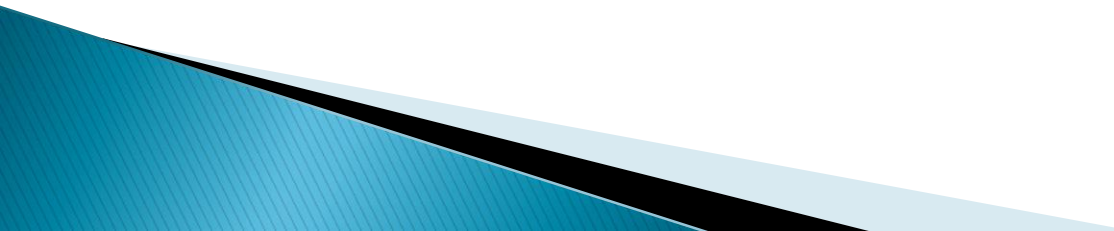


# Middleware e Sistemas Distribuídos

Obtido da URL: <http://www.cin.ufpe.br/~locb/SemRedes/>



# Roteiro

- ▶ Sistemas Distribuídos
  - ▶ Middleware
  - ▶ Tipos de Middleware
  - ▶ Principais Middlewares
  - ▶ Estudo de Caso
  - ▶ Conclusões
- 

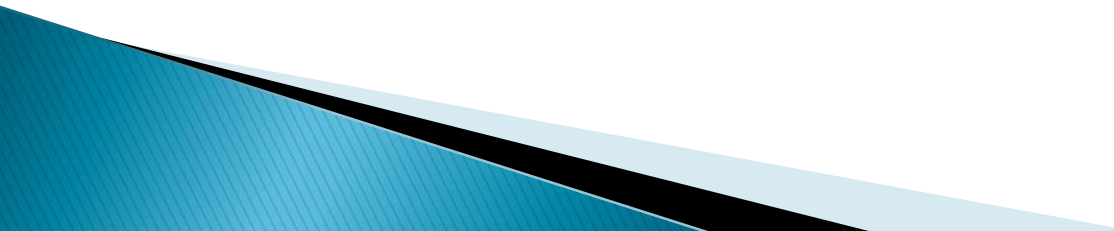
# Sistemas Distribuídos

## ► Sistemas Distribuídos são...

### ◦ Vantagens

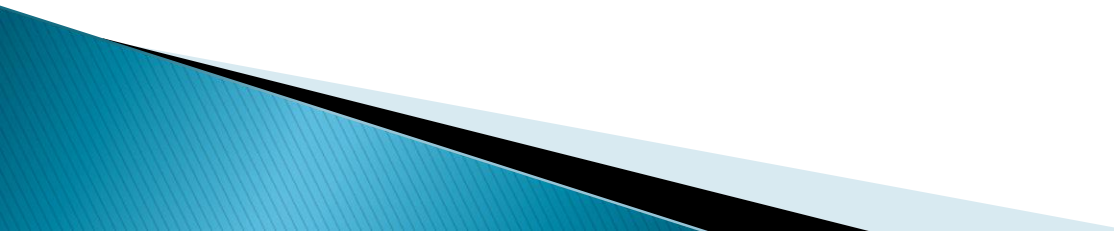
- Disponibilidade
- Escalabilidade
- Tolerância a falhas
- Custo benefício

### ◦ Desvantagens

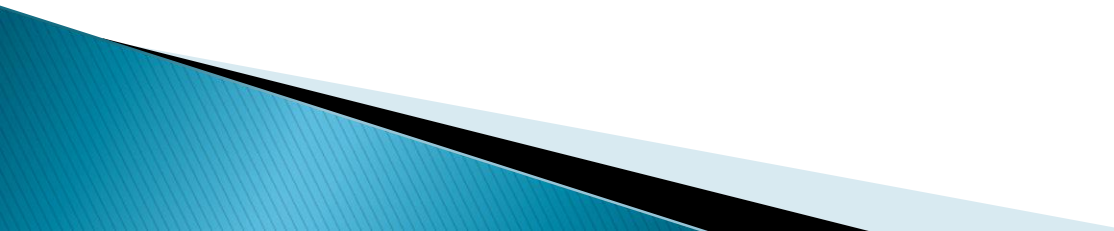
- Mais fontes de falhas
  - Complexidade
  - Heterogeneidade
  - Segurança
- 

