

# Sistemas Distribuídos

## Aula III – Comunicação I

---

PROF. DR. RAFAEL TEIXEIRA SOUSA

# Comunicação entre Processos

---

“Coração” de qualquer Sistema Distribuído

Como processos em diferentes máquinas trocam informações?

Desejável obter modelos onde a complexidade da comunicação seja transparente para o desenvolvedor

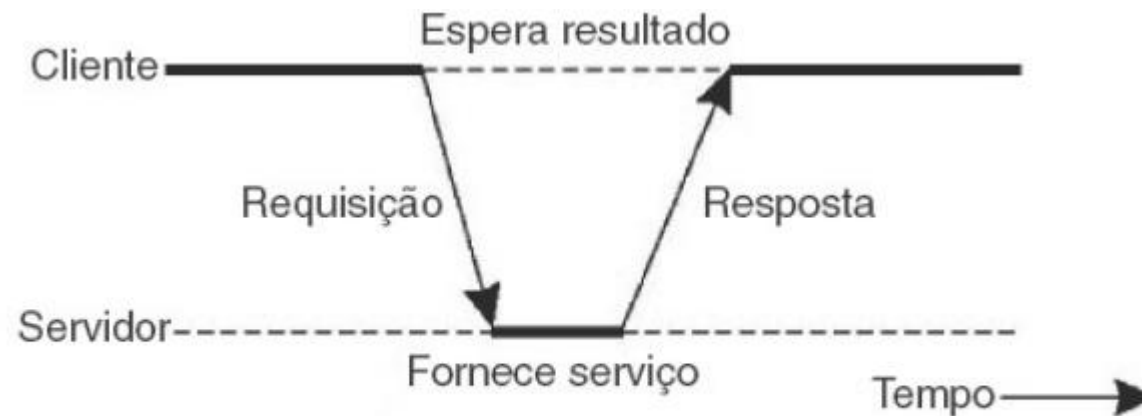
# Modelo Cliente-Servidor

---

Participantes são divididos em

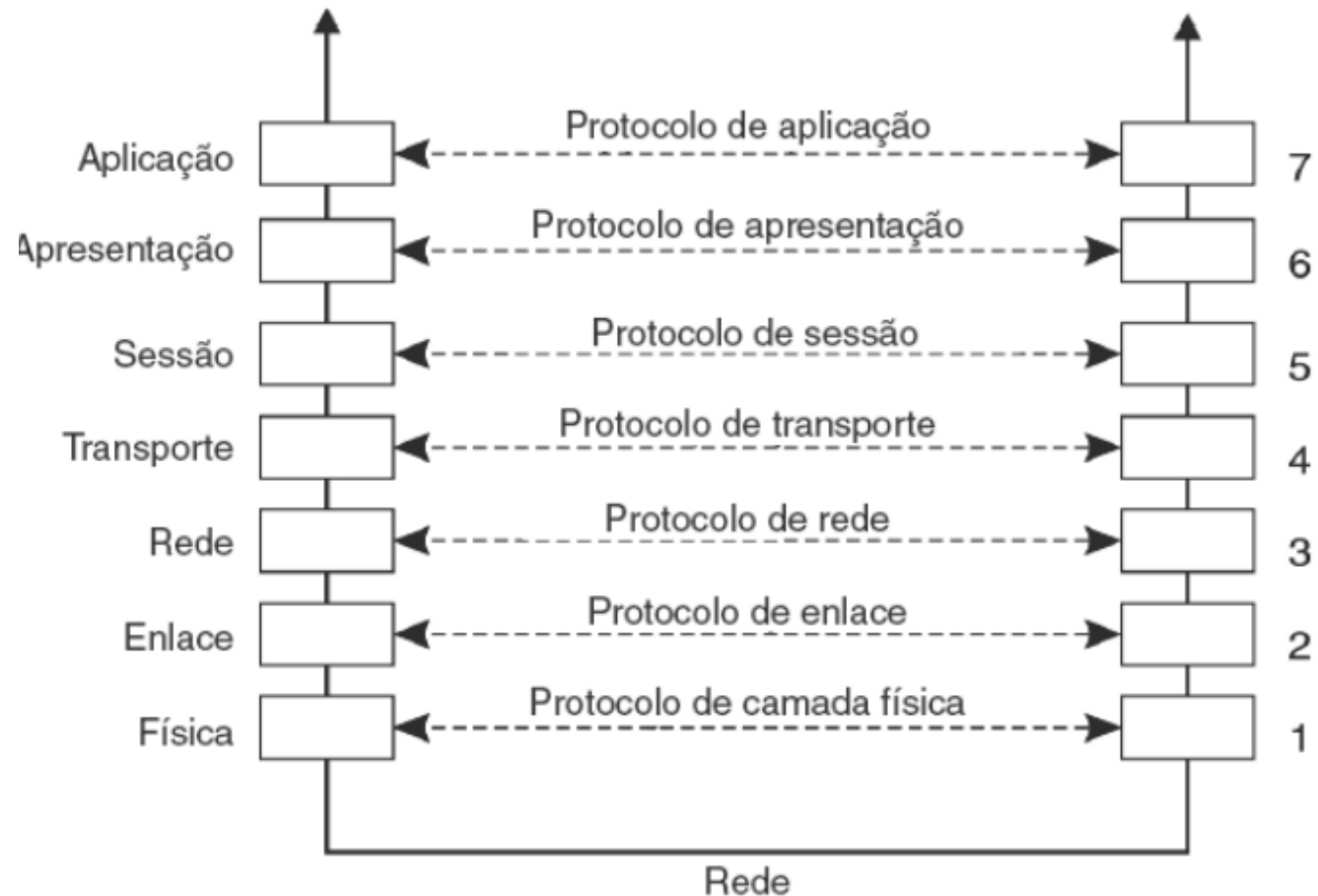
- Servidores
- Clientes

Comportamento requisição-resposta

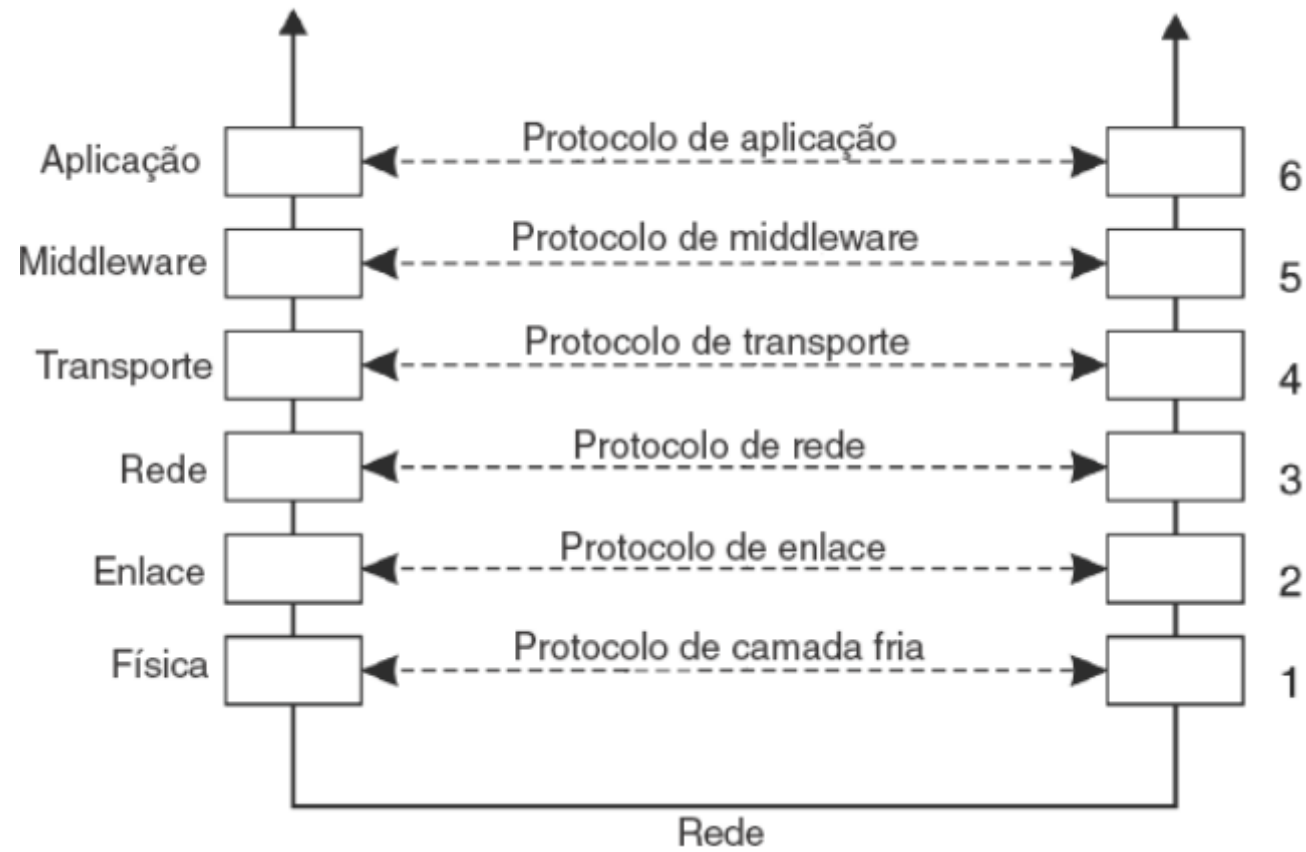


**Figura 2.3** Interação geral entre um cliente e um servidor.

# Protocolos em Camadas



# Camada de Middleware



# Sockets

---

Permite comunicação full-duplex entre processos cliente/servidor

