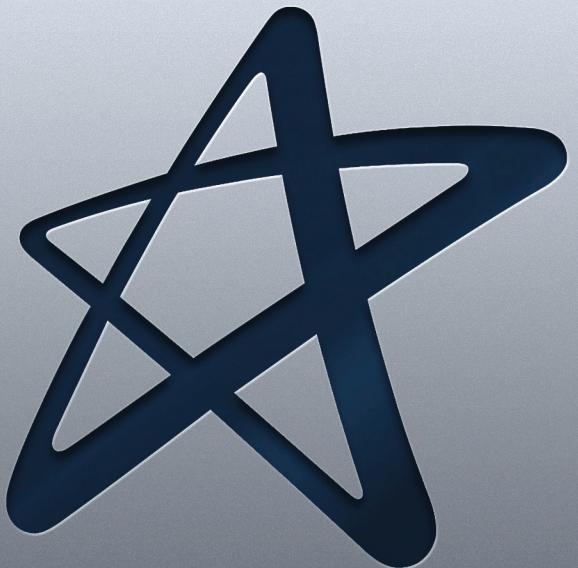


Arquitetura de Sistemas Distribuídos



Cruzeiro do Sul Virtual
Educação a distância

Material Teórico



Comunicação em Sistemas Distribuídos

Responsável pelo Conteúdo:

Prof. Me. Max D'Angelo Pereira

Revisão Textual:

Prof.^a Dr.^a Selma Aparecida Cesarin

UNIDADE

Comunicação em Sistemas Distribuídos



- Introdução;
- Conceitos de Redes;
- Chamadas Remotas;
- Corba;
- Web Services.



OBJETIVO DE APRENDIZADO

- Apresentar as Tecnologias de Comunicação mais utilizadas para Aplicações Distribuídas.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:

Determine um horário fixo para estudar.

Mantenha o foco! Evite se distrair com as redes sociais.

Procure manter contato com seus colegas e tutores para trocar ideias! Isso amplia a aprendizagem.

Seja original! Nunca plágie trabalhos.

Aproveite as indicações de Material Complementar.

Conserve seu material e local de estudos sempre organizados.

Não se esqueça de se alimentar e de se manter hidratado.

Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Introdução

Nesta Unidade, você será apresentado(a) às Tecnologias mais utilizadas na comunicação para Aplicações Distribuídas.

Você irá conhecer os conceitos e o funcionamento das soluções existentes para implementação de aplicações que se comuniquem por meio de um modelo de arquitetura distribuída.

Atualmente, é comum encontrarmos aplicações que possuam seus recursos de *software* espalhados, fisicamente, em diferentes computadores. Por exemplo, um Caixa Eletrônico possui uma parte de *software* instalado fisicamente no *hardware* do próprio Caixa Eletrônico, porém ele depende de comunicação com um servidor que atenda às suas solicitações.