# Sistemas Distribuídos

## ▶ Transparência

- Local
  - Acesso
  - Falha
  - Tecnologia
  - Concorrência
- 

# Middleware

- ▶ “Um middleware pode ser visto como uma camada de software intermediária localizada entre o sistema operacional e a aplicação.”
  - ▶ Desenvolvimento de Sistemas distribuídos mais fácil e ágil.
- 

# Middleware

## Requisitos de um Middleware

- ▶ Permitir Comunicação
  - Uso de protocolos de comunicação
  - Marshalling e Unmarshalling
  - IDL para garantir (un)marshalling dos dados

# Middleware

## Requisitos de um Middleware

- ▶ Confiança na execução de requisições
  - Melhor esforço
    - Sem garantias
  - Pelo menos uma
    - Potencialmente mais de uma
  - No máximo uma
    - Só uma vez

# Middleware

## Requisitos de um Middleware

### ► Permitir Escalabilidade

- Medida de Capacidade de adaptação
- Replicação de componentes como solução
  - Transparência de acesso
  - Transparência de localização
  - Transparência de migração
  - Transparência de replicação



# Middleware

## Requisitos de um Middleware

- ▶ Lidar com Heterogeneidade
  - Não restringe o sistema a uma só tecnologia
  - Permite que componentes legados sejam integrados a novos componentes.

# Middleware

## ▶ Serviços de Middleware

- Ciclo de vida
  - Gerenciamento do Ciclo de vida dos objetos
- Serviço de nomes
  - Permite referenciar objetos pelo nome
- Relacionamento
  - Cria associações dinamicamente entre objetos
- Transação
  - Faz gerenciamento de transações

# Middleware

## ▶ Serviços de Middleware

- Negócio
  - Permite localização de serviços
- Segurança
  - Oferece funcionalidades de segurança
- Tempo
  - Serviço de sincronização de relógios
- Evento
  - Registra interesse de componentes por eventos

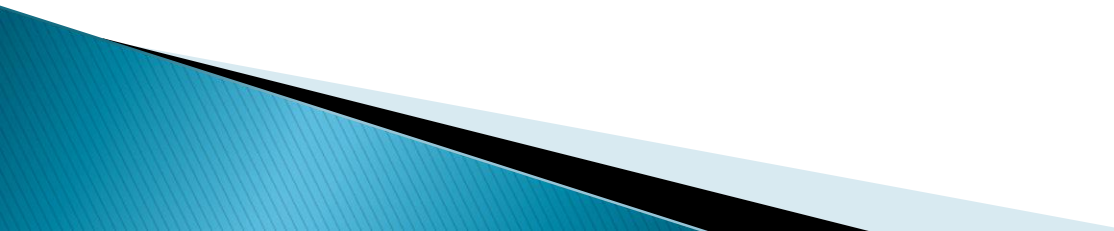
# Tipos de Middleware

- ▶ Middleware Transaccional
  - Suporte a transações síncronas
  - Coordena requisições entre clientes e servidores
  - Pode suportar as propriedades ACID

# Tipos de Middleware

## Middleware Transacional

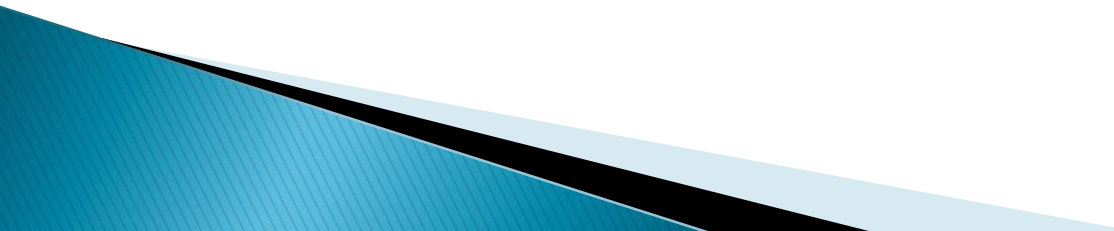
### ► Vantagens

- Componentes se mantêm consistentes
  - Bastante confiável
  - Boa performance
  - Escalonamento e priorização de solicitações
- 

