Desenvolvimento de Aplicações Distribuídas

Desafios para a Construção de Sistemas Distribuídos

Prof. Dr. Fábio Favarim
favarim@utfpr.edu.br

Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

7 de Agosto de 2018

Objetivos da Aula

- Conhecer os desafios para a construção de sistemas distribuídos e as Transparências desejáveis em sistemas distribuídos;
- Conhecer o que é middleware e sua importância;
- Conhecer as arquiteturas de sistemas distribuídos.

Sumário

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Introdução aos Sistemas Distribuídos
Revisão da Primeira Aula

- **Algumas aplicações são nativamente distribuídas**
  - mensagens instantâneas, sistema acadêmico, navegação web
- **Desempenho**
  - Tarefas podem ser distribuídas por diversos computadores
  - Facilidade para aumentar a capacidade de processamento (número de CPU)
  - Vantagem financeira: Desempenho *vs* Custo
- **Algumas aplicações são críticas**
  - Continuará funcionando mesmo diante de alguma falha

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Introdução aos Sistemas Distribuídos
Revisão da Primeira Aula

Dificuldades em sistemas distribuídos

- Compartilhamento
  - Dados e processamento
- Descoberta de serviços
  - Como localizar os serviços e depois como invocá-los?
- Complexidade para codificação
  - Dimensão do sistema e funcionamento não determinístico

Introdução aos Sistemas Distribuídos
Revisão da Primeira Aula

Falhas nos sistemas

- **Sistema não distribuído**
  - Falha tudo ou nada
  - O usuário fica ciente da falha
  - Estratégia de recuperação: fechar e abrir novamente
- **Sistema distribuído**
  - Falha pode ser parcial
  - Falha pode não ser percebida pelo usuário
  - Diferentes estratégias para lidar com falhas

Desafios para a Construção de Sistemas Distribuídos

Os SD são facilmente encontrados em qualquer lugar nos dias de hoje, mas projetos ambiciosos podem pedir requisitos que trazem muitos desafios para o seu desenvolvimento:

- Heterogeneidade
- Abertura
- Concorrência
- Segurança
- Escalabilidade
- Tratamento de falhas
- Transparência

Desafios para a Construção de Sistemas Distribuídos
Heterogeneidade

Permitir ao sistema funcionar mesmo levando em conta a diversidade de hardware e software.

- Um sistema distribuído deve considerar:
  - diferentes tipos de rede
  - diferentes tipos de hardware (diferentes representações de dados, diferente código máquina)
  - diferentes sistemas operacionais (diferentes interfaces para os protocolos de comunicação)
  - diferentes linguagens de programação (diferentes representações de estruturas de dados como arrays ou registos,...)
- **Como tratar do problema?**

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Desafios para a Construção de Sistemas Distribuídos

Ser Aberto / Abertura (Openness)

Permitir ao sistema ser estendido ou implementado de diferentes maneiras

- **Exemplos:**
  - **Computador pessoal** – É possível adicionar periféricos de diferentes fabricantes
  - **Rede TCP/IP** – computadores com diferentes arquiteturas e sistemas operacionais conseguem interagir sem problema
- Para isso é importante que:
  - publicação de **interfaces**
  - documentação e especificação
  - código aberto
  - protocolos e formatos padronizados usados na Internet (*Request For Comments* – RFCs) podem ser obtidas em `www.ietf.org`

Sistema Distribuído Aberto

- Faz uso de mecanismos de comunicação padronizados e torna público interfaces para acesso aos recursos compartilhados
- Pode ser construído com hardware e software heterogêneos, possivelmente de diferentes fabricantes

Desafios para a Construção de Sistemas Distribuídos
Concorrência

Permitir que recursos compartilhados possam ser acessados por diversos usuários/processos simultaneamente.

- Questões:
  - Necessário coordenar o acesso aos recursos compartilhados: hw, sw, dados
  - **Problema:** ocorrência de conflitos $\Longrightarrow$ podem gerar inconsistências em ambientes concorrentes.
    - Dois usuários estão no mesmo momento acessando a mesma tabela no Banco de Dados
    - O sistema deverá garantir o acesso concorrente ao mesmo tempo que garante a consistência dos dados (geralmente protegidos por *locks*)

Desafios para a Construção de Sistemas Distribuídos
Escalabilidade

Capacidade do sistema permanecer efetivo mesmo quando há um aumento no número de recursos e usuários.

- Um sistema e seu software não deveria mudar severamente seu comportamento quando sua escala (seu "**tamanho**") cresce
- Deve-se tomar cuidado com serviços, dados e algoritmos centralizados:

Conceito	Exemplo
Serviços Centralizados	Um único servidor para todos os usuários
Dados Centralizados	Uma única lista telefônica online
Algoritmos Centralizados	Fazer roteamento com base em informações completas

- Podem se tornar gargalos, pontos únicos de falhas e saturar a rede onde residem.

Desafios para a Construção de Sistemas Distribuídos
Tratamento de Falhas

Prevenir

Para que falhas não afetem outros componentes do sistema (física, software e humanas)

Tolerar

Na Web, por exemplo, os browsers são projetados para tolerar falhas. Quando um servidor não pode ser conectado, o browser não fica eternamente tentando estabelecer uma conexão, ao invés disso, ele encerra a tentativa de conexão e em seguida avisa o usuário sobre a desistência;

Recuperação

Dependendo do software, um projeto adequado permite que um sistema possa recuperar um estado consistente de dados até o momento antes da falha. Ex: transações usado em BDs.

Desafios para a Construção de Sistemas Distribuídos
Tratamento de Falhas

Redundância

Os sistemas podem também oferecer componentes redundantes para evitar falhas. Ex: BDs com replicação de dados, Servidores de DNS e Roteadores.

