Cap. 02 – Arquiteturas de Sist. Distribuídos

2.1 – Estilos Arquiteturais

2.2 – Arquiteturas de Sistemas Distribuídos

- 2.2.1 Arquiteturas Centralizadas
- 2.2.2 Arquiteturas Descentralizadas
- 2.2.3 Arquiteturas Híbridas

2.3 – Arquiteturas *vs* Middleware

- 2.3.1 Interceptadores
- 2.3.2 Abordagem para Software Adaptativo
- 2.3.3 Discussão

Cap. 02 – Arquiteturas de Sist. Distribuídos

2.4 – Auto Gerenciamento em Sistemas Distribuídos

- 2.4.1 Modelo de Controle com Realimentação
- 2.4.2 Monitoramento de Sistema com Astrolabe
- 2.4.3 Estratégias de Replicação Diferenciadas
- 2.4.4 Reparo Automático de Componente no JADE

Referências Bibliográficas

- Andrew S. Tanenbaum; Maarten van Steen - Distributed Systems: Principles and Paradigms, Prentice-Hall, 2007, ISBN-10: 0132392275, ISBN-13: 9780132392273

  … Lectures dos autores Andrew S. Tanenbaum e Maarteen van Steen (“www.cs.vu.nl” e “www.distributed-systems.net”)

- George Coulouris; Jean Dollimore; Tim Kindberg – Sistemas Distribuídos: Conceitos e Projeto, Bookman, 4th Edition, 2007, ISBN 9788560031498

- Notas de Aula do Prof. Ricardo Anido do Instituto de Computação (IC) da UNICAMP - “www.ic.unicamp.br/~ranido”

Cap. 02 - Arquiteturas de Sistemas Distribuídos

- "**sistema distribuído**" – são frequentemente constituídos de partes complexas de software, componentes que estão dirpersos por definição em múltiplas máquinas.
- … para dominar/controlar esta complexidade, é crucial que estes sistemas sejam devidamente organizados.
- … esta organização é principalmente sobre os componentes de "software" que compõem o sistema distribuído.
- **Arquitetura de Software** de Sistema Distribuído – nos informa como estes componentes de "software" estão organizados e como devem interagir entre si bem como com outros componentes.
- … concepção/implantação efetiva de um sistema distribuído requer a instanciação e associação de componentes de software em máquinas reais, o que pode ser feito de diferentes maneiras.

2.1 – Estilos Arquiteturais

- **Arquitetura de Software** de Sistema Distribuído – nos informa como estes componentes de “software” estão organizados e como devem interagir entre si bem como com outros componentes.
- … instanciação final de uma **Arquitetura de Software** é comumente referenciada como **Arquitetura de Sistema**.
- … como discutido no Cap. 01 um dos objetivos de um sistema distribuído é o separar aplicações da plataforma subjacente através da camada de “middleware”.
- … adoção de uma camada como esta é uma decisão arquitetural importante e o seu principal propósito é o de prover transparência de distribuição (Cap. 01)

