SISTEMAS DISTRIBUÍDOS COM JAVA

EQUIPE:
DAVI FERREIRA DE LIMA
ANTÔNIO CRISTIANO
MARCELO RAMALHO
ANTÔNIO JURACY
ARQUIMEDES
GABRIEL

PROFESSOR:
MARCELLUS CYSNE

INTRODUÇÃO

Os avanços tecnológicos, a popularização da Internet e a evolução das redes de computadores, resultaram no surgimento de aplicações distribuídas, cada vez mais aumenta a necessidade de compartilhamento de informações. Em outras palavras, aumenta a necessidade de interoperabilidade entre sistemas.

"Interoperabilidade define se dois componentes de um sistema, desenvolvidos com ferramentas diferentes, de fornecedores diferentes, podem ou não atuar em conjunto." (Lichun Wang, Instituto Europeu de Informática – CORBA Workshops)

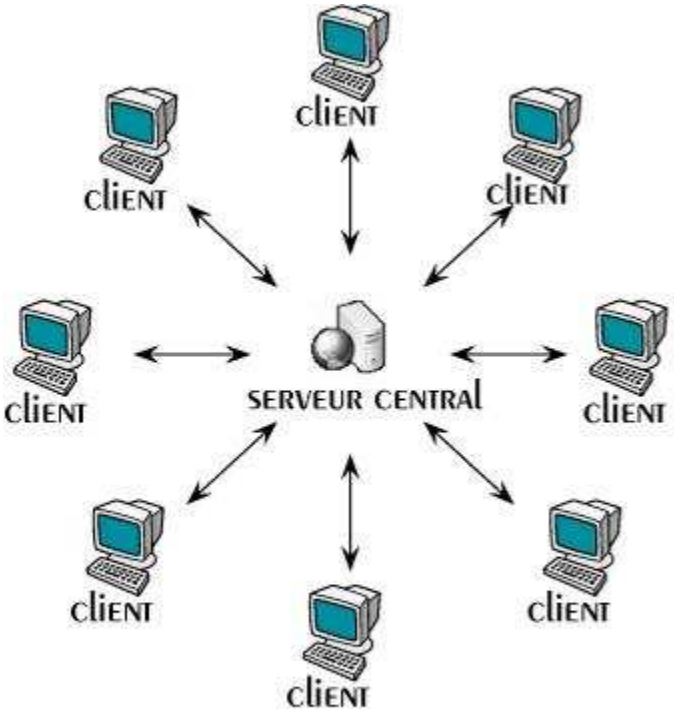
INTRODUÇÃO

- Em principio, as redes de computadores eram baseadas na comunicação entre **cliente e servidor**. Mais adiante, com o surgimento da programação **orientada a objetos**, surgiram novos **middlewares**, que possuem como função possibilitar que as aplicações possam ser escritas de modo mais **independente** possível do hardware e do sistema operacional, permitindo assim que um mesmo código de aplicação possa ser carregado e executado em diferentes equipamentos receptores.
- Em resumo, o **middleware** é um software capaz de interpretar os aplicativos e traduzi-los na linguagem do sistema operacional em que ele reside. Exemplos como, CORBA, DCOM e **RMI** (JAVA), onde o processamento passou a ser repassado para vários servidores.