Desafios para a Construção de Sistemas Distribuídos
Segurança

- Questão problemática em sistemas distribuídos devido à facilidade de compartilhamento de recursos.
- Segurança costuma ser subdividida em 3 partes:

Confidencialidade

Proteção contra acesso não autorizado;

Integridade

Proteção contra alteração não autorizada

Disponibilidade

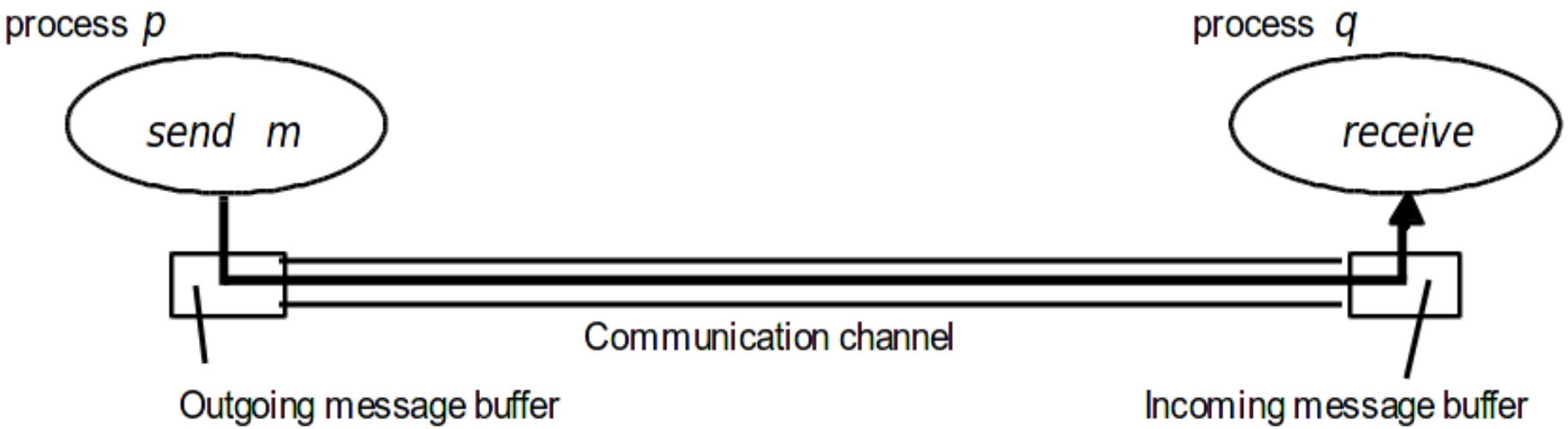
Proteção contra interferência no acesso ao recurso

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Desafios para a Construção de Sistemas Distribuídos
Segurança