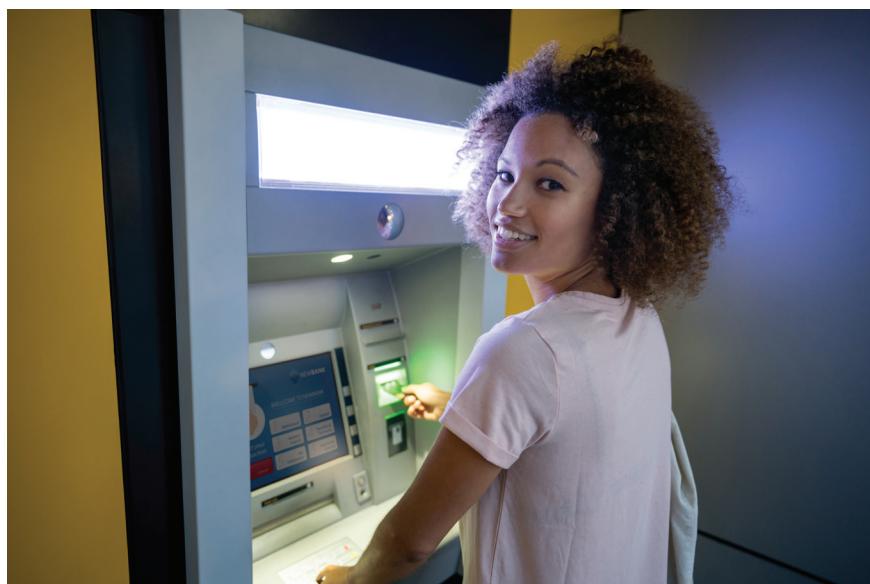


Figura 1
Fonte: Getty Images

Então, pense em uma operação de saque. Essa funcionalidade depende da camada de apresentação (telas), que está fisicamente instalada e processa no *hardware* do Caixa Eletrônico.

Quando um cliente interage com as telas e sua senha é solicitada, essa senha precisa ser validada pelo servidor, pois, por questões de segurança, não se armazena o Banco de Dados de senhas localmente em cada Caixa Eletrônico. Então, precisa existir um ou mais Protocolos de comunicação que permitam que o Caixa Eletrônico se comunique com o(s) servidor(es) e solicite a validação da senha.

O servidor irá processar a solicitação e devolver o resultado para o Caixa Eletrônico (neste caso, no papel de cliente). Se a senha estiver correta, permite-se que a operação de senha prossiga; caso contrário, uma mensagem informativa será exibida na tela do Caixa Eletrônico e a operação se encerra.



Como aplicações em Computadores com Sistemas Operacionais e *hardwares* heterogêneos conseguem comunicar-se?

E se você estivesse em uma situação na qual precisa falar com outra pessoa que não entende o idioma Português – digamos que ela só fale Alemão e você só fale Português, mas um amigo em comum comprehende os dois idiomas?

A solução mais simples parece ser pedir ao amigo em comum que faça a tradução para vocês. Bem, de certo modo, as Tecnologias que estudaremos a seguir são Protocolos que irão realizar a função de tradutor para as aplicações de *softwares* que precisam estabelecer uma Comunicação.



Figura 2
 Fonte: Getty Images

Conceitos de Redes

Quando você pensar em Sistemas Distribuídos, é importante que tenha em mente que a comunicação é a chave para esse tipo de Sistema.

O processamento, memória e demais recursos computacionais utilizados pelas Aplicações Distribuídas estão “espalhados”, sendo que a comunicação entre eles permite que troquem informações e compartilhem recursos.

Antes de iniciar o entendimento das tecnologias que permitem que as aplicações sejam implementadas sob uma arquitetura ou modelo distribuído, você precisa recordar alguns conceitos fundamentais de Comunicação de Redes de Computadores.

Redes de Computadores

A Comunicação de Rede, ou Interligação de Rede, define um conjunto de Protocolos (isto é, regras e padrões) que permitem que os Programas de Aplicativos conversem entre si sem considerar o *hardware* e os sistemas operacionais em que são executados. A Comunicação de Rede permite que os programas aplicativos se comuniquem independentemente de suas conexões físicas de Rede.

A tecnologia de interligação de Redes, denominada TCP/IP, recebe o nome de seus dois principais Protocolos: TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*). Para entender o TCP/IP, você deve estar familiarizado com os seguintes termos:

- **Host:** é um computador ou outro dispositivo conectado a uma Rede;
- **Cliente:** um processo que solicita serviços na Rede;

- **Servidor:** um processo que responde a uma solicitação de serviço de um cliente;
- **Datagrama:** unidade básica de informações, consistindo de um ou mais pacotes de dados, que são transmitidos pela *Internet* no nível de transporte;
- **Pacote:** Unidade ou Bloco de uma transação de dados entre um computador e sua Rede. Um pacote, geralmente, contém um cabeçalho de Rede, pelo menos um cabeçalho de Protocolo de alto nível e blocos de dados. Geralmente, o formato dos Blocos de Dados não afeta o modo como os pacotes são manipulados.



