1. INTRODUÇÃO

Sistema distribuído –> sistema que corre em varias maquinas independentes e potencialmente localizadas em pontos geograficamente. Podendo ser heterogéneas e administradas por organizações diferentes. Partilham estado através de redes de interligação.

Plataformas de suporte a SDist

- ➔ **Problemas:** comunicação exclusiva por mensagem
  : modelo de faltas mais complexos
  : conhecimento parcial do estado do sistema
  : distribuição do sistema
  : segurança
  : heterogeneidade
  : desempenho
- ➔ **Vantagens:** adequado a repartição geográfica
  : modularidade
  : extensibilidade
  : maior disponibilidade
  : melhor desempenho
  : custo mais reduzido

2. PROGRAMAÇÃO DA COMUNICAÇÃO

Comunicação: interacção entre um processo *emissor* e um processo *receptor*.
Canal: abstracção dos mecanismos de transporte que suportam a transferência de mensagens.
Porto: extremidade de um canal.
Protocolo: definição das mensagens e do respectivo encadeamento que permite a comunicação.

Concretização de canal

Tipo de canal: com ligação (fiável, bidireccional e garante sequencialidade)
: sem ligação (não fiável)
: com capacidade de armazenamento de fila de mensagens

Portos

Possuem dois tipos de identificadores (o do modelo computacional e o do protocolo de transporte)

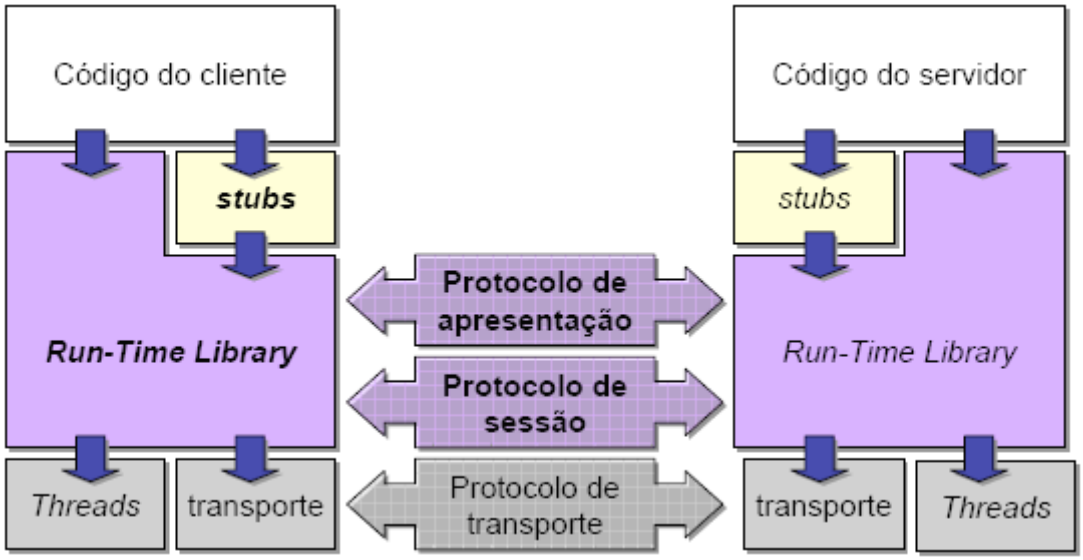
Semântica de envio: assíncrona
síncrona (garantia de entrega)
Pedido Resposta (RPC)

Semântica de recepção: ler de forma não bloqueante
bloquear a espera de mensagem
bloquear a espera de múltiplos eventos

3.CHAMADAS DE PROCEDIMENTOS REMOTOS

Benefícios do RPC:

Adequado ao fluxo de execução das aplicações.
Simplifica tarefas fastidiosas e delicadas.
Simplifica a divulgação de serviços.
Esconde diversos detalhes do transporte.