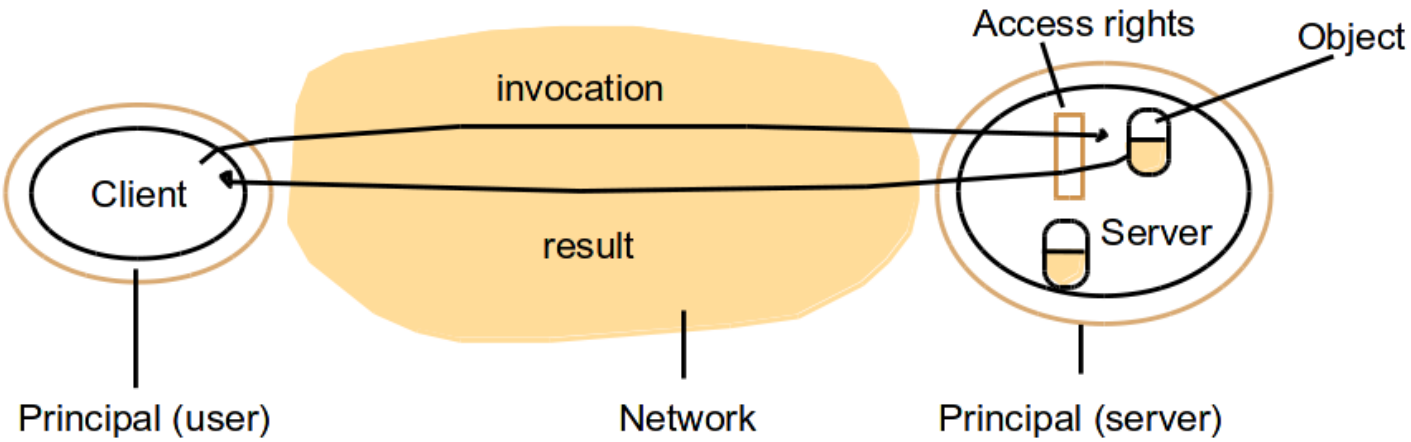
Comunicação normal

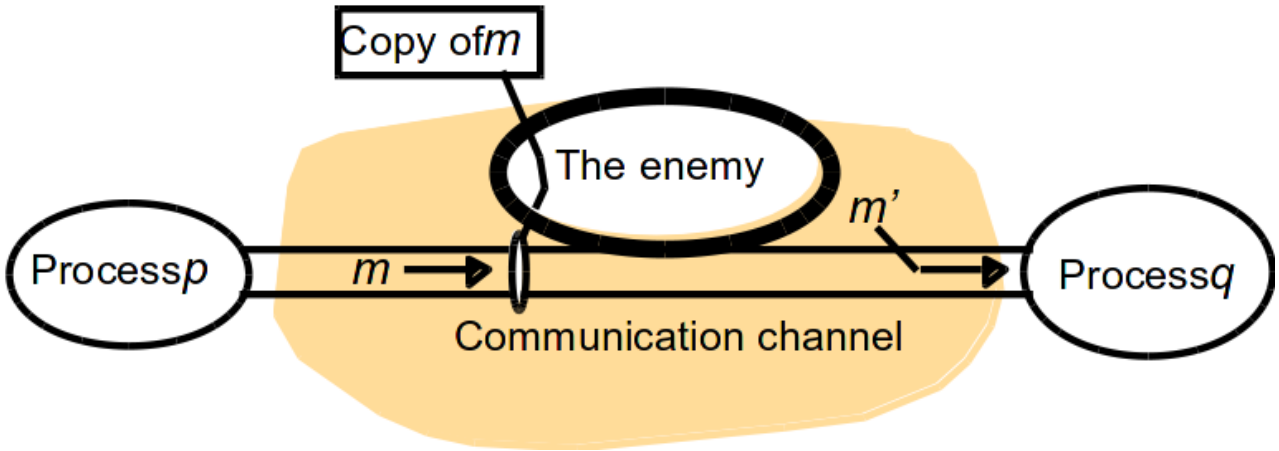
Através de canais de comunicação não seguros.



Desafios para a Construção de Sistemas Distribuídos
Segurança

Interceptação/Alteração da mensagem



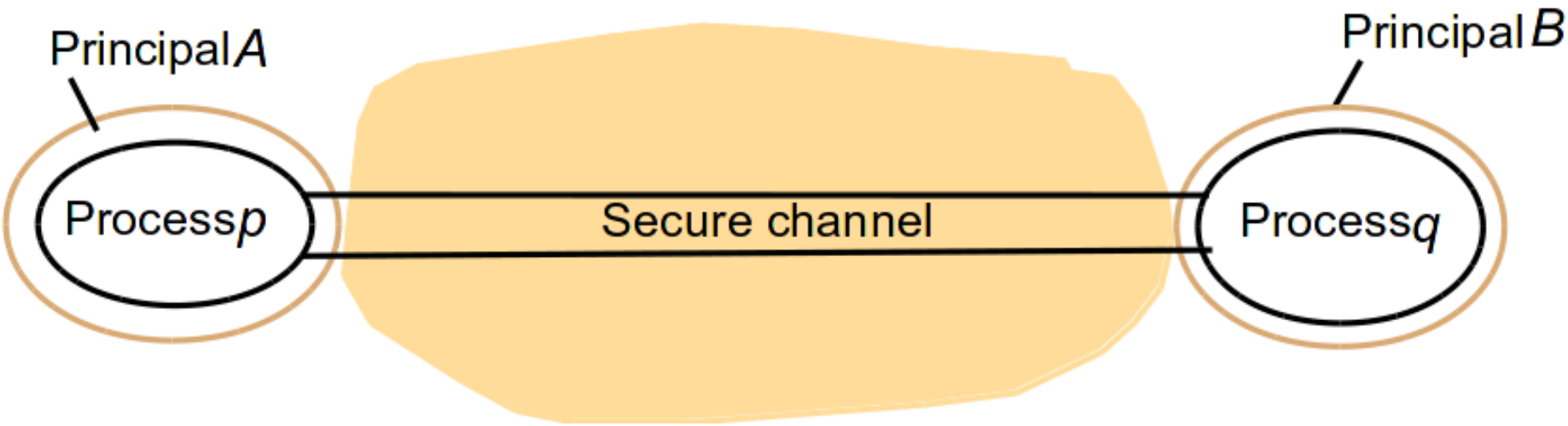


Desafios Sistemas Distribuídos - Segurança

Segurança

Comunicação "Segura"

Através de canais de comunicação seguros.



Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Entidades remotas e locais podem ser acessadas usando operações idênticas

- Esconder as diferenças para representação dos dados
  - Como os dados serão representados em diferentes arquiteturas de máquinas e sistemas operacionais
  - Ex: Convenções para nomes de arquivos, bem como sua manipulação em diferentes sistemas operacionais
    - Usuários e aplicações não precisam ter ciência disto

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Esconder a localização dos recursos

- Entidades podem ser acessadas sem o conhecimento de sua localização.
- Usuário não consegue determinar onde um recurso está localizado fisicamente no sistema
- Ex: `http://www.pb.utfpr.edu.br/favarim`
  - Onde está localizado o servidor web da UTFPR?

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Permitir que o sistema seja reconfigurado para melhorar o desempenho de acordo com a carga

- Novos servidores *web* poderiam ser agregados para ajudar no balanceamento de carga em períodos críticos
- Ex: Último dia para entrega do IRPF

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Esconder a mudança de localização enquanto os recursos estão sendo acessados

- Entidades podem se deslocar sem afetar a operação dos usuários ou programadores.

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Esconder que os recursos são replicados

- Recursos podem ser replicados para aumentar a disponibilidade ou desempenho
- Ex: Quando acessa a página do facebook ou google, em qual servidor você está se conectando?
  - No mais próximo do teu computador ou no servidor no EUA?

