Objetos Distribuídos

CORBA

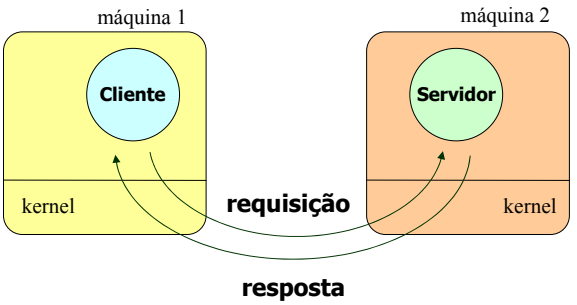
Sumário...

- Comunicação entre processos
  - Sockets
  - RPC
  - RMI
- Arquitetura OMA
- Vantagens
- IDL

Sumário...

- Arquitetura CORBA
- Interoperabilidade
- Processo de implementação CORBA
- Serviços CORBA:
  - Serviço de nomes
  - Serviço de eventos
- Considerações finais

Comunicação entre processos



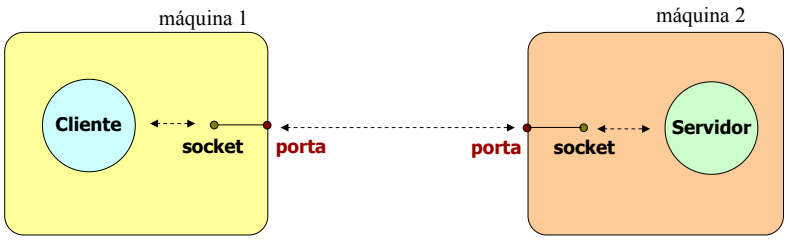
Sockets

- API da camada de transporte
- Domínio
  - Internet: TCP ou UDP
- Classes
  - Socket, ServerSocket, ObjectInputStream, ObjectOutputStream, DatagramSocket, DatagramPacket, InetAddress
- Métodos
  - accept, writeObject, readObject, send, receive, close

Sockets

- Para que o cliente faça a conexão com o servidor é necessário conhecer:
  - Endereço do servidor
  - Porta do serviço desejado

Sockets

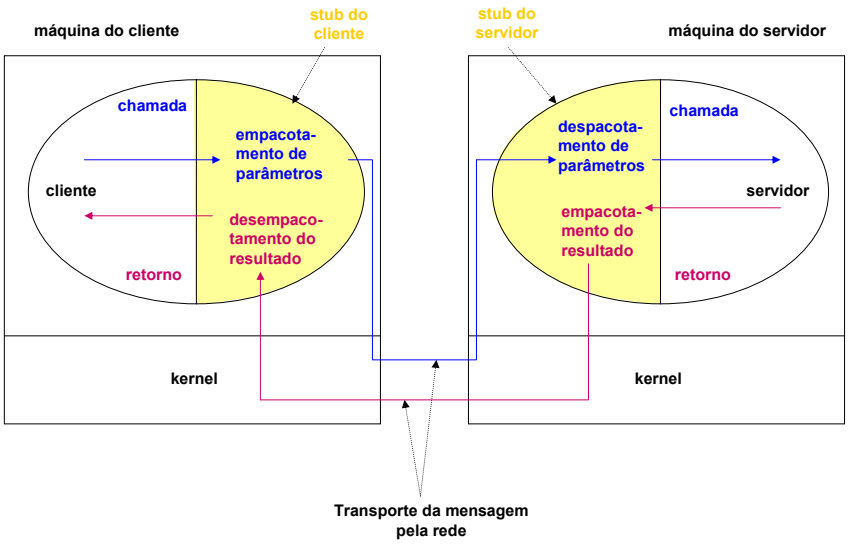


RPC

- Programas podem chamar procedimentos localizados em outras máquinas
- Parâmetros são passados por meio de mensagens
- Stub do cliente e stub do servidor montam mensagens