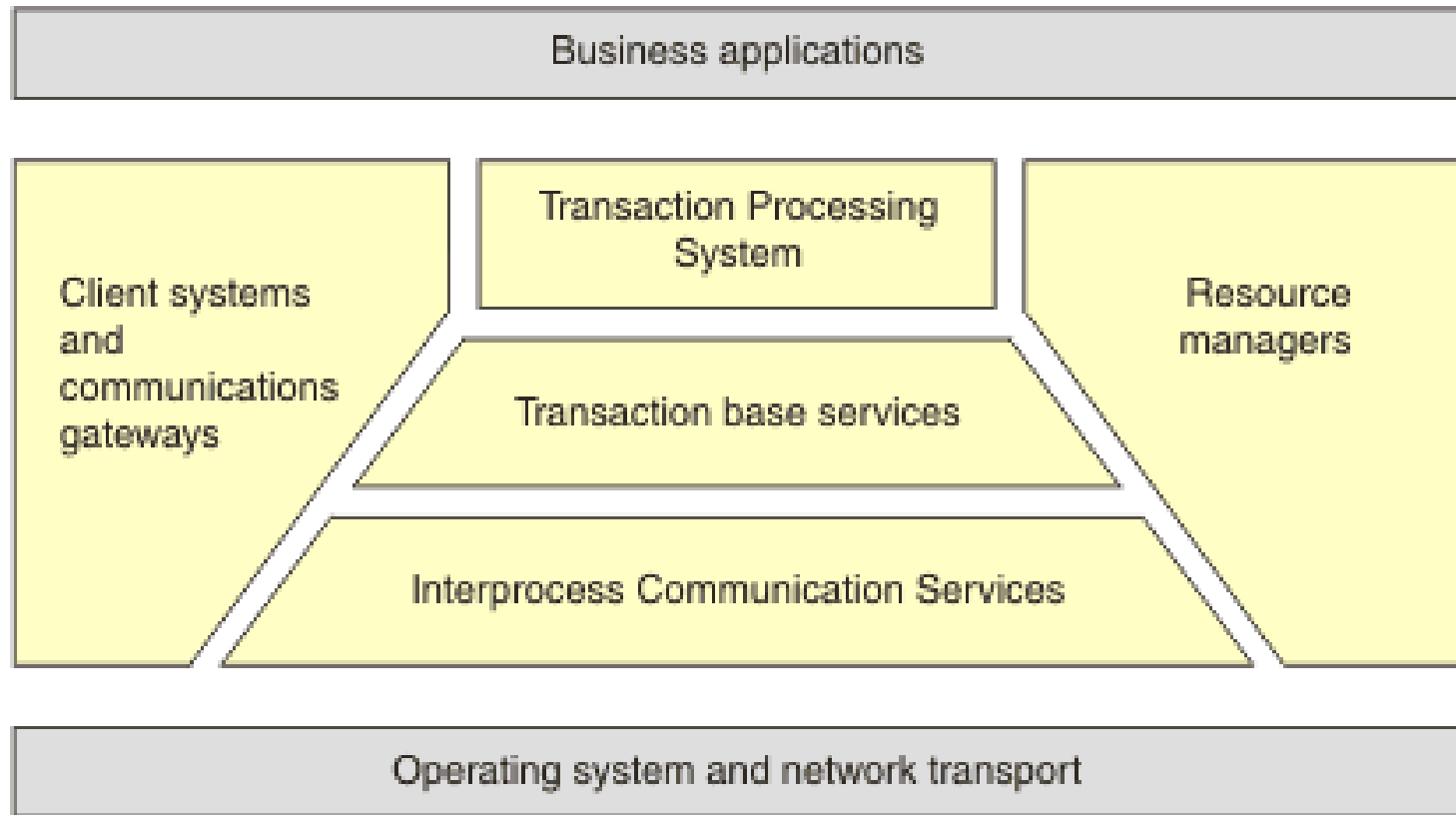
# Tipos de Middleware

## Middleware Transacional

### ► Desvantagens

- Ausência de padronização para descrever serviços
  - Executa numa menor quantidade de plataformas
  - Bloqueios desnecessários
  - Marshalling e unmarshalling implementadas manualmente
- 

# Tipos de Middleware



# Tipos de Middleware

- ▶ Middleware Orientado a Mensagens (MOM)
  - Message queuing
    - Comunicação indireta
    - Assíncrona
    - Mensagens enviada para filas
  - Message Passing
    - Comunicação direta
    - Síncrona
    - Destaque para o modelo publish–subscribe



