Projeto de Software

# Arquiteturas de Sistemas Distribuídos

**Lesandro Ponciano**

2024

# Questões Motivadoras da Aula

1) O que é arquitetura de sistemas distribuídos?

2) Quais são os principais estilos arquitetônicos?

3) Quais são as principais arquiteturas de sistemas distribuídos?

4) Qual a relação entre arquitetura e *middleware*?

# Objetivos da Aula

- Contextualizar
  - arquitetura de software
  - arquitetura de software distribuído
- Discutir os principais estilos (padrões) arquitetônicos
- Contextualizar os softwares adaptativos

# Software Distribuído

- Sistemas distribuídos são frequentemente constituídos de partes de software
- As partes estão dispersas em múltiplas máquinas
- É crucial que este software esteja devidamente organizado
  - Arquitetura de Software

# Arquitetura de Software

"Software architecture is the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time"

(Garlan and Perry, IEEE TSE, April 1995)

- A arquitetura de um software é uma estrutura de componentes interconectados através de interfaces
  - Componentes são compostos de componentes menores e interfaces
  - A interação entre componentes ocorre através de suas interfaces

# Elementos Arquitetônicos

- Questões para se entender os elementos fundamentais de um sistema distribuído
  1) Quais são as entidades que estão se comunicando?
     - Objetos, componentes, *web services*
  2) Qual é o paradigma de comunicação utilizado?
     - Comunicação entre processos
     - Invocação de remota
     - Protocolos requisição-resposta
  3) Quais funções e responsabilidades na arquitetura global?
     - Cliente-servidor, peer-to-peer (P2P)
  4) Qual a localização (ou posicionamento) na infraestrutura distribuída física?
     - Serviços em vários servidores, cache, código móvel (ex.: applet), agentes móveis

# Estilos Arquitetônicos

- Também chamados "padrões arquitetônicos"

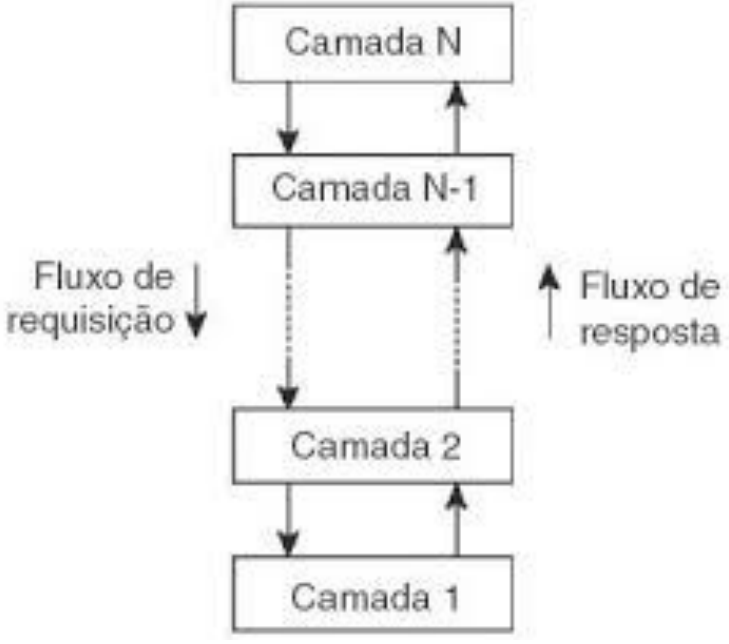
- Definem
  - a forma como os componentes se conectam
  - como os dados são trocados entre os mesmos
  - como são configurados conjuntamente no sistema

- Principais estilos
  1) Arquitetura em camadas
  2) Arquitetura baseada em objetos
  3) Arquiteturas centradas em dados
  4) Arquitetura baseada em eventos

# 1 Arquiteturas em Camadas

- Componentes são organizados hierarquicamente em camadas
  - Componente da Camada “N” pode invocar componentes da Camada “N-1”
  - Fluxo de controle passa de camada em camada