RPC versus mensagens

Positivo	Negativo
▪ A interface do serviço encontra-se claramente especificada e não é apenas um conjunto de mensagens ▪Mecanismo de estabelecimento da ligação entre o cliente e o servidor automático ▪ As funções do cliente e do servidor são consistentes, o sistema garante que são correctamente emparelhadas ▪ O modelo de invocação de uma função e respectiva sincronização simplificam a programação ▪ Os dados são automaticamente codificados e descodificados resolvendo o problema da heterogeneidade ▪ As Excepções são um bom mecanismo para tratamento do erro	•O servidor deve estar a executar-se para o RPC poder ter sucesso •A sincronização pode dar origem a estrangulamentos • Só são bem suportadas as interacções 1-para-1 (ou seja não suporta difusão) •Existem mais níveis de software que implicam maior overhead na execução

Passos da interacção RPC

- ➔ **Cliente:** estabelecimento da sessão (*binding*) [identificação do servidor, localização, estabelecimento de um canal de transporte, autenticação]
  : chamada de procedimento remoto.
  : terminação da sessão.

- ➔ **Servidor:** registo (nome do porto de transporte e registo da identificação num servidor de nomes)
  : esperar por pedidos de criação de sessões
  : esperar por invocações de procedimentos
  : terminar sessão.

Infra-estrutura de suporte do RPC

- ➔ **No desenvolvimento:** linguagem de especificação de interfaces (*IDL*)
  : compilador de *IDL* (produz os *stubs*)

- ➔ **Na execução:** serviço de nomes
  : biblioteca de suporte à execução do RPC (suporta o registo de serviços, o binding, protocolo de execução de RPC e controlo global da interacção cliente-servidor.

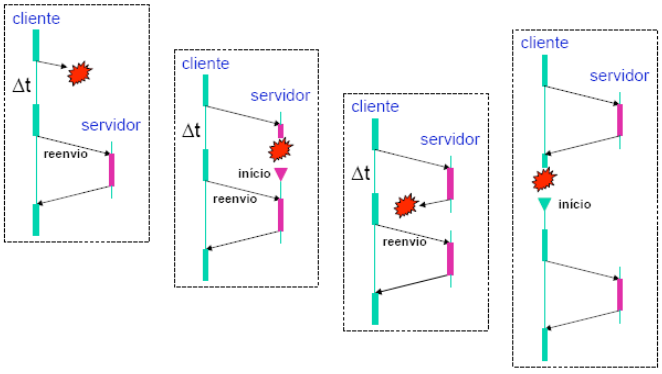
RPC IDL – linguagem própria para a especificação de interfaces para RPC (é declarativa e depende dos protocolos de ligação e do tratamento da heterogeneidade)
- permite definir tipos de dados, protótipos de funções, interfaces, identificadores.

Entraves à transparência do RPC

- ➔ **IDL** (diferente das linguagem de programação usadas)
- ➔ **Passagem por parâmetros** (semânticas não suportadas pelo RPC)
- ➔ **Execução do procedimento remoto** (fluxos de execução complexo, tolerância a faltas e notificação das mesmas)
- ➔ **Desempenho** (depende da infra-estrutura de comunicação).

Semânticas de execução

Idealmente a semântica de execução do RPC seria a de um processo local, no entanto o modelo de faltas é diferente do de um processo local.



➔ **Tratamento de faltas –** em geral não são contempladas falhas de maquinas ou de servidores, as faltas são consideradas intermitentes e que as maquinas são *fail-silent*.

