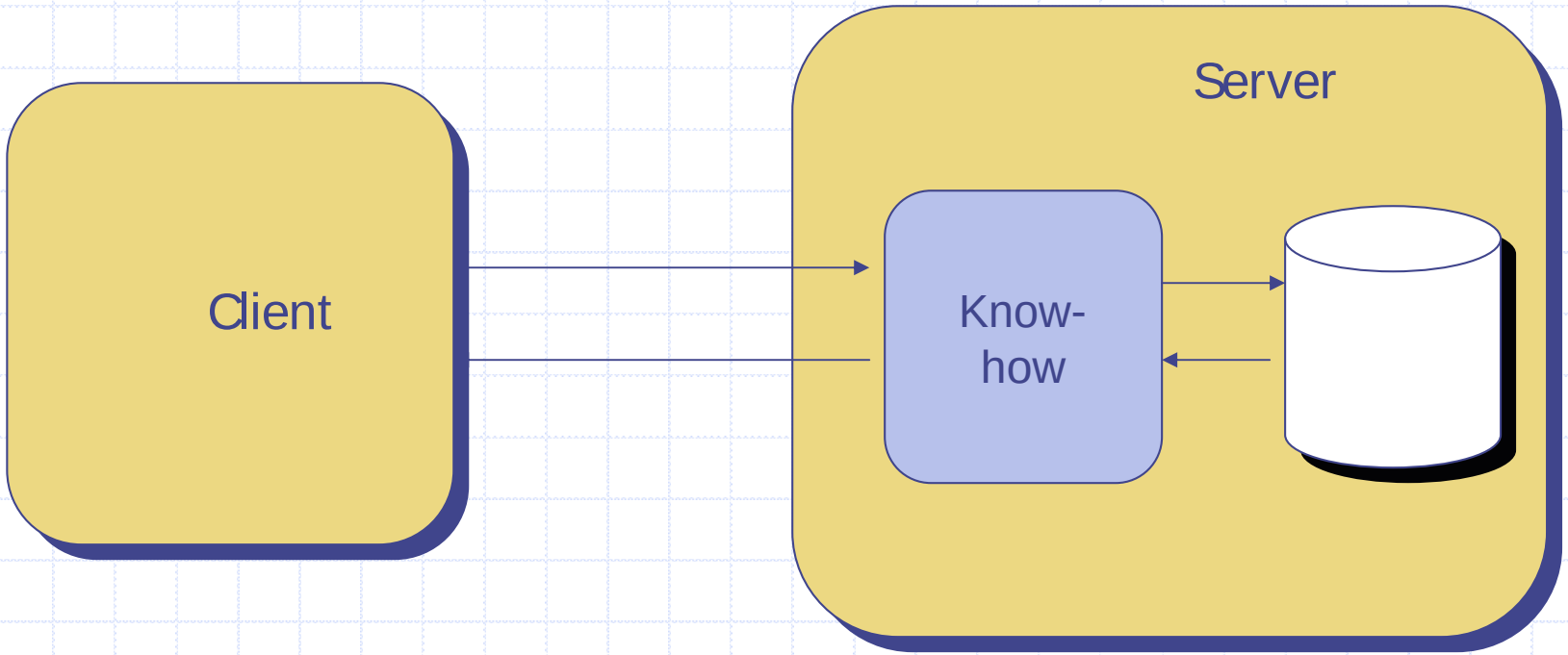
Paradigmas de Computação Distribuída

Computação na Rede:

- Cliente/Servidor,
- Software de Código Móvel.

Paradigma Cliente/Servidor



Paradigma Cliente/Servidor

- O servidor anuncia um conjunto de serviços que ele provê acesso para alguns recursos (por exemplo, banco de dados).
- O código que executa esses serviços é hospedado localmente pelo servidor.
- O servidor detém o know-how.

Paradigma Cliente/Servidor

- O próprio servidor processa o serviço e assim, tem a capacidade do processador.
- Se o cliente está interessado em acessar um recurso hospedado pelo servidor, o cliente usa um ou mais dos serviços providos pelo servidor.

Paradigma Cliente/Servidor

- O cliente necessita decidir qual dos serviços ele deve usar.
- O servidor tem: o know-how, os recursos e o processador.
- A maioria das aplicações distribuídas e dos sistemas distribuídos são baseados neste paradigma.

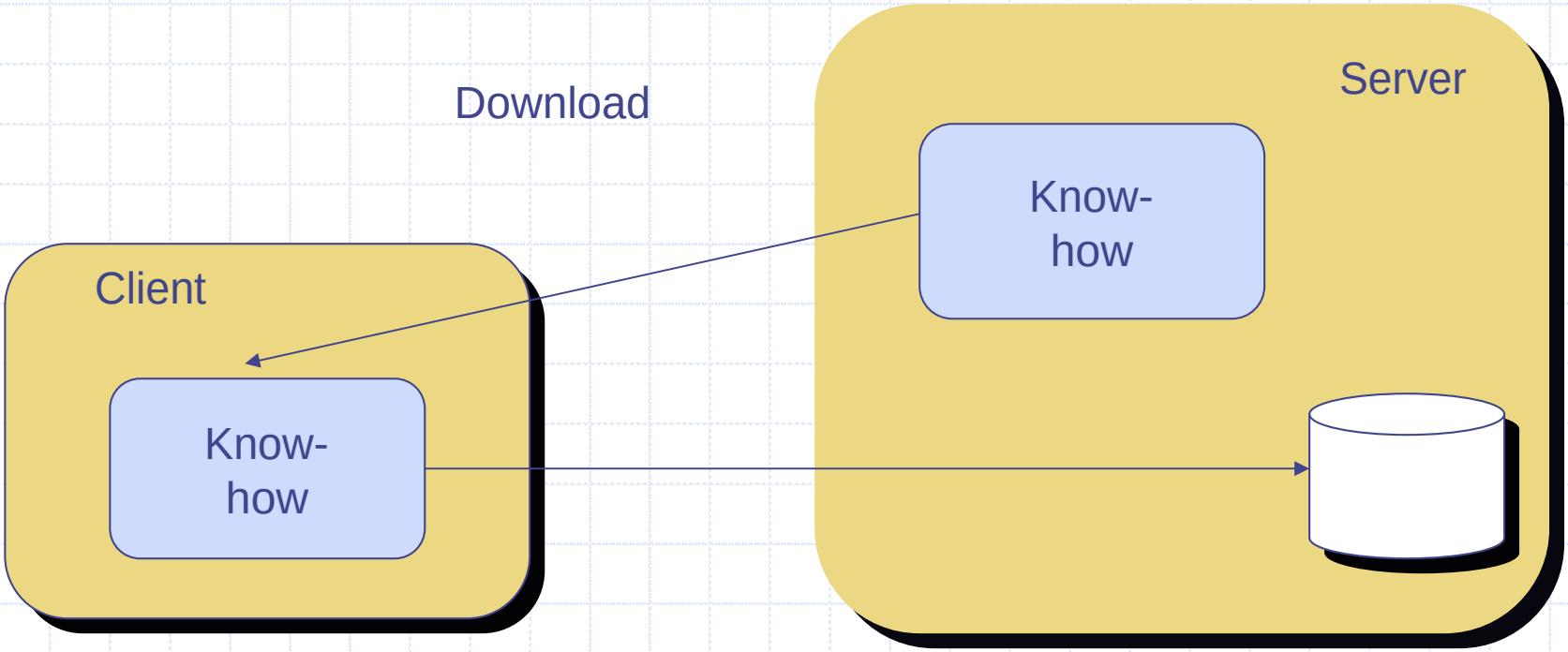
Paradigmas Cliente/Servidor

- Cliente/Servidor com Processos
- Cliente/Servidor com Objetos Distribuídos
  (CORBA, DCOM, RMI, ...)
- Cliente/Servidor com Objetos para Web
  (RPC/XML, SOAP/XML)

Paradigma do Software de Código Móvel

- Código Sob-Demanda
- Agentes Móveis

Paradigma do Código Sob-Demanda



Paradigma do Código Sob-Demanda

- Primeiro obtém-se o know-how quando necessita-se. Supõe-se que um cliente é incapaz para executar sua tarefa por causa da falta de código (know-how).
- Um host na rede provê o código necessário.
- A computação é executada no cliente.

Paradigma do Código Sob-Demanda

- O cliente utiliza a capacidade do processador, bem com os recursos locais.
- O cliente não necessita o código pre-instalado, porque todo o código necessário será “donwloaded”.
- O cliente tem os recursos e o processador e o host tem o know-how.

Paradigma do Código Sob-Demanda

- Applets em Java são “donwloaded” in Web browsers e executam localmente.
- Servlets são “uploaded” para servidores Web remotos e executam lá.

O que é um Agente de Software

Agente de Software

- Perspectiva do usuário final:

  Agentes são definidos como programas que assistem pessoas e agem em nome delas. Pessoas delegam trabalho a eles (agentes).

Agente de Software

- Podem ser encontrados em sistemas operacionais, bases de dados, redes, ...
- Agentes podem ser construídos em diferentes tipos e operam em muitos cenários e configurações.

Agente de Software

- Quais propriedades esses programas compartilham, e que constituem a essência de ser um agente ?

Agente de Software

- O fato que eles são hospedados em um ambiente.
- Habilidade para interagirem com seu ambiente de execução e agirem assincronamente e autonomamente sobre ele.

Agente de Software

- Nenhum ambiente é requerido entregar informação ao agente ou consumir quaisquer que seja sua saída.
- Agentes, simplesmente, agem continuamente no sentido de realização de suas tarefas.

Agente de Software

- Em contraste aos objetos (estáticos) da programação orientada a objeto, agentes são entidades ativas que trabalham de acordo com o seguinte princípio: “Não nos chame, nós chamaremos você!”

Agente de Software

- Perspectiva de sistema

Um agente é um programa que está situado dentro de algum ambiente de execução.

Agente de Software

- Possui as seguintes **propriedades obrigatórias**:
  - **reativa**: sente mudanças no ambiente e age de acordo a essas mudanças.
  - **autônomo**: tem controle sobre suas próprias ações.
  - **dirigido à meta**: é pro-ativo.
  - **continuidade temporal**: está continuamente executando.

Agente de Software

- Possui qualquer das seguintes **propriedades ortogonais**:
  - **comunicativo**: capaz de se comunicar com os outros agentes.
  - **móvel**: pode migrar de um host a outro.
  - **adaptativo**: aprende de acordo com a experiência prévia.
  - **acreditável**: o usuário pode acreditar no que ele realiza.

Agente de Software

- Agente Estacionário

  Um agente estacionário executa somente sobre o sistema onde ele começa sua execução.

- Se ele necessita informação que não está sobre aquele sistema, ele necessita interagir com um agente em um outro diferente sistema, ele tipicamente usa um mecanismo de comunicação tal como uma chamada remota de procedimento (RPC).

Agente de Software

- Agente Móvel

  Em contraste, um agente móvel não está limitado ao sistema onde ele começa a sua execução.
- Ele é livre para viajar entre hosts na rede. Criado em um ambiente de execução, ele pode transportar seu estado e código com ele, para um outro ambiente de execução na rede, onde ele retoma sua execução.

Agente de Software

- Estado:

  os valores dos atributos do agente que auxiliam a ele determinar o que fazer quando ele retoma a execução em seu destino.