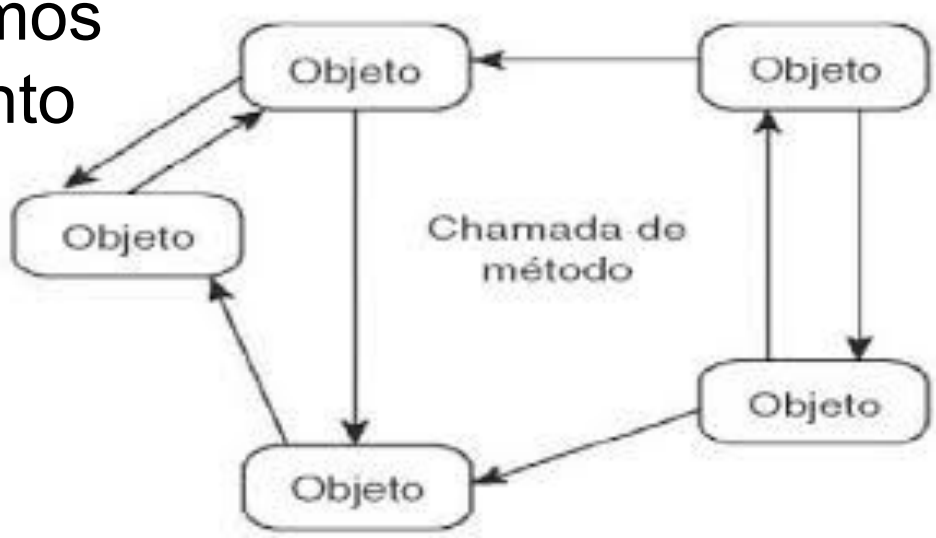
- Largamente utilizado em redes de computadores e sistemas cliente-servidor



2

# Arquiteturas Baseadas em Objetos

- Cada componente é um objeto
- Objetos se conectam uns aos outros através de mecanismos de chamada de procedimento (remoto)
- Amplamente utilizados em software de grande porte



3

# Arquiteturas Centradas em Dados

- Processos desacoplados no espaço e no tempo
  - não precisam se encontrarem ativados quando a comunicação iniciar
  - permite alcançar a transparência de distribuição

4

# Arquiteturas Baseadas em Eventos

- Processos se comunicam essencialmente através da propagação de eventos
  - eventos opcionalmente podem carregar dados
- A propagação de eventos vem geralmente associada com um *publish/subscribe system*
  - Requisição-resposta
  - Comunicação entre produtores e consumidores de informações

# Arquiteturas Baseadas em Eventos

# Arquitetura de Sistema

1) Arquiteturas centralizadas
   - Camadas de aplicação
   - Arquiteturas multi-divididas

2) Arquiteturas descentralizadas
   - Arquiteturas peer-to-peer estruturadas
   - Arquiteturas peer-to-peer não estruturadas

3) Arquiteturas Híbridas
   - Sistemas distribuídos colaborativos

# 1 Arquiteturas Centralizadas

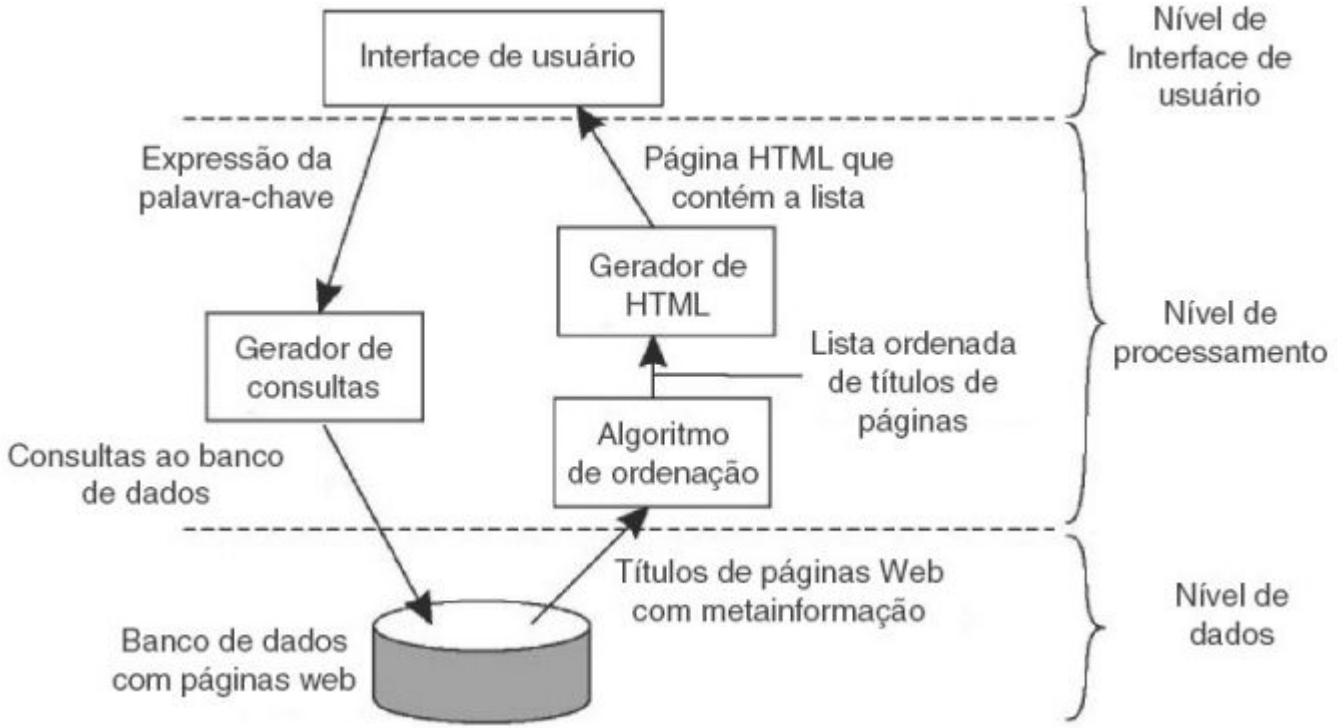
- Comportamento cliente-servidor ou requisição - resposta
  - Há clientes que requisitam serviços por meio do envio de requisições e aguardando respostas
  - Há um servidor que implementa um serviço específico