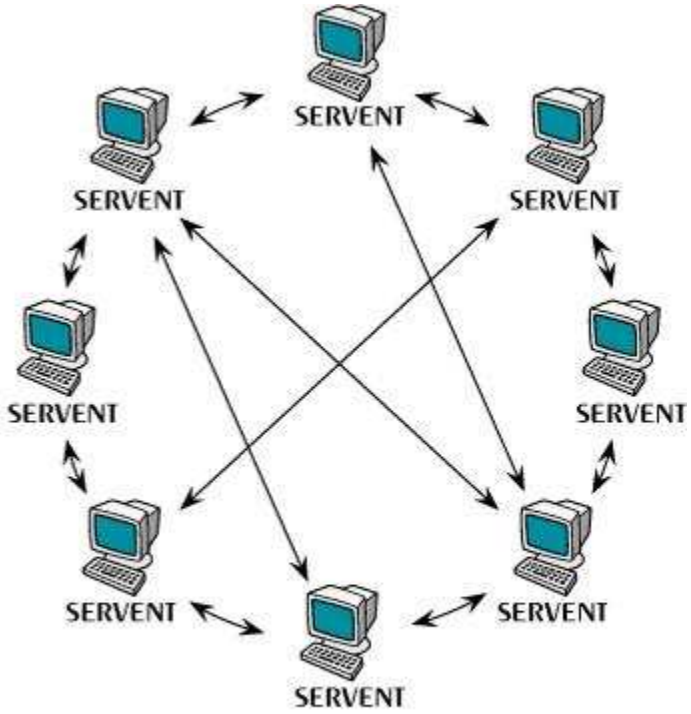
Sun
ORACLE

Java

INTRODUÇÃO



Architecture client-serveur



Architecture pair-à-pair

Sun
ORACLE

Java

LINGUAGEM JAVA

Linguagem de programação orientada a objetos

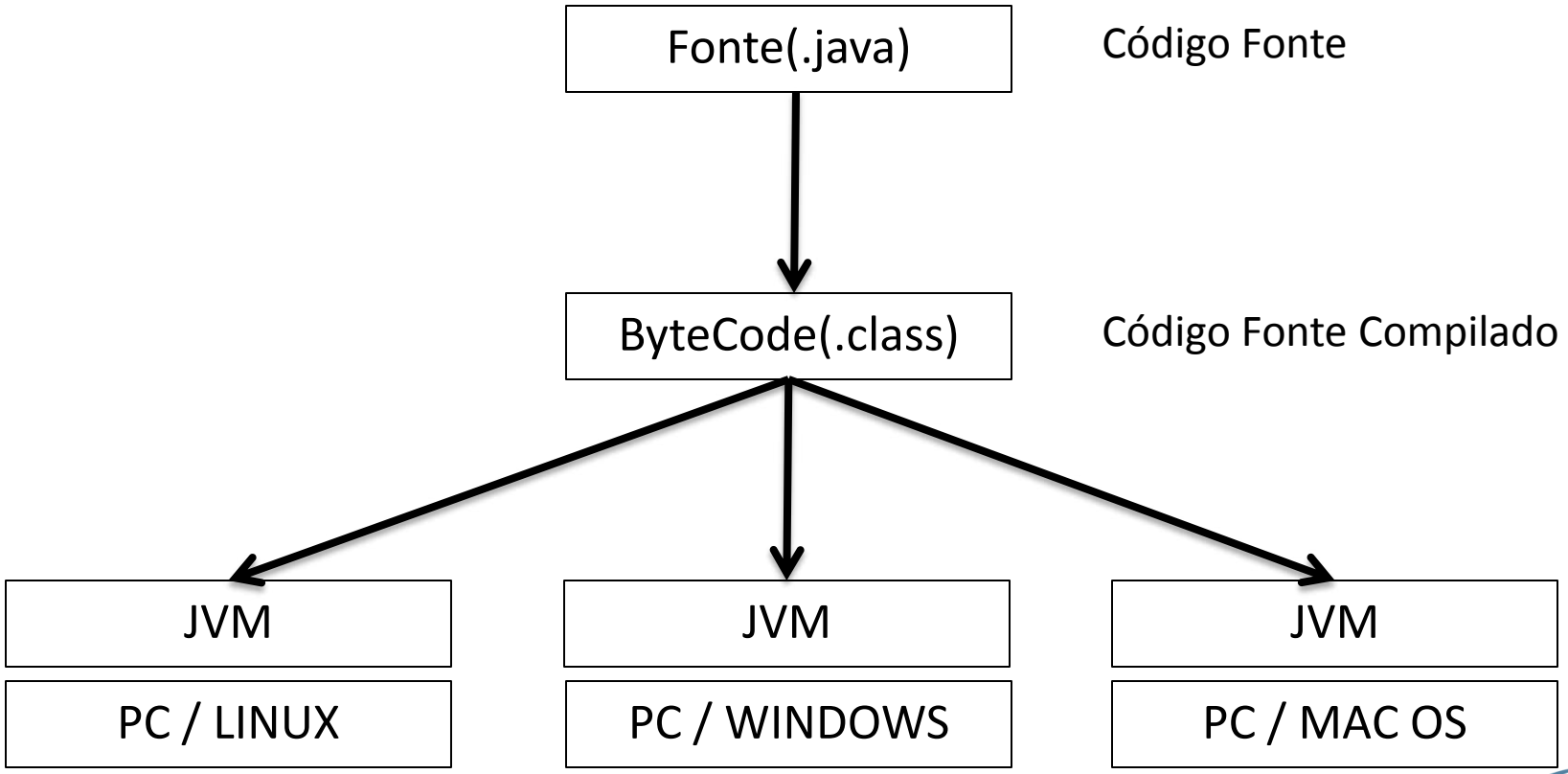
- *Familiar (sintaxe parecida com C)*
- *Simples e robusta (minimiza bugs, aumenta produtividade)*
- *Suporte nativo a threads (+ simples, maior portabilidade)*
- *Dinâmica (módulos, acoplamento em tempo de execução)*
- *Com coleta de lixo (menos bugs, mais produtividade)*
- *Independente de plataforma*
- *Segura (vários mecanismos para controlar segurança)*
- *Código intermediário de máquina virtual interpretado*
- *(compilação rápida - + produtividade no desenvolvimento)*
- *Sintaxe uniforme, rigorosa quanto a tipos (código mais*
- *simples, menos diferenças em funcionalidades iguais)*

Sun

ORACLE

Java

LINGUAGEM JAVA



Sun
ORACLE

Java

PRODUTOS E APIs DO JAVA

Java possui uma coleção de APIs (bibliotecas) padrão que podem ser usadas para construir aplicações