- Código:

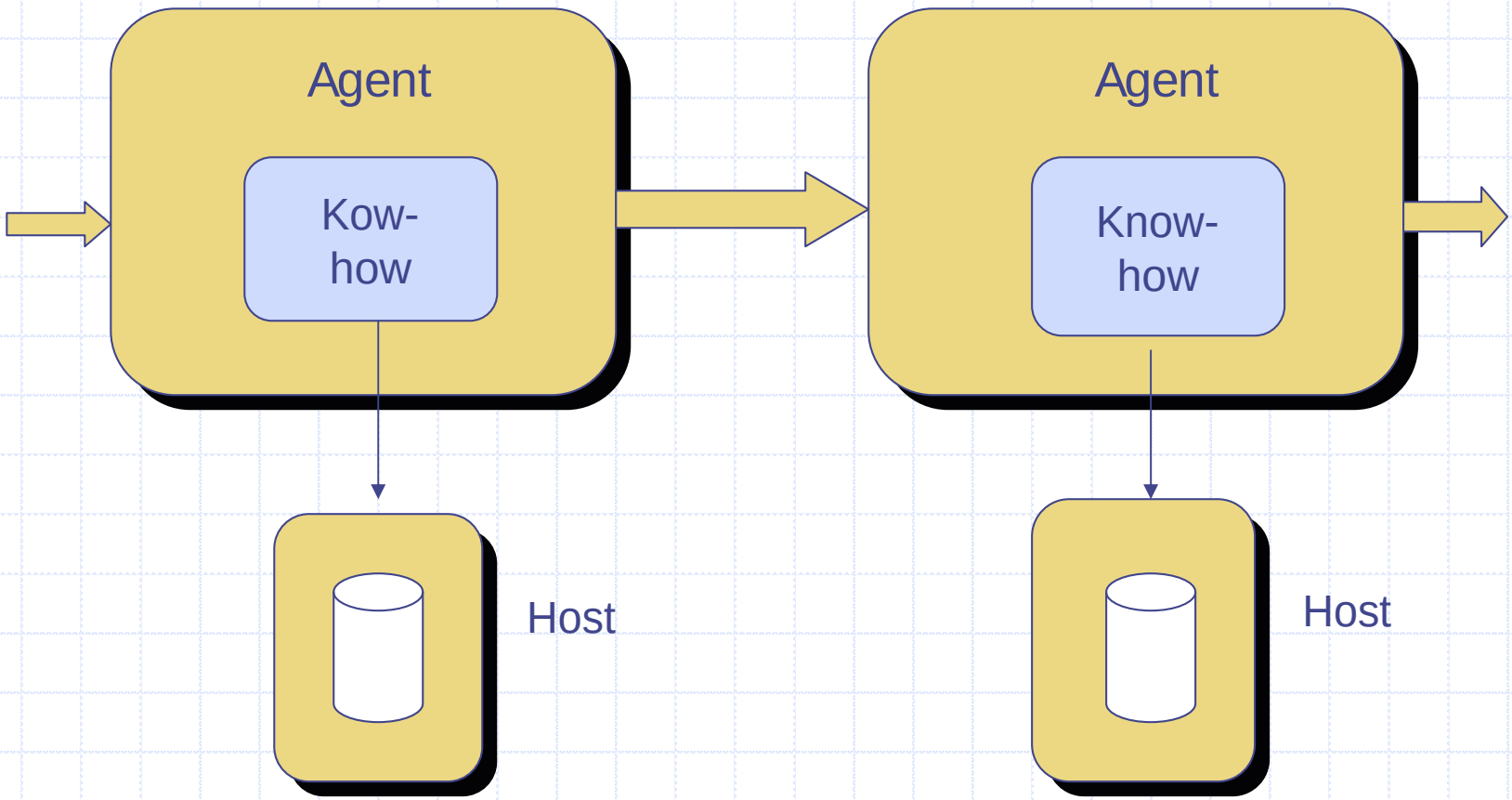
  em um contexto orientado a objeto, o código da classe necessário para o agente executar.

Agente de Software

- Agente Móvel

Um agente móvel pode ser movido para um sistema que contém um objeto, com o qual o agente deseja interagir e assim ter a vantagem de estar no mesmo host como o objeto.

Paradigma dos Agentes Móveis



Paradigma dos Agentes Móveis

- Know-how (na forma de agentes móveis) não está amarrado a um único host, mas ao contrário está disponível através da rede.

Sete Boas Razões para se usar Agentes Móveis

- Reduzem a carga na rede.
- Sobrepujam a latência da rede.
- Encapsulam protocolos.
- Executam assincronamente e autonomamente.
- Adaptam-se dinamicamente.
- Naturalmente heterogêneos.
- Robustos e tolerantes a falhas.

Reduzem carga na rede

- Sistemas distribuídos, frequentemente, confiam em protocolos de comunicação que envolvem múltiplas operações para se conseguir uma dada tarefa.
- Isto é, especialmente verdade quando medidas de segurança são habilitadas. O resultado é o crescimento do tráfego de rede.

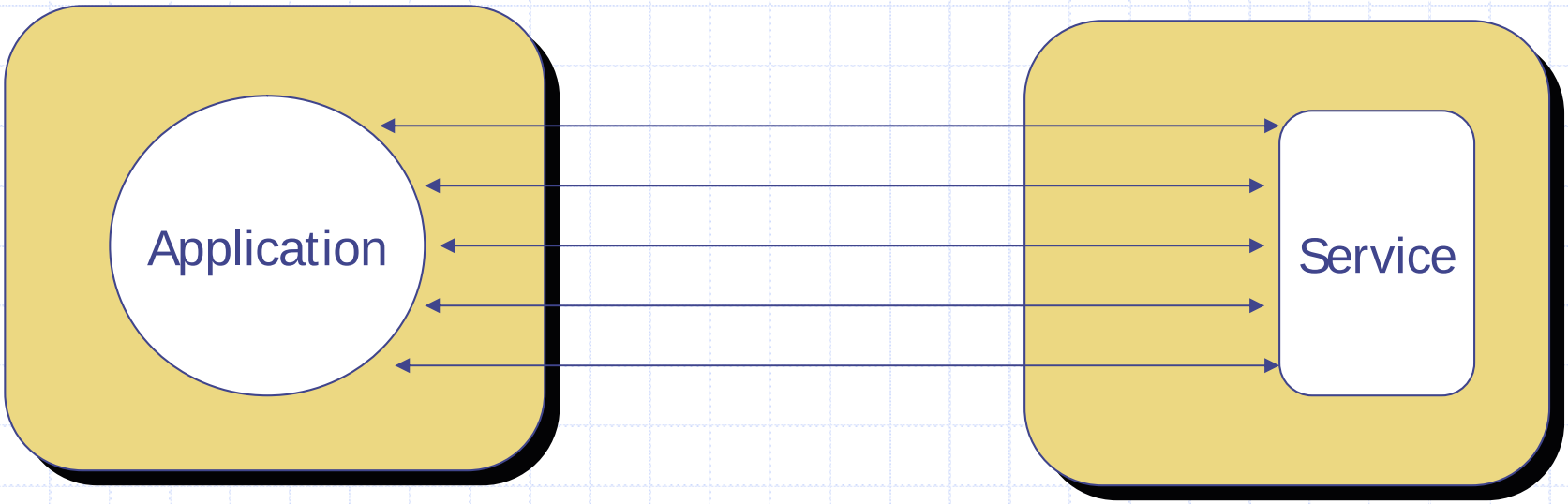
Reduzem carga na rede

- Agentes móveis permitem o empacotamento da conversação e o despacho deles para um host de destino, onde as interações podem tomar lugar localmente.
- Eles são também úteis quando são usados para reduzir o fluxo de dados na rede.

Reduzem carga na rede

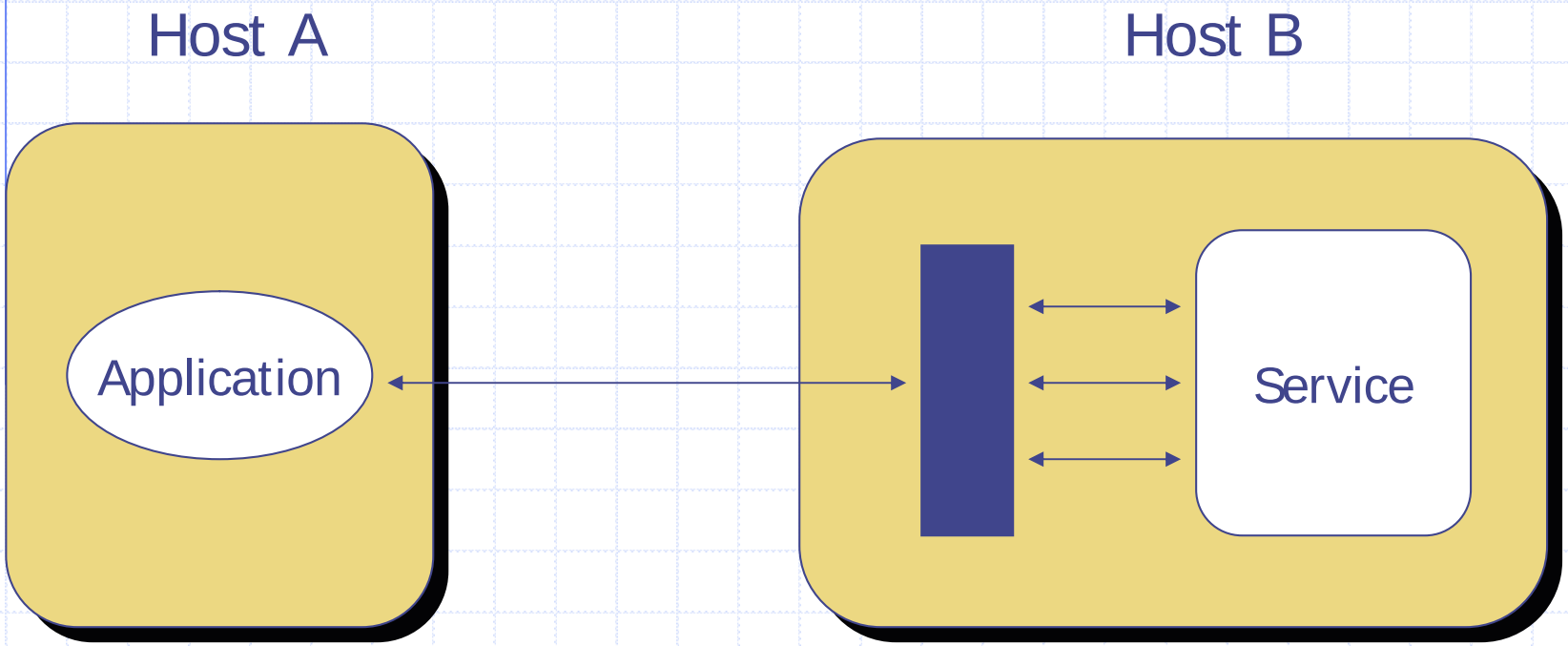
- Quando grandes volumes de dados são armazenados em hosts remotos, estes dados devem se processados na localidade dos dados, ao contrário do que transferidos sobre a rede.
- O lema é simples: mover as computações aos dados, ao contrário do que mover dados às computações.

Reduzem a carga na rede



RPC-Based Approach

Reduzem carga na rede



Mobile Agent-Based Approach

Sobrepujam latência

- Sistemas críticos de tempo real, tais como robots em processos de manufatura, necessitam responder em tempo real a mudanças em seus ambientes. Controlar tais sistemas em uma rede de fábrica de um substancial tamanho envolve latências significativas.

Sobrepujam latência

- Para sistemas críticos de tempo real, tais latências não são aceitáveis.