RPC

- Heterogeneidade
  - Representação padronizada
- Otimização do processo de passagem de parâmetros
  - Parâmetros de entrada, de saída ou de entrada e saída
- Ligação dinâmica
  - Similar ao servidor de nomes (portmapper)



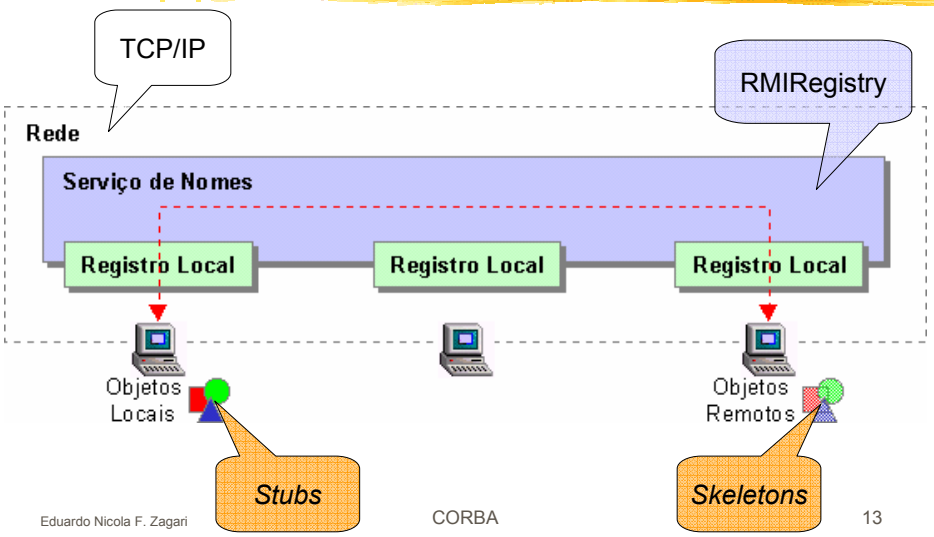
RPC

- Problemas
  - Suporte apenas a integração procedural de serviços de aplicação
    - Não fornece abstração de orientação a objeto (polimorfismo, herança etc.)
  - Não permite troca de mensagens assíncronas ou invocação dinâmica

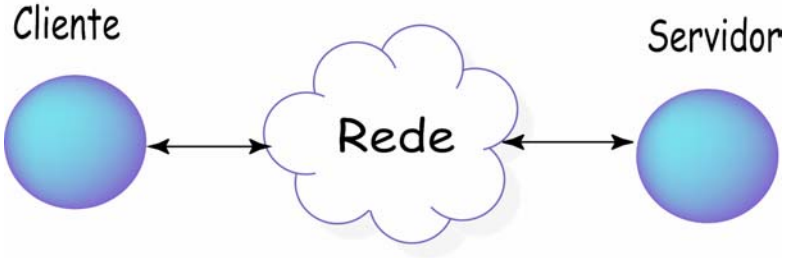
RMI

- Modelo de distribuição de objetos Java
  - Semântica dos objetos Java
  - Provê uso de objetos remotos
- Tecnologia Java-to-Java
  - Permite invocar objetos residentes em outras JVM
- Problema
  - Limitado a Java