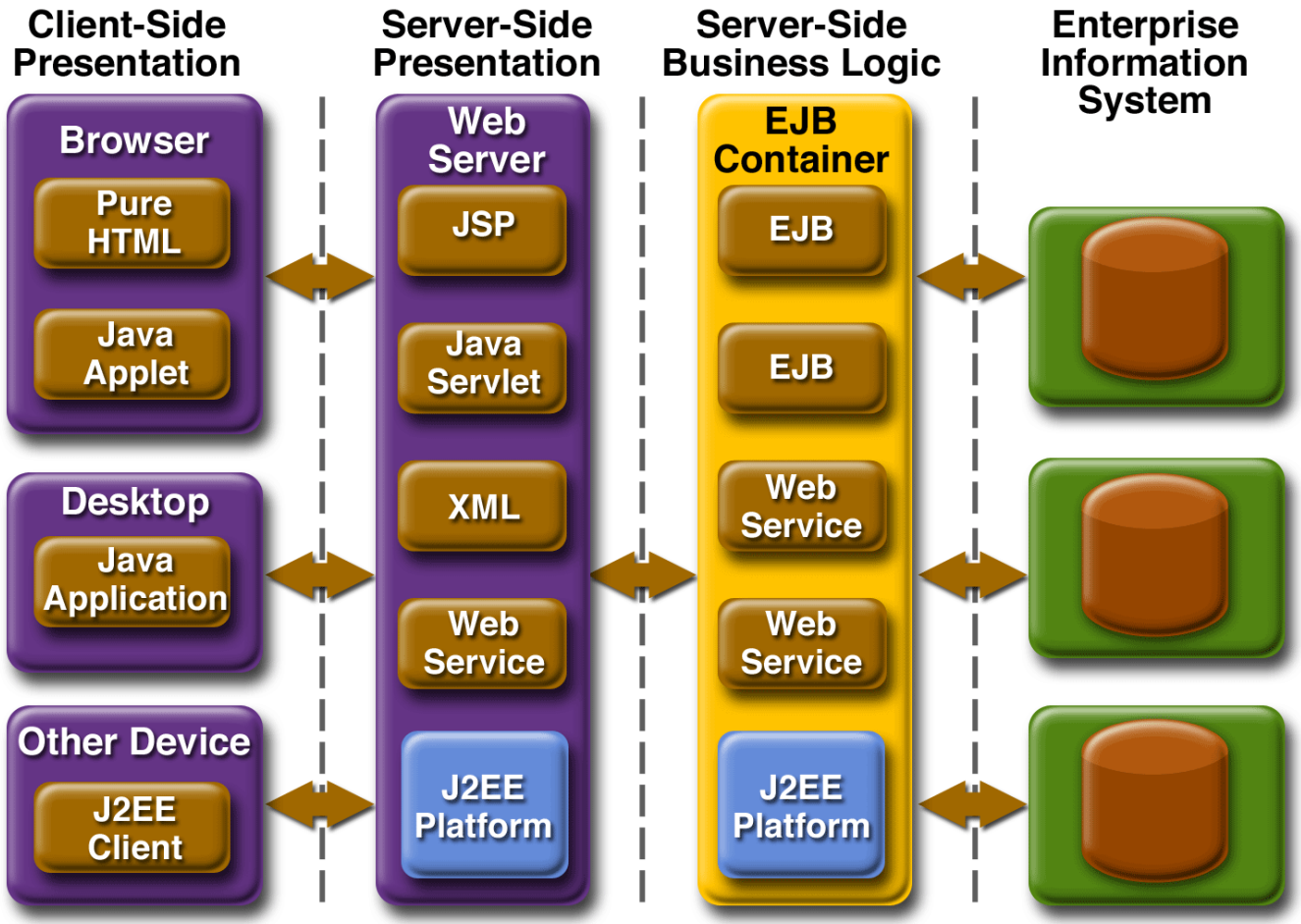
- *Organizadas em pacotes* ***(java.*, javax.* e extensões)***

*usadas pelos ambientes de execução (**JRE**) e de desenvolvimento (**SDK**)*

As principais APIs são distribuídas juntamente com os produtos para desenvolvimento de aplicações

- ***Java 2 Standard Edition (J2SE):*** *ferramentas e APIs essenciais para qualquer aplicação Java (inclusive GUI)*
- ***Java 2 Enterprise Edition (J2EE): ferramentas e APIs para o desenvolvimento de aplicações distribuídas e não distribuídas.***
- ***Java 2 Micro Edition (J2ME):*** *ferramentas e APIs para o desenvolvimento de aplicações para aparelhos portáteis*