## Combinação

- Um endereço IP
- Um nº de uma porta do protocolo de transporte(TCP/UDP)

Informações são string de bytes, sem significado aparente

# Sockets

---

Não existe a transparência de distribuição

- Toda a comunicação está explícita
- Funções mais sofisticadas devem ser feitas na camada de aplicação

**Por que não oferecer comunicação de alto nível,  
independente da aplicação?**

# Solução

---

## Middleware de comunicação

### Tipos:

- Chamadas de Procedimento Remoto
- Comunicação orientada a Mensagens
- Comunicação orientada a fluxo



# Tipos de Comunicação - Persistência

---

## Persistente

- Mensagem é armazenada pelo middleware de comunicação durante o tempo que for necessário para entregá-la ao receptor

## Transiente

- Mensagem é armazenada somente durante o tempo em que a aplicação remetente e a aplicação receptora estiverem executando

# Tipos de Comunicação - Sincronização

---

## Assíncrona

- Remetente continua sua execução imediatamente após ter apresentado sua mensagem para transmissão

## Síncrona

- Remetente é bloqueado até saber que sua requisição foi aceita

## Pontos de Sincronização

- Até que o middleware avise
- Até que a requisição chegue no receptor
- Até que o receptor responda

# Tipos de Comunicação - Granularidade

---

## Discreta

- Partes se comunicam por mensagens e cada mensagem forma uma unidade de informação completa

## Fluxo

- Várias mensagens que estão relacionadas uma com as outras
- Relacionadas pela ordem ou pela relação temporal