➔ **Semânticas –** *talvez* (o *stub* retorna erro se não obtiver resposta, pode ter executado ou não o pedido)

pelo-menos-uma-vez (o *stub* repete o pedido ate receber uma resposta – funções *idempotentes*)

no-maximo-uma-vez (o protocolo de controlo identifica pedidos para detectar as repetições no servidor, mantém estado)

exactamente-uma-vez (a actividade é controlada por monitores transaccionais)

HETEROGENEIDADE DE TIPOS DE DADOS

- Nos sistemas distribuídos a heterogeneidade é a regra
- Os formatos de dados são diferentes
  - Nos processadores (ex.: little endian, big endian, apontadores)
  - Nas estruturas de dados geradas pelos compiladores
  - Nos sistemas de armazenamento
  - Nos sistemas de codificação
- As mensagens entre as máquinas apenas podem enviar tipos básicos: caracteres, inteiros, reais, etc.
- As redes transmitem bytes entre as máquinas. Todos os restantes tipos têm de ser serializados para os tipos básicos

No modelo OSI a camada de Apresentação era responsável por resolver este problema. No caso dos RPC's este problema é resolvido por técnicas de compilação.

Destas duas soluções a heterogeneidade da comunicação passou a ser resolvida em tempo de execução.

⇨ **Estrutura das mensagens** pode ser **implícita** (apenas contem os dados a transmitir) ou **explícita** (auto-descritiva).

⇨ **Politica de conversão de dados** pode existir uma representação para a qual todos convertem (**canónica**) ou o então **o-receptor-converte**.

Fluxos de execução

1. **Fluxos de execução simples** um pedido de cada vez ou vários pedidos em paralelo (biblioteca de RPC tem de suportar paralelismo).
2. **Fluxos de execução complexos** chamadas em ricochete, chamadas assíncronas.
3. **Fluxos de execução alternativos** chamadas sem retorno, RPC's locais, RPC em difusão.

4.REMOTE METHOD INOCATION – RMI

Nome dado ao RPC num sistema e objectos (a IDL é orientada aos objectos, C++, Java).

CORBA

O seu modelo conceptual é uma síntese do modelo cliente-servidor e as arquitecturas de objectos.

A sua estrutura inicial era composta por:

Object Request Broker – serviço que auxilia a invocação de métodos remotos. O seu papel é localizar o objecto, activa-lo (se necessário) e enviar o pedido do cliente ao objecto.

Serviços de Suporte ao ciclo de vida dos objectos

no corba 2.0 foram acrescentados:

IDL – linguagem OO com suporte a herança.

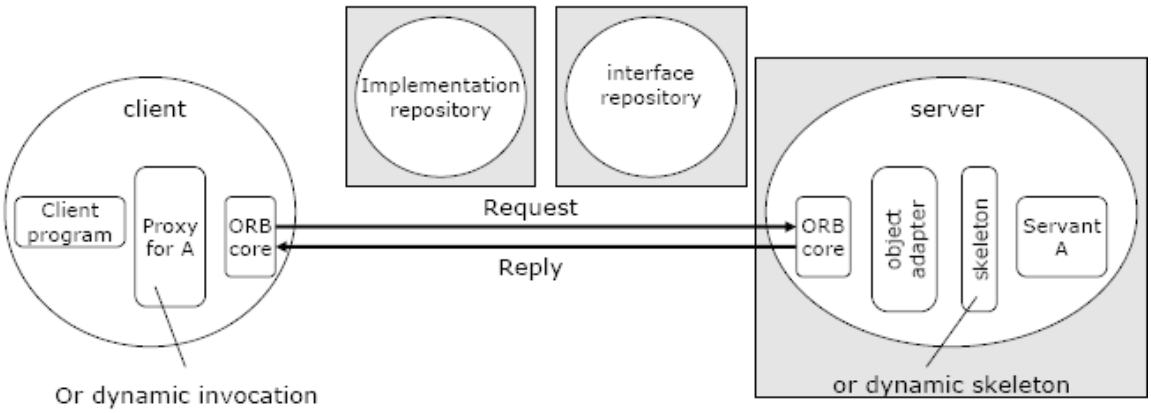
General Inter Orb Protocol (orb é o runtime da invocação remota)

Internal Inter Orb Protocol

Modelo de objectos – cada objecto implementa uma interface descrita na IDL Corba.

Cada objecto Corba pode ser invocado remotamente através de uma referência remota (os objectos residem no servidor).

Apenas os argumentos de tipos primários são passados por valor.