# Tipos de Middleware

## Middleware Orientado a Mensagens (MOM)

### ► Vantagens

- Suporta comunicação em grupo de forma atômica
- Confiabilidade
- Amplo suporte a protocolos de rede

# Tipos de Middleware

## Middleware Orientado a Mensagens (MOM)

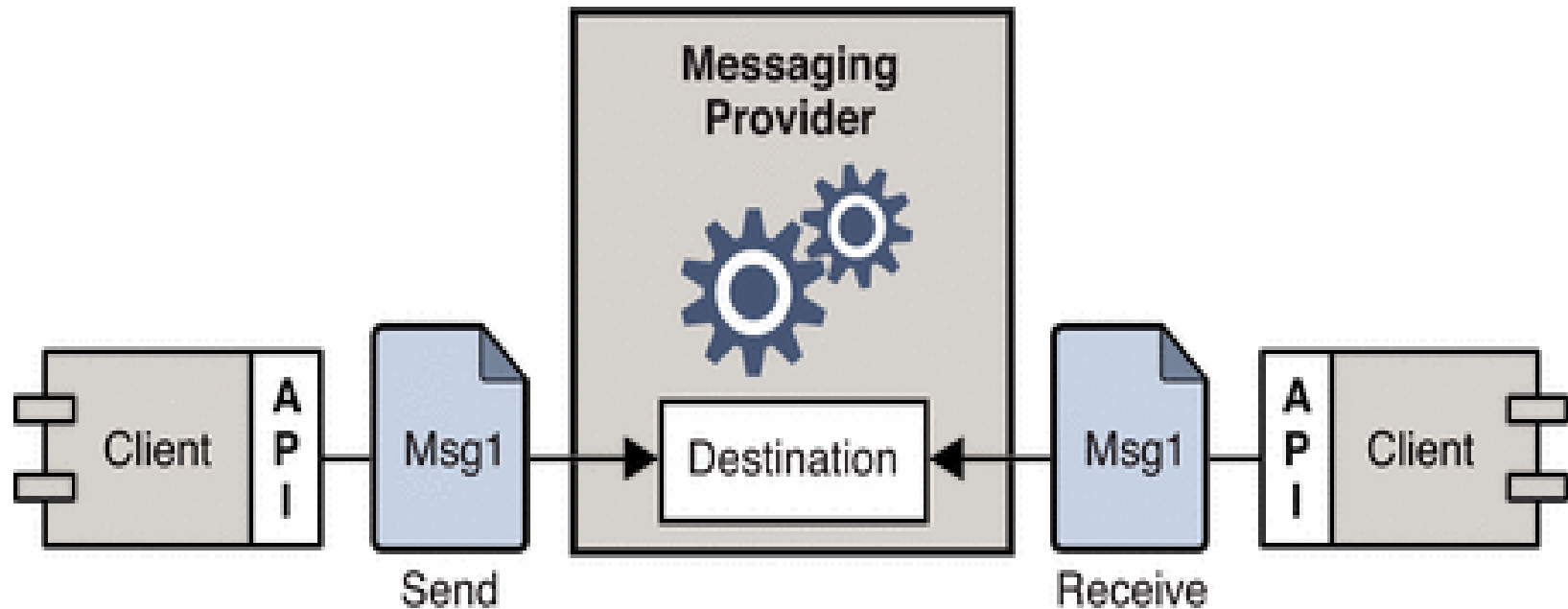
### ► Desvantagens

- Escalabilidade e heterogeneidade limitadas
- Pouca portabilidade por falta de padronização

### ► Uso

- Aplicações cuja disponibilidade da rede ou de todos os componentes não seja um problema

