DEC VAX

(*) a seconda dell'architettura, si passa da un modo all'altro via hw e/o sw

Altri casi di endianness

- **Bit in un byte/word**
  - little-endian sembra più naturale per un intero (l'indice del bit corrisponde con il peso del bit)
  - big-endian sembra più naturale per un razionale in virgola fissa
  - in un dispositivo seriale la scelta viene fatta a livello data link
- **Formato di una data**
  - little-endian (Europa): Day Month Year
  - big-endian (ISO 8601): Year Month Day
  - middle-endian (USA): Month, Day, Year

Altri casi di endianness

- Sistema di numerazione arabo è big-endian
  - numeri da sinistra a destra
- Il sistema di numerazione inglese ed italiano è big-endian:
  - 23: twenty three
  - 23: venti tre
- Il sistema di numerazione tedesco e olandese sono big-endian con l'eccezione dei multipli di dieci
  - 23: drei und zwanzig (tre e venti)

Implicazioni di endianness

- **Portabilità dei programmi**
  - BMP memorizza i dati in little endian integer
- **Endianess nelle comunicazioni (NUXI problem)**
  - IP definisce un **network byte order** di tipo big-endian
  - Nei dispositivi seriali, endianness è definito a livello data link
- **Debugging**
  - molti considerano big-endian più intuitivo (significatività dei byte è ordinata secondo l'ordine testuale)

Approcci alla rappresentazione esterna

- I dati sono tradotti in un formato esterno convenzionale prima di essere trasmessi e convertiti nel formato locale dopo essere ricevuti

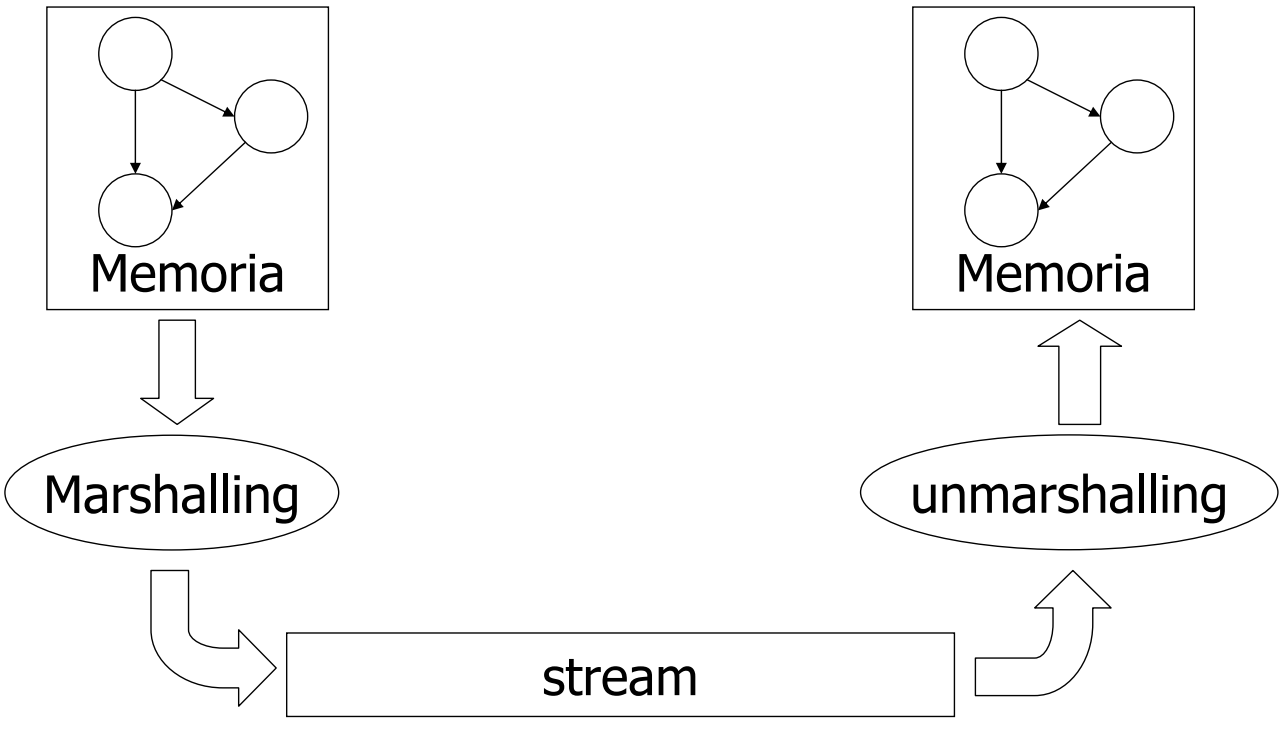
  Osservazione. Se le due piattaforme utilizzano lo stesso formato di rappresentazione, la conversione da/a formato esterno può essere omessa