# RPC

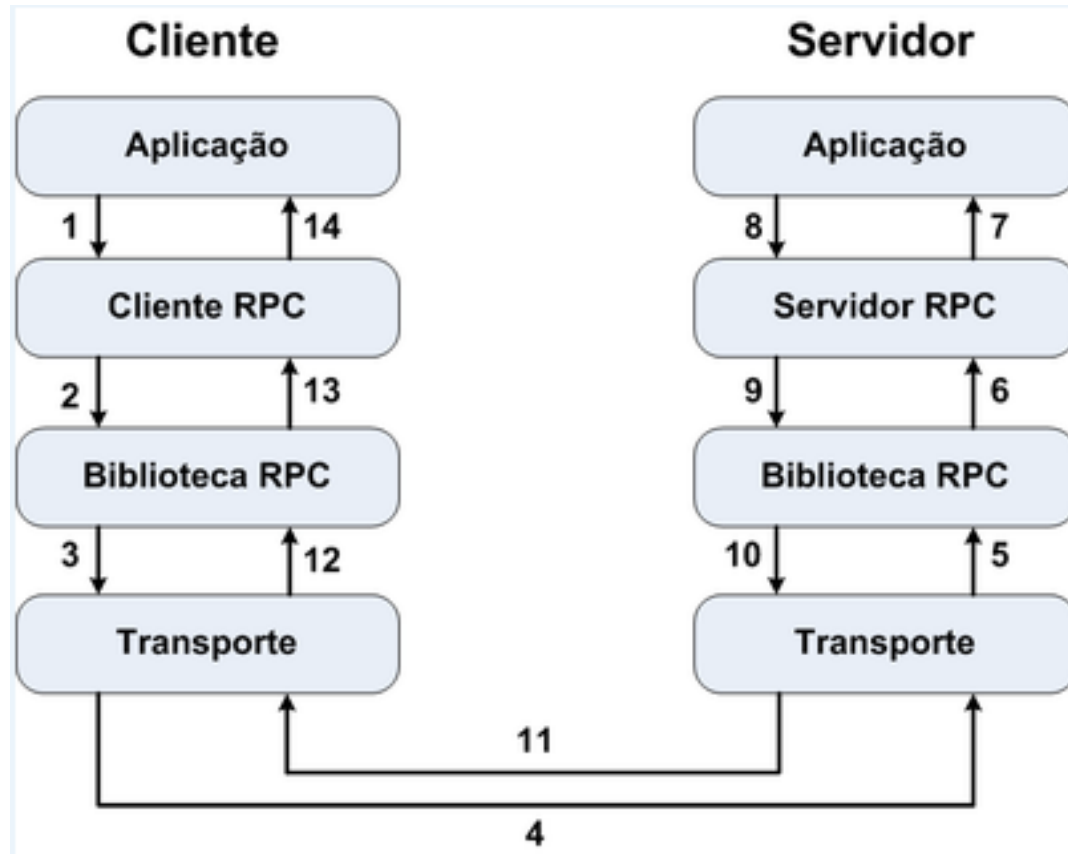
---

Permite a processos chamar procedimentos localizados em outras máquinas

A ideia é fazer com que essa chamada se seja transparente para o desenvolvedor

Desenvolvedor não precisa mais se preocupar mais com detalhes de implementação de rede

# RPC



# RPC - Problema

---

Arquiteturas das máquinas podem ser diferentes

Espaço de endereçamento diferentes

Passagem de parâmetro

# Modelo

---

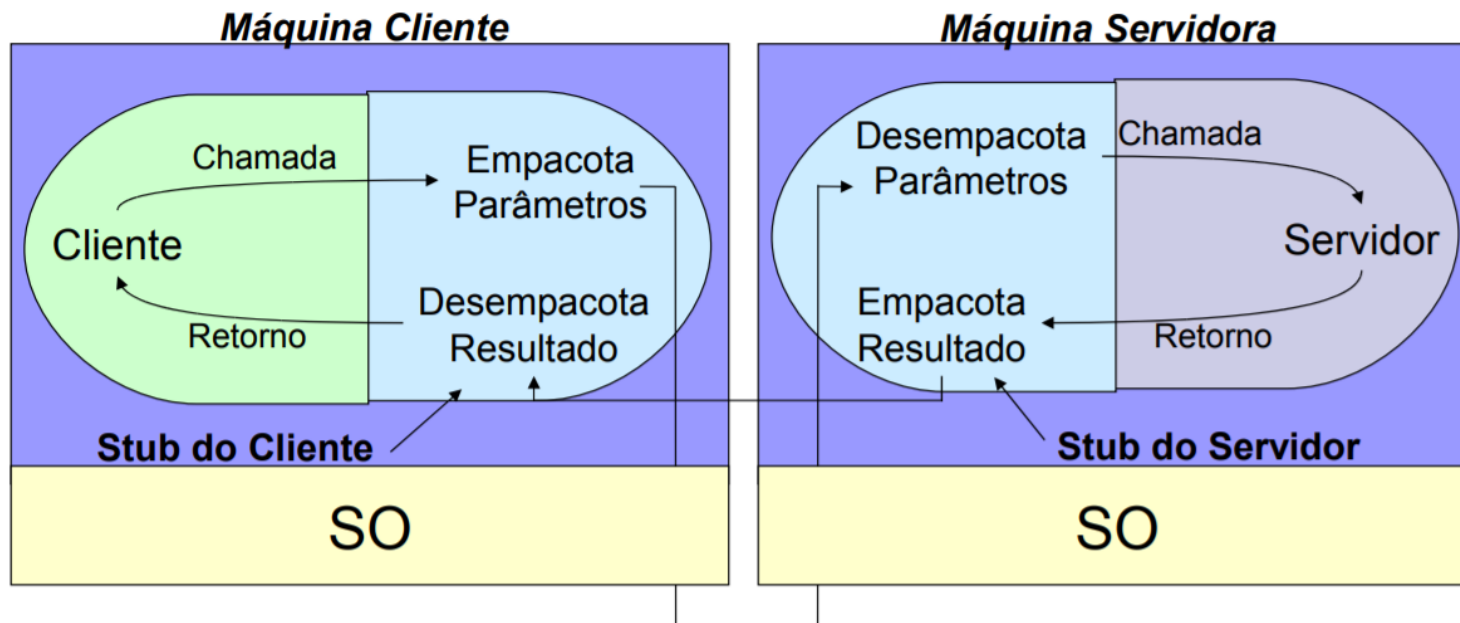
Uma chamada remota difere da chamada local em alguns pontos