RMI



Cliente/Servidor

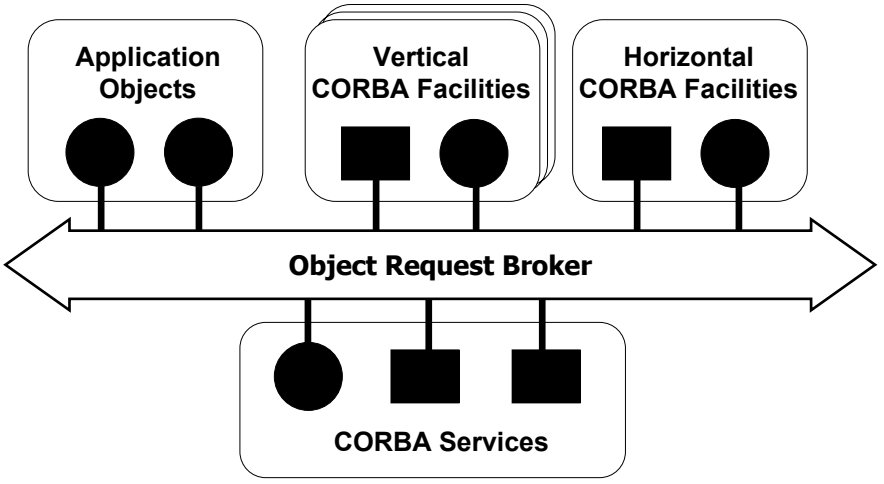


Sockets
RPC
Java RMI
CORBA

OMG

- Object Management Group: consórcio que produz e mantém especificações para obter interoperabilidade entre aplicações empresariais
  - www.omg.org
- Membros
  - Boing, Borland, Cisco, Citigroup, Fujitsu, GE, HP, IBM, Motorola, Nasa, Nokia, Oracle, Sun, Unisys, Visa, universidades, institutos de pesquisa...
- Especifica a OMA (Object Management Architecture)

OMA



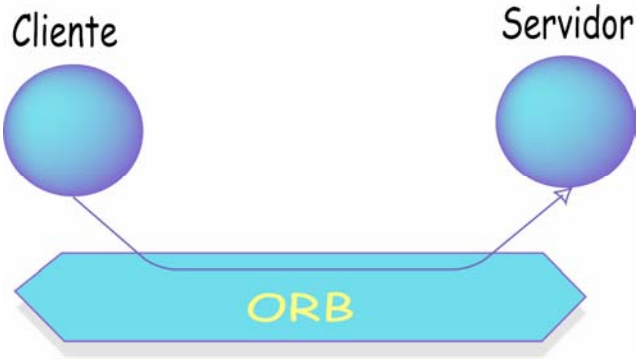
OMA

- CORBA Services: fornecem funcionalidades básicas similares às chamadas de sistema UNIX
- CORBA Facilities (horizontal): recursos úteis a diferentes domínios de negócio. Ex.: impressão, internacionalização, agente móvel
- CORBA Domain (vertical): serviços característicos de um determinado nicho. Ex.: saúde, telecomunicações
- Objetos de aplicação: objetos customizados para aplicações individuais (não precisam ser padronizadas)
- Object Request Broker: comunicação
  - Broker: intermediário, agente, corretor

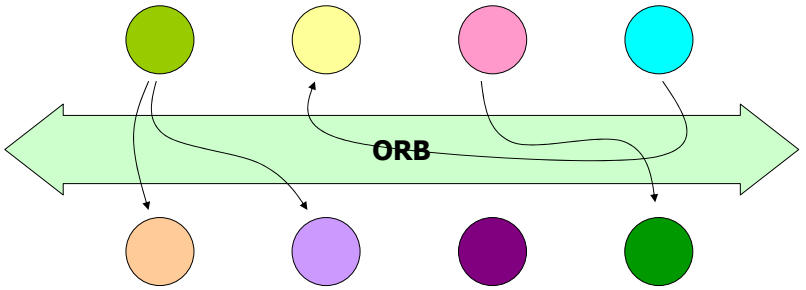
CORBA

- Aplicações CORBA são compostas por objetos (unidades individuais de software que combinam funcionalidades e dados)
- CORBA (Common Object Request Broker Architecture) é a especificação da arquitetura de um sistema que fornece interoperabilidade entre objetos em um ambiente heterogêneo

CORBA



CORBA



Vantagens

- Portabilidade: rodar em diferentes plataformas
- Interoperabilidade: comunicação entre diferentes ORBs
- Transparência na comunicação cliente/servidor
- Independência de linguagem de programação
- Coexistência com aplicações legadas
- Uso do paradigma de orientação a objetos: encapsulamento, herança, polimorfismo...