- Agentes móveis oferecem uma solução, porque eles podem ser despachados de um controlador central e agir localmente, e diretamente executar as funções do controlador.

Encapsulam Protocolos

- Quando dados são trocados em um sistema distribuído, cada host contém o código que implementa o protocolo, para adequadamente, codificar dados para serem transmitidos e decodificar dados que são recebidos.

Encapsulam Protocolos

- Contudo, quando protocolos precisam acomodar novos requisitos para melhorar sua eficiência ou segurança, é uma tarefa muito trabalhosa, senão impossível, fazer o “upgrade” adequado para o código do protocolo.

Encapsulam Protocolos

- Como resultado, protocolos, frequentemente, tornam-se legados, e **agentes móveis**, por outro lado, podem mover-se para hosts remotos para estabelecer “canais” baseados sobre protocolos.

Executam assincronamente e autonomamente

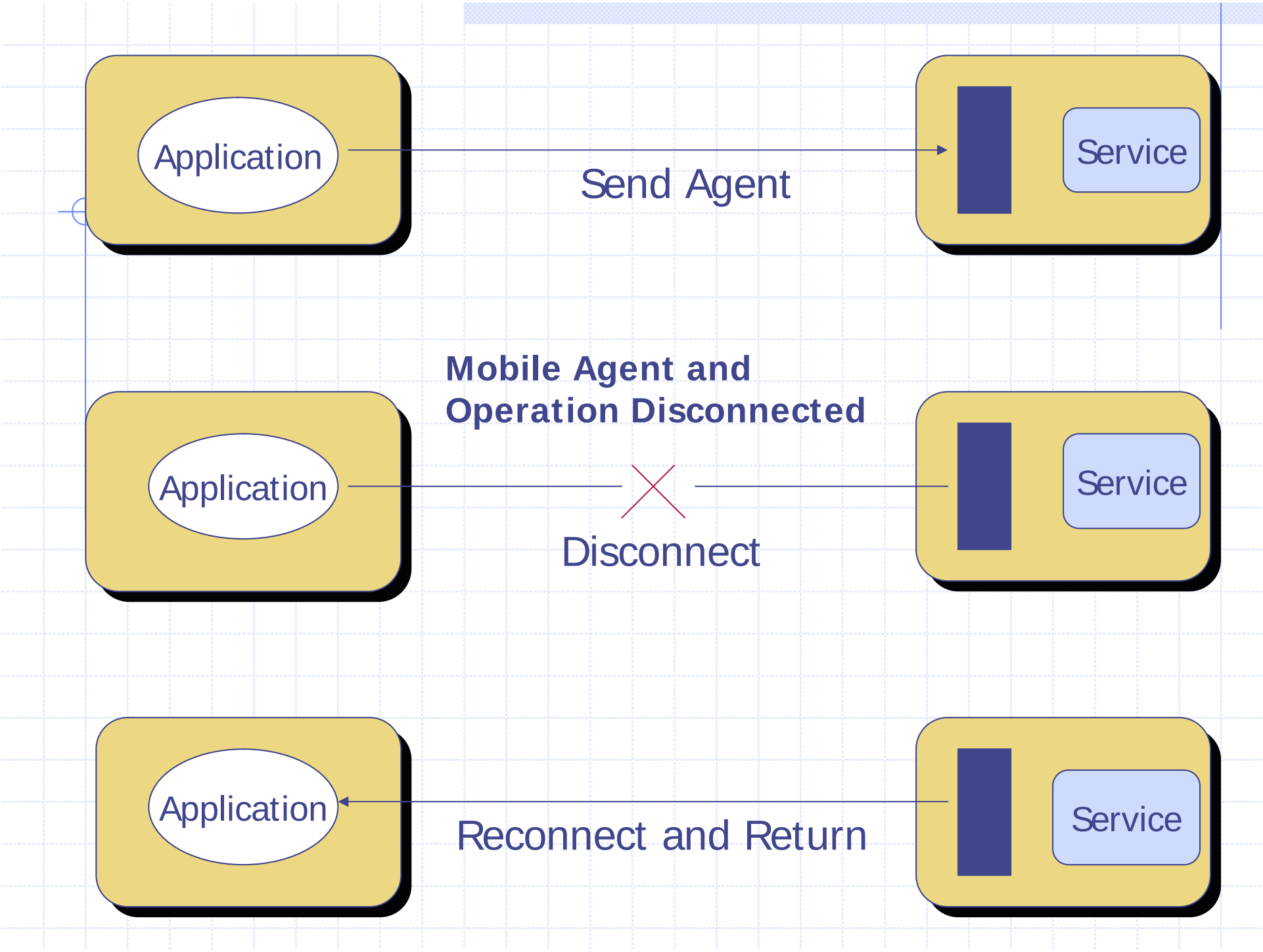
- Frequentemente, dispositivos móveis devem confiar em conexões de rede frágeis ou caras. Tarefas que requerem conexão aberta continuamente entre um dispositivo móvel e uma rede fixa, não será tecnicamente e economicamente viável.

Executam assincronamente e autonomamente

- Para resolver este problema, tarefas podem ser embutidas dentro de agentes móveis, os quais podem então ser despachados na rede.

Executam assincronamente e autonomamente

- Após, serem despachados, os agentes móveis tornam-se independentes do processo que os criou e podem operar assincronamente e autonomamente. O dispositivo móvel pode reconectar-se em um tempo posterior para coletar o agente.


Application
Send Agent
Service
Mobile Agent and
Operation Disconnected
Application
Service
Disconnect
Application
Reconnect and Return
Service

Se adaptam dinamicamente ...

- Agentes móveis têm a habilidade para sentir seu ambiente de execução e reagir autonomamente a mudanças.

- Múltiplos agentes móveis podem possuir a única habilidade para se distribuirem eles próprios entre os hosts na rede, assim como manter a configuração para resolver um problema particular.

Naturalmente heterogêneos

- A computação em rede é fundamentalmente heterogênea, da perspectiva do hardware e do software.
- Porque agentes móveis são geralmente independentes da camada de transporte e do computador e são dependentes sobre seu ambiente de execução, eles proporcionam condições ótimas para integração de sistemas.

Robustos e Tolerantes a Falhas

- A habilidade para agentes móveis reagirem dinamicamente a eventos e situações desfavoráveis, fazem ele mais fácil para construir sistemas distribuídos robustos e tolerantes a falhas.

Robustos e Tolerantes a Falhas

- Se um host está sendo desligado, todos os agentes executando sobre aquela máquina serão advertidos e dado tempo para despachá-los e continuarem suas operações sobre outro host na rede.

Domínios de Aplicação

Áreas onde agentes móveis são utilizados.

Domínios de Aplicação

- Alguns domínios para os quais a mobilidade de código pode trazer os maiores benefícios:
  - Recuperação de Informação
  - Documentos Ativos
  - Serviços Avançados de telecomunicações
  - Controle de dispositivos remotos
  - Gerência de Fluxos de trabalho
  - Redes Ativas
  - Comércio Eletrônico
  - Gerência de Rede

Recuperação de Informação

- Agrupamento de informações satisfazendo determinados critérios a partir de fontes de informação dispersas através das redes.

- Contribuição da mobilidade de código:
  - Maior eficiência através da migração do processo de busca para locais próximos às fontes de informação.
  - Freqüentemente considerada a aplicação, para motivar o uso do padrão AM.

Documentos Ativos

- Dados passivos estendidos com a capacidade de executar programas relacionados ao seu conteúdo.

- Contribuição da mobilidade do código:
  - Encapsulamento de código e estado dentro de documentos, e suporte para extração e execução dos mesmos no lado do cliente (padrão CsD).
  - Aplicação tipica: uso de formulários gráficos para compor e submeter consultas a um BD remoto.
  - Tecnologia típica: WWW, Java, ...

Serviços Avançados de Telecomunicações

- Infra-estruturas de comunicação especializadas para suportar, gerenciar e monitorar serviços como video-conferência, video sob demanda e outros.
  - Aplicações: tele-medicina, educação à distância.
- Contribuição da Mobilidade do código:
  - Facilidade de customização e re-configuração dinâmica.
  - Suporte a clientes móveis através de componentes autônomos (operações desconectadas).

Controle de Dispositivos Remotos

- Configuração e monitoramento do status de uma rede de dispositivos.

- Contribuição da mobilidade do código:
  - Co-locação de componentes de monitoramento junto aos dispositivos monitorados para reportarem a evolução do status dos dispositivos.
  - Envio de componentes de gerência para configurar dispositivos remotos localmente (maior desempenho e flexibilidade).

Gerência de Fluxos de Trabalho

- Suporte à cooperação entre pessoas e ferramentas envolvidas num processo de negócio ou processo de engenharia.

- Contribuição da Mobilidade de código:
  - Encapsulamento de atividades como componentes móveis (padrão AM).
  - Exemplo de documentos compartilhados.

Comércio Eletrônico

- Suporte à realização de transações de negócio através da rede:
- Contribuição da mobilidade de código:
  - Customização do comportamento dos participantes e dos protocolos de negociação.
  - Migração de componentes da aplicação para os locais próximos às fontes de informação relevantes para a transação.
    - suporte a clientes móveis e operações desconectadas.

Redes Ativas

- Aumento da flexibilidade das redes através de mecanismos que permitam programá-las de acordo com as necessidades de cada aplicação.

  - A maioria baseada em mecanismos de suporte à mobilidade.
  - Espectro de soluções delimitado por duas abordagens extremas:
    - switch programável
    - cápsula (código de manipulação encapsulado em cada pacote transmitido pela rede)

Software de Código Móvel

Tecnologias,

Padrões de Arquiteturas

Objetivo Geral

- Introduzir conceitos e discutir temas ligados ao desenvolvimento de software de código móvel.

Objetivos Específicos

- Discutir a importância da mobilidade de código como uma alternativa às abordagens tradicionais.
- Oferecer uma visão geral do espectro de tecnologias disponíveis.

Objetivos Específicos

- Apresentar os principais padrões de arquitetura utilizados no projeto de sistemas de código móvel.

- Discutir os benefícios da mobilidade de código no contexto de vários domínios de aplicação.

Mobilidade de Código: Histórico

- Anos 70: processamento de jobs remotos em ambientes mainframes (JCL).
- Anos 80: controle de impressoras (PostScrit).
- Anos 80-90: migração de processos e objetos no nível de sistemas operacionais.
- Anos 90-00: Sistemas de Código Móvel.

Sistemas de Código Móvel

- Baseados nas técnicas e nos mecanismos originalmente utilizados ao nível de SO.
- Várias inovações e melhorias com respeito à mobilidade de código tradicional:
  - projetados para redes de grande porte;
  - execução em ambientes heterogêneos;
  - localização dos componentes conhecida pela aplicação;
  - Mobilidade sob controle do programador;
  - não apenas para balancear a carga.

Agentes Móveis

- O que é um “agente móvel” ?

  - Um componente de software que é capaz de atingir um objetivo realizando ações e reagindo a eventos em um ambiente dinâmico [Maes, 1994].

  - Um programa que pode migrar através dos nodos de uma rede carregando o seu estado de execução, nos momentos e locais de sua própria escolha [Kotz & Gray, 1999].

Agentes Móveis

- Outras definições:
  - Múltiplas definições e interpretações em diversas áreas da computação (IA, SD, ES, ...).
  - Confusão terminológica e semântica.