Pacotes são o meio de troca usado na camada comunicação de Rede.

Protocols

Um Protocolo é um conjunto de regras ou padrões que cada host deve seguir para permitir que outros *hosts* recebam e interpretem mensagens enviadas para eles.

Existem dois tipos gerais de Protocolos de transporte, apresentados a seguir.

Protocolo não orientado à conexão

Um **Protocolo** não orientado à conexão é um Protocolo que trata cada datagrama como independente de todos os outros. Cada datagrama deve conter todas as informações necessárias para sua entrega. Um exemplo de tal Protocolo é o *User Datagram Protocol* (UDP).

O UDP é um Protocolo da camada de transporte criado diretamente na camada IP e usado para envio de um datagrama diretamente de uma aplicação para aplicação que estejam em *hosts* baseados em TCP/IP.

O UDP não garante a entrega de dados e, portanto, é considerado não confiável. Aplicações que exigem entrega confiável de fluxos de dados devem usar o TCP.

Protocolo orientado à conexão

Um **Protocolo orientado à conexão** requer que os *hosts* estabeleçam uma conexão lógica entre si antes que a comunicação possa ocorrer. Uma troca orientada por conexão inclui três fases:

- Iniciar a conexão;
- Transferir dados;
- Finalizar a conexão.

Um exemplo de um Protocolo desse tipo é o *Transmission Control Protocol* (TCP).

O TCP fornece um meio confiável para a entrega de pacotes entre *hosts* em uma Internet. TCP divide um fluxo de dados em datagramas, envia cada um

individualmente usando IP e remonta os datagramas no nó de destino. Se algum datagrama for perdido ou danificado durante a transmissão, o TCP detecta isso e retransmite os datagramas ausentes. O fluxo de dados recebido é, portanto, uma cópia confiável do original.

A Internet e a Web funcionam graças ao TCP. Ele é complementado pelo *Internet Protocol* (IP), sendo normalmente chamado de TCP/IP. O TCP é um Protocolo da Camada de Transporte e muitas Aplicações Distribuídas são implementadas com o uso do TCP.

Portas TCP/IP e Soquetes

Em uma Rede TCP/IP, todos os dispositivos devem ter um endereço IP. O endereço IP identifica o dispositivo, por exemplo, o computador. No entanto, um endereço IP sozinho não é suficiente para executar aplicativos de Rede, pois um computador pode executar vários aplicativos e ou serviços. Assim como o endereço IP identifica o computador, a porta de Rede identifica o aplicativo ou o serviço em execução no computador.



O uso de portas permite que os Computadores/Dispositivos executem vários serviços / aplicativos.

Se você usar uma analogia de um prédio de apartamentos, o endereço IP corresponde ao endereço da rua. Todos os apartamentos compartilham o mesmo endereço. No entanto, cada apartamento também tem um número que corresponde ao número da porta.

Soquete

Um soquete é a combinação de endereço IP mais porta utilizada em uma conexão entre dois computadores. Com um soquete, é possível identificar unicamente um aplicativo na Rede de comunicação IP.

Chamadas Remotas

RPC

RPC (*Remote Procedure Call*) ou Chamada de Procedimento Remoto define um Protocolo que permite que um Programa solicite a execução de uma rotina de outro programa por meio de uma Rede de Computadores.

Além de permitir que seja executado uma rotina que se encontra em outro computador, um servidor, por exemplo, o programador não precisa codificar em

detalhes como essa comunicação deve ocorrer, pois esses detalhes ficam encapsulados pelo próprio Protocolo RPC.

O Protocolo RPC pode ser implementado sobre diferentes Protocolos de transporte. Não cabe ao RPC especificar como a mensagem é enviada de um processo para outro, mas somente especificá-la e interpretá-la. A sua implementação depende, portanto, de sobre qual Protocolo de Transporte vai operar.