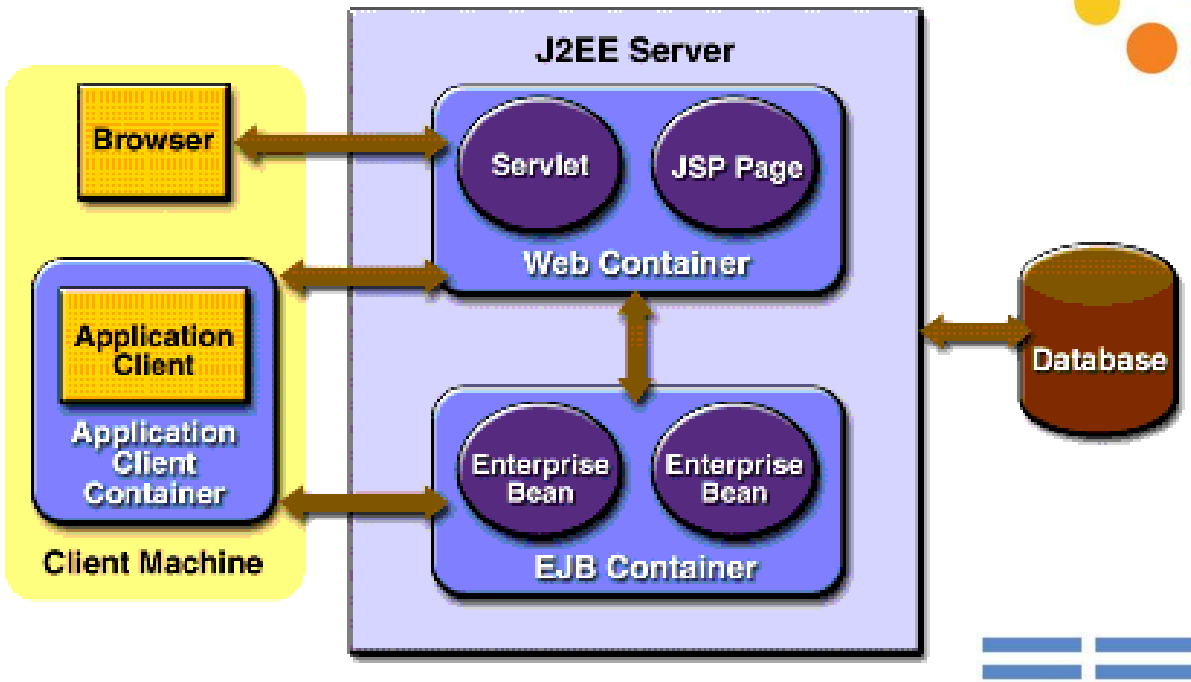
API J2EE



Sun
ORACLE

Java

CONTAINER J2EE



JBoss™

APACHE GERONIMO

IBM®

Sun ORACLE

Java™

SISTEMAS DISTRIBUÍDOS

- Uma definição de sistemas distribuídos é aquela na qual os componentes de hardware e software localizados em computadores interligados por rede, comunicam e coordenam suas ações somente através da troca de mensagens (COULOURIS, 2001).

O QUE FAZ UMA APLICAÇÃO DISTRIBUÍDA?

- Obtém dados de fontes remotas:

  Páginas HTML,arquivos de imagens,dados relacionais e semi-estruturados(XML),etc;
- Acessa continuamente informações de conteúdo dinâmico:

  Cotação de ações,notícias,monitoramento remoto de sistemas;
- Envia dados para fontes remotas:

  Servidores de Arquivos

O QUE FAZ UMA APLICAÇÃO DISTRIBUÍDA?

- Interage com outras aplicações de modo “ponto-a-ponto”(P2P)
- Jogos,bate-papo,compartilhamento de arquivos
- Busca informações na Web
- Realiza transações de Comércio Eletrônico
- E muito mais!
- Computação móvel/ubíqua(J2ME)
- TV interativa(ex.:SBTVD)
- Trabalho colaborativo

SISTEMAS DISTRIBUÍDOS
- WEB SERVICE -

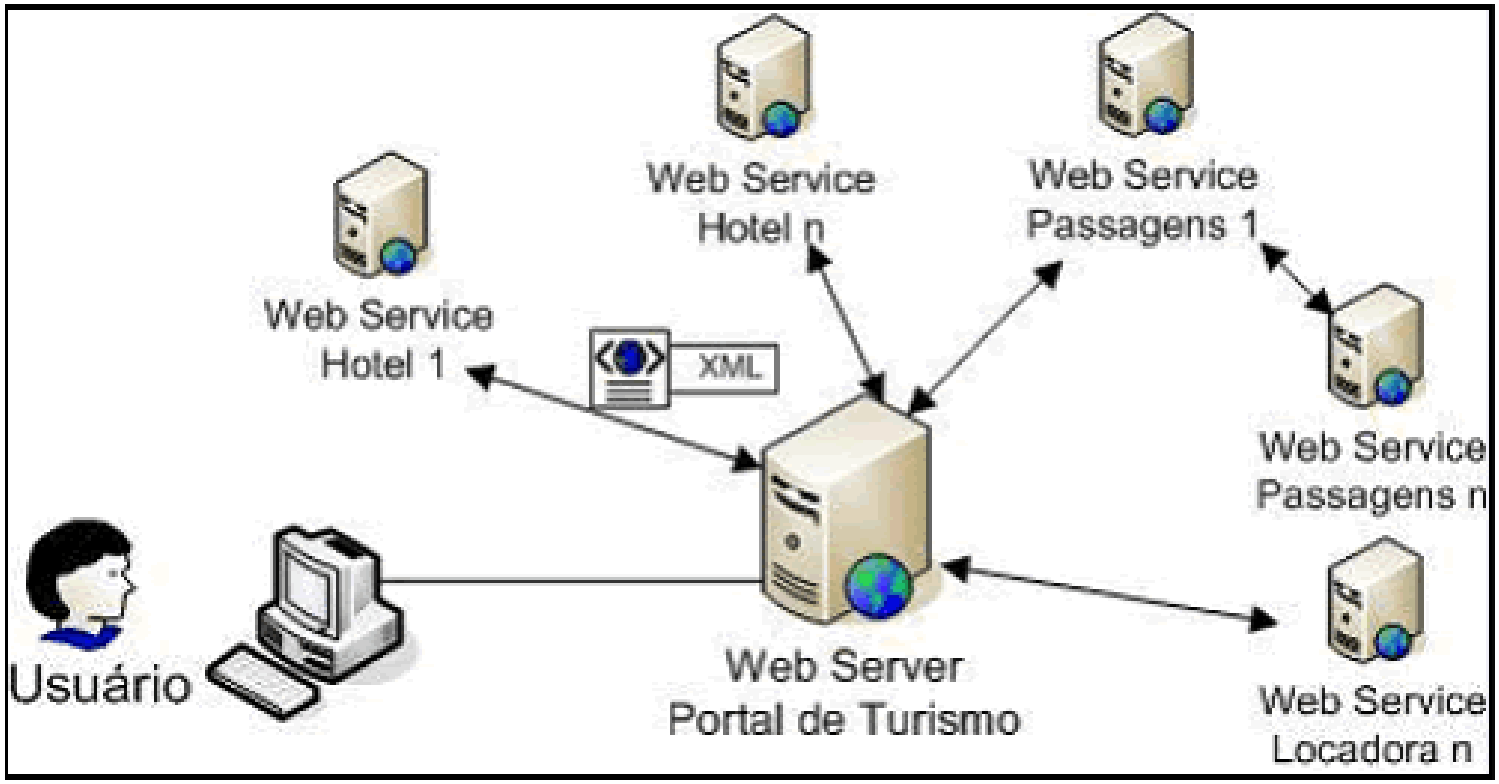
- O principal objetivo dos Web Services é proporcionar a interoperabilidade entre sistemas distribuídos, independente da plataforma e da linguagem de programação utilizada por eles, disponibilizando uma melhor interligação destas aplicações.