- I dati sono trasmessi nel formato locale del mittente insieme con una indicazione del formato utilizzato. Se necessario, il ricevente converte nel suo formato locale

Marshalling e unmarshalling

- **Marshalling** è l'operazione che traduce i dati dal loro formato locale al formato esterno
- **Unmarshalling** è l'operazione opposta al marshalling e consiste nel tradurre i dati dal formato esterno di rappresentazione nel formato locale
- marshalling → trasmissione → ricezione → unmarshalling

Marshalling & unmarshalling



Marshalling e unmarshalling

- Il processo di marshalling/unmarshalling viene eseguito dallo strato di middleware
  - il problema deve essere risolto correttamente ed efficientemente
  - il problema è troppo complicato per essere lasciato al programmatore
- Il processo di marshalling può convertire i dati applicativi sia in uno stream di bit sia in uno di caratteri
  - lo stream di bit è preferito perché più compatto
  - HTTP utilizza lo stream di caratteri

L'approccio di Corba e di Java

- **Corba's Common Data Representation**
  - CDR definisce un formato esterno di rappresentazione sia per i tipi semplici sia per quelli strutturati che può essere utilizzato da una varietà di linguaggi di programmazione
- **Java's serialization**
  - La serializzazione definisce sia un formato esterno di rappresentazione sia un metodo per "appiattire" oggetti comunque complessi e può essere utilizzata solo nel linguaggio Java

CORBA CDR: i tipi

- **Corba definisce**
- **15 tipi primitivi** (**primitive types**)
  - short (bit), long (32-bit), unsigned short, unsigned long, float (32-bit), double (64-bit), char, boolean, octet, any
- **6 tipi composti** (**constructed types**)
  - sequence, string, array, struct, enumerated, union

CORBA CDR: i tipi semplici

- CDR definisce una rappresentazione per big-endian e little-endian
- I dati sono trasmessi nel formato del mittente che è specificato nel messaggio stesso
- Nello stream, un valore di tipo primitivo di *n*-byte è allineato ad un multiplo di n

CORBA CDR: i tipi composti

Tipo	***Rappresentazione***
sequence	<lenght(*)><elements in order> [(*) ulong]
string	<lenght(*)><chars(**) in order> [(*) ulong; (**) also wide chars]
array	<elements in order>
struct	<elements in the order of declaration of the components>
enumerated	<index of the literal declaration>
union	<type tag><value of the selected member>

CORBA CDR: esempio

```
struct Persona {
      string name;
      string place;
      long year;
};
```