# Camadas de Aplicação

- Nem sempre é clara a distinção entre cliente e servidor
- Muitas aplicações cliente-servidor visam dar suporte ao acesso de usuários ao banco de dados
- Nesse caso, defende-se a implementação em camadas com distinção em três níveis
  - Nível de interface do usuário
  - Nível de processamento
  - Nível de dados

# Exemplo

- Organização simplificada de um mecanismo de busca na internet em três camadas diferentes



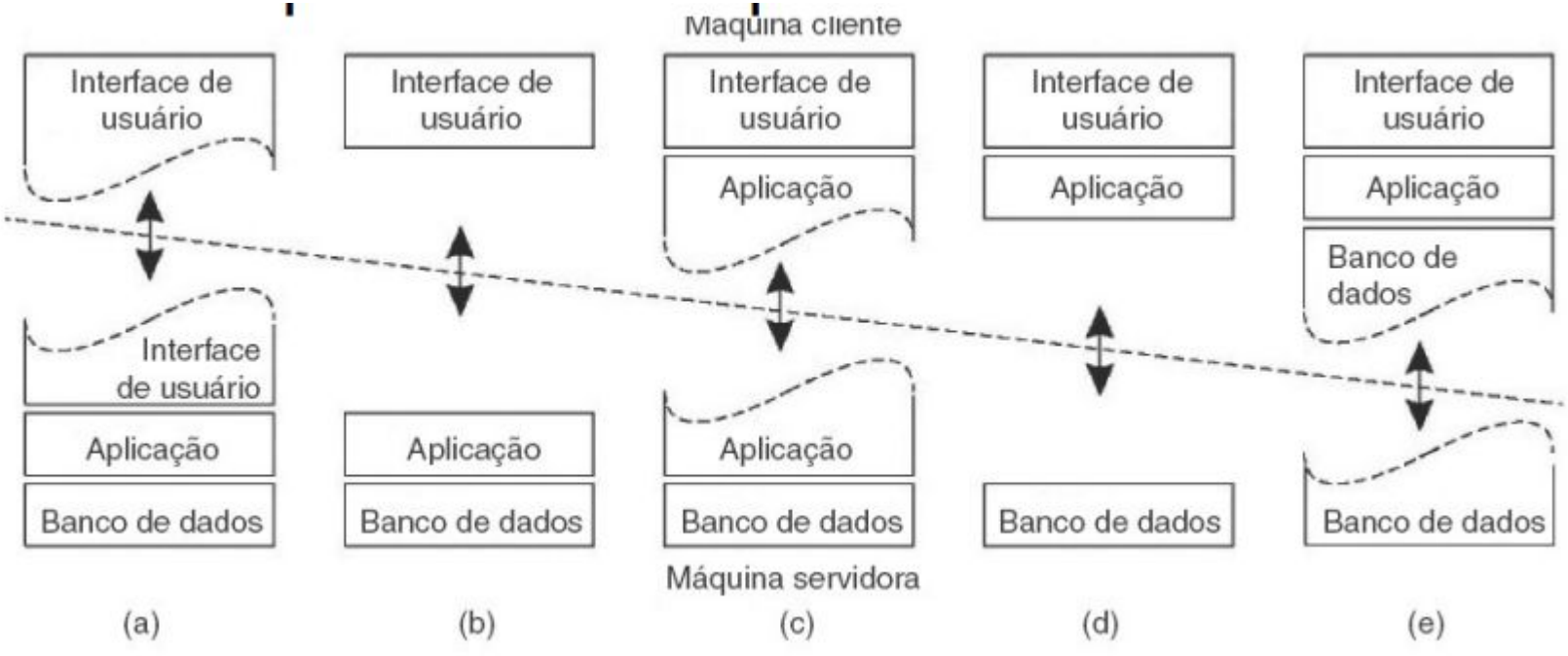
# Arquiteturas Multi-divididas

- A distinção em níveis lógicos sugere várias possibilidades para distribuição física da aplicação cliente-servidor por várias máquinas