  Esta interligação tem como princípio facilitar os processos de negócios, proporcionando a softwares isolados passarem a funcionar de forma conjunta com os demais. Um projeto bem elaborado busca a diminuir custos, aumentar a produtividade e uma maior oportunidade de rendimento. Mas, diversos quesitos devem ser levados em conta, para evitar que esta comunicação de plataformas possa trazer prejuízos em vez de benefícios.

Sun
ORACLE

Java

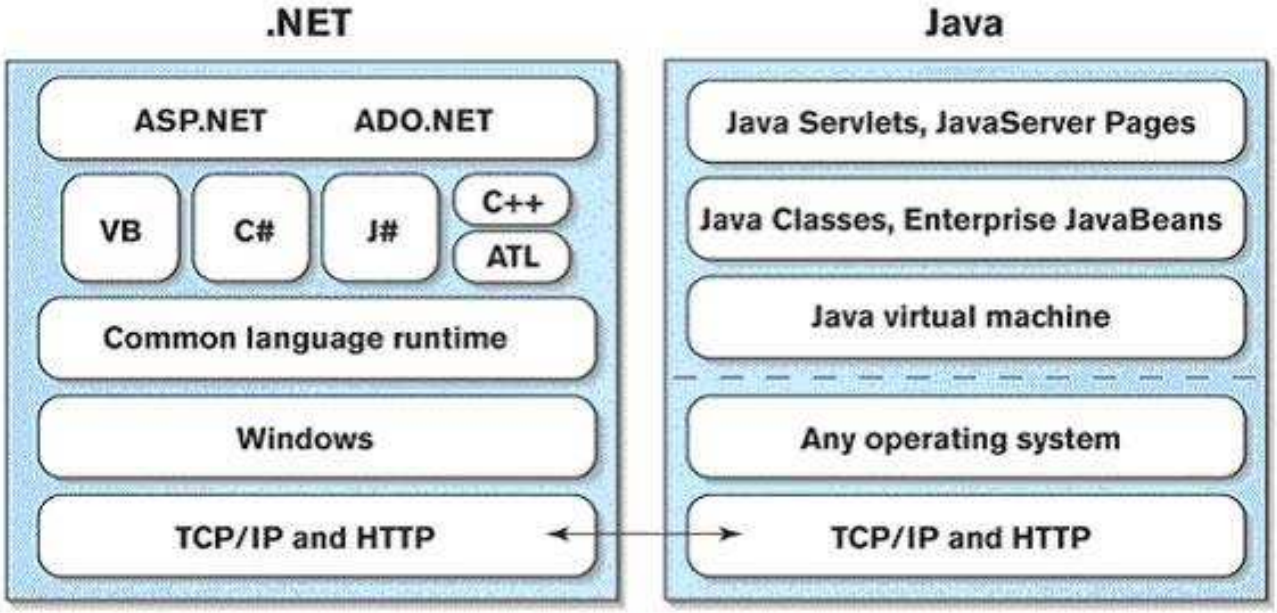
SISTEMAS DISTRIBUÍDOS
- WEB SERVICE -

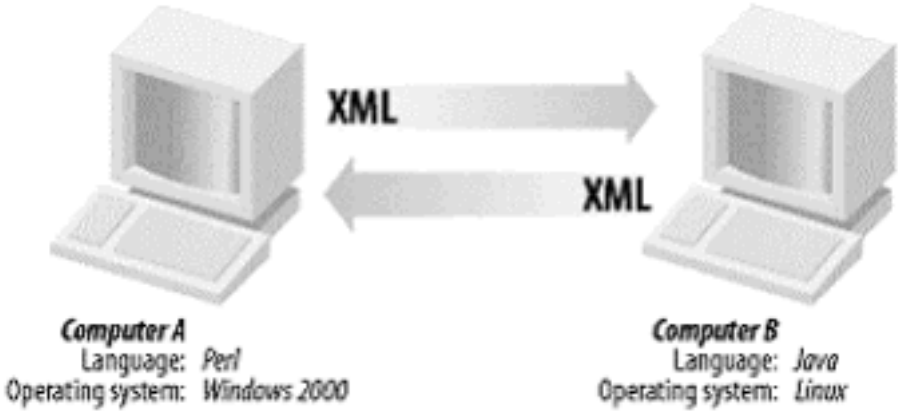


Sun
ORACLE

Java

SISTEMAS DISTRIBUÍDOS
- WEB SERVICE -





Sun
ORACLE

Java