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Esconder que o recurso pode estar sendo compartilhado de forma concorrente com outros usuários

- Dois usuários estão acessando simultaneamente a mesma tabela no Banco de Dados

Transparência

Esconder dos usuários o fato que o **processamento e os recursos estão fisicamente distribuídos** por diversos computadores

- O usuário deve enxergar como um único sistema

Tipo

- Acesso
- Localização
- Desempenho
- Mobilidade ou Migração
- Replicação
- Concorrência
- Falhas

Esconder as falhas de um recurso, bem como sua recuperação

- Servidor web possui 3 réplicas para melhorar o desempenho
- Se uma réplica falhar, o serviço continua acessível
- Quando a réplica que falhou for recuperada, nada muda na forma de acesso para o usuário

Transparência

- É **impossível** prover um sistema distribuído **completamente transparente**
  - Impossível esconder a latência na transmissão pela Internet
  - Nem sempre será possível esconder falhas do sistema
- **Em alguns casos é desejável que o usuário tenha ciência**
  - Ex: O usuário precisa saber onde está a impressora para onde mandou o trabalho

Arquiteturas de Sistemas Distribuídos

Arquitetura de Software

- Originalmente, se referia à estruturação do software em camadas ou em módulos em um único computador;
- Atualmente, se refere à estruturação tanto no mesmo ou em computadores diferentes.

A arquitetura de um sistema distribuído:

- é a sua estrutura em termos de componentes especificados separadamente
- demonstra como os seus componentes interagem para chegar a um objetivo
- trata dos papéis funcionais e os padrões de comunicação

Arquiteturas de Sistemas Distribuídos

Plataforma

- Termo utilizado para fazer referência a camadas de baixo nível de hardware e software que provêem serviços a camadas superiores;
- Normalmente, em termos de computadores, uma plataforma está relacionada ao tipo de arquitetura de processador e o sistema operacional utilizado.
  - Intel/Windows, Intel/Linux, PowerPC/MacOS, etc.

Arquiteturas de Sistemas Distribuídos

Middleware

- termo aplicado para uma camada de software que tem o objetivo de providenciar uma abstração na programação através do ocultação das diferenças (heterogeneidade) existentes nos ambientes de rede, hardware, sistemas operacionais e linguagens de programação.
- oferece um modelo programação uniforme para os programadores, simplificando bastante o desenvolvimento de aplicações distribuídas;
- auxilia na obtenção das transparências desejáveis em SD;
- Exemplos: RPC, RMI, CORBA, Serviços Web.

Arquiteturas de Sistemas Distribuídos

Papéis de processos no SD:

- **Processo servidor** - provê serviços, aceita pedidos de outros processos
- **Processo cliente** - utiliza serviços
- **Processo par** - Papel igualitário (prove e usa serviços)

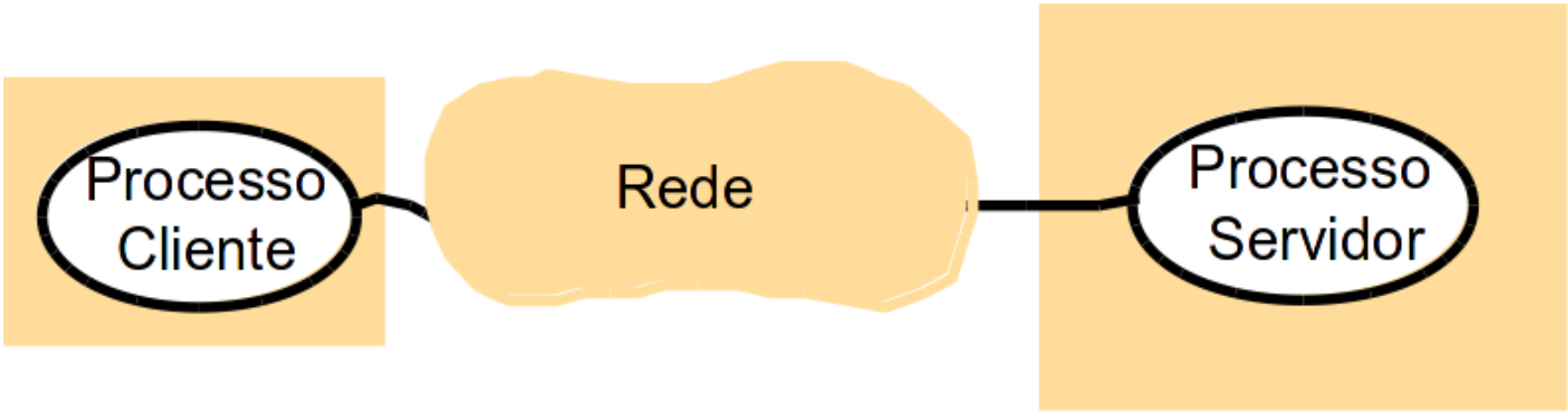
Arquiteturas de Sistemas Distribuídos

Arquiteturas Existentes:

- **Cliente/Servidor e variações**
  - Mais usada em sistemas distribuídos
- **Peer-to-Peer (P2P)**
  - Todos os envolvidos desempenham funções semelhantes, interagindo cooperativamente como pares (peers), sem distinção entre processos clientes e servidores.
- **Híbrida:** Cliente/Servidor + P2P

Arquitetura Cliente/Servidor

- Processos clientes invocam serviços aos servidores;
- Arquitetura mais comum e utilizada em SD.