CORBA

- Cada objeto que fornece um serviço deve descrever sua interface
- A linguagem IDL (Interface Definition Language) da OMG é utilizada para especificar interfaces
  - Qualquer cliente que deseja invocar uma operação em um objeto deve usar a interface para especificar a operação que ele deseja e para fazer o marshalling dos argumentos
  - Quando a invocação chega ao objeto alvo a mesma definição de interface é usada para fazer o unmarshalling dos argumentos
  - A interface é usada ainda para fazer o marshalling e unmarshalling dos resultados

IDL

- A definição de interface IDL é independente de linguagem e pode ser mapeada para várias linguagens
  - C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python e IDLscript
- Separação entre interface e implementação permite a interoperabilidade
  - A interface de cada objeto é bem definida enquanto a implementação do objeto fica escondida do resto do sistema
- * Não é uma linguagem de programação

Tipos IDL

- Todo objeto CORBA possui um tipo igual ao nome da interface
- short, unsigned short, long, unsigned long, long long, unsigned long long (não existe int)
- float, double, long double
- boolean
- char,string
- any, void
- struct, enum, union, sequence

Passagem de parâmetros IDL

- Os parâmetros em IDL devem ter uma direção
  - in: parâmetro passado de cliente para o servidor
  - out: o parâmetro é retornado do servidor para o cliente
  - inout: o parâmetro é passado do cliente para o servidor e depois retornado do servidor para o cliente sobrescrevendo o valor original
- Os modos de passagem de parâmetros são usados para otimizar a troca de dados entre cliente e servidor

Módulos IDL

- IDL define o escopo (limite) em que as definições de tipos podem ser usadas
  - *namespaces* em C++
  - packages em Java

Exemplo de interface IDL

```
module Banco {
   interface Conta {
        attribute float saldo;
        readonly attribute string titular;
        void deposito( in float quantia, out float saldo );
        void saque( in float quantia, out float saldo );
   };

   interface Gerencia {
        Conta abertura();
   };
};
```

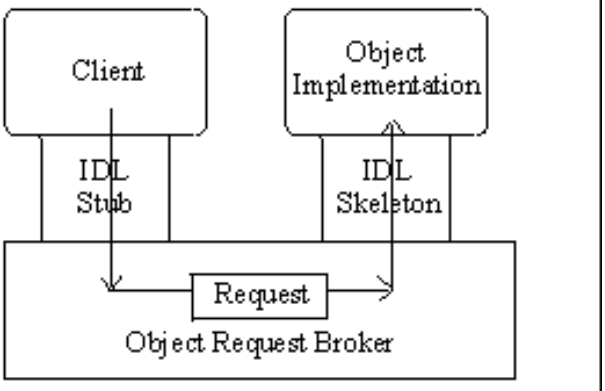
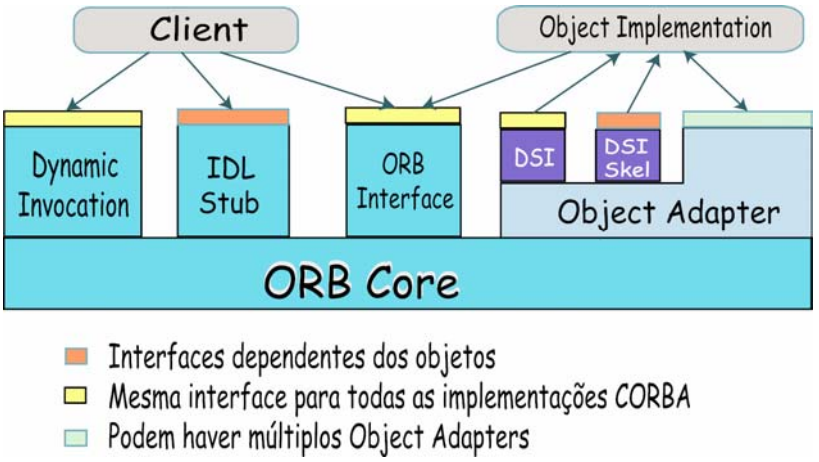