Confusão terminológica

- Agente de Software
- Agente Inteligente
- Agente Móvel
- Agente Autônomo
- Agente de Informação

Sistemas de Código Móvel e Agentes Móveis

- Duas situações distintas:
  - Sistema de comunicação móvel
  - Sistema de agentes que são móveis.

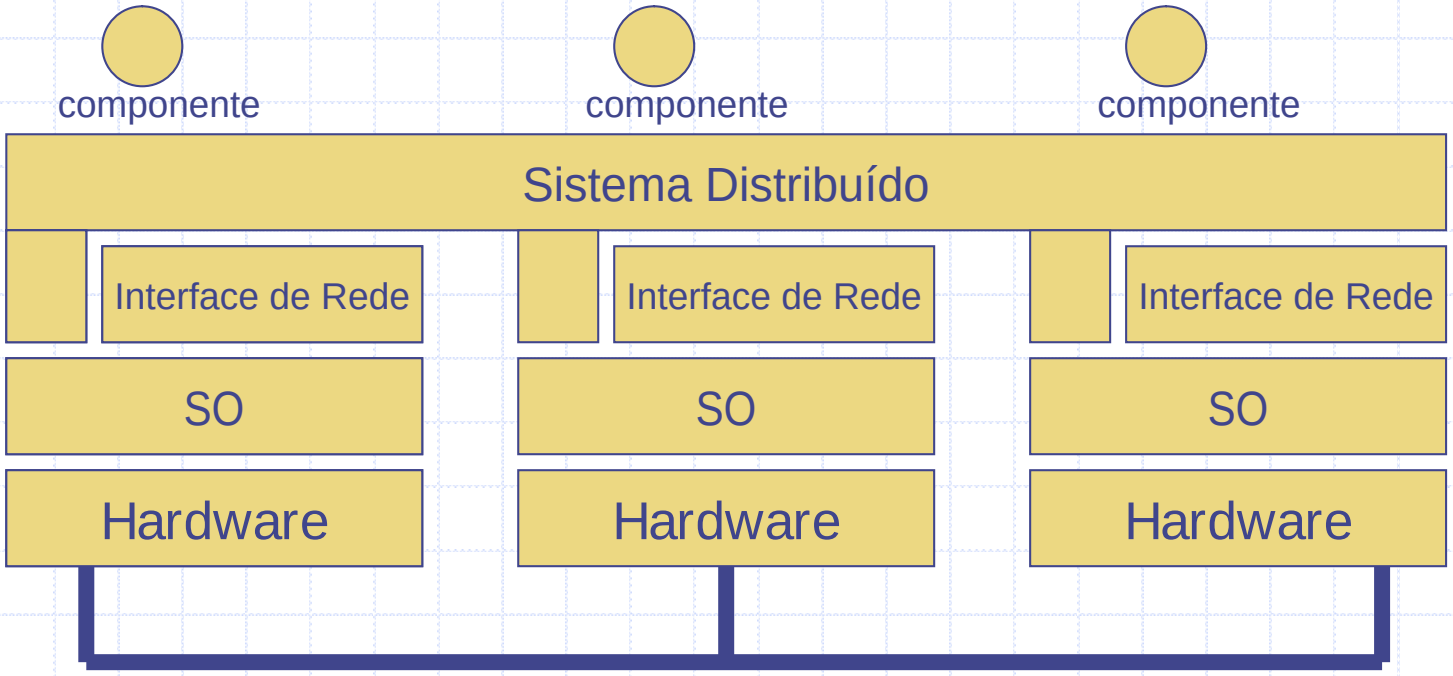
Conceitos Básicos

Ambiente de execução.

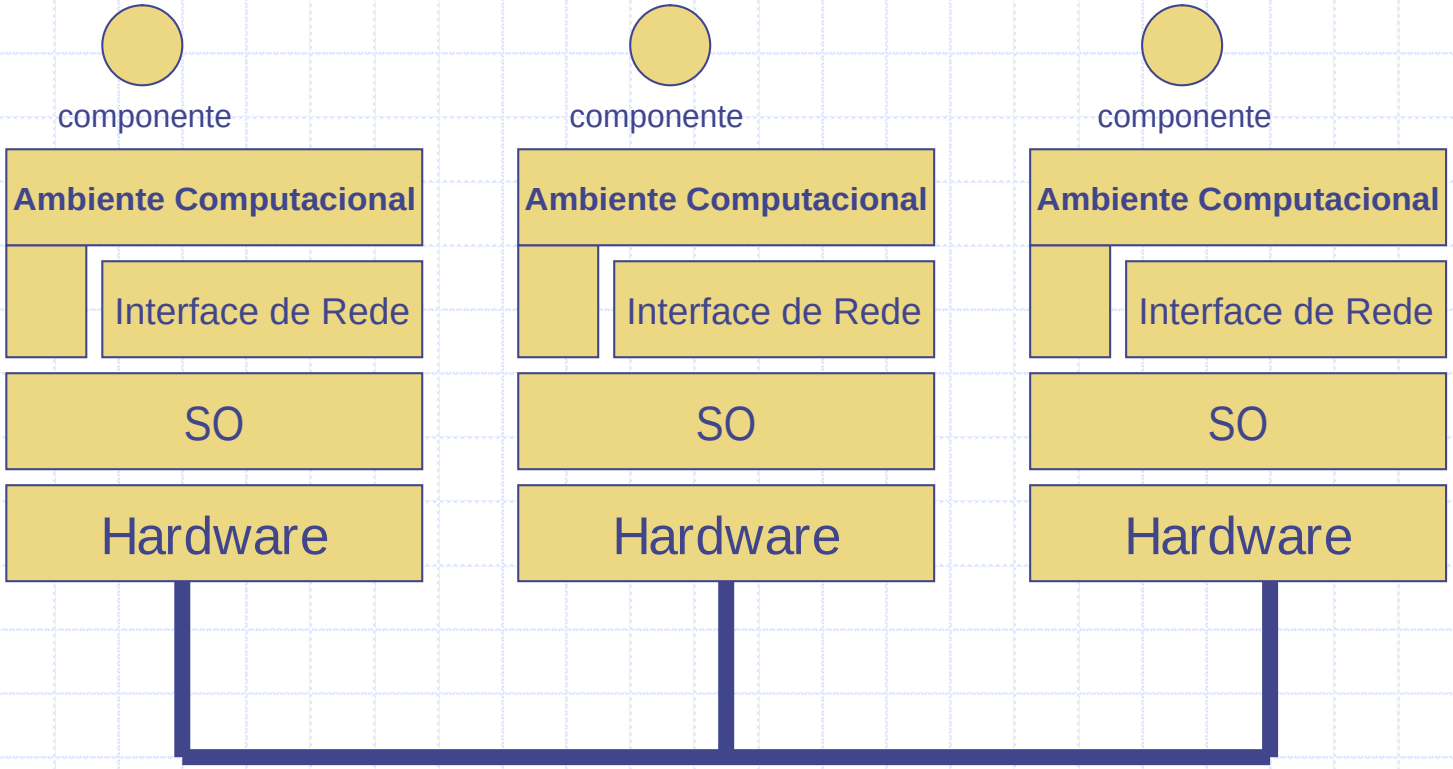
Mecanismos de de gerência de mobilidade.

Mecanismos de Gerência do espaço de dados.

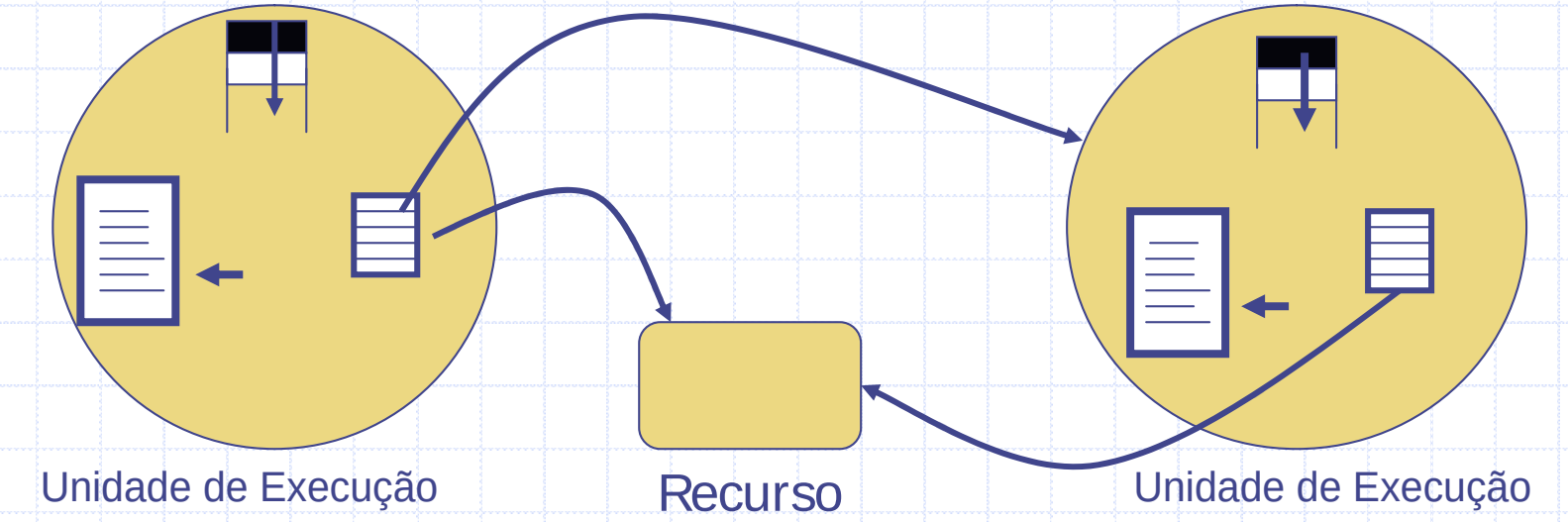
Ambiente de Execução Distribuído



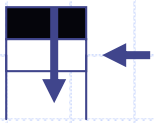
Ambiente de Execução de um Sistema de Código Móvel



Estrutura Interna de Sistema de Código Móvel



Segmento de Código



Estado de Execução

(pilha e contador de programa)

Espaço de dados

Mecanismos de Mobilidade

- Permitem re-alocar uma unidade de execução (UE) para diferentes ambientes computacionais (AC).
- Porção da unidade de execução a ser re-alocada, determinada pela composição dos mecanismos de suporte à mobilidade disponíveis no sistema de código móvel.

Mecanismos de Mobilidade

- Gerência de Mobilidade
- Gerência do Espaço de Dados

Mecanismos de Gerência de Mobilidade

- Mobilidade Forte

  Migração: Reativa, Pró-Ativa

  Clonagem Remota: Reativa, Pró-Ativa

Mecanismos de Gerência de Mobilidade

- Mobilidade Fraca

  Envio de Código: Auto-Contido, Fragmento

  Captura de Código: Auto-Contido, Fragmento

Mobilidade Forte

- Re-alocação tanto do código quanto do estado de execução de uma unidade de execução, para um ambiente computacional diferente.

Mobilidade Forte

- Classificação quanto ao tipo:

Migração – suspensão da unidade de execução, no ambiente computacional de origem; continuação da unidade de execução no novo ambiente computacional.

Clonagem Remota – criação de uma cópia da unidade de execução no ambiente computacional de destino.

Mobilidade Forte

- Classificação quanto ao controle:

Pró-ativa – tempo e destino da re-alocação determinados autonomamente pela unidade de execução.

Reativa – re-alocação sob controle externo de uma unidade de execução diferente (gerente de mobilidade).