SISTEMAS DISTRIBUÍDOS
- P2P-

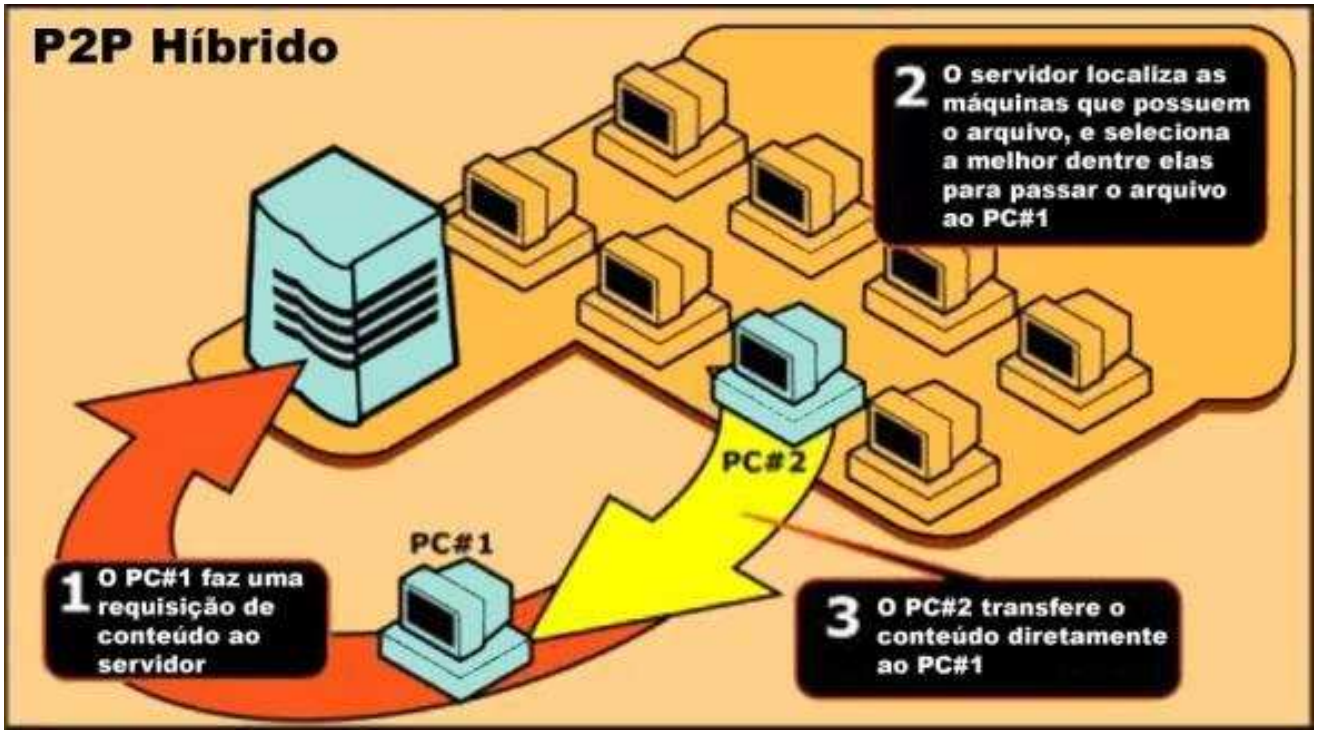
- São usados servidores para tarefas como autenticação de usuários, serviços de diretório e mapeamento de recursos disponíveis. A idéia básica é que os nós possam contatar algum servidor para iniciar a transação, ou para algum dos serviços acima. Em seguida, o servidor devolve ao nó inicial alguma informação pertinente que permita a ele conectar-se diretamente com outro nó e efetuar a transação. É o modelo usado na grande maioria dos sistemas P2P.

  Kazaa, Napster, eMule, ICQ, Gnutella são alguns softwares que utilizam P2P para oferecer os serviços aos seus inúmeros usuários.