- Uma organização simples (dois tipos de máquinas)
  - Clientes com programas que implementam o nível de interface com o usuário
  - Servidor que contém programas que implementam o nível de processamento e de dados

# Alternativas de Organização

- Alternativas de organizações cliente-servidor com dois tipos de máquinas (cliente e servidor) e diferenciação entre o que executa em cada tipo



# Arquitetura em 3 Divisões Físicas

Servidor de aplicação agindo como cliente

# 2 Arquiteturas Descentralizadas

- Distribuição horizontal
  - Cliente e servidor podem ser divididos em partes logicamente equivalentes
  - Cada parte trabalha em sua porção do conjunto de dados

- Esse é o caso dos sistemas peer-to-peer (P2P)
  - Comportamento simétrico: cada máquina pode servir como um cliente ou como um servidor

# Sistemas P2P

- Formados por um conjunto de nós organizados em um *overlay* ou rede de sobreposição

- Rede de sobreposição
  - Nós são processos
  - Enlaces são canais de comunicação possíveis

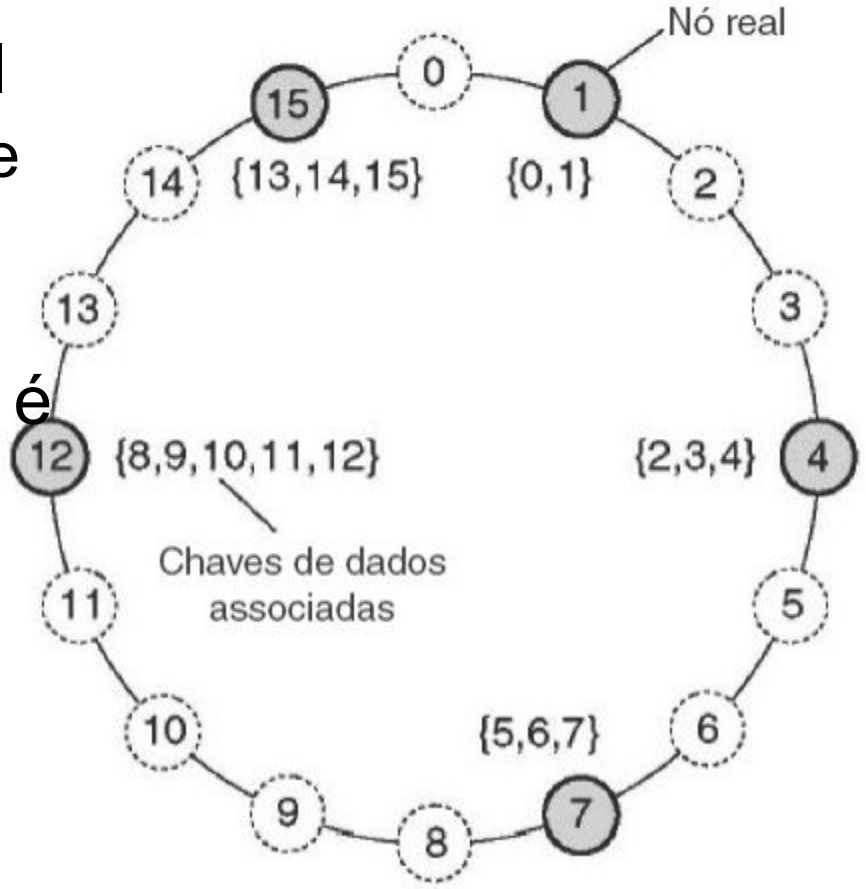
- Arquiteturas estruturadas ou  não estruturadas

# Arquiteturas P2P Estruturadas

- Procedimento determinístico de estruturação
  - Tabela de hash distribuída (DHT, *Distributed Hash Table*)
- DHT
  - Itens de dados recebem uma chave aleatória (de 128 a 160 bits)
  - Nós do sistema recebem um número no mesmo espaço de identificadores
- Sistema em DHT mapeia de forma eficiente e determinística itens de dados a nós da rede

# Mapeamento

- Organização lógica em anel
  - Nós têm informações sobre seu predecessor
- Item de dados com chave k é mapeado para o nó que tenha identificador id ≥ k