Uma chamada de procedimento remoto é uma técnica de comunicação entre processos utilizada para aplicativos baseados em cliente/servidor. Também é conhecido como chamada de sub-rotina ou chamada de função.

Um cliente tem uma mensagem de solicitação que o RPC traduz e envia para o servidor. Essa solicitação pode ser um procedimento ou uma chamada de função para um servidor remoto. Quando o servidor recebe a solicitação, ele envia a resposta necessária de volta para o cliente.

A sequência de eventos em uma chamada de procedimento remoto acontece da seguinte maneira:

- A aplicação do cliente efetua uma chamada remota;
- O RPC do cliente se encarrega de enviar a mensagem ao servidor e coloca os parâmetros na mensagem;
- A mensagem é enviada do cliente para o servidor pelo Sistema Operacional do cliente;
- O Sistema Operacional do servidor recebe e repassa a mensagem para o RPC do servidor;
- Os parâmetros são removidos da mensagem pelo RPC do servidor;
- Em seguida, a rotina da aplicação do servidor é chamada pelo RPC do servidor.

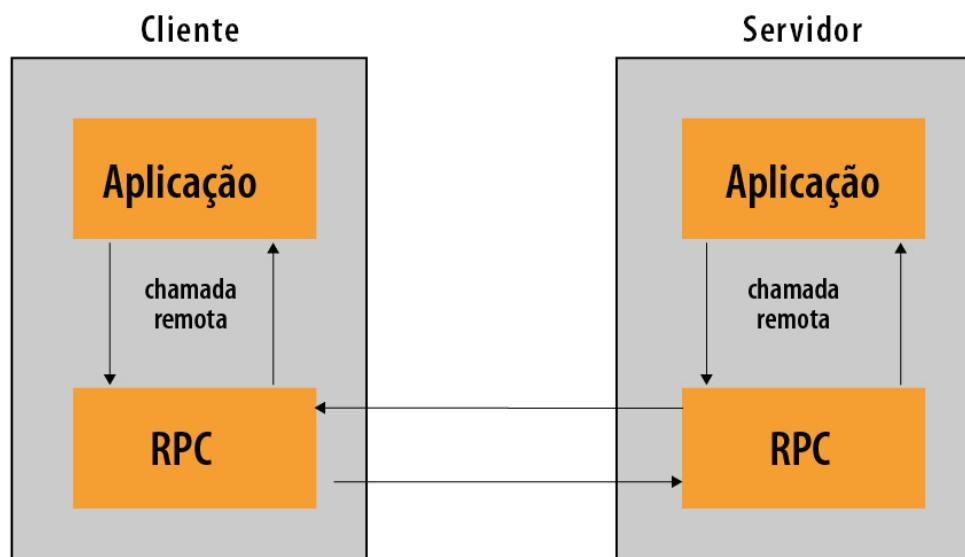


Figura 3 – Esquema de funcionamento de uma chamada remota com RPC

Fonte: Acervo do Conteudista

Vantagens

Algumas das vantagens do RPC são as seguintes:

- Suportam modelos orientados a processos e orientados a *threads*;
- O mecanismo interno de passagem de mensagens do RPC está oculto para o usuário;
- Pouco esforço é necessário para escrever o código em RPC;
- Podem ser usadas no ambiente distribuído, bem como no ambiente local.

Desvantagens

Algumas das desvantagens do RPC são as seguintes:

- A chamada de procedimento remoto é um conceito que pode ser implementado de diferentes maneiras. Não é um padrão;
- Não há flexibilidade no RPC para arquitetura de *hardware*;
- Há um aumento nos custos de processamento;
- As RPCs são nomeadas por meio de interfaces. Uma interface descreve e identifica exclusivamente uma rotina específica, informando seus parâmetros e tipos de argumentos. Dessa forma, o cliente precisa conhecer a interface antes de usá-la.