... 2.1 – Estilos Arquiteturais

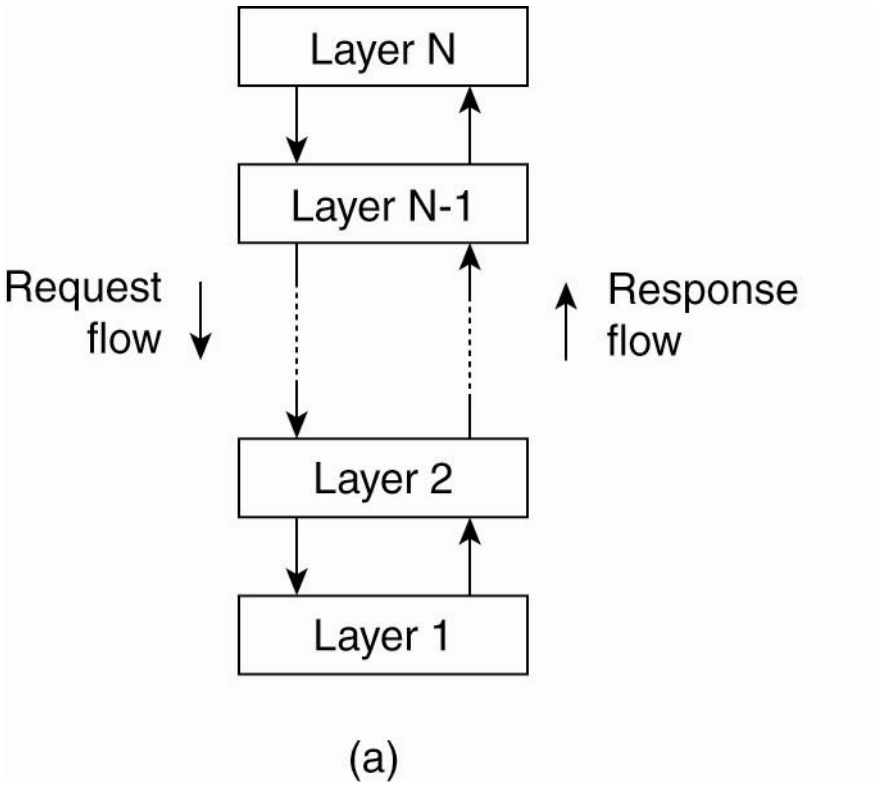
- **Estilo Arquitetural** – define a forma como os componentes se conectam uns com os outros, como os dados são trocados entre os mesmos bem como são configurados conjuntamente no sistema;
- … **componente** (“**component**”) – unidade modular com interfaces bem definidas que são necessárias a sua operação e passíveis de substituição no seu ambiente.
- … **conector** (“**connector**”) – mecanismo responsável pela mediação da comunicação, coordenação e cooperação entre os componentes.

- **Idéia** - organizar o sistema em componentes logicamente diferentes e distribuir estes componentes em diversas máquinas.

... 2.1 – Estilos Arquiteturais

- "**arquitetura em camadas**" – componentes são organizados em camadas de modo que um componente da Camada "N" invoque componentes da Camada "N-1", mas não outro fora desta ordem.

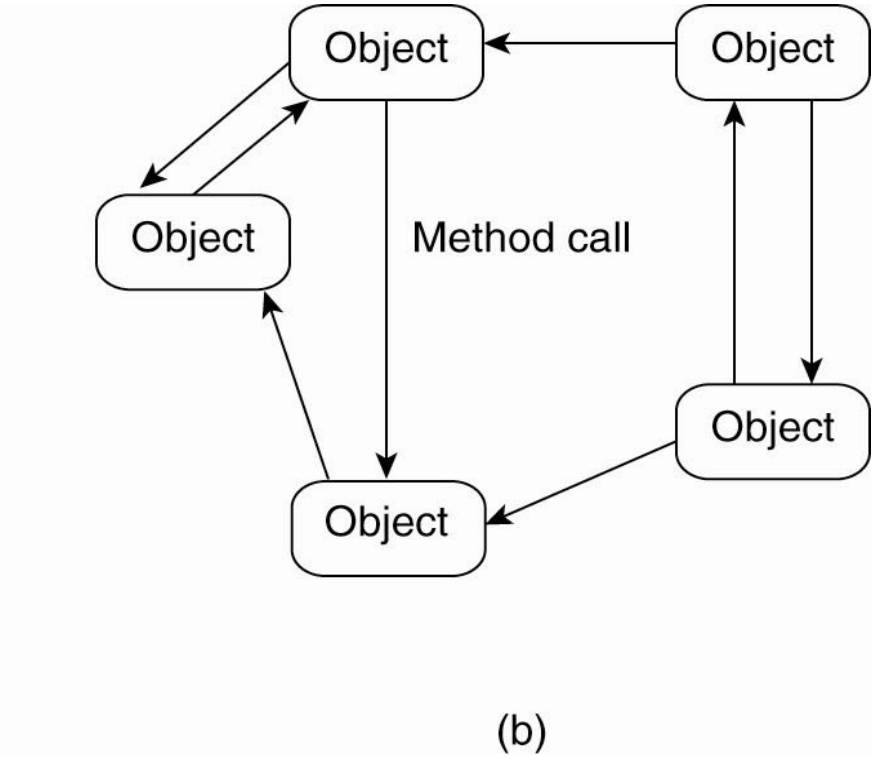
- Obs.: … fluxo de controle passa de camada em camada, na ordem em que forma a estrutura hierárquica;
- … largamente utilizado em redes de computadores e sistemas cliente-servidor.