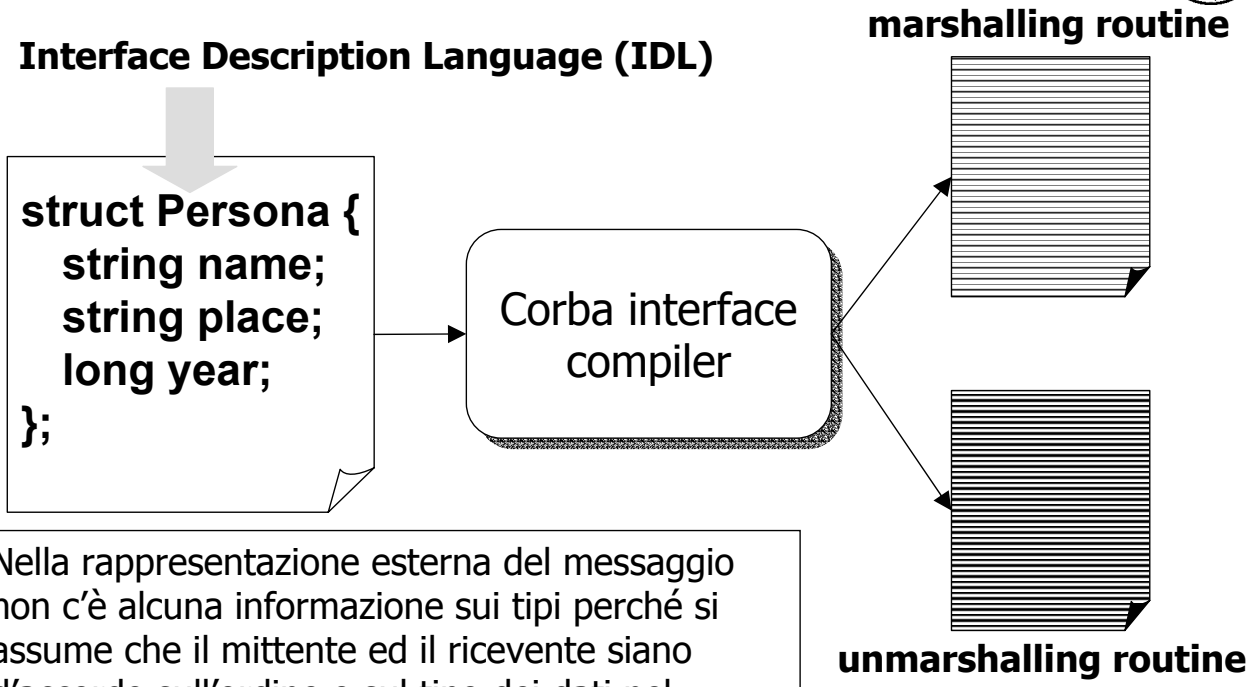
Commenti:

- **ulong** sono allineati ai multipli di 4
- nell'esempio nessuna distinzione tra big- e little-endian
- nessuna indicazione di tipo nello stream

{"Smith", "London", 1934}

index in the sequence of bytes	**stream**	**note**
0-3	5	lenght of string (ulong)
4-7	"Smit"	string
8-11	"h___"	
12-15	6	lenght of string (ulong)
16-19	"Lond"	string
20-23	"on__"	
24-27	1934	ulong

CORBA: marshalling, unmarshalling



Nella rappresentazione esterna del messaggio non c'è alcuna informazione sui tipi perché si assume che il mittente ed il ricevente siano d'accordo sull'ordine e sul tipo dei dati nel messaggio

Java serialization

- Java assume che il processo che deserializza *non conosca* a priori gli oggetti in forma serializzata, perciò nello stream devono essere inserite informazioni (*nome* e *numero di versione*) sulle classi che permettono a tale processo di caricare dinamicamente le classi necessarie
  - il numero di versione serve per rilevare modifiche alla classe
- La **reflection** permette di realizzare serializzazione e deserializzazione in modo generico senza fare uso di routine di marshalling & unmarshalling

Java serialization

- Gestione delle classi
- Quando serializza un oggetto, Java scrive sullo stream
  - la classe dell'oggetto (nome e numero di versione);
  - il numero delle variabili istanza;
  - per ciascuna variabile istanza il tipo(*) ed il nome;
  - (*) la procedura è ricorsiva.
  - Ad ogni classe è associato un'**handle** per evitare che ogni classe sia scritta per più di una volta sullo stream

Java serialization: tipi primitivi

- I tipi primitivi sono convertiti in un formato esterno standard
- Le stringhe ed i caratteri sono convertiti nel formato Universal Transfer Format (UTF)
  - ASCII chars non sono modificati
  - Unicode chars sono rappresentati su più byte
  - Le stringhe sono precedute dal numero di caratteri che le compongono
  - operazione **ObjectOutputStream.writeUTF**

Java serialization: example

```
public class Persona implements Serializable {
    private string name;
    private string place;
    private long year;
    public Persona(String aName, String aPlace, int aYear) {
        name = aname;
        place = aPlace;
        year = aYear;
    }
    // other methods
};
```

```
Persona p = New Persona("Smith", "London", 1934);
```


Java serialization: example

<table>
<tr><th colspan="4">valori serializzati</th><th>note</th></tr>
<tr><td>Person</td><td colspan="2">vn1 (version number)</td><td>h0 (handle)</td><td>nome, numero di versione e handle della classe</td></tr>
<tr><td>3</td><td>int year</td><td>java.lang. String name</td><td>java.lang. String place</td><td>numero, tipo e nome delle variabili istanza</td></tr>
<tr><td>1934</td><td>5 "Smith"</td><td colspan="2">6 "London"</td><td>valore delle variabili istanza</td></tr>
</table>