Implementation Repository – armazena as tabelas que relaciona os object adapters (rotina de despacho que recebe as mensagens e invoca os stubs apropriados) e os pathname dos ficheiros que contem a implementação dos objectos.

Interface Repository – da informação sobre as interfaces registadas.

Invocação – por omissão é usada a semântica *no-maximo-uma-vez*, e como politica de conversão de dados é utilizada *o-receptor-converte*.

JAVA RMI

Utilizado para RPC's entre objectos JAVA distribuídos.

Pressupõe que se usa um ambiente JAVA de linguagem única.

A passagem de objectos depende da sua localização, se o objecto for local é passado por valor (tem que ser serializável), caso contrario é passado por referência.

5. WEB SERVICES

Motivação

Protocolo muito simples para garantir a interoperação entre plataformas de múltiplos fabricantes
Resolver todo o tipo de heterogeneidade de dados e informação com XML
Mensagens codificadas em texto para passar através das firewalls
Permitir utilizar RPC ou MOM sistemas de comunicação síncronos e assíncronos
Usar de forma directa o HTTP e HTTPS como protocolos de transferência de informação
Usar URL e URI como referências remotas para objectos
Permitir a transferência de todo o tipo de informação desde estruturas de dados a documentos estruturados e informação multimédia.
Eliminar a distinção de sistemas para transferência de documentos e sistemas para transferência de dados

Arquitectura dos WebServices

Serviço de directório para registo e pesquisa dos serviços **–> UDDI**
Protocolo de pedido-resposta para invocação de serviços –> **SOAP**
Uma especificação da interface do serviço –> **WSDL**

XML – eXtensible markup language

-> O XML pretende resolver o problema de heterogeneidade entre dados, utilizado um formato canónico e baseia-se numa estrutura explícita (utilizando tags como o HTML).

Sintaxe – o XML apresenta regras simples mas rígidas:

1. Existe um elemento raiz único;
2. Todos os elementos têm de fechar;
3. Fecham pela ordem inversa em que abrem;
4. os nomes são case sensitive;

Benefícios do XML

Tecnologia não proprietária
Independente das plataformas – o formato é texto
Compatível com o HTTP – o XML tem uma sintaxe mais restritiva que o HTML, mas pode ser usado como o HTML pelo que passa pelas firewalls
Internacional – formato texto que usa UTF-8 ou UTF-16 para representar os caracteres
Extensível – Novas tags podem ser adicionadas por necessidade de extensão. Para evitar duplicações existem name spaces a que um documento pode ser associado.
A interpretação das mensagens depende de tags e não de posição do campo na mensagem
Permite a transformação automática dos dados - As transformações são especificadas usando XML Stylesheet Language Transformatiom (XSLT)
Auto definida – A estrutura de um documento XML tem uma meta-descrição com base Document Type Definition (DTD) ou actualmente com XML schema
Ferramentas genéricas – ferramentas Open Source em Java e ferramentas já existentes para SGML.

SOAP – simple object access protocol

Características:

Protocolo de comunicação distribuída que permite o envio de qualquer tipo de informação.

Define o protocolo de pedido resposta

O protocolo de representação de dados é baseado em XML

Referencias remotas baseadas em URI.

Protocolo extensível.

No SOAP toda a informação está incluída dentro de uma mensagem o envelope do SOAP.

A mensagem tem um ou mais cabeçalhos (headers) e um corpo (body).

Extensões às mensagens são os SOAP headers

A heterogeneidade é tratada pelo XML assim como convenções para representar tipos de dados abstractos ou tipos de dados complexos

Uma interacções do tipo pedido - resposta

Um mecanismo de ligação entre as mensagens SOAP e o protocolo HTTP, o mais usado na Internet, contudo o SOAP pode utilizar outros protocolos de transporte como SMTP

Um mecanismo para tratar as faltas – SOAP fault

Na base estão os protocolos de comunicação que podem ser usados: HTTP, SMTP, FTP, ou API de comunicação como os sockets

A segurança pode aparecer a qualquer nível, por exemplo, utilizar SSL na comunicação ou assinaturas digitais nos headers.