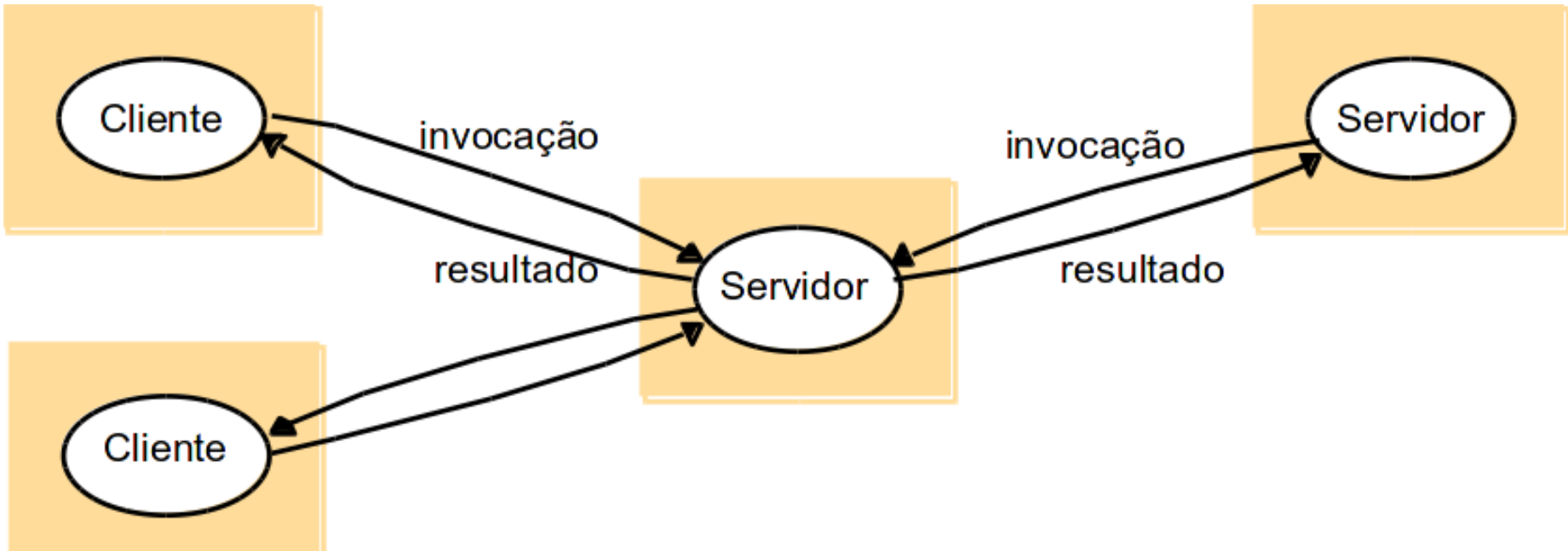


Variação: Arquitetura Cliente/Servidor

Servidor como cliente de outro servidor

Neste modelo, um servidor pode se tornar um cliente para solicitar serviços de outro servidor. Ex:

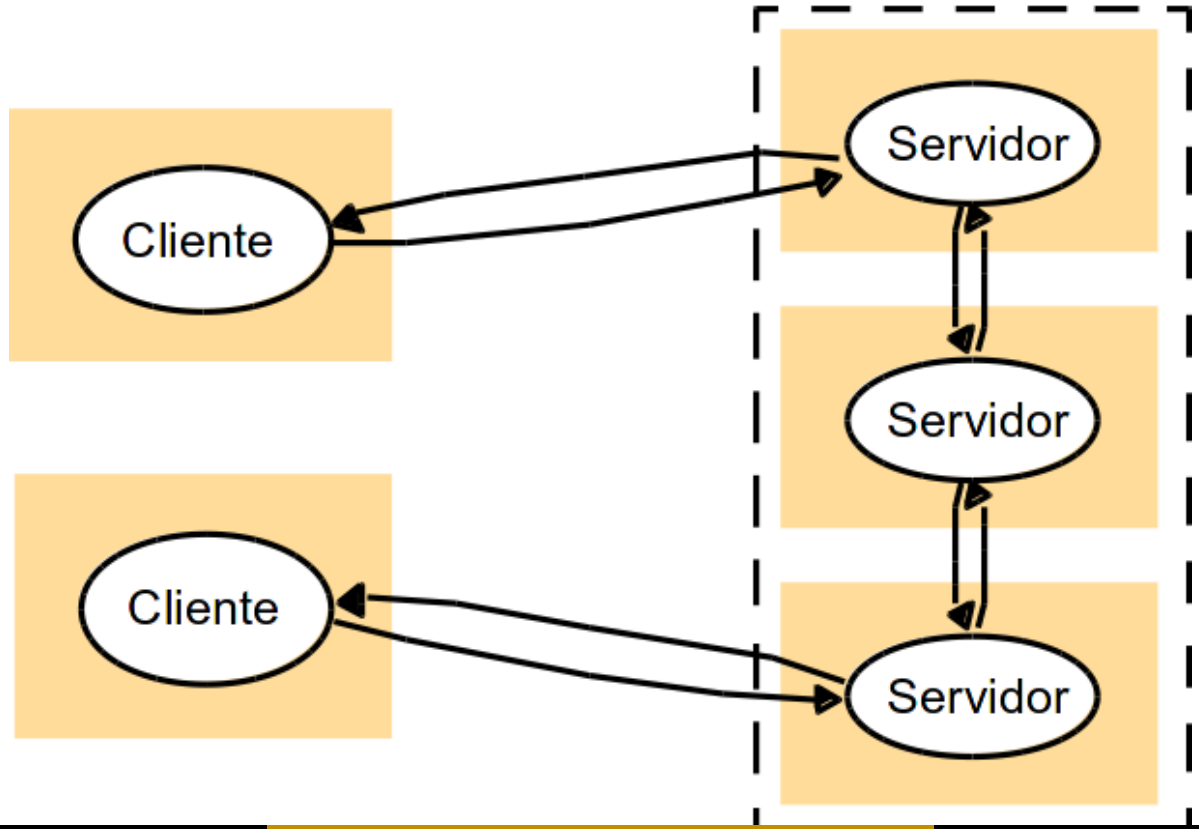
- Um servidor de email busca no DNS o endereço IP de outro servidor de email;
- Um serviço de busca on-line de passagens em várias companias aéreas.