... 2.1 – Estilos Arquiteturais

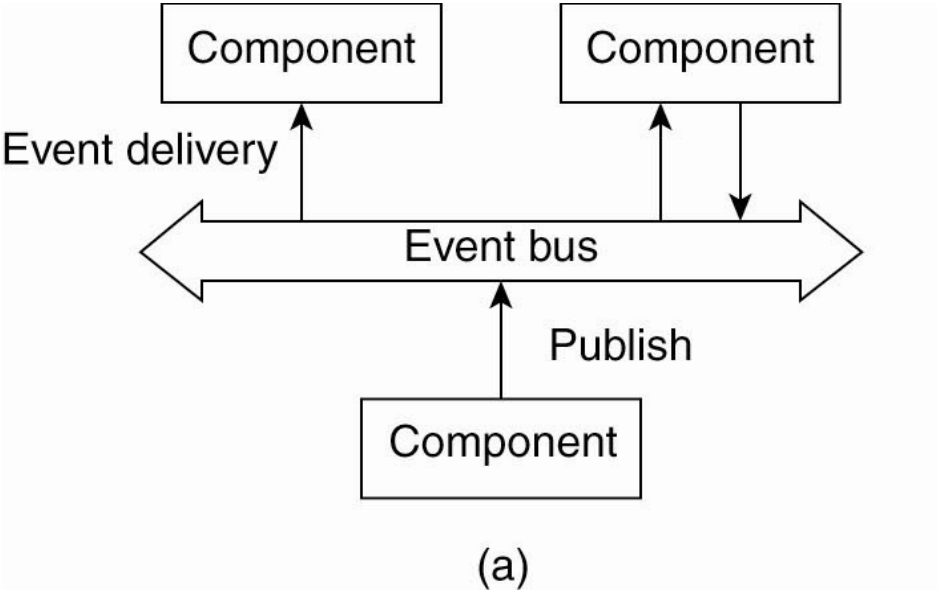
- "**arquitetura baseada em objetos**" – cada componente é um objeto que se conectam uns aos outros através de mecanismos de chamada de procedimento (remoto).

- … esta arquitetura combina com a arquitetura cliente-servidor já descrita;
- … arquitetura baseada em objetos e camadas ainda formam os mais importantes estilos para a maioria dos sistemas de software;
- … largamente utilizado em sistemas distribuídos.



... 2.1 – Estilos Arquiteturais

- "**arquitetura baseadas em evento**" – processos se comunicam essencialmente através da propagação de eventos, que opcionalmente podem carregar dados.
- … em sistemas distribuídos a propagação de eventos vem geralmente associada com o que conhecemos por "publish/ subscribe system".