Figure 1: A request passing from client to object implementation

Arquitetura CORBA



Stub

- Gerado pelo compilador IDL
- Existe um stub para cada interface
- Ligado ao programa que deseja invocar estaticamente um método do servidor através da interface associada
- Mapeia o objeto do lado servidor em um objeto nativo na linguagem do cliente
- Age com um proxy para objetos remotos do servidor fazendo marshaling de métodos e parâmetros a serem transmitidos via ORB

DII (Dynamic Invocation Interface)

- Interface de invocação dinâmica
- Invocações DII diferem de invocações usando stub da mesma maneira que scripts diferem de programas, ou seja, eles são interpretados em tempo de execução e não compilados previamente.
- Usando DII, um cliente pode invocar uma operação em um novo tipo de objeto que ele acabou de descobrir (usando, por exemplo, Trader Service)
- Existe somente um DII servindo cada instância de cada tipo de objeto
- Verifica no repositório de interface (IFR - Interface Repository) a descrição das interfaces

Skeleton

- Stub do servidor
- Gerado pelo compilador IDL
- Existe um skeleton para cada interface
- Ligado ao servidor
- Fornece interface estática para a chamada de métodos de uma implementação de objeto
- Faz unmarshalling de métodos e parâmetros que vêm do cliente via ORB

DSI (Dynamic Skeleton Interface)

- Interface de skeleton dinâmico: é um mecanismo de ligação em tempo de execução para a criação de interfaces de servidor durante a execução do sistema
- Fornece uma maneira de entregar requisições de um ORB para uma implementação de objeto cujo tipo não era conhecido em tempo de compilação
- Análogo ao DII mas do lado servidor
- Torna possível a inspeção de parâmetros de uma requisição para determinar o objeto alvo e seu método

POA (Portable Object Adapter)

- Um adaptador de objeto é um mecanismo que conecta uma requisição usando uma referência a objeto com o código adequado para servir a requisição
  - Adaptador de objeto portável é um tipo particular de adaptador de objetos
- Responsável pela gerência de objetos do servidor e referências a objetos relacionadas
- Usa o repositório de implementação (IR – Implementation Repository) para localizar um nome de servidor e executar o comando, identificar o modo de ativação e determinar as permissões de execução e acesso
  - Os modos de ativação do servidor definem as características de tempo de execução de um servidor, incluindo se um servidor pode conter um ou mais objetos e se vários clientes podem acessar o mesmo objeto

POA (Portable Object Adapter)

- Políticas
  - ThreadPolicy: define se a implementação ORB ou a do programador cuidará da criação de threads
  - LifespanPolicy: especifica a criação transiente ou persistente de objetos CORBA
  - IdUniquenessPolicy: determina se cada servant pode se ligar a uma ou várias identificações de objeto
  - IdAssignment: define se o POA gera ID de objetos automaticamente
  - ImplicitActivationPolicy: seleciona se um objeto CORBA é criado automaticamente ou não