RMI

O RMI (*Remote Method Invocation*) é uma Tecnologia suportada pela plataforma Java que auxilia o desenvolvimento de Aplicações Distribuídas. O RMI permite ao programador chamar métodos de objetos remotos, ou seja, que estão alocados em Máquinas Virtuais Java (JVM), distintas sem necessidade de codificação de detalhes de comunicação de Rede, tornando o processo muito semelhante às invocações de objetos locais (mesma JVM).



Importante!

RMI é uma interface de programação que permite a execução de chamadas remotas no estilo RPC, porém o RPC é para Linguagens Procedurais e o RMI é para Linguagem Java e orientado a objetos.

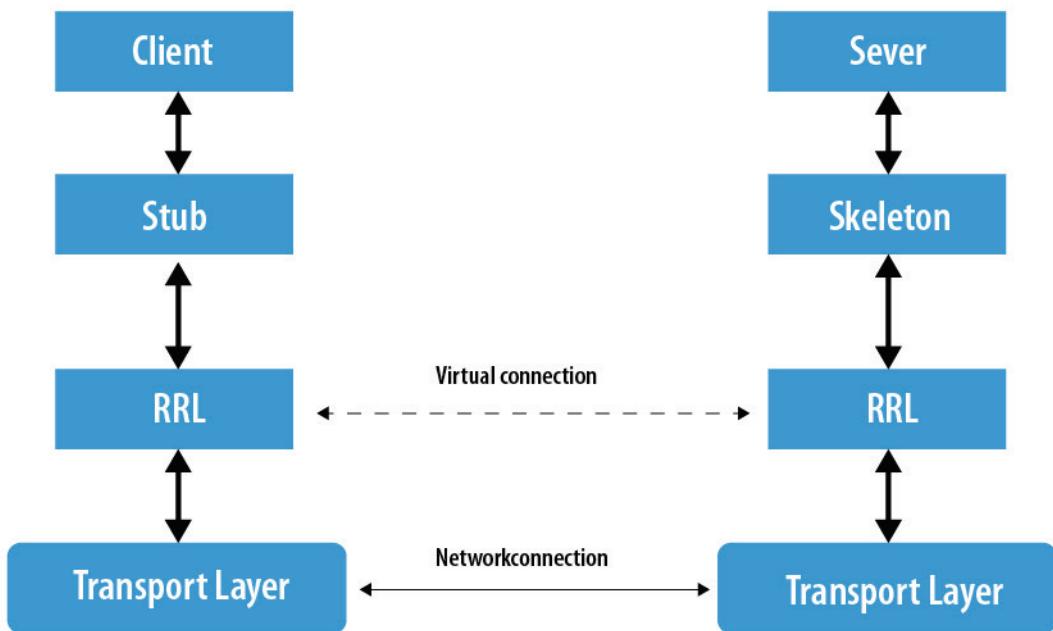


Figura 4 – Arquitetura de uma aplicação cliente/servidor com RMI

Fonte: Acervo do Conteudista

Veja a explicação de cada componente separadamente:

- **Transport Layer:** Esta camada conecta o cliente e o servidor. Ela gerencia a conexão existente e configura novas conexões;
- **Stub:** Um *stub* é uma representação (*proxy*) do objeto remoto no cliente. Ele reside no Sistema do cliente e age como um *gateway* para o programa cliente;
- **Skeleton:** Este é o objeto que reside no lado do servidor. O *stub* se comunica com o *skeleton* para passar a solicitação para o objeto remoto;
- **RRL (Remote Reference Layer):** É a camada que gerencia as referências feitas pelo cliente para o objeto remoto.

Corba

O CORBA (Common Object Request Broker Architecture) é um padrão proposto pelo OMG (Object Management Group), em 1991. É um consórcio de mais de 800 Empresas que trabalham em um padrão independente de Plataforma e Linguagem para Programação Distribuída.