# Operações

- Para encontrar um dado k: Lookup(k)
  - Cada nó manterá atalhos para outros nós
  - O(log(N)) número de etapas, onde N é o número de nós que participam da rede de sobreposição
- Para se juntar ao sistema
  - Gerar Id
  - Contatar succ(id) e seu predecessor
  - Entrar no anel
- Para sair do sistema
  - Informar o predecessor e sucessor
  - Transferir dados para o sucessor

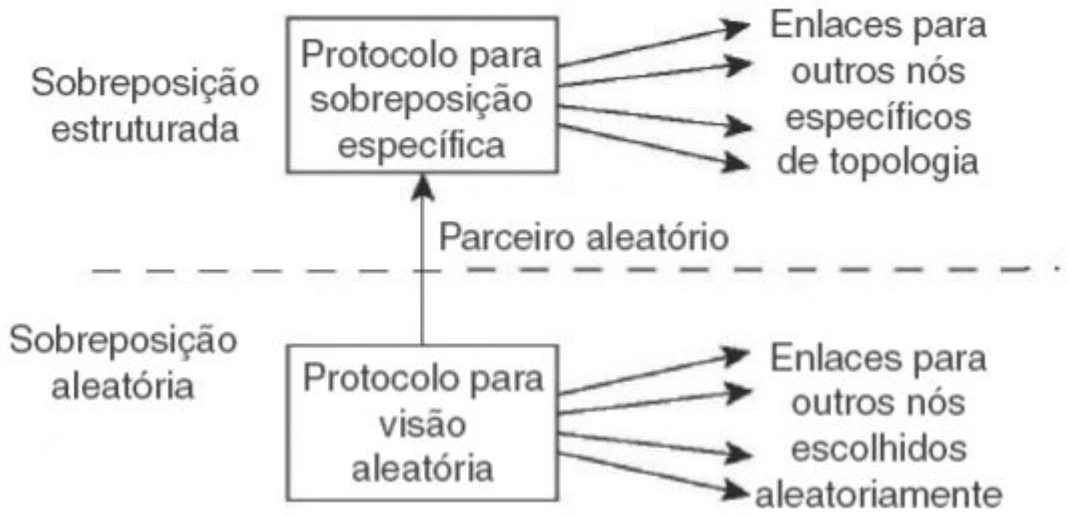
# P2P não Estruturado

- Rede de sobreposição parecida com um grafo aleatório
  - Cada nó mantém uma lista de *c* vizinhos (visão parcial)
  - um vizinho é um nó vivo escolhido de forma aleatória no conjunto de nós vigentes no momento

- Características
  - Dados são colocados de forma aleatória nos nós
  - Para encontrar um dado é necessário inundar a rede de sobreposição

# Gerenciamento de Topologia em P2P

- Abordagem de duas camadas para construir e manter topologias específicas de sobreposição usando técnicas de sistemas P2P não estruturados



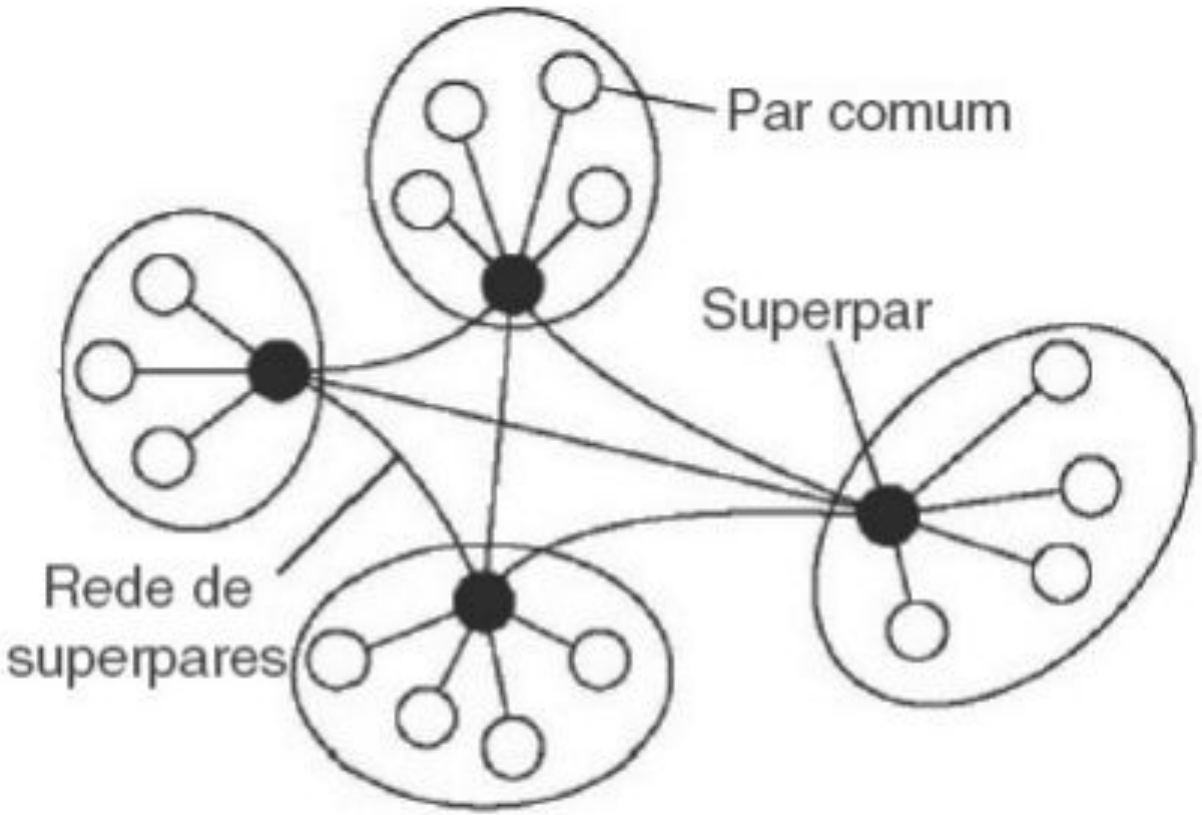
# Superpares (*superpeers*)