WSDL – web service definition language

`port Type` – Descreve a interface abstracta de um Web service.

- Atenção porque o termo "port" é usado com um sentido totalmente diferente dos sockets, semelhante a interface de Java

`message` – assinatura das operações descrevendo o nome e os parâmetros da operação

`types` – colecção de todos os tipos de dados usados na especificação.

Estes elementos são reutilizáveis porque definem entidades abstractas e não a concretização de um serviço

UDDI - Universal Description Discovery & Integration

Definição de um conjunto de serviços que suportam a descrição e a localização de:

- Entidades que disponibilizam Web Services (empresas, organizações)
- Os Web Services disponibilizados
- As interfaces que devem ser utilizadas para aceder aos Web Services

Baseada em standards Web: HTTP, XML, XML Schema, SOAP

Norma definida por um consórcio alargado: Accenture, Ariba, Commerce One, Fujitsu, HP, i2 Technologies, Intel, IBM, Microsoft, Oracle, SAP, Sun e Verisign

Versão actual: UDDI Version 3.0, 19 Jul 200

JAX-RPC

Java API for XML-based Remote Procedure Call

Em JAX-RPC, uma chamada remota de procedimento é efectuada utilizando o protocolo SOAP (que define a estrutura e regras de representação da informação).

A API do JAX-RPC esconde a complexidade da utilização de SOAP do programador.

No servidor, o programador especifica os procedimentos remotos definindo uma interface em Java e criando uma classe que implemente esta interface.

No cliente, o programador cria uma proxy (objecto local que representa o serviço) que é invocado para executar os métodos.

Um cliente JAX-RPC pode aceder a Web services definidos noutras plataformas (devido à utilização de HTTP, SOAP e WSDL).

6. GESTÃO DE NOMES

⇨ Associamos nomes a objectos para:

1. Identifica-los
2. Localiza-los
3. Partilha-los
4. Simplificar a interface com o cliente
5. Simplificar a gestão do sistema

Conceitos Base

Identificador => mecanismo de discriminação de um objecto, atribuído por uma autoridade (se permitir localiza-lo chama-se de endereço).

Nome => mecanismo de discrição de um objecto, atribuído por um humano (normalmente permite obter um identificador para um objecto).

Autoridade => gere o objecto.

Espaço de nomes => conjunto de regras que define um universo de nomes admissíveis.

Contexto => domínio onde se considera válido um determinado espaço de nomes

Directório => tabela que num contexto descreve as associações entre nomes e objectos.

Propriedades dos nomes

Unicidade referencial => num determinado contexto um nome só pode estar associado a um objecto.

Âmbito => um nome tem o mesmo significado em todos os contextos onde o espaço de nomes é válido (**GLOBAL**), c.c. é **LOCAL**.

Homogeneidade => formado por uma única componente (c.c. é heterogéneo)

Pureza => um nome é **puro** se o algoritmo de resolução não o utilizar para localizar o objecto a si associado.

Serviços de Directório

Os serviços de nomes tinham por objectivo efectuar a tradução de nomes em identificadores de objectos

A sua estrutura era constituída por pares <nome, atributo>

Serviços mais complexos podem armazenar relações entre nomes e múltiplos atributos e permitir a pesquisa por atributos

Os primeiros são normalmente designados servidores de nomes e os segundos por serviços de directórios.

Permite genericamente dois tipos de serviços de procura:

- White-pages: capacidade de procura por nome
- Yellow-pages: capacidade de procura por conteúdo semântico associado aos nomes

Um directório é composto por **esquema** (mapa lógico de base de dados), **classes**, **atributos**, **valores** e **objecto.**

Arquitectura de Serviço de Nomes e Directório

Um serviço de nomes permite **o registo de associações, a distribuição das associações, resolução de nomes** e **resolução inversa.**