Mobilidade Fraca

- Transferência de código através de ambientes computacionais.
- Código transferido, ligado dinamicamente a uma unidade de execução já existente ou usado como segmento de código para uma nova unidade de execução.

Mobilidade Fraca

- Dimensões de classificação:

  - Direção da transferência
  - Natureza do código
  - Sincronização
  - Momento de execução no destino

Mobilidade Fraca

- Direção da Transferência

Envio de código – a unidade de execução envia o código para o ambiente computacional de destino.

Captura de código – a unidade de execução captura o código do ambiente computacional destino.

Mobilidade Fraca

- Natureza do código

Auto-contido – segmento de código completo, usado para criação de uma nova unidade de execução.

Fragmento – segmento de código usado para ligação dinâmica a uma unidade de execução já existente.

Mobilidade Fraca

- Sincronização

  Síncrona – a unidade de execução requisitante é suspensa até o código transferido ser executado.

  Assíncrona – a unidade de execução não é interrompida durante a transferência do código.

Mobilidade Fraca

- Momento de execução no destino:

  Imediato – o código é executado imediatamente após seu recebimento.

  Programado – a execução está sujeita à satisfação de uma determinada condição (primeiro evento externo gerado pela aplicação).

Gerência do Espaço de Estados

- Reorganização das ligações a recursos externos acessíveis a uma unidade de execução quando de sua re-alocação.
  - Anulação de ligações
  - Restabelecimento de novas ligações
  - Migração de recursos para o ambiente computacional de destino.

- Escolha dependente da natureza dos recursos, das formas de ligação e dos requisitos impostos pela aplicação.

Recursos

- Entidades que podem ser compartilhadas entre múltiplas unidades de execução (variáveis de ambiente, arquivos, dispositivos de hardware entre outros).

Recursos

- Tipos de recursos:
  - Transferível Livre – pode migrar (dados em geral)
  - Transferível fixo - migração tecnicamente possível, mas sujeita aos requisitos impostos pela aplicação (dados muito volumosos ou confidenciais)
  - Intransferível – migração tecnicamente impossível (impressoras)

Recursos

- Modelado como uma tripla:

  Recurso = [I, V, T]

  onde

  I é o identificador único do recurso,

  V é o valor (conteúdo) do recurso,

  T é o tipo do recurso.

- Valor e tipo afetam a capacidade de migração.

Ligações

- Meios pelos quais um recurso pode estar acessível a uma unidade de execução.
- Formas de ligação:
  - Por identificador
  - por valor
  - por tipo

Formas de Ligação

- Por identificador:

  A unidade de execução requer ligação permanente a um recurso.

- Por Valor:

  A unidade de execução requer ligação com um recurso de determinado tipo e valor.

- Por tipo:

  A unidade de execução requer ligação com um recurso de determinado tipo, independentemente do valor.

Recursos e Formas de Ligação

- Um mesmo recurso pode estar acessível a diferentes unidades de execução, através de diferentes formas de ligação.

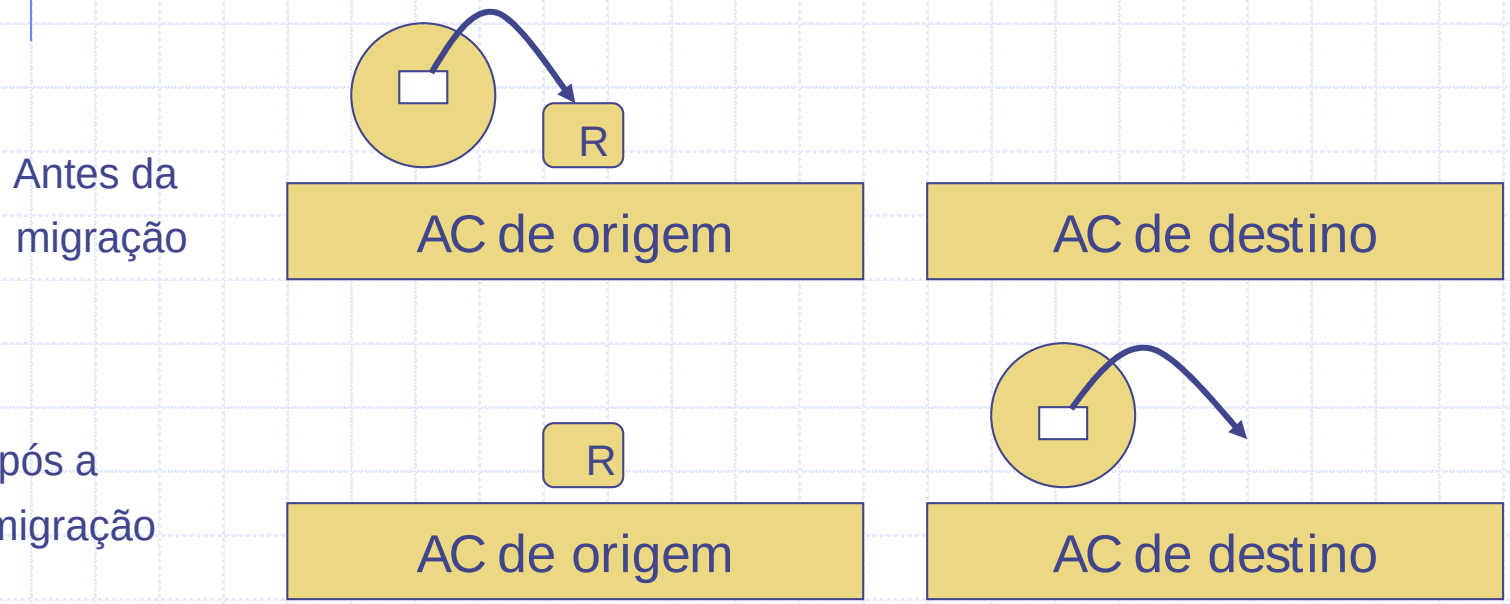
Mecanismos de Gerência do Espaço de Dados

- Remoção de Ligação
- Por deslocamento
- Referência Remota
- Por cópia
- Religação

Remoção de Ligação

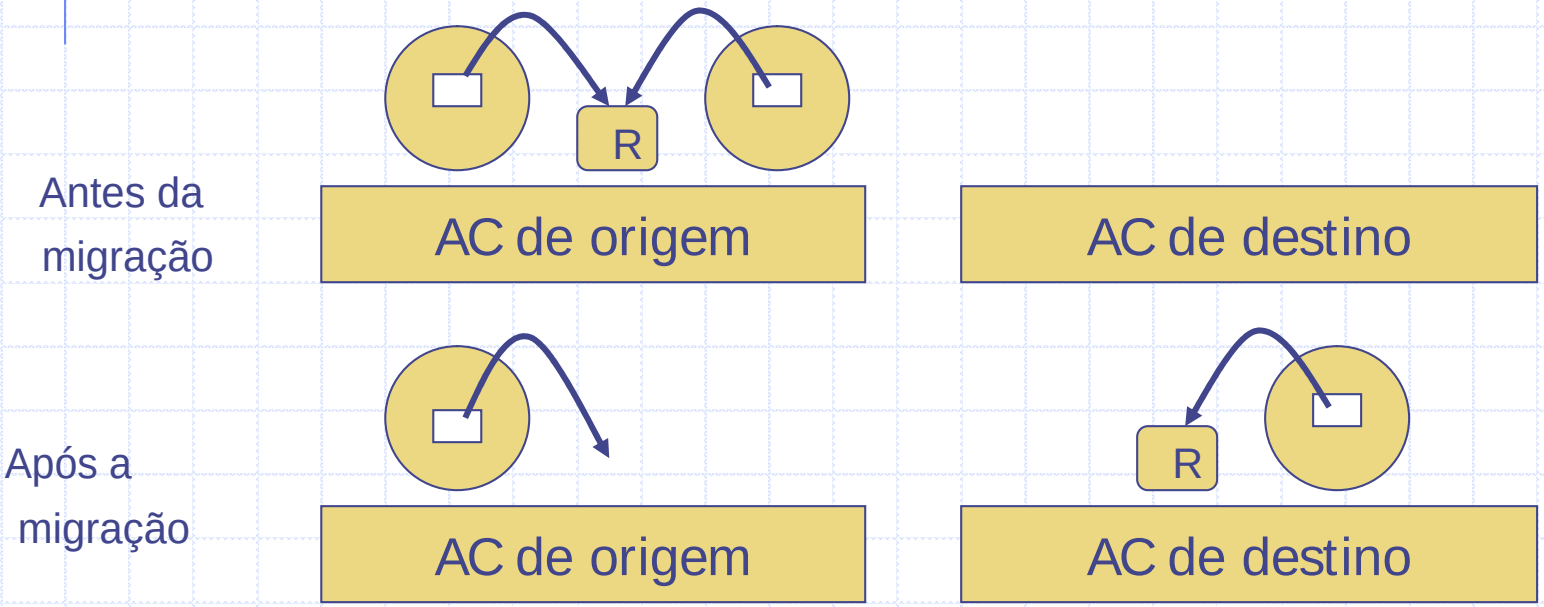
Ligação descartada após migração

Independente da forma de ligação e do tipo de recurso.



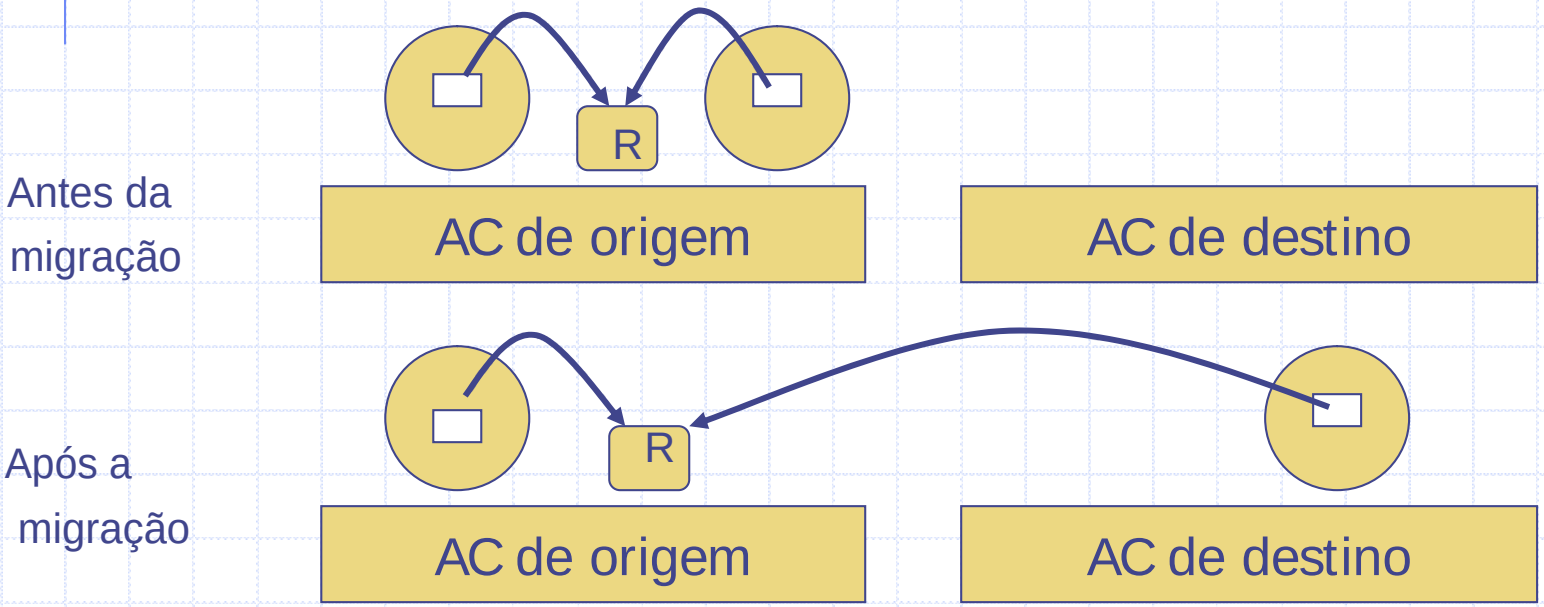
Por deslocamento

- Migração do recurso junto com a UE
- Ligação por identificador (recuso transferível livre)