- A medida que a rede cresce, localizar itens de dados em sistemas P2P não estruturados pode ser problemático
  - inundar a rede

- Solução: superpares
  - Nós que mantêm o índice de dados ou que agem como nós intermediários que possuem dados para disponibilizar os recursos a nós vizinhos
  - Toda comunicação de/para um par comum ocorre por meio daquele superpar associado ao par

- Novo problema: Seleção do líder (superpar)

# Superpares

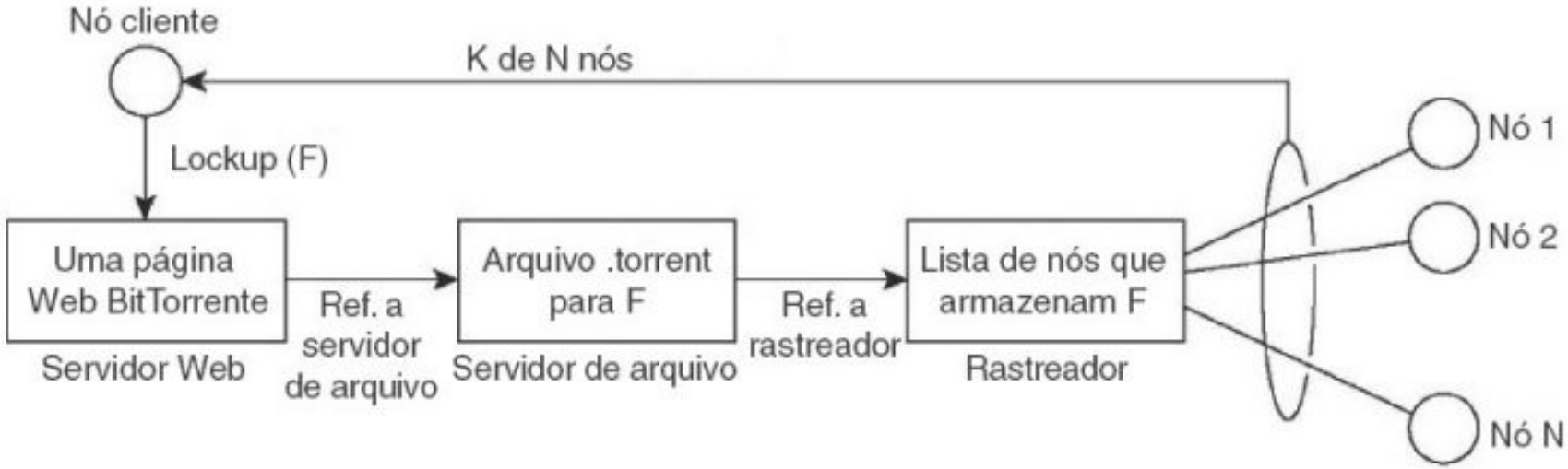


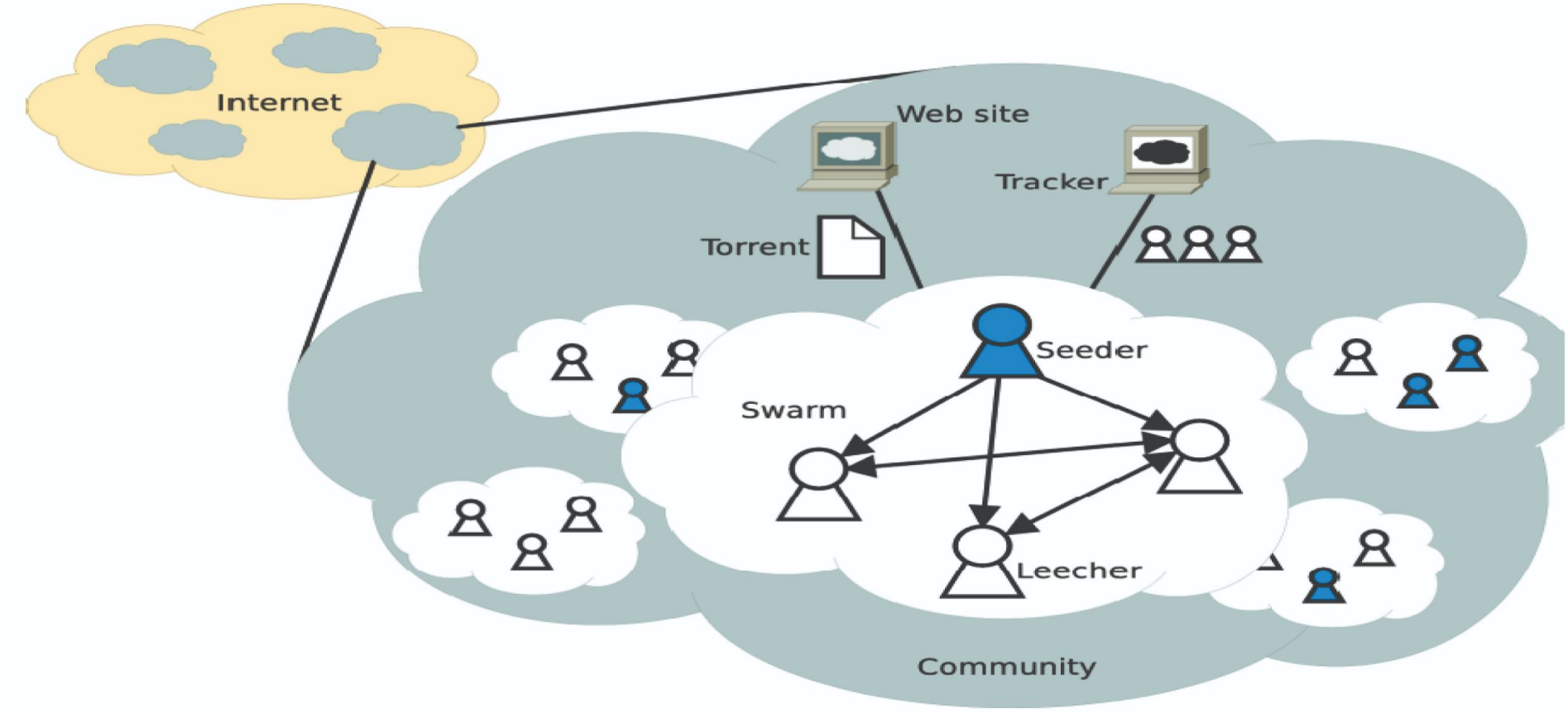
# 3 Arquiteturas Híbridas

- Sistemas distribuídos nos quais solução cliente-servidor é combinada com arquiteturas descentralizadas

- Exemplo: Sistemas distribuídos colaborativos
  - Principal objetivo é iniciar a troca de informações
  - Após adição do nó na rede, a distribuição dos dados é feita de forma descentralizada
  - BitTorrent

# BitTorrent


Nó cliente
K de N nós
Lockup (F)
Uma página Web BitTorrente
Servidor Web
Ref. a servidor de arquivo
Arquivo .torrent para F
Servidor de arquivo
Ref. a rastreador
Lista de nós que armazenam F
Rastreador
Nó 1
Nó 2
Nó N


Internet
Web site
Tracker
Torrent
Seeder
Swarm
Leecher
Community

# Arquitetura *versus* Middleware

- Middleware é “software” que oferece serviços para as aplicações tendo por base serviços disponíveis no sistema operacional
  - middleware interage com o sistema operacional, mas não é parte dele
- Sistemas de middleware seguem um estilo arquitetônico específico
- Ideia principal: ser simples de configurar, adaptar e personalizar conforme necessidade da aplicação
  - Solução: Interceptores

# Interceptores

- Interceptador é um software que interromperá o fluxo de controle usual e permitirá que seja executado um outro código