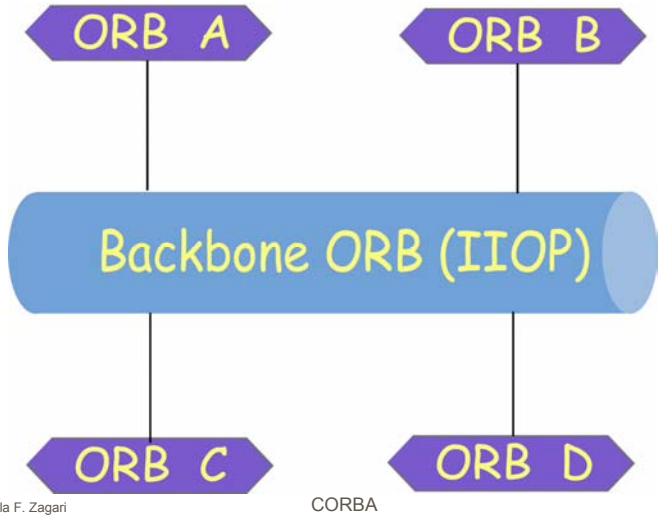
CORBA

- Um cliente pode localizar um objeto servidor obtendo uma referência ao objeto diretamente do servidor requisitando o objeto
  - pelo nome via serviço de nomes CORBA
  - para uma referencia a objetos que casa com algum critério via serviço de trader CORBA
- A especificação CORBA define um formato de referência a objeto interoperável (IOR - Interoperable Object Reference) que pode ser trocada entre diferentes implementações CORBA

CORBA

- Depois de obter a referência o cliente pode invocar métodos no objeto servidor usando diferentes modos:
  - **Synchronous:** o cliente envia a requisição e fica bloqueado até o retorno da resposta
  - **One-way/Poll:** o cliente envia um requisição e não espera a reposta mas fica fazendo poll até que o serviço tenha terminado
  - **One-way/Callback:** o cliente envia um requisição passando uma referência de objeto para callback. Quando o servidor termina ele invoca um método no objeto do cliente
  - **One-way/Event Service:** o cliente pede para ser notificado pelo serviço de eventos ao final do processamento e envia a requisição ao servidor. Quando o servidor termina ele avisa o servidor de eventos que notifica o cliente

Interoperabilidade



CORBA

- Interoperabilidade entre ORBs
  - GIOP (General Inter-ORB Protocol): define formatos de mensagens padronizados, representação de dados comum para o mapeamento de tipos IDL em mensagens e um formato de referência de objetos interoperável.
  - IIOP (Internet Inter-ORB Protocol): define a troca de mensagens GIOP sobre redes TCP/IP

CORBA

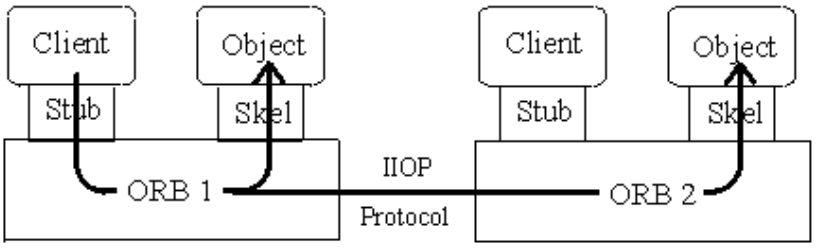


Figure 2: Interoperability uses ORB-to-ORB communication

Relação entre Protocolos

Processo de implementação CORBA

1. Especificar a interface IDL do servidor
2. Executar o compilador IDL que gera uma interface em código nativo, stub, e skeleton. Stub e skeleton pode ser para linguagem de programação diferentes. Nesta etapa o compilador pode escreve uma descrição de interface para o repositório de interface
3. Implementar o servidor
4. Compilar o servidor e ligar ao skeleton que foi gerado pelo compilador IDL. O resultado é um programa executável que aceita invocações de métodos via CORBA.

Processo de implementação CORBA

5. Registrar o servidor no repositório de implementação especificando nome, comando de execução, modo de ativação e permissões de execução e acesso. O servidor está pronto para ativação.
6. Implementar o cliente. Escrever as chamadas a métodos do objeto do servidor como ser fossem locais.
7. Compilar o programa cliente e ligar ao stub gerado pelo compilador IDL
8. Quando o cliente está sendo executado ele usa o ORB para se ligar ao objeto servidor e obter uma referência ao objeto.
9. Usando a referência, o cliente invoca métodos no objeto servidor