Sun
ORACLE

Java

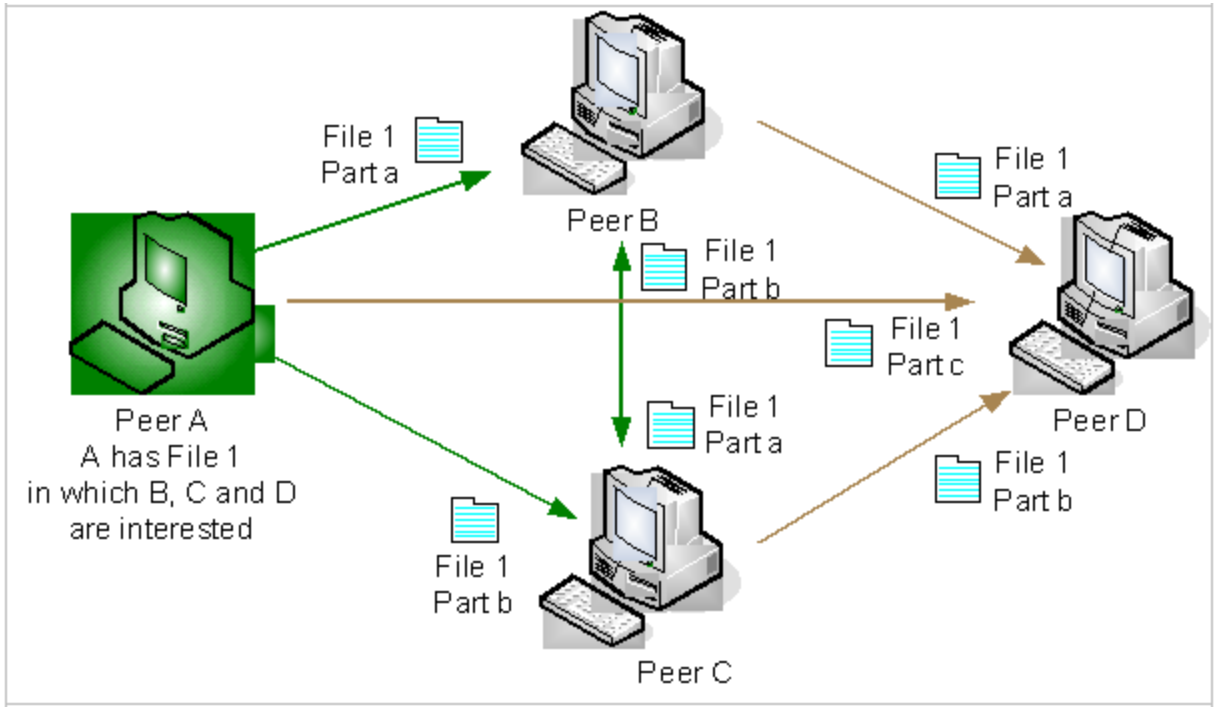
SISTEMAS DISTRIBUÍDOS
- P2P-



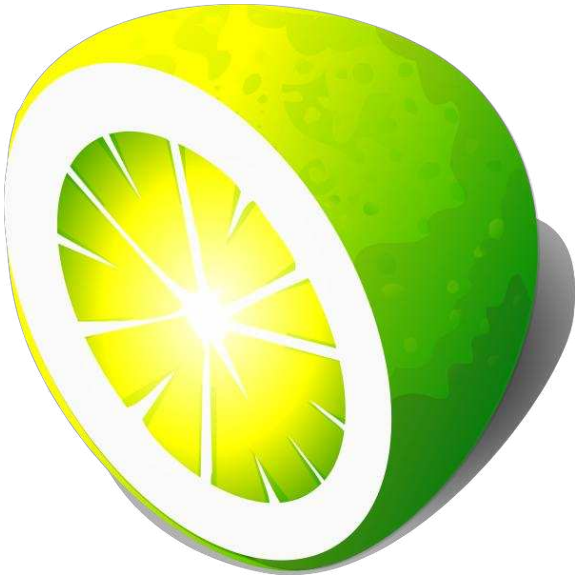
Sun
ORACLE

Java

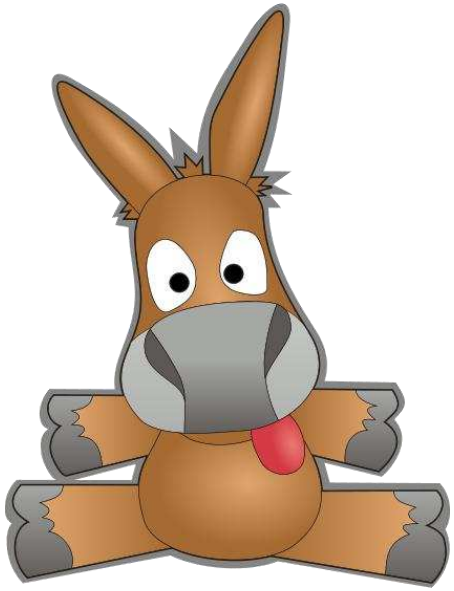
SISTEMAS DISTRIBUÍDOS
- P2P-



SISTEMAS DISTRIBUÍDOS
- P2P-



LIMEWIRE



EMULE

NAPSTER

Sun
ORACLE

Java

EXEMPLO DE CÓDIGO
- ACESSANDO OBJETOS REMOTOS -

```
import java.net;
import java.rmi;

public interface NomeDoLivro extends Remote
{
    String imprimeNomeDoLivro() throws RemoteException;
}
```

EXEMPLO DE CÓDIGO
- ACESSANDO OBJETOS REMOTOS -

```
import java.net;
import java.rmi;
import java.rmi.server.*;
import java.rmi.registry.*;

public class NomeDoLivroImpl extends UnicastRemoteObject implements NomeDoLivro
{
	public NomeDoLivroImpl() throws RemoteException
	{
		super();
	}

	 public  String imprimeNomeDoLivro() throws RemoteException
	{
		return “Sistemas Distribuidos com Java: Conceito RMI”;
	}

}
```

EXEMPLO DE CÓDIGO
- ACESSANDO OBJETOS REMOTOS -

```
import java.net;
import java.rmi;
import java.rmi.server.*;
import java.rmi.registry.*;

public class NomeDoLivroServer
{
	public static void main(String args[])
	{
		System.setSecurityManager(new RMISecurityManager());
		try
		{
			NomeDoLivroImpl obj = new NomeDoLivroImpl();
			Naming.rebind("NomeDoLivroServer",obj);
		}catch(Exception e)
			{
				System.out.println(e.getMessage());
			}
	}
}
```

EXEMPLO DE CÓDIGO
- ACESSANDO OBJETOS REMOTOS -

```
import java.net;
import java.rmi;
import java.rmi.server.*;
import java.rmi.registry.*;

public class NomeDoLivroClient
{
	public static void main(String args[])
	{
		System.setSecurityManager(new RMISecurityManager());
		try
		{
		NomeDoLivro obj = (NomeDoLivro ) Naming.lookup(“rmi://server/NomeDoLivroServer”) ;
		 System.out.println(obj. imprimeNomeDoLivro() );
		}catch(Exception e)
				{
					System.out.println(e.getMessage());
				}
	}
}
```

EXEMPLO DE CÓDIGO
- ACESSANDO OBJETOS REMOTOS -

E o que isso tudo faz?