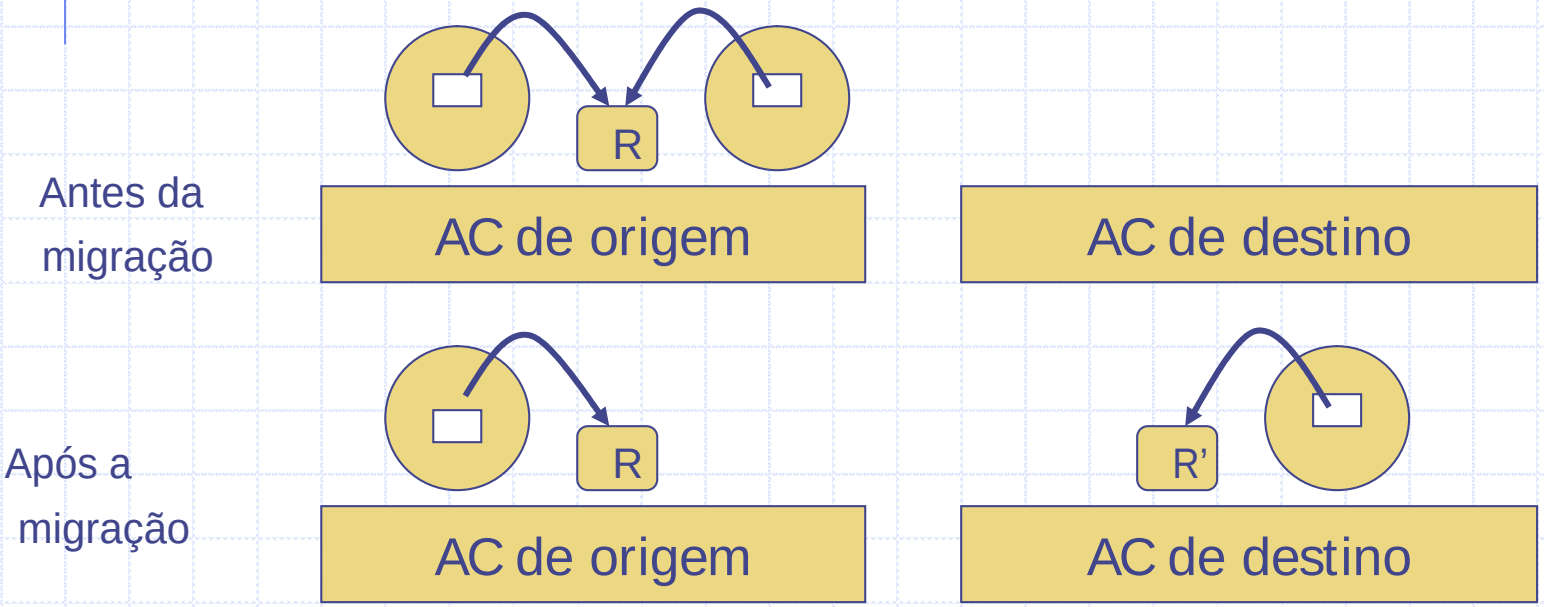
Referência remota

- Ligação modificada para uma referência via rede
- Ligação por identificador(recurso intransferível ou transferível fixo)



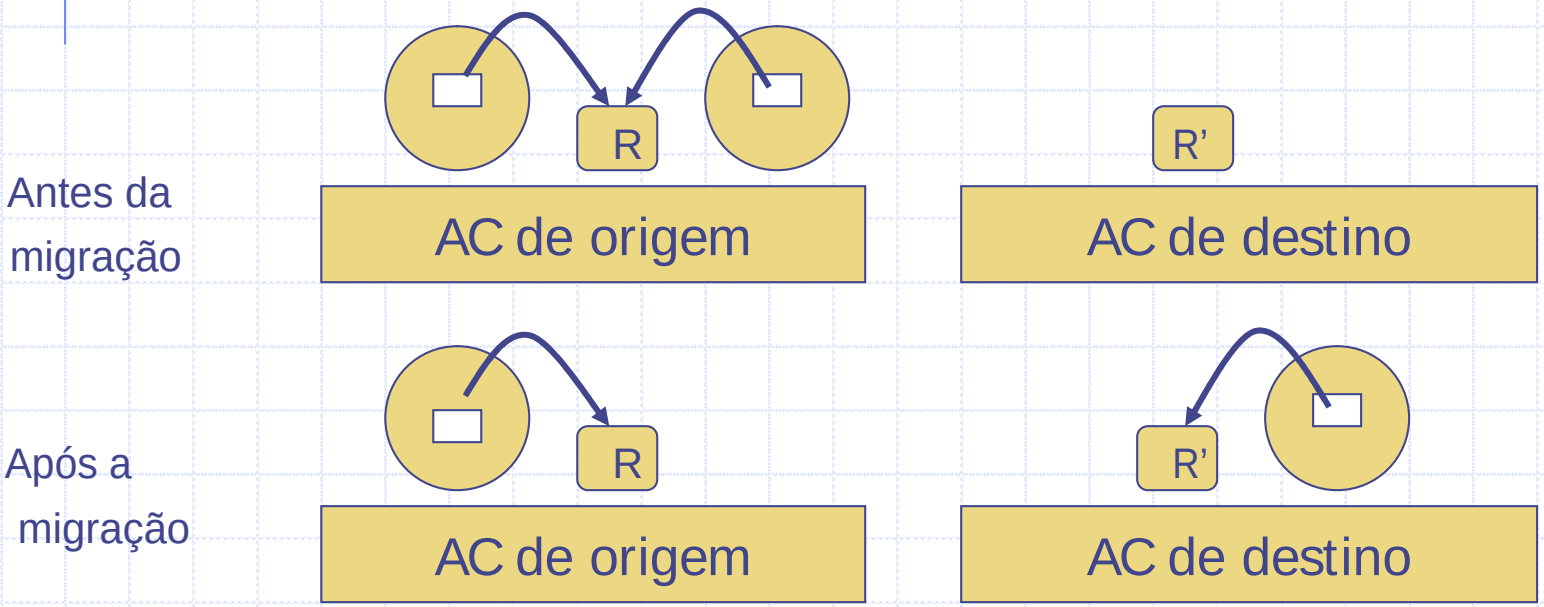
Por cópia

- Migração do recurso junto com a UE
- Ligação por valor (recurso transferível livre)



Religação

- Ligação restabelecida para um recurso de mesmo tipo no AC de destino
- Ligação por um tipo(independente do valor)



Tecnologias

Extensões de Java

Extensões de TCL

Outras linguagens e extensões

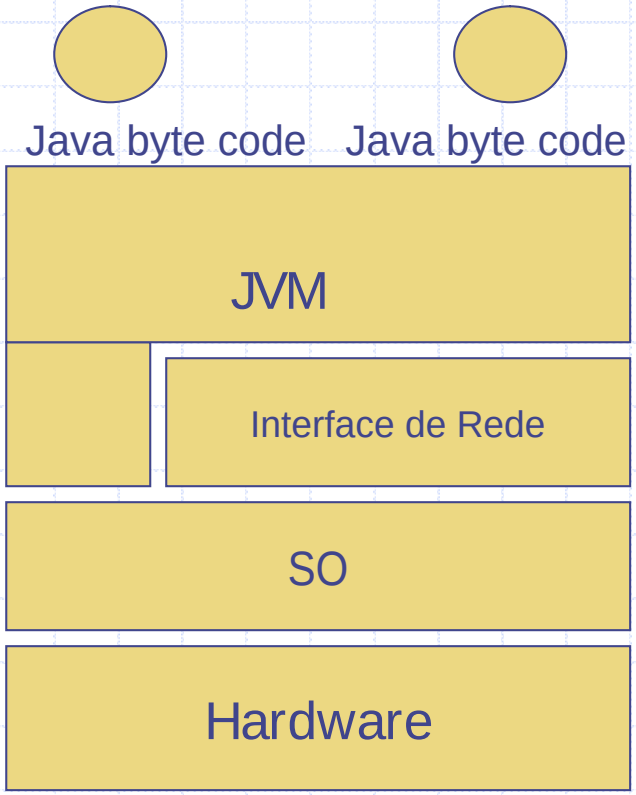
Tecnologias Baseadas em Java

- Applets
- Aglets
- Mole
- Sumatra
- MuCode
- Concordia
- Voyager
- Ajanta
- AgentSpace
- Jamp
- JavaSeal
- JumpingBeans
- ...

Java

Origem	Sun Microsystems (EUA)
Descrição	Linguagem Orientada a Objeto de propósito geral, com suporte à mobilidade fraca.
Unidade De Execução	Java Byte Code – forma intermediária independente de plataforma, gerada pelo compilador Java.
Ambiente Computacional	Java Virtual Machine (JVM) – interpretador para a forma intermediária, disponível em múltiplas plataformas.

Java: Ambiente de execução



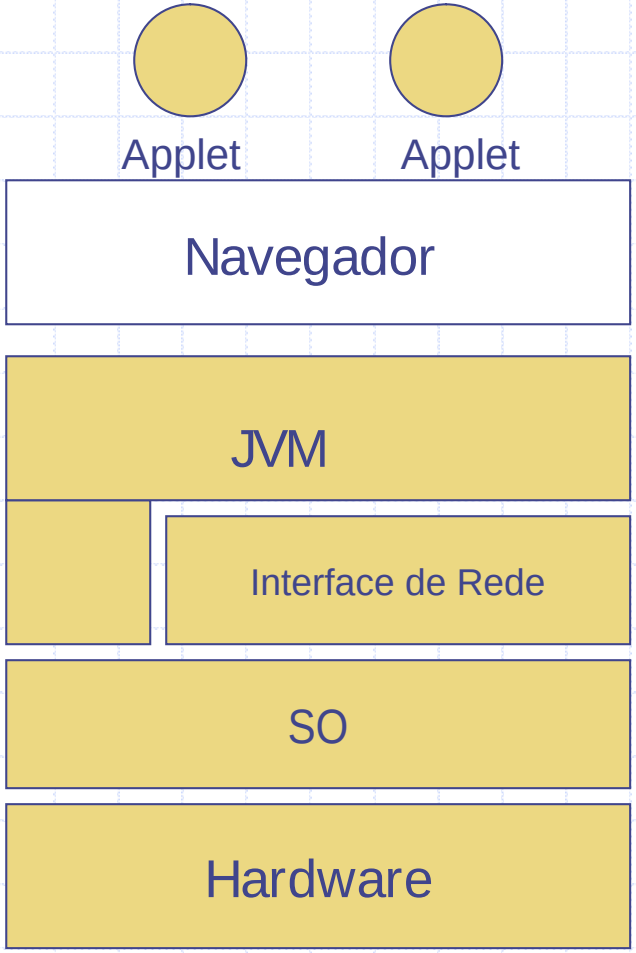
Java: Mecanismos de Mobilidade

Gerência de Mobilidade	Carregador de classe – mecanismo programável para captura e ligação dinâmica de classes (locais e remotas) em uma JVM: - Assíncrona - Execução Imediata ou programável
Gerência do Espaço de Dados	Não há – código capturado sempre executado no início, não possuindo estado de execução, nem ligações para recursos externos.