- Tratamento de erros
- Variáveis globais
- Desempenho
- Autenticação

# RPC - Transparência

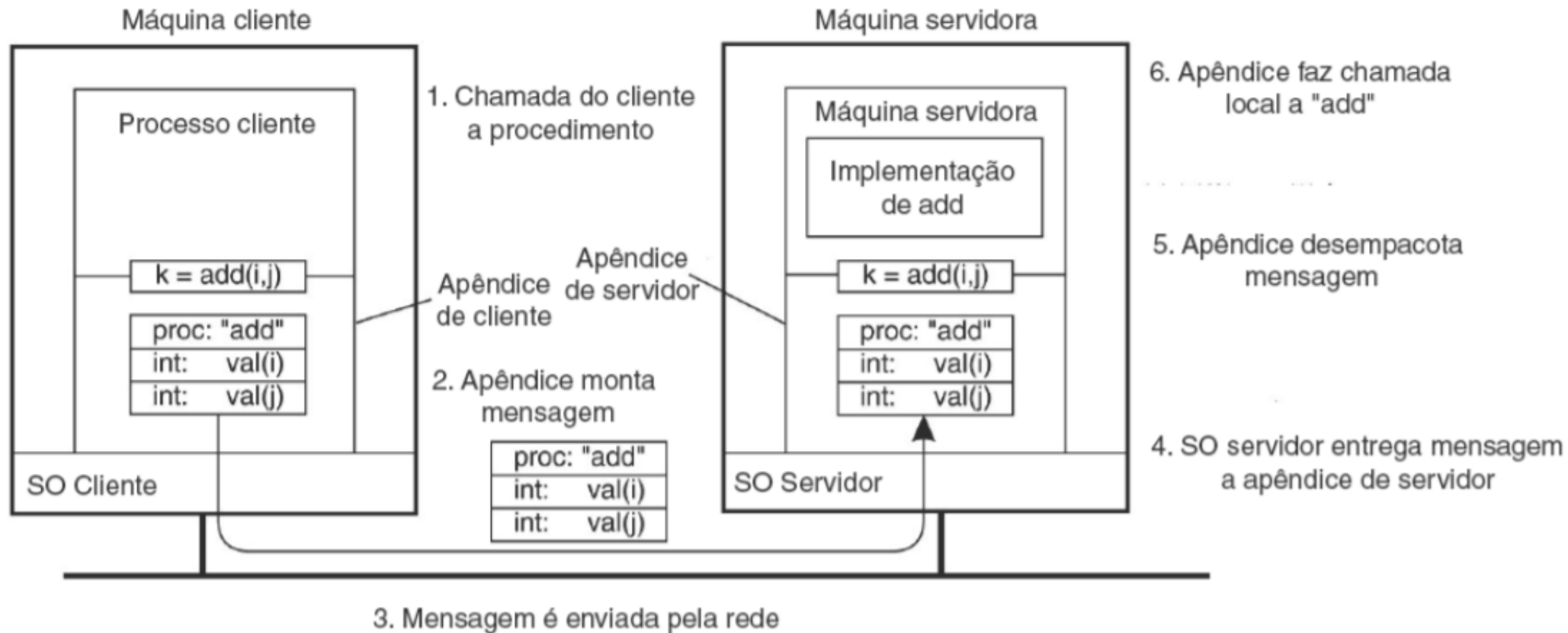
Alcançada através do uso dos stubs

- Stub Cliente
- Stub Servidor ou Skeleton





# Passagem de Parâmetros



# Passagem de parâmetros por referência

---

Como são passados os ponteiros se só fazem sentido dentro do espaço de endereçamento do processo?