Quando a classe do servidor e executada, ela usa o metodo rebind da classe Naming do pacote java.rmi para se registrar no registro RMI daquele servidor:

NomeDoLivroImpl obj = new NomeDoLivroImpl();

Se tudo der certo, os clientes que foram usar esta classe conseguirao acessa-la atraves de uma URL com o protocolo RMI, como argumento do metodo lookup:

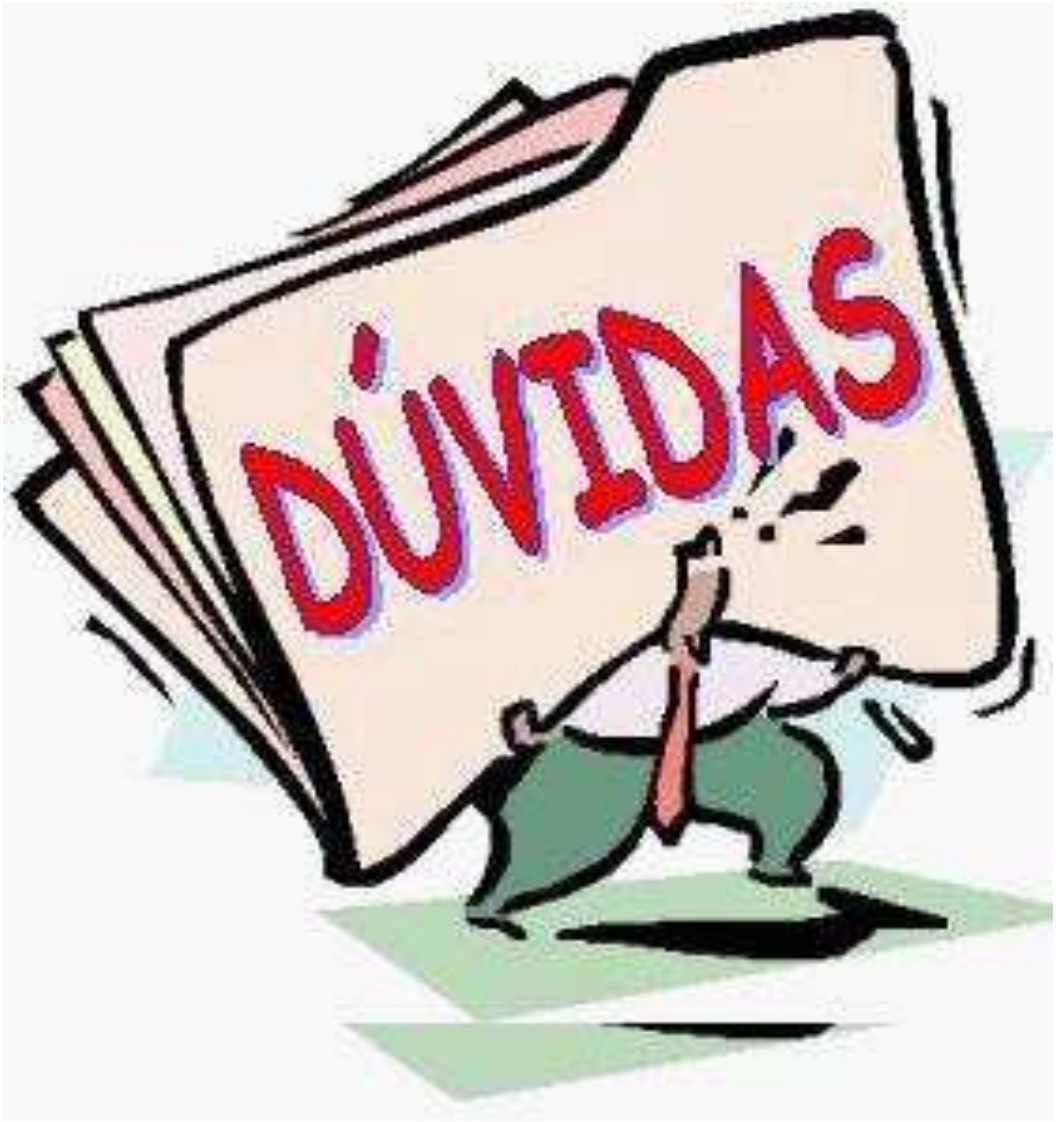
NomeDoLivro obj = (NomeDoLivro) Naming.lookup(“rmi://server/NomeDoLivroServer”) ;

Nesse momento, o cliente encontra disponivel um “stub” da aplicacao servidor e faz o download do mesmo para o micro local.Comeca entao a conversa, momento em que trabalhamos com o objeto como se ele fosse “local” .

System.out.println(obj. imprimeNomeDoLivro());

Ao chamarmos um metodo do objeto remoto, todo o ciclo de execucao mencionado anteriormente ocorre.Entao,com a chamada acima, o que e passado e:

```
public  String imprimeNomeDoLivro() throws RemoteException
    {
        return “Sistemas Distribuidos com Java: Conceito RMI”;
    }
```



Sun
ORACLE

Java

OBRIGADO !!!!