Applets

Origem	Sun Microsystems (EUA)
Descrição	Integração de Java com tecnologias WWW - Navegadores estendidos para incluir um interpretador Java de uso restrito (sand box). - código Java descarregado junto com páginas HTML. - Apresentação dinâmica acesso interativo a servidores Web.
Unidade De Execução	Applet
Ambiente Computacional	Navegador

Applets: Ambiente de Execução



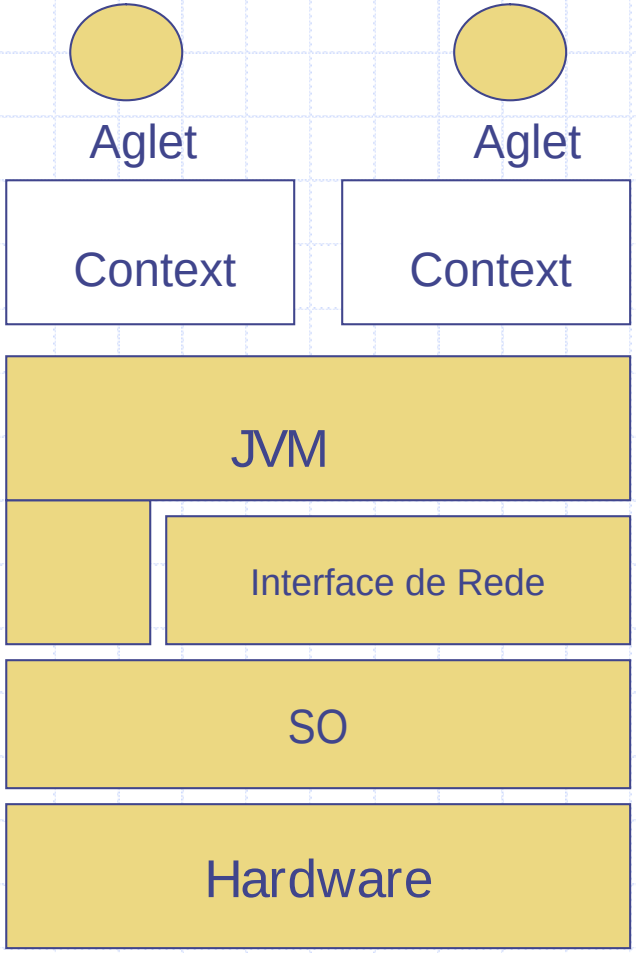
Applets: Mecanismos de Mobilidade

Gerência de Mobilidade	Captura de código auto-contido (segmento de código da applet) pelo navegador. - Assíncrono - Execução Imediata
Gerência do Espaço de Dados	Não há – applet sempre executada do início, não sendo considerado o estado de execução, nem ligações para recursos externos.

Aglets

Origem	IBM Tokio Research Laboratory (Japão)
Descrição	API Java com suporta à mobilidade fraca. - Migração do segmento de código e dos objetos de uma thread (aglet) - Context como uma abstração da JVM para organização, execução e migração de threads.
Unidade De Execução	Aglet
Ambiente Computacional	Context

Aglets: Ambiente de Execução



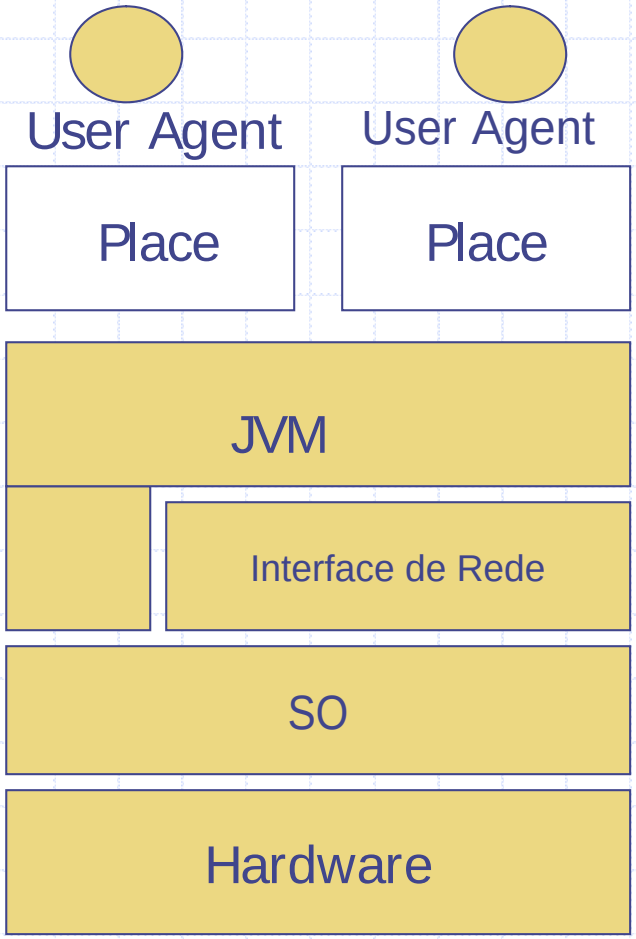
Aglets: Mescanismos de Mobilidade

Gerência de Mobilidade	dispatch – envio do segmento de código de aglet requisitante para o contexto de destino. - Assíncrono - Execução imediata
	retract – captura do segmento de código de um aglet que migrou de volta para o contexto do aglet requisitante. - Síncrono - Execução imediata
Gerência do Espaço de Dados	Por cópia – recursos referenciados pelo aglet copiados e transferidos junto com o seu segmento de código

Mole

Origem	University of Stuttgart (Alemanha)
Descrição	API Java com suporta à mobilidade fraca. - Migração do segmento de código e dos objetos de uma thread (user agent) - Place como uma abstração da JVM para organização, execução e migração de threads e para acesso a serviços de SO.
Unidade De Execução	User Agent
Ambiente Computacional	Place

Mole: Ambiente de Execução



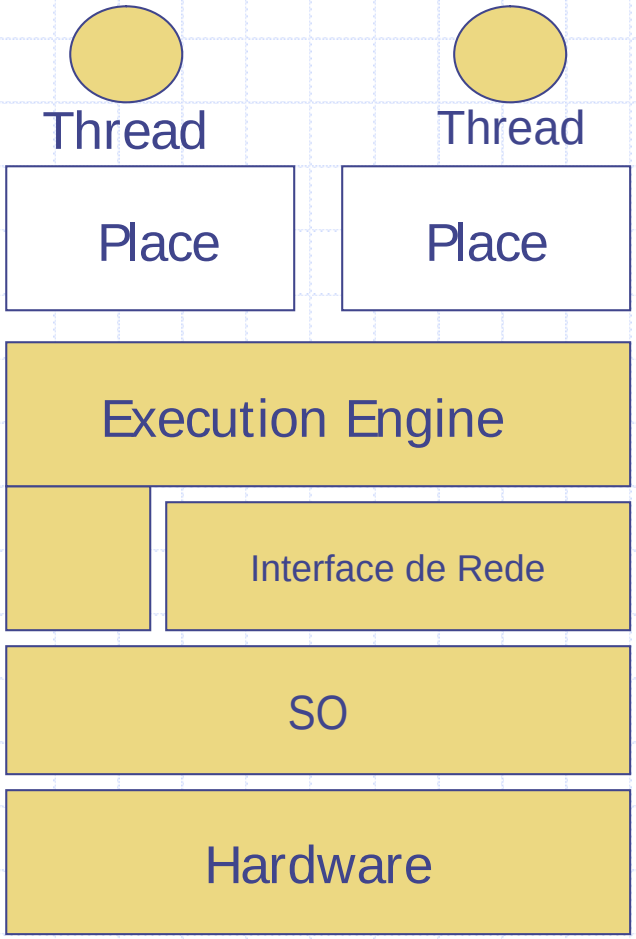
Mole: Mecanismos de Mobilidade

Gerência de Mobilidade	Envio de Código Auto-Contido (segmento de código da thread) - Assíncrono - Execução imediata
Gerência do Espaço de Estados	Por deslocamento – migração de código e dados determinada pelo fecho transitivo sobre todos os objetos referenciados pelo agente (agent island)
	Remoção de ligação – referências entre agentes e recursos externos removidas quando da migração.

Sumatra

Origem	University of Mariland (EUA)
Descrição	Extensão de java com suporte à mobilidade de código forte. - Objetivo inicial: implementação de programas auto-adaptáveis as mudanças nos recursos da rede (resource-aware mobile programs) - Execution Engine como uma extensão da JVM com suporte à migração, clonagem remota e envio do segmento de código de threads.
Unidade De Execução	Thread
Ambiente Computacional	Execution Engine

Sumatra: Ambiente de Execução



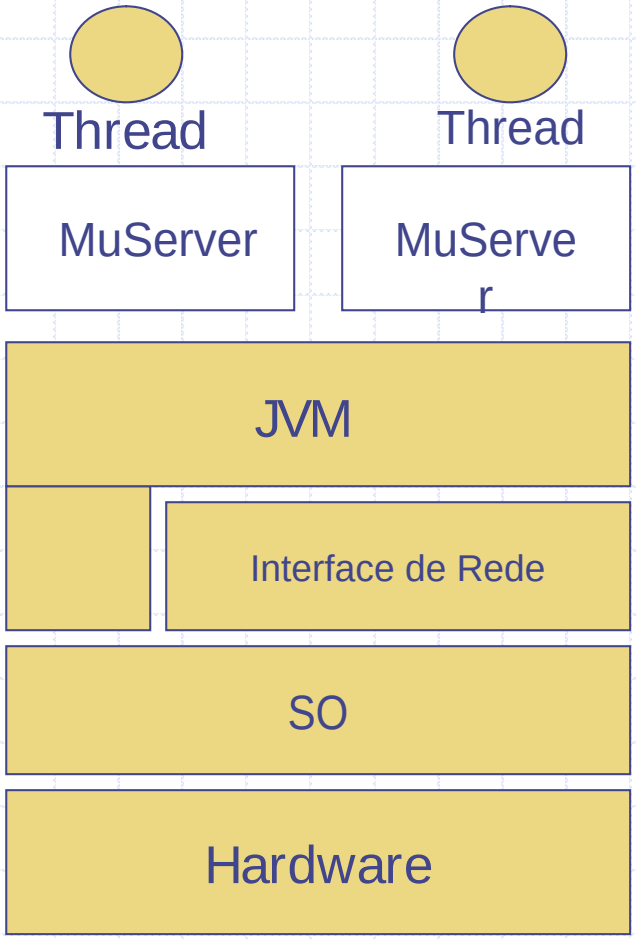
Sumatra: Mecanismos de Mobilidade

Gerência de Mobilidade	`go` – migração pro-ativa
	`rexec` – clonagem remota pró-ativa
	`downloadClass` – envio de código auto-contido (segmento de código da classe) - Sincrono - Execução imediata
Gerência do Espaço de Estados	Por deslocamento – transferência em separado de grupos de objetos criados e mantidos dinamicamente pela aplicação
	Referência Remota – ligações para objetos transferidos convertidos para referências remotas no AC de origem.