# Tipos de Middleware



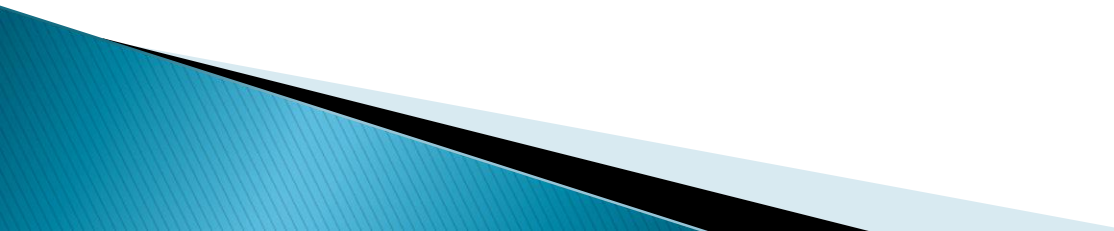
# Tipos de Middleware

- ▶ Middleware Orientado a Objetos (MOO)
  - Evolução dos middlewares procedurais
  - Interação por invocação de métodos
  - Comunicação tipicamente síncrona
  - IDLs para descrever serviços

# Tipos de Middleware

## Middleware Orientado a Objetos (MOO)

### ► Vantagens

- Grande suporte a heterogeneidade
  - Marshalling e unmarshalling automáticos
  - Versatilidade
- 