Variação: Arquitetura Cliente/Servidor

Serviço fornecido por múltiplos servidores

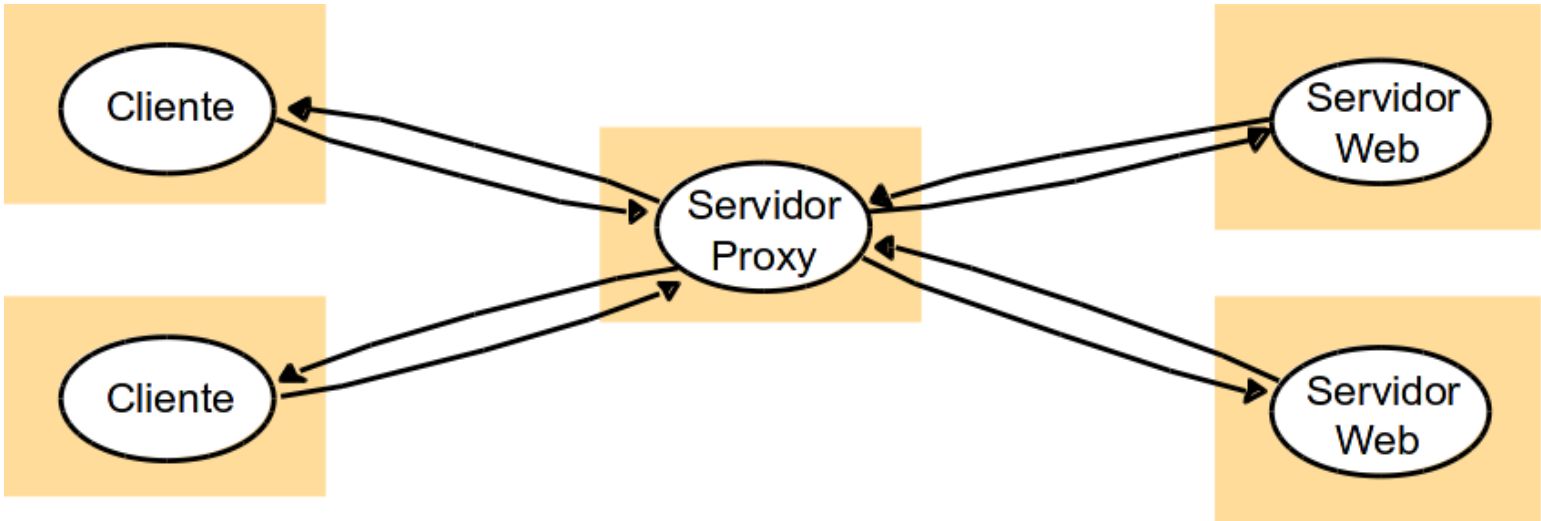
Neste modelo, os serviços são implementados em vários servidores (réplicas) que interagem entre si para oferecerem os serviços solicitados pelos processos clientes.



Variação: Arquitetura Cliente/Servidor

Servidor Proxy

- Servidores proxy mantêm cópias de dados solicitados anteriormente.
- Quando um cliente faz uma solicitação a um servidor proxy, primeiro é verificado se os dados estão presentes localmente, caso contrário a informação é buscada efetivamente na rede.
- O propósito geral é garantir desempenho e disponibilidade de dados sem aumentar a carga ou tráfego na rede utilizada.

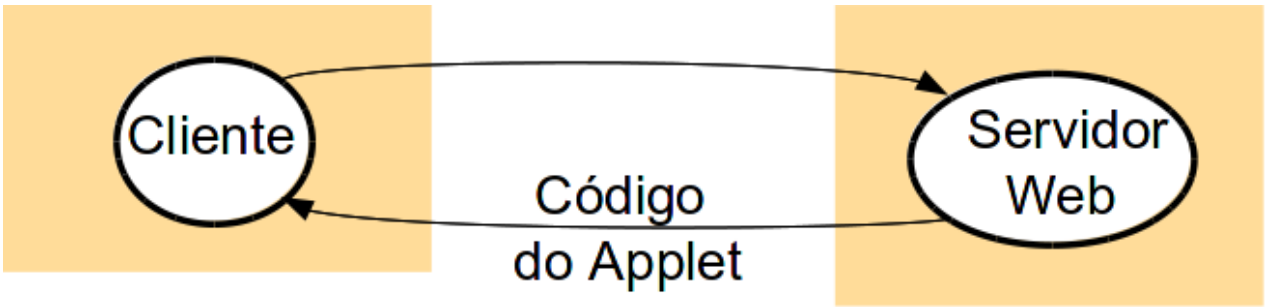


Variação: Arquitetura Cliente/Servidor

Código Móvel

Seu funcionamento é baseado de modo a permitir que os códigos sejam transferidos do servidor para o cliente, onde são executados. (vantagem: sem atraso/sem tráfego). Ex:applets

a) cliente requisita o download do código do applet



b) cliente interage com o applet

Variação: Arquitetura Cliente/Servidor

Código Móvel

Através de códigos móveis é possível oferecer serviços que não podem ser dado normalmente pela Web (modelo **pull** de comunicação). Ex:

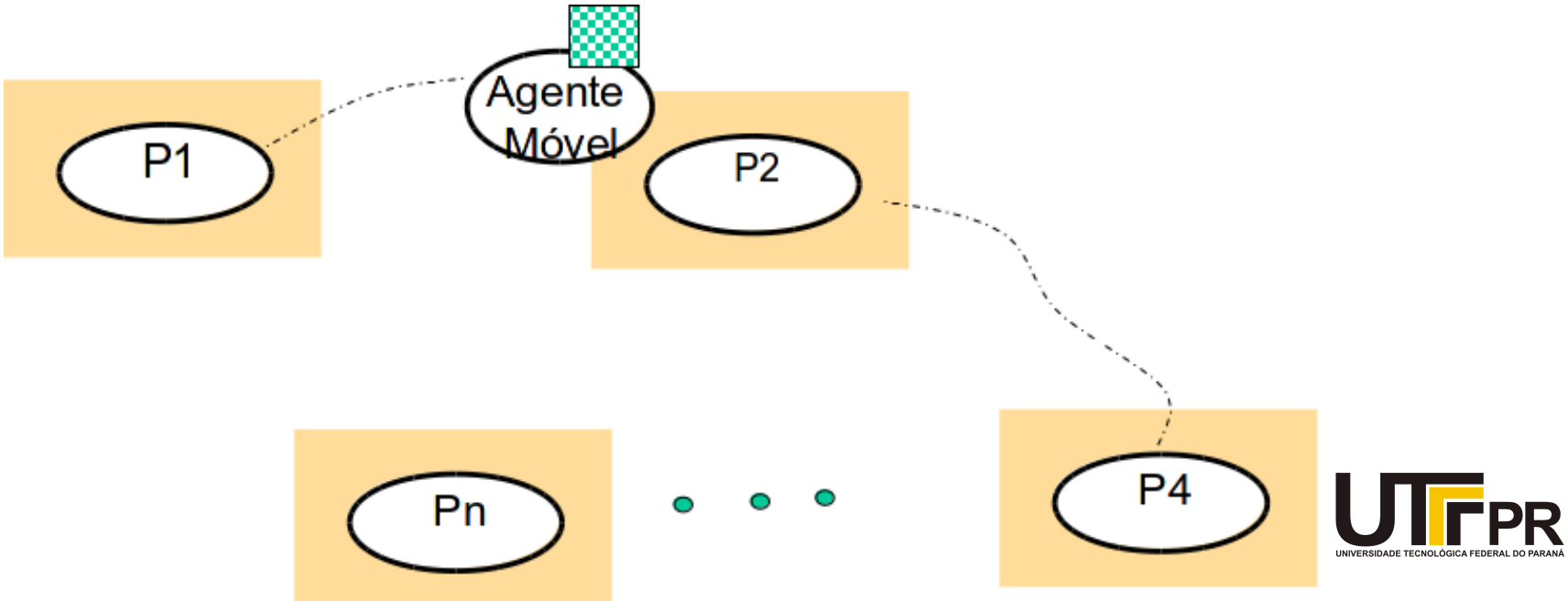
- Serviços de mensagens do tipo **push**
- O servidor inicia a comunicação

Variação: Arquitetura Cliente/Servidor

Agentes Móveis

São programas executáveis (**códigos e dados**) que trafegam de um computador ao outro para executar alguma tarefa. Ex: Aplicações que necessitem coletar diversas informações em muitas máquinas.

- A vantagem é a redução do tráfego de rede, devido a redução de número de chamadas remotas por parte de códigos fixos;



Variação: Arquitetura Cliente/Servidor

Agentes Móveis

Cuidados com a segurança:

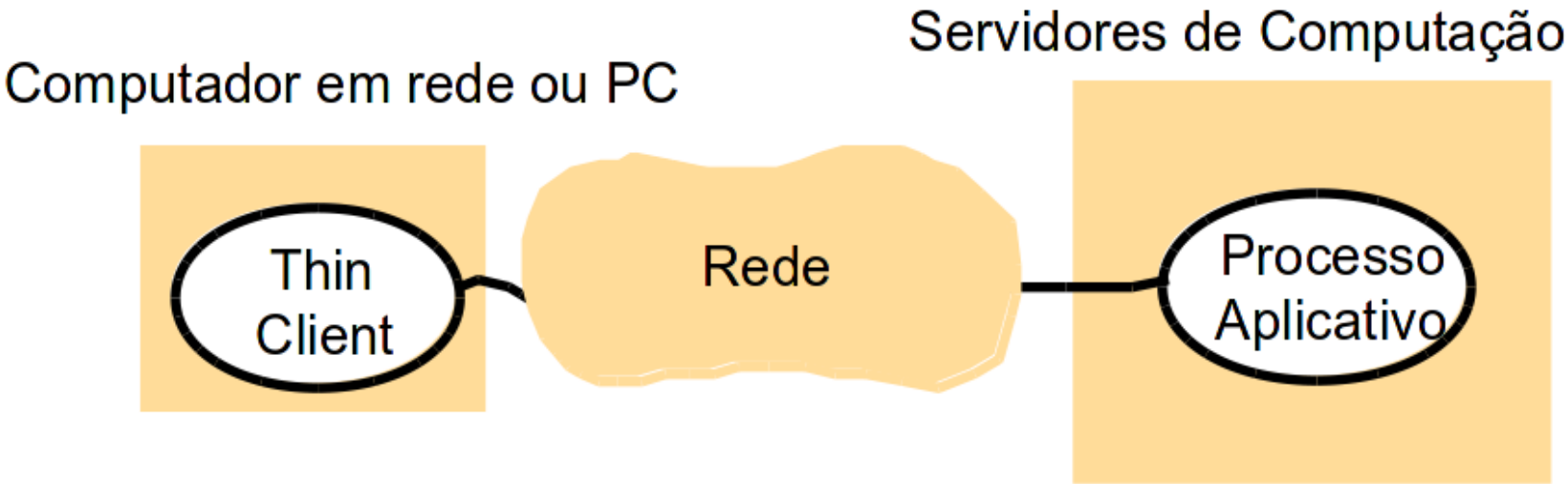
- O ambiente deve preocupar como e quais recursos pode prover aos agentes móveis.
- Por outro lado, os agentes também podem sofrer problemas e não completarem suas tarefas, pois não conseguiram ter acesso autorizado aos recursos necessários.