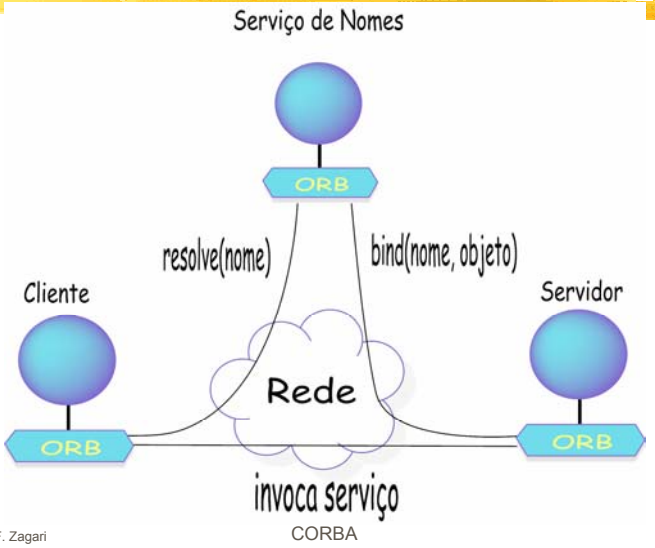
CORBA Services

- Additional Structuring Mechanisms for the OTS
- Collection Service
- Concurrency Service
- Enhanced View of Time
- Event Service
- Externalization Service
- Licensing Service
- Life Cycle Service
- Lightweight Services
- Management of Event Domains
- Naming Service
- Notification Service
- Notification / JMS Interworking
- Persistent State Service
- Property Service
- Query Service
- Relationship Service
- Security Service
- Telecoms Log Service
- Time Service
- Trading Object Service
- Transaction Service

Serviço de nome (Naming Service)

- Permite a associação entre nomes e referências de objetos remotos de objetos CORBA dentro de contextos de nomes
- Contexto de nome: escopo do nome (o nome deve ser único no contexto)
  - Os contextos podem ser aninhados gerando um espaço de nomes hierárquico (árvore de contextos)
- Um nome pode ser resolvido iniciando de qualquer contexto de nome
  - É obtida uma referência de objeto remoto ou outro contexto de nome
  - A operação é repetida até se obter um referência de objeto

Serviço de nome (Naming Service)



Serviço de eventos (Event Service)

- Define interface que permite objetos produtor (suppliers) comunicar notificações a outros objetos consumidores (consumers)
- As notificações são comunicadas como argumentos ou resultados de invocações de métodos remotos
- Modos
  - Push: empurrada do produtor para o consumidor
  - Pull: puxada do produtor pelo consumidor
- Canais de eventos: objetos CORBA que permitem a comunicação entre múltiplos produtores e múltiplos consumidores

Serviço de notificação (Notification Service)

- Estende serviço de eventos definindo novas características:
  - Notificações podem ser estruturas de dados
  - Consumidores podem definir filtros
  - Mecanismo para que produtores descubram os eventos de interesse dos consumidores
  - Consumidores podem descobrir tipos de eventos oferecidos em um canal
  - Configuração de propriedades do canal
    - Prioridade, confiabilidade, ordenação

Implementações CORBA

- Borland Enterprise Server (Borland)
- ChorusORB (Sun)
- Component Broker/DSOM (IBM)
- Hardpack (Lockheed Martin)
- ILU (Xerox Parc)
- JavaIDL (JavaSoft)
- MICO (University of Frankfurt)
- OAK (Camros Software)
- ObjectDirector (Fujitsu)
- omniORB2 (AT&T Laboratories)
- ORBacus (Object-Oriented Concepts)
- Orbix (Iona Technologies)
- ROBIN (Jim Pangburn, FermiLab)
- TAO (Washington University)
- VisiBroker (Borland)
- Voyager ORB (ObjectSpace)

Implementações CORBA

- MICO
- OmniORB2
- Jacorb
- ORBit
- ORBacus
- OmniBroker
- TAO
- JavaIDL

- Orbix
- Visibroker
- CorbaPlus
- Oak
- OrbixWeb

Considerações finais

- Curva de aprendizado
  - Receita de bolo
- Alto nível de abstração
- Riqueza de tipos
- Heterogeneidade
- Integração de sistemas legados
- ORB dentro de browser
- Exemplos não refletem sistemas em produção