# Tipos de Middleware

## Middleware Orientado a Objetos (MOO)

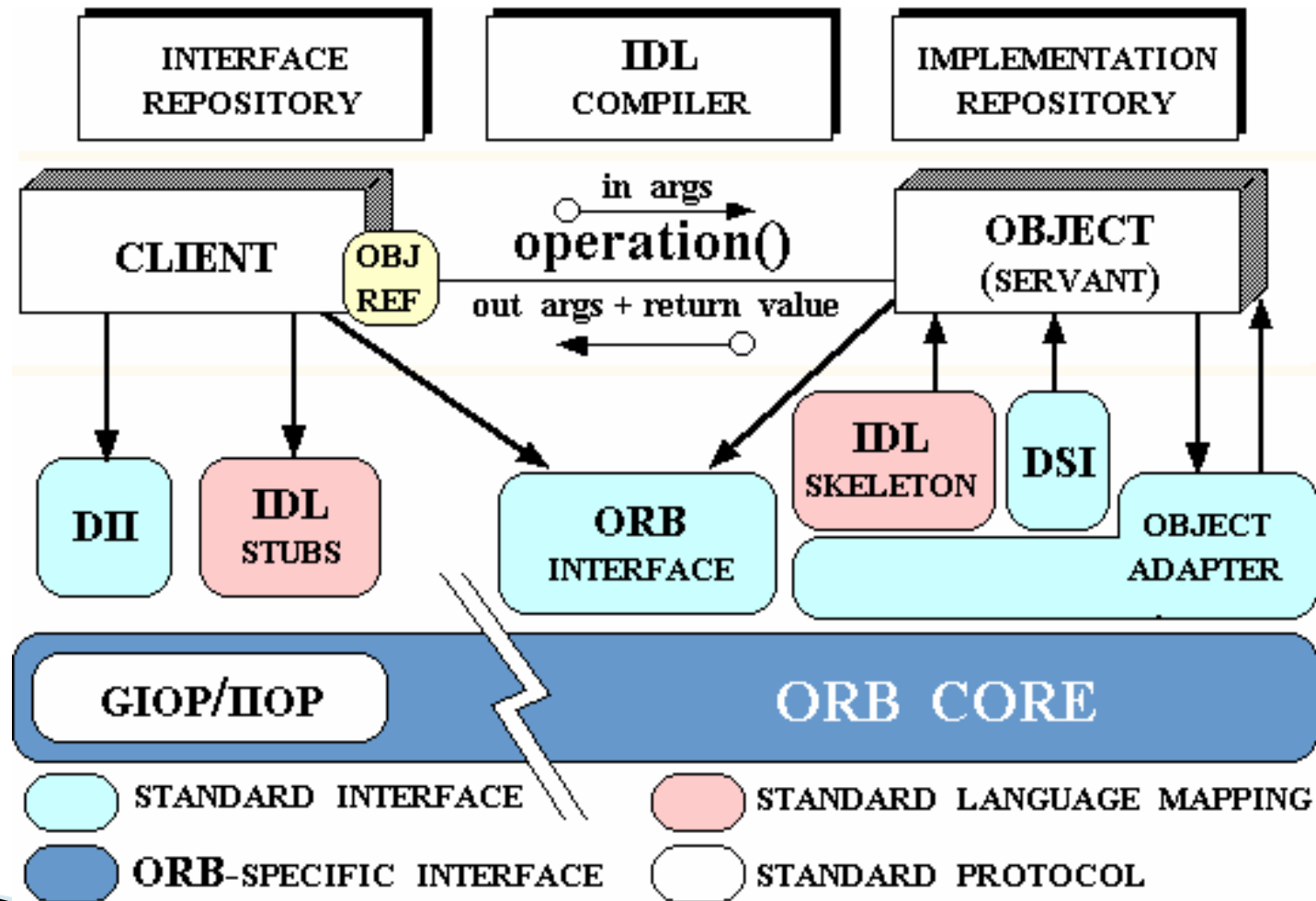
- ▶ Desvantagens

- Pouca Escalabilidade

- ▶ Uso

- Aplicações que não precisam de grande escalabilidade

# Tipos de Middleware

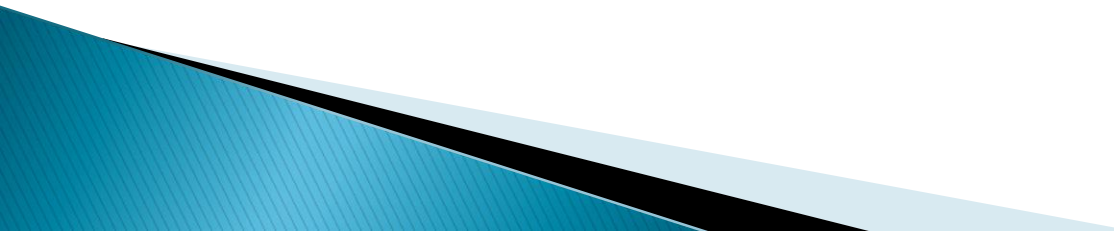


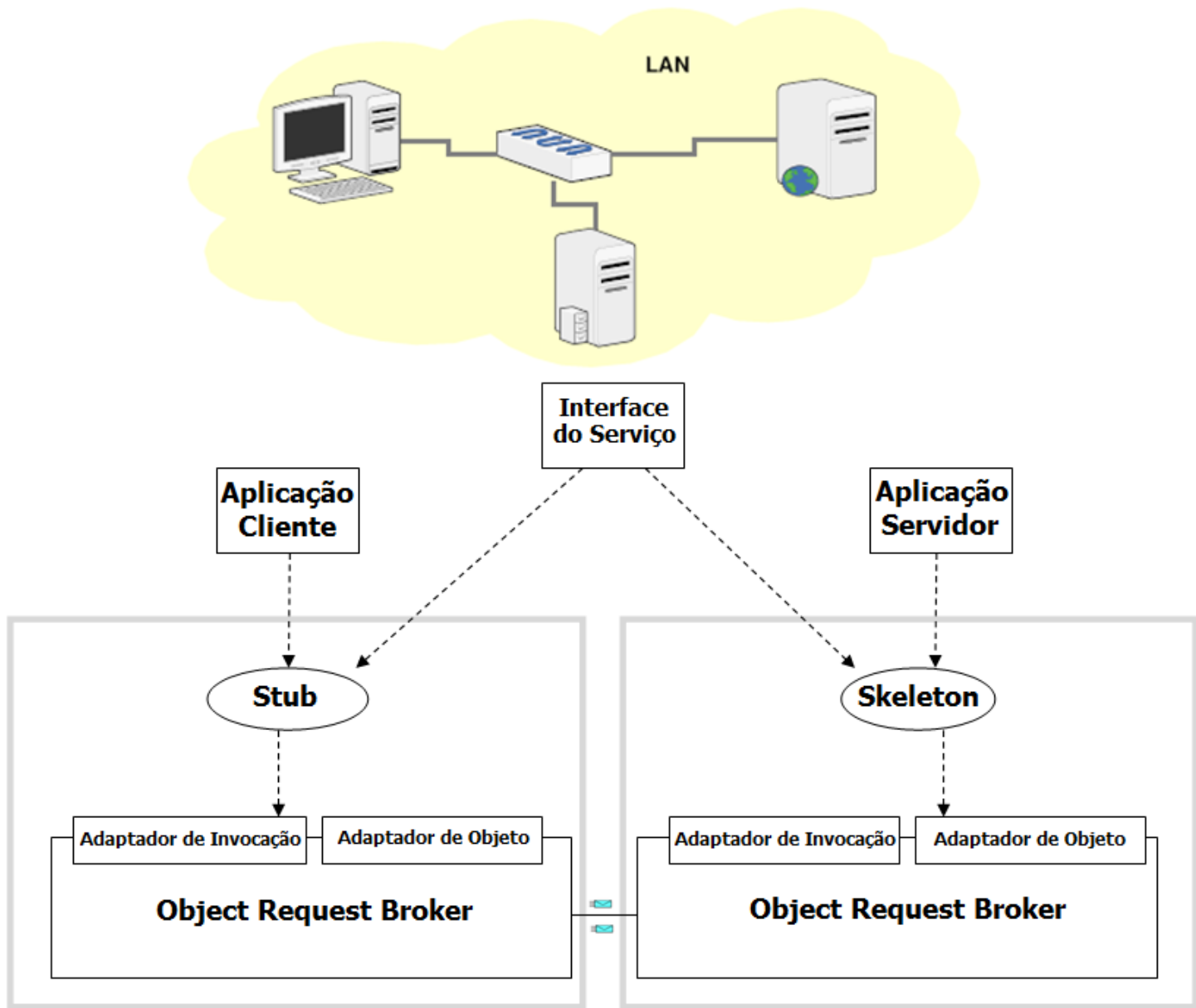
# Principais Middlewares

- ▶ Transacionais
  - Tuxedo (BEA)
  - CICS (IBM)
  - Encina (Transarc)
- ▶ MOM
  - MQSeries (IBM)
  - JMS (Sun)
- ▶ MOO
  - CORBA (OMG)
  - COM (Microsoft)
  - RMI
  - ICE
- ▶ MPI é baseado em troca de mensagens e não em invocações remotas. Trata-se de um padrão na comunidade de alto desempenho, amplamente usado pela comunidade C++.

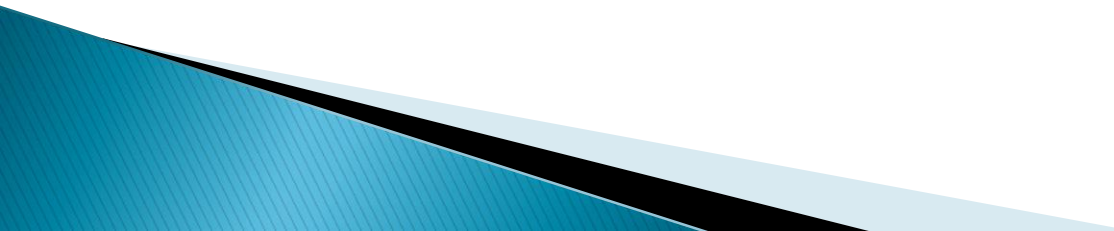


# Estudo de Caso – CORBA

- ▶ Especificado pela OMG
  - ▶ Especialização do Object Management Architecture (OMA)
  - ▶ Transforma modelos abstratos de objetos em formas concretas
- 



# CORBA – IDL

- ▶ Interface Definition Language (IDL)
    - Linguagem declarativa
    - Define interfaces de objetos com independência de linguagem
    - Separa a interface da implementação de um objeto
    - Permite a herança de interfaces
- 

# CORBA – IDL

## ▶ Language Mappings

- Gera o Stub e Skeleton
- Mapeia tipos da IDL para tipo da linguagem alvo
- Language Mapping é extremamente dependente das linguagens de programação utilizadas.

# CORBA – ORB

## ▶ Object Request Broker

- É o componente mais importante de CORBA
- Transmite invocação de operações do cliente para o servidor
- Trata de todas as tarefas associadas à invocação de um método

# CORBA – ORB

## ▶ Object Request Broker

- Lida com todas as heterogeneidades do ambiente
  - Localidade
  - Linguagem de programação
  - Sistema operacional
  - Hardware
  - Meios de comunicação

# CORBA – Adaptadores

- ▶ Adaptadores de Invocação
  - Usados indiretamente pelo cliente
  - Separados do ORB
- ▶ Adaptadores de Objetos
  - Fornece um ambiente para instanciar objetos e passar requisições (ciclo de vida)
  - Também separados do ORB

# Conclusões

- ▶ Middlewares objetivam resolver a maioria das complexidades inseridas pelo uso de vários computadores
  - ▶ Criam um ambiente de desenvolvimento de aplicações distribuídas de mais alto nível
  - ▶ Permitem reuso de componentes
- 