Variação: Arquitetura Cliente/Servidor

Thin Clients

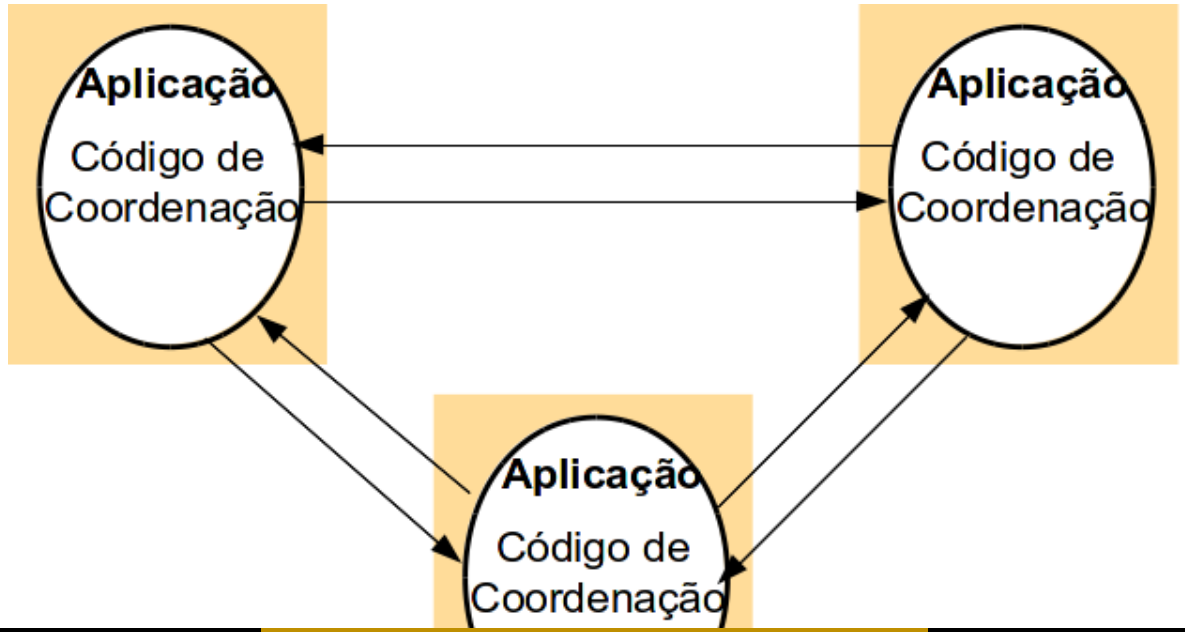
O termo "*thin client*" consiste da arquitetura em que os aplicativos são executados em um computador remoto (servidor).

- Diferente da arquitetura de computadores de rede, os aplicativos são todos executados nos servidores e não baixados;
- Servidores devem ser poderosos, capazes de executar diversos processos em paralelo para atender inúmeros clientes.



Arquitetura Peer-to-Peer

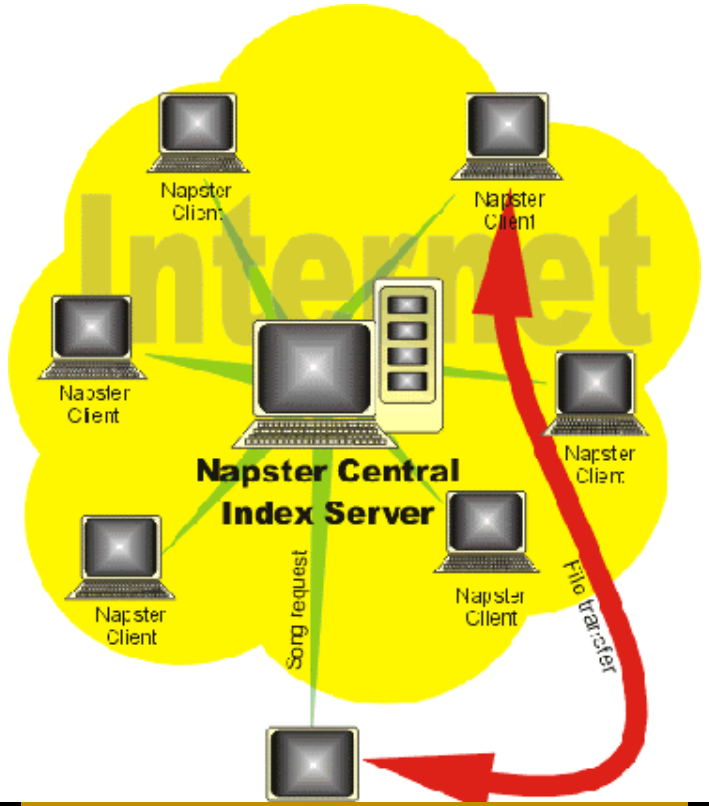
- Nesta arquitetura todos processos interagem com regras **semelhantes**;
- Trabalham de forma **cooperativa** para desempenhar atividades computacionais;
- Não há distinção entre cliente e servidor - papel igualitário.
- Exemplo: GnuTella



Arquitetura Hibrida

Combina aspectos da arquitetura cliente/servidor e peer-to-peer. Ex:

- Mensagem instântanea
- Algumas arquiteturas de compartilhamento de arquivos: napster, bittorrent.



Próxima aula

- Revisão: Programação de aplicações locais orientadas a objetos em java