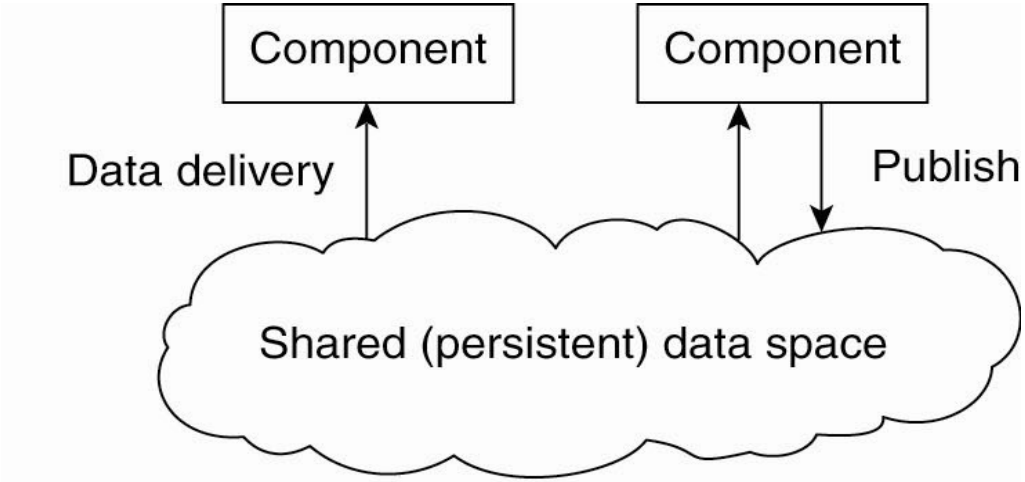


... 2.1 – Estilos Arquiteturais

- … principal vantagem do sistema baseado em eventos é o de que os processos são fracamente acoplados, ou seja, a princípio não precisam referenciar um ao outro para se comunicarem.
- … esta modalidade de troca de informações é referenciada como sendo desacoplada no espaço (“anonimato” entre os objetos) ou “**referentially decoupled**”.
- Arquiteturas baseadas em Evento podem ser combinadas com Arquitetura Centrada em Dados – o que é conhecido por Espaco de Dados Compartilhado (desacoplamento no espaco e tempo).

... 2.1 – Estilos Arquiteturais

- essência – processos são desacoplados no espaço e também no tempo, ou seja, eles não precisam se encontrarem ativados quando a comunicação iniciar.
- … o que torna estas arquiteturas importantes para sistemas distribuídos é que elas tem também por objetivo alcançar a transparência de distribuição.



(b)

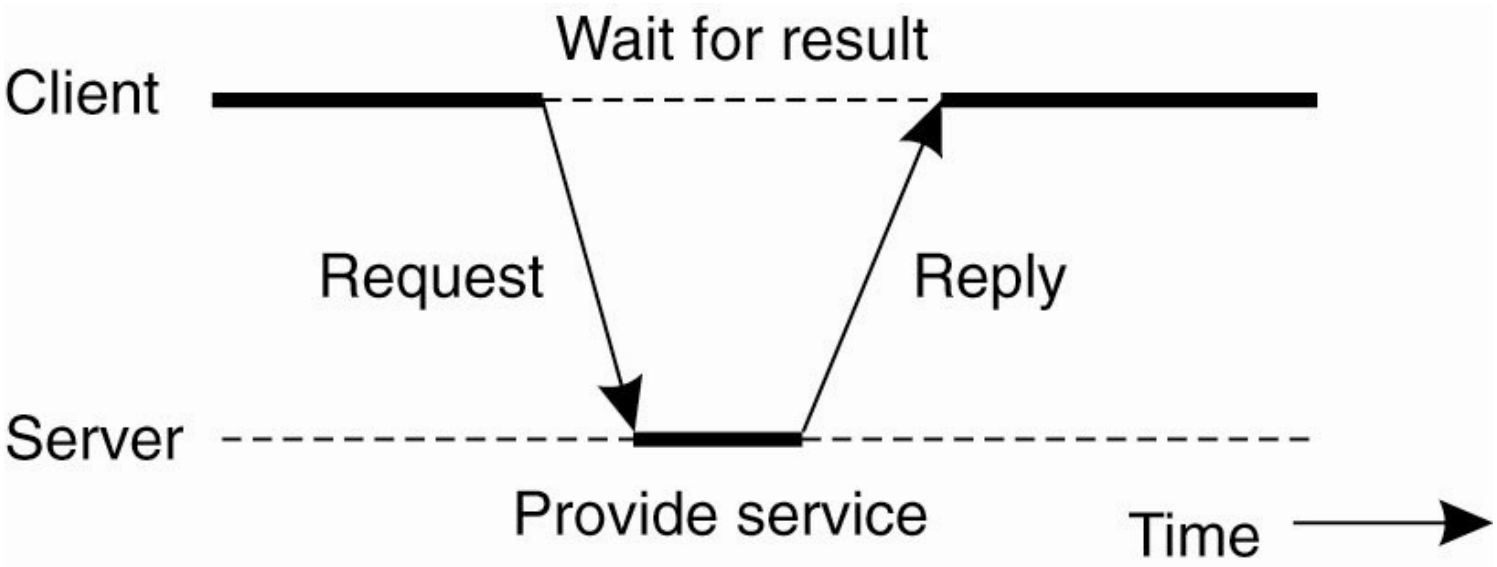
... 2.1 – Estilos Arquiteturais

- Arquitetura de Software de Sistema Distribuído – nos informa como estes componentes de “software” estão organizados e como devem interagir entre si bem como com outros componentes.
- … instanciação final de uma Arquitetura de Software é comumente referenciada como Arquitetura de Sistema.

- **Arquitetura de Sistema** – decidir quais são os componentes de software, suas interações e localizações significa gerar uma instância da **Arquitetura de Software**.
- … dentres as abordagens possíveis, destacam-se: Arquitetura Centralizada, Descentralizada e Híbrida.

2.2.1 – Arquiteturas Centralizadas

- Modelo Cliente/Servidor: processos em um sistema distribuído são divididos em 02 grupos – os que oferecem serviços (servidores) e os que usam serviços (clientes).
- … clientes e servidores espalhados na rede interagem através de mensagens do tipo “request/replay”.

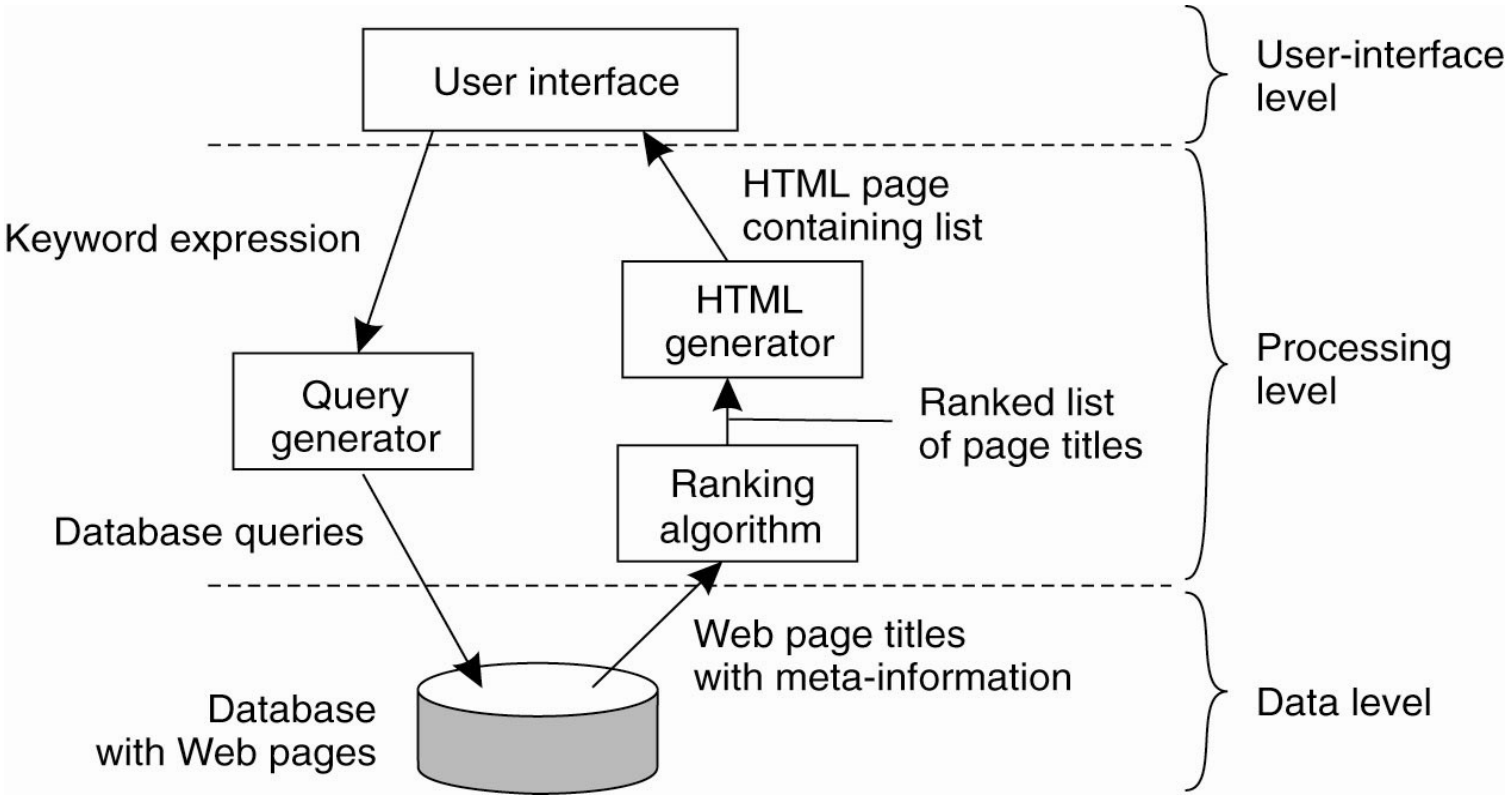


… 2.2.1 – Arquiteturas Centralizadas

- … considerando que mensagens “request/replay” não sejam perdidas ou danificadas, o protocolo “request/replay” funciona bem mesmo sobre um protocolo não orientado a conexão (LAN);
- … infelizmente, tornar o protocolo resistente a falhas ocasionais de transmissão não é trivial – cliente não pode detectar qual das mensagens se perdeu (e.g., “request” ou “reply”).
- operação **idempotente** – pode ser repetida inúmeras vezes sem prejuízos, mas por outro lado, nem todas mensagens do tipo “request” são idempotentes => não há solução simples para tratar a perda de mensagens.
- alternativa – utilização de protocolo orientado a conexão confiável, o que por outro lado pode comprometer a performance.

Aplicação em Camadas

- Considerando que muitas das Aplicações Cliente/Servidor estão direcionadas para acesso a banco de dados, há um consenso de que o estilo arquitetural é o de camadas.



… Aplicação em Camadas

- "**User Interface Leve**l" - contém o que é necessário para interagir com o usuário, ou seja, define a interface de comunicação;
- "**Processing Level**" – contém a aplicação (sem os dados);
- "**Data Level**" – contém os dados que o cliente quer acessar.

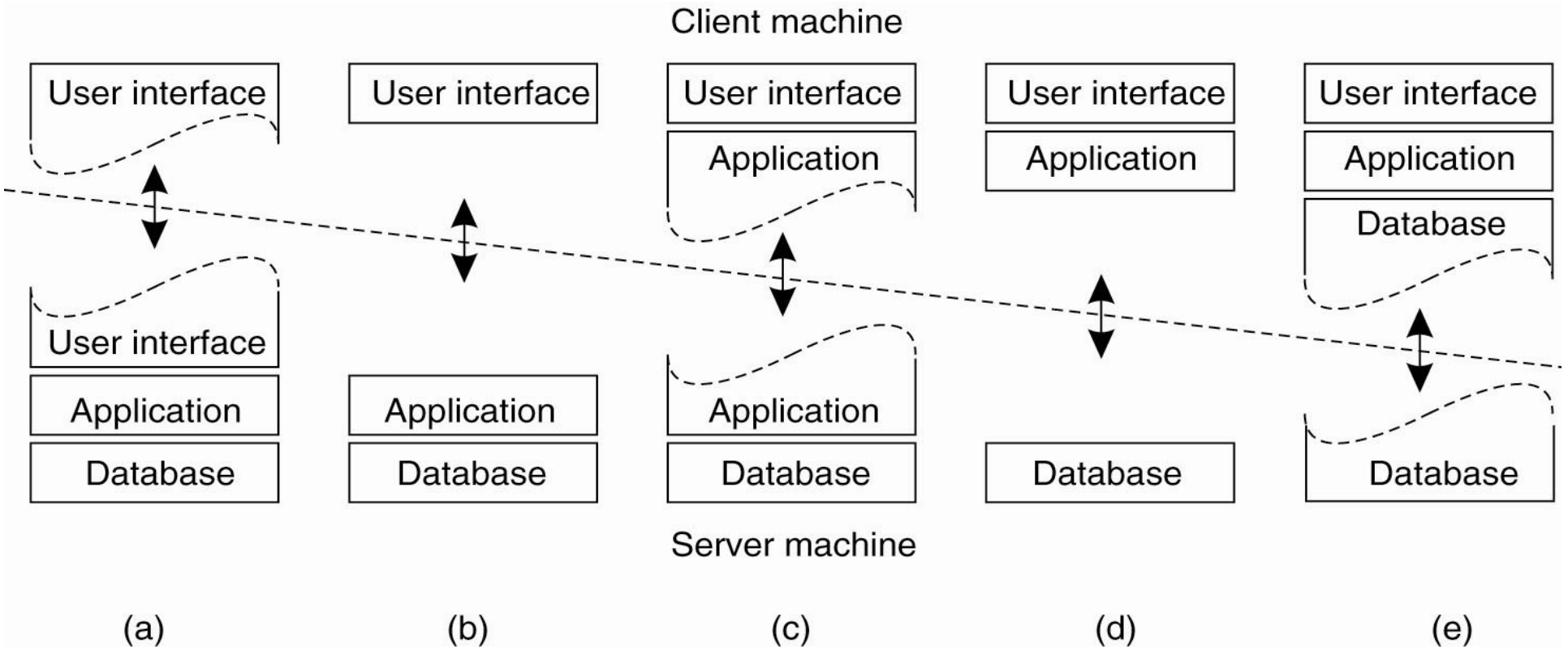
- Observação: … estas camadas são encontradas em muitos sistemas distribuídos de informação e utilizam tecnologia tradicional de banco de dados e de aplicações.

Arquitetura em Multicamadas

- Conclusão - distinção entre 03 níveis lógicos, sugere inúmeras possibilidades para distribuir a aplicação ou partes do cliente/servidor sobre diferentes máquinas.
- … organização mais simples é a de termos 02 tipos:
  - máquina cliente – contém somente programas implementando o nível de interface do usuário ou parte deste nível;
  - máquina servidor – contém programas que implementam os níveis de processamento e de dados.
- Nesta abordagem, praticamente tudo é manipulado pelo servidor enquanto o cliente é essencialmente não mais que um terminal 'burro' com provavelmente uma interface gráfica.

… Arquitetura em Multicamadas

- Uma outra abordagem consiste em distribuir os programas na camada de aplicação discutidos anteriormente em diferentes máquinas, como mostrado na figura abaixo.