CORBA não é uma Linguagem de Programação, nem é uma ferramenta de software. É uma especificação para integrar componentes de aplicativos distribuídos. Ele define os métodos padrão para acessar objetos remotos e para comunicação entre eles, independentemente do Sistema Operacional ou da Linguagem de Programação.

Um Sistema baseado em CORBA é um conjunto de objetos que separam servidores de clientes com uma interface de programador bem definida.

Eles fornecem vários serviços, como localizar um servidor pelo nome e facilitar a comunicação entre um servidor e um cliente, incluindo a passagem de argumentos para os métodos e obter resultados de suas chamadas remotas.

Uma implementação específica do ambiente CORBA, bem como objetos que fornecem serviços e clientes, pode ser feita em uma das várias linguagens de programação. Para garantir sua interoperabilidade, eles devem estar em conformidade com a especificação definida na Linguagem específica e universal – o IDL (*Interface Definition Language*).

A especificação é usada para gerar vários arquivos de origem na Linguagem de Programação de destino, usada para implementar objetos CORBA.

Existem várias implementações do ambiente CORBA disponíveis no Mercado. Eles são mais ou menos compatíveis devido às diferenças nas versões da especificação CORBA. O Java SDK forneceu essa implementação desde a versão 1.3.

O ORB (*Object Request Broker*) é uma parte fundamental da implementação do CORBA.

É um software projetado para garantir a comunicação entre objetos na Rede. Em particular, permite a localização de objetos remotos, passando argumentos para métodos e retornando os resultados de chamadas remotas. Pode redirecionar uma solicitação para outro agente do ORB.

O CORBA define regras gerais para a comunicação entre agentes de maneira independente da Linguagem, já que todos os Protocolos são definidos na Linguagem IDL.

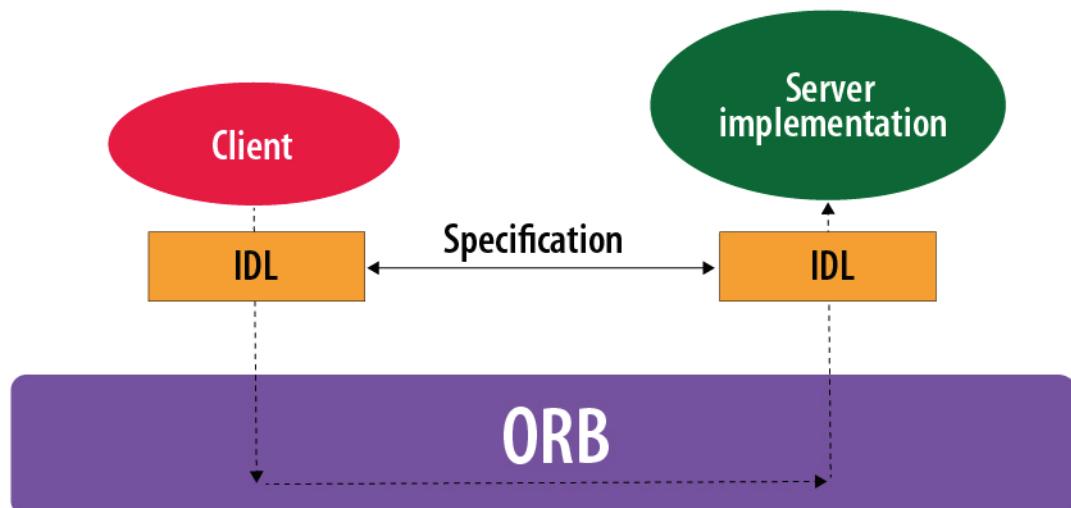


Figura 5 – Um cliente envia uma solicitação para um servidor através da ORB

Fonte: Acervo do Conteudista



Você Sabia?

Existem Tecnologias que podem ser utilizadas como alternativa ao CORBA.

A troca de mensagens via *sockets* ou utilizando RPC, RMI que possui uma restrição de Linguagem de implementação com *Java* e o DCOM que, basicamente, é a implementação CORBA da *Microsoft*.