MuCode

Origem	Politecnico de Milano (itália)
Descrição	API Java com suporte à mobilidade de código fraca. - Objetivo inicial: oferecerem comjunto mínimo de primitivas de suporte à mobilidade de código. - MuServer como uma abstração da JVM para a criação e cópia de threads, e re-alocação de classes. - Base para a implantação e customização de SCMs com diferentes estratégias de mobilidade.
Unidade De Execução	Thread
Ambiente Computacional	MuServer

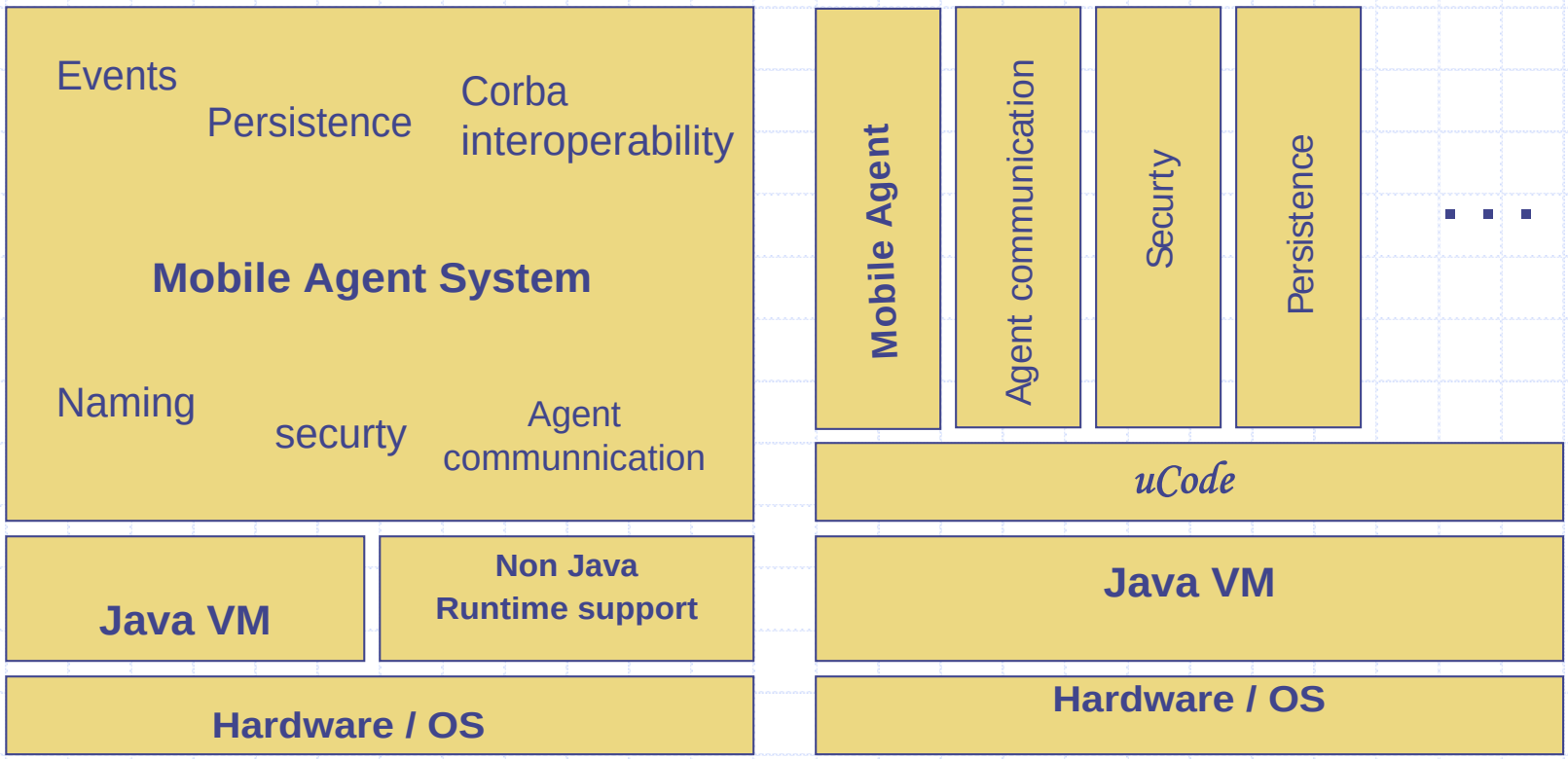
MuCode: Ambiente de Execução



MuCode: Um Ambiente de execução

Gerência de Mobilidade	Código Auto-Contido	`rSpawnThread` – criação de uma nova thread no AC de destino. `rCopyThread` – criação de uma cópia da thread requisitante no AC de destino. - Síncrono - Execução Imediata ou Programada
	Fragmento de Código	`fetchClass` – captura de uma classe do AC de destino. `shipClass` – envio de uma classe para o AC de destino. - Síncrono ou Assíncrono
Gerência do Espaço de Dados	Por deslocamento – transferência conjunta de um grupo de objetos criados e mantidos dinamicamente pela aplicação.	

MuCode vs. Agentes Móveis



Current Mobile Agent Systems

uCode

Outros SCMs baseados em Java

- Protótipos Acadêmicos:
  - Ajanta – University de Minnesota (EUA)
  - AgentSpace – Universidade Técnica de Lisboa (Portugal)
  - Jamp – UFMG (Brasil)
  - JavaSeal – Université de Genève (Suiça)
- Produtos Comerciais:
  - Concordia – Mitsubishi Research (EUA)
  - JumpingBeans – Aramira (EUA)
  - Voyager – ObjectSpace (EUA)

Padrões de Arquitetura

Metodologia, arquitetura, padrões.

Padrões para software de código móvel e avaliação dos padrões.

Arquitetura

- “Decomposição de um sistema de software em termos de componentes de alto nível e as interações entre elas.”
- Principal resultado da fase de projeto.

Padrões de Arquitetura

- “Abstrações e estruturas de referência que podem ser instanciadas para produzir arquiteturas de software com características similares.”
- Independente de tecnologia

Elementos da Arquitetura de um Software de Código Móvel

- Componentes
- Interações
- Sites

Componentes

- Código - capturam o conhecimento para realizar uma determinada computação.
  - Recurso – representam dados ou dispositivos usados durante a computação.
  - Computação – execução do código.

Componentes

- Interações

  Eventos envolvendo dois ou mais componentes.

- Sites

  Sistemas que hospedam componentes e dão suporte à execução dos mesmos.

Padrões de Arquitetura para Software de Código Móvel

- Coordenação e re-alocação dos componentes necessários para realizar um serviço.
- Características:
  - Localização dos componentes antes e depois da execução do serviço.
  - Componente de computação responsável pela execução do código do serviço.
  - Local onde o serviço é de fato realizado.

Padrões de Arquitetura para Software de Código Móvel

- Quatro padrões principais:

  - Cliente/Servidor (C/S)
  - Avaliação Remota (AR)
  - Código sob Demanda (CsD)
  - Agente Móvel (AM)

Padrões de Arquitetura para Software de Código Móvel

- Apresentação baseada num cenário ilustrado através de metáforas
- Elementos do cenário

  A – componente de computação que solicita um serviço.

  SA – site onde A está hospedado.

  B – componente de computação que estará envolvido na realização do serviço.

  SB – site onde B está hospedado

  c – código necessário para a realização do serviço

  r – recursos usados durante a execução do serviço
- Metáfora básica:

  Duas vizinhas, Luiza e Cristina, precisam interagir e cooperar para preparar um bolo de chocolate.

Padrões de Arquitetura para Software de Código Móvel

- Elementos envolvidos na metáfora:

  - Luisa e Cristina (componentes de computação)
  - Casas da Luisa e da Cristina (sites)
  - Bolo (resultado do serviço)
  - Receita (código para executar o serviço)
  - Ingredientes (recursos transferíveis livres)
  - Forno (recurso transferível fixo)
  - Batedeira de Bolo (componente de computação que realiza o serviço)

Padrão Cliente/Servidor

- Cenário:

Antes		Depois	
SA	SB	SA	SB
A	B(c,r)	A	B(c,r)

Padrão Cliente/Servidor

- Metáfora:

Luisa gostaria de preparar um bolo de chocolate, mas ela não conhece a receita, e não tem um forno, nem os ingredientes necessário em casa. Felizmente ela sabe que a sua vizinha Cristina sabe fazer um bolo de chocolate delicioso e tem uma cozinha bem equipada na sua casa. Como Luisa sabe que Cristina adora fazer bolo para os amigos, ela lhe telefona e pede: “você poderia me fazer um bolo de chocolate?”. Cristina prepara o bolo e o entrega na casa de Luisa.

Padrão Avaliação Remota

- Cenário:

Antes		Depois	
SA	SB	SA	SB
A(c)	B(r)	A	B(c,r)

Padrão Avaliação Remota

- Metáfora

Luisa gostaria de preparar um bolo de chocolate. Ela conhece a receita, e não tem um forno, nem os ingredientes necessário em casa. Felizmente ela sabe que a sua vizinha Cristina tem ambos na sua casa, mas não sabe fazer um bolo de chocolate. Como Luisa sabe que Cristina adora aprender novas receitas, ela lhe telefona e pede: “você poderia me fazer um bolo de chocolate?”. A receita é a seguinte: ....” . Cristina prepara o bolo e o entrega na casa de Luisa.

Padrão Código Sob Demanda

- Cenário:

Antes		Depois	
SA	SB	SA	SB
A(r)	B(c)	A(c,r)	B

Padrão Código Sob Demanda

- Metáfora:

  Luisa gostaria de preparar um bolo de chocolate. Ela tem um forno e todos os ingredientes necessário em casa, mas lhe falta a receita. No entanto, Luisa sabe que a sua vizinha Cristina tem a receita e que ela já a emprestou para todos os amigos. Luisa então telefona para Cristina e pede: “você poderia me emprestar sua receita de bolo de chocolate?” Cristina lhe descreve a receita pelo telefone e Luisa então prepara o bolo de chocolate na sua própria casa.

Padrão Agente Móvel

- Cenário:

Antes		Depois	
SA	SB	SA	SB
A(c)	(r)	---	A(c,r)

Padrão Agente Móvel

- Metáfora:

Luisa gostaria de preparar um bolo de chocolate. Ela tem a receita e todos os ingredientes necessários em casa, mas lhe falta um forno. No entanto, Luisa sabe que a sua vizinha Cristina tem um forno em sua casa, e que ela não se importa em emprestá-lo. Luisa então prepara a massa do bolo de chocolate na sua casa, e em seguida vai com a massa até a casa de Cristina, onde ela assa o bolo.

Avaliação dos Padrões

- Oferecem várias abstrações para representar as ligações entre os componentes, locais de execução e código, e suas re-configurações dinâmicas.
- Modelam explicitamente o conceito de localização a nível de arquitetura.
  - Interações entre componentes de mesma localização têm custo desprezível quando comparadas a interações envolvendo comunicação via rede.
- Modelam explicitamente o conceito de mobilidade de componentes.
  - Mudanças de localização podem alterar dinamicamente a qualidade e os custos das alterações.

Avaliação dos Padrões

- AR e AM permitem a execução de código numa máquina remota.
  - Interações locais com componentes localizados na máquina remota.
- CsD permite que componentes de computação capturem código de outros componentes remotos.
  - Flexibilidade para estender dinamicamente o seu comportamento e os tipos de interação que eles suportam.
- Escolha do melhor padrão depende do tipo de aplicação e dos critérios que se quer otimizar.
  - No. de interações, custos de CPU, tráfego na rede, ...