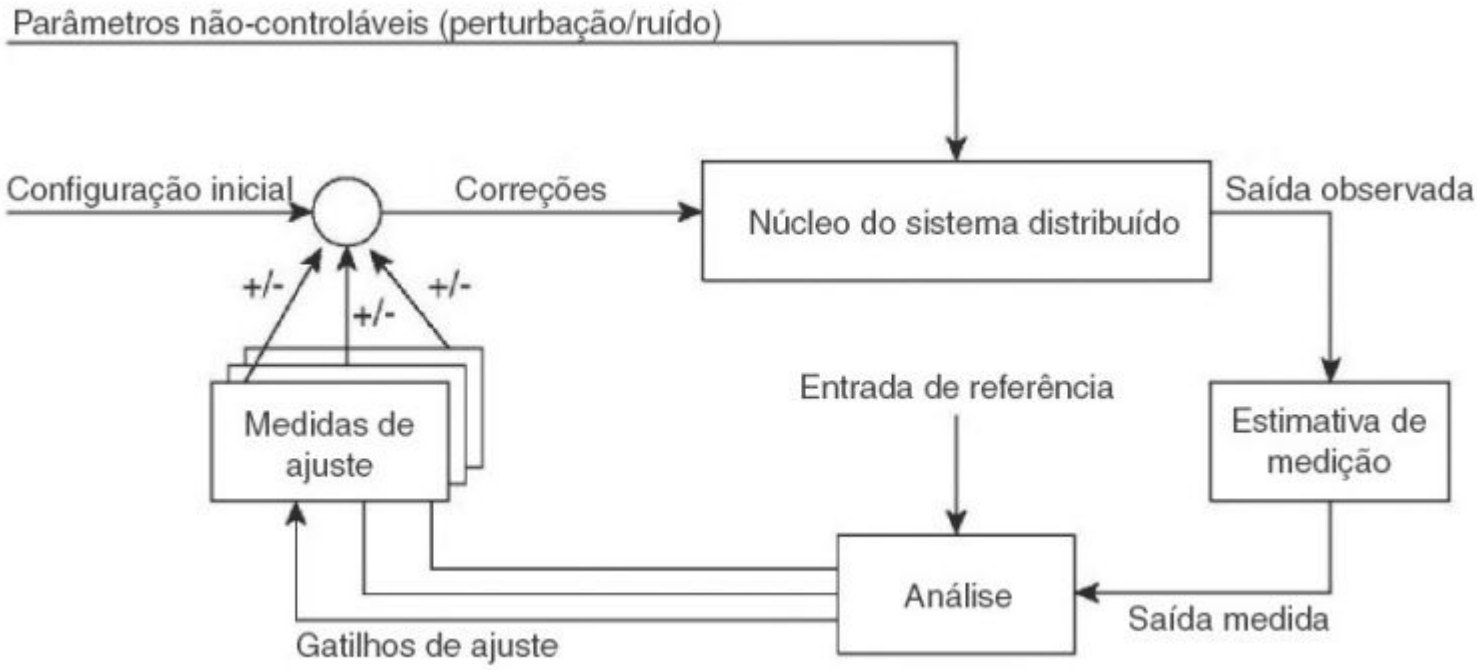
- Exemplo: Objeto A chama um método do objeto B, que está em uma máquina diferente de A
  - É oferecida a A uma interface local idêntica à de B
  - Chamada de A é transformada (pelo middleware) em uma chamada genérica
  - A chamada genérica é transformada em mensagem e enviada a B

# Autogerenciamento

- Sistemas distribuídos precisam fornecer soluções gerais de blindagem contra aspectos indesejáveis inerentes a redes
- O objetivo é suportar o maior número possível de aplicações
- Solução
  - Construir sistemas onde seja possível fazer monitoração e ajustes
  - Sistemas distribuídos adaptativos ou autogerenciados

# Autogerenciamento

- Organização lógica de um sistema de realimentação de controle



# Atividade de Fixação

- O que é arquitetura de sistemas distribuído?
- Quais são os principais estilos arquitetônicos e as principais arquiteturas de sistemas distribuídos?
- Qual a relação entre arquitetura e middleware?
- O que é o auto-gerenciamento e por que ele é necessário?

# Referências

TANENBAUM, Andrew S.; Steen, Maarten van. Sistemas Distribuídos: princípios e paradigmas - 2a edição. Pearson 416 ISBN 9788576051428. (Capítulo 2).

SISTEMAS distribuídos conceitos e projeto. 5. Porto Alegre Bookman 2013 ISBN 9788582600542. (Capítulo 2)

# Projeto de Software

**Prof. Dr. Lesandro Ponciano**

https://orcid.org/0000-0002-5724-0094