Em Síntese

Quais as vantagens de CORBA em relação a abordagens mais tradicionais, como *sockets* e RPC?

A comunicação em Sistemas distribuídos através de *sockets* ou mesmo de RPC é feita de forma não padronizada, sem flexibilidade na hora do endereçamento de objetos e de referências no servidor.

Já o CORBA disponibiliza meios de endereçamento transparentes e exatos, tornando o trabalho do programador da aplicação mais simples.

Além disso, CORBA utiliza entidades intermediárias para a comunicação dos seus *stubs*, ao contrário dos métodos tradicionais, que não possuem uma facilidade deste tipo.

DCOM

DCOM (*Distributed Component Object Model*) é a solução da *Microsoft* para computação distribuída. Ele permite que um aplicativo cliente inicie remotamente um objeto de servidor DCOM em outra máquina e invoque seus métodos. Então, funcionalmente, é semelhante ao CORBA e ao RMI. No entanto, diferentemente do RMI, que é dependente de *Java*, o DCOM é independente de Linguagem e da Plataforma – assim como o CORBA.

A principal diferença entre o DCOM e o CORBA está na implementação. O DCOM é um padrão binário, enquanto o CORBA é apenas uma especificação: ele define as interfaces de como os clientes podem usar os objetos, mas não define a própria implementação.

Assim, diferentes fornecedores estão livres para implementar ORBs CORBA de muitas maneiras diferentes, o que torna o CORBA possivelmente mais flexível.

Só porque o DCOM é um padrão binário, no entanto, não significa que não seja independente de plataforma. Porém o alvo do DCOM continua sendo as aplicações que estão sob o Sistema Operacional da *Microsoft*, o *Windows*.

CORBA x DCOM

- DCOM é um padrão binário e CORBA é uma especificação. As IDLs são incompatíveis;
- DCOM IDL não oferece suporte a exceções, enquanto CORBA IDL oferece;
- DCOM somente permite herança de interface, enquanto CORBA permite herança de implementação;
- Ambas suportam invocação de interface estática e dinâmica;
- O DCOM visa, principalmente, a PCs com soluções corporativas sendo secundário, enquanto o CORBA é exatamente o oposto.

Web Services

Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. *Web Service* é baseado em padrões abertos (XML, SOAP, HTTP etc.) que interagem com outros aplicativos da Web com o objetivo de trocar dados.

O padrão *Web Service* é mantido pela W3C (*World Wide Web Consortium*).

Um *Web Service* é uma entidade de *software* independente de Linguagem, baseada em padrões, que aceita solicitações formatadas de outras entidades de *software* em máquinas remotas por meio de Protocolos de comunicação e utiliza Protocolos neutros, sem um proprietário ou fornecedor.