Solução-> Copiar/Restaurar

- Copia o vetor para a mensagem e envia ao servidor
- Stub do servidor, chama o servidor com um ponteiro para este vetor
- Modificação feita pelo servidor é armazenada diretamente no vetor que está no stub
  - Ao enviar o vetor de volta ao stub do cliente, o vetor é copiado de volta ao cliente

# XDR

---

## External Data Representation

Empacota e desempacota dados em uma arquitetura de maneira independente para que o dados sejam transferidos entre sistemas de computadores heterogêneos

# Tipos XDR

---

Tipo de Dado	Tam.	Descrição
Int	32 bits	Inteiro binário sinalizado de 32 bits
Unsigned int	32 bits	Inteiro binário não sinalizado de 32 bits
Bool	32 bits	Valor booleano (false ou true), representado por 0 ou 1
Enum	Arbitrário	Tipo de enumeração com valores definidos por inteiros (ex.: red=1, blue=2)
Hyper	64 bits	Inteiro sinalizado de 64 bits
Unsigned hyper	64 bits	Inteiro não sinalizado de 64 bits
Float	32 bits	Numero de ponto flutuante com precisão simples
Doublé	64 bits	Número de ponto flutuante com precisão dupla
Opaque	Arbitrário	Dado não convertido, i.e, dado na representação nativa do remetente
String	Arbitrário	String de caracteres
Fixed array	Arbitrário	Um array de tamanho fixo de qualquer outro tipo de dado
Counted array	Arbitrário	Um array no qual o tipo tem um limite superior fixo, mas arrays individuais podem variar no tamanho
Structure	Arbitrário	Um dado agregado, como uma struct da linguagem C
Discriminated union	Arbitrário	Uma estrutura de dados que permite uma de várias formas alternativas, como uma union da linguagem C.
Void	0	Usado se nenhum dado está presente onde um item de dado é opcional (ex.: dentro de uma structure)
Symbolic Constant	Arbitrário	Uma constante simbólica e um valor associado
Optional data	Arbitrário	Permite zero ou uma ocorrência de um item

# Funções XDR

---

Função	Argumentos	Tipo de dado convertido
Xdr_bool	Xdrs, ptrbool	Booleano (int em C)
Xdr_bytes	Xdrs, ptrstr, strsize, maxsize	Byte String contado
Xdr_char	Xdrs, ptrchar	Caracter
Xdr_double	Xdrs, ptrdouble	Ponto flutuante de precisão dupla
Xdr_enum	Xdrs, ptrint	Variável de tipo de dado enumerado (um <i>int</i> em C)
Xdr_float	Xdrs, ptrfloat	Ponto flutuante de precisão simples
Xdr_int	Xdrs, ip	Inteiro de 32 bits
Xdr_long	Xdrs, ptrlong	Inteiro de 64 bits
Xdr_opaque	Xdrsptrchar, count	Bytes enviados sem uma conversão
Xdr_pointer	Xdrs, ptrobj, objsize, xdrobj	Um ponteiro (usado em listas encadeadas ou árvores)
Xdr_short	Xdrs	Inteiro de 16 bits
Xdr_string	Xdrs, ptrstr, maxsize	Uma string em linguagem C
Xdr_u_char	Xdrs, ptruchar	Inteiro não sinalizado de 8 bits
Xdr_u_int	Xdrs, ptrint	Inteiro não sinalizado de 32 bits
Xdr_u_long	Xdrs, ptrulong	Inteiro não sinalizado de 64 bits
Xdr_u_short	Xdrs, ptrushort	Inteiro não sinalizado de 16 bits
Xdr_union	Xdrsptrdiscrim, ptrunion, choicefcn, default	União discriminada
Xdr_vector	Xdrs, ptrarray, size, elemsize, elemproc	Array de tamanho fixo
Xdr_void	--	Não é uma conversão (uso: denotar a parte vazia de uma estrutura de dados)

# XML - JSON

---

## XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

## JSON

```
{ "empinfo" :
  {
    "employees" : [
      {
        "name" : "James Kirk",
        "age" : 40,
      },
      {
        "name" : "Jean-Luc Picard",
        "age" : 45,
      },
      {
        "name" : "Wesley Crusher",
        "age" : 27,
      }
    ]
  }
}
```