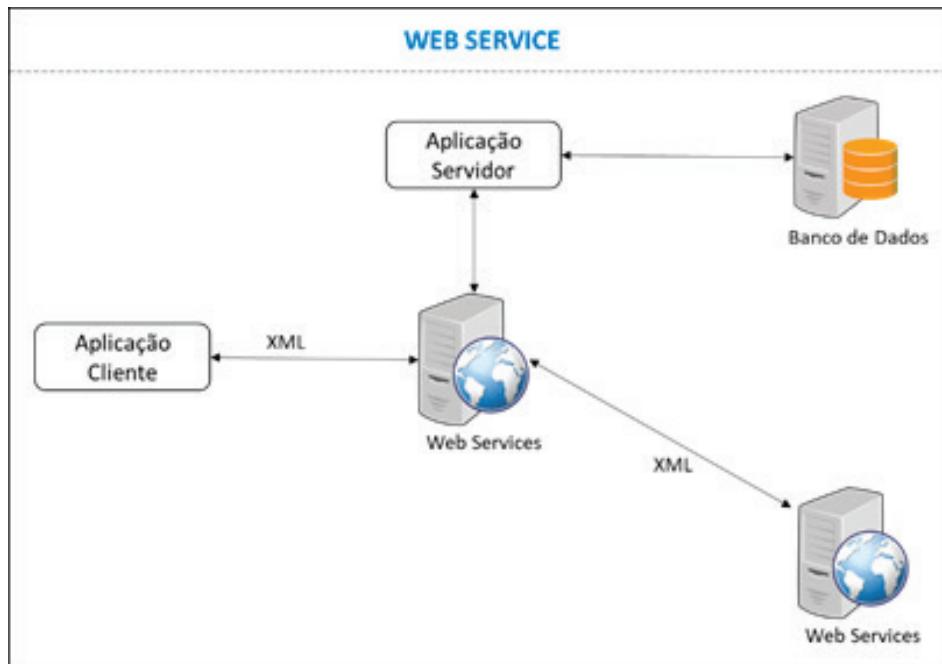


Figura 6 – Exemplo de um sistema *Web Service*

Fonte: Acervo do Conteudista

Na Figura acima, você pode perceber que temos uma aplicação que necessita realizar uma solicitação para o servidor.

Em outras Aplicações Distribuídas, seria necessário conhecer em detalhes todo o formato da interface do retorno da comunicação, porém, com Web Services, os dados serão comunicados por meio do formato XML (*Extensible Markup Language*), que facilita a troca de informações entre aplicações, pois é autodocumentado, o próprio formato descreve a sua estrutura e nomes de campos, assim como valores válidos.

Um *Web Service* pode se comunicar com outros serviços expostos ou se comunicar com uma aplicação servidora (*back-end*) para transações com SGBD (Sistemas Gerenciadores de Banco de Dados).



Importante!

O XML (*eXtensible Markup Language*) é uma recomendação do W3C para a criação de Linguagens de Marcação que permitem a estruturação, a descrição e o intercâmbio de dados.

Benefícios

- **Fracamente acoplado:** cada serviço existe independentemente dos outros serviços que compõem o aplicativo. Partes individuais do aplicativo podem ser modificadas sem afetar áreas não relacionadas;
- **Facilidade de integração:** permitem trocar dados entre Organizações e Departamentos, sem necessidade de unificar a plataforma de *software* (Sistemas Operacionais, Linguagens ou Banco de Dados);
- **Reutilização de código:** um serviço é codificado e pode ser usado repetidamente por vários aplicativos diferentes.

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Sites

Corba Platform Category – Specifications associated

<http://bit.ly/2005r1l>

Vídeos

Modelo OSI e TCP/IP - Como funciona o processo de comunicação em redes

Modelo OSI e TCP/IP – Como funciona o processo de comunicação em Redes. Redes Brasil. Acesso em: 10 mar. 2019.

<https://youtu.be/oz8gvGIUKFw>

Web Services (O que é, motivos para uso, como funciona, Protocolos SOAP/REST)

Web Services (O que é, motivos para uso, como funciona, Protocolos SOAP/REST). Canal TI. Acesso em: 10 mar. 2019.

<https://youtu.be/RIB5wcuFvcc>

Leitura

RMI – Remote Method Invocation

MARCIANO, Carlos Eduardo. **RMI – Remote Method Invocation.** GTA – UFRJ. Acesso em: 10 mar. 2019.

<http://bit.ly/2Q2uQYA>

Referências

COLOURIS, George *et al.* **Sistemas Distribuídos:** Conceitos e Projeto. 5.ed. Porto Alegre: Bookman, 2013.

JAVA RMI INTRODUCTION. Disponível em: <www.tutorialspoint.com>. Acesso em: 10 mar. 2019.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Sistemas distribuídos:** princípios e paradigmas. 2. ed. São Paulo: Pearson Prentice Hall, 2007.

TRANSPORT PROTOCOLS FOR SOCKETS. IBM Knowledge Center, 24 de outubro de 2014. Disponível em: <https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.cbcpx01/cbc1p2327.htm>.



Cruzeiro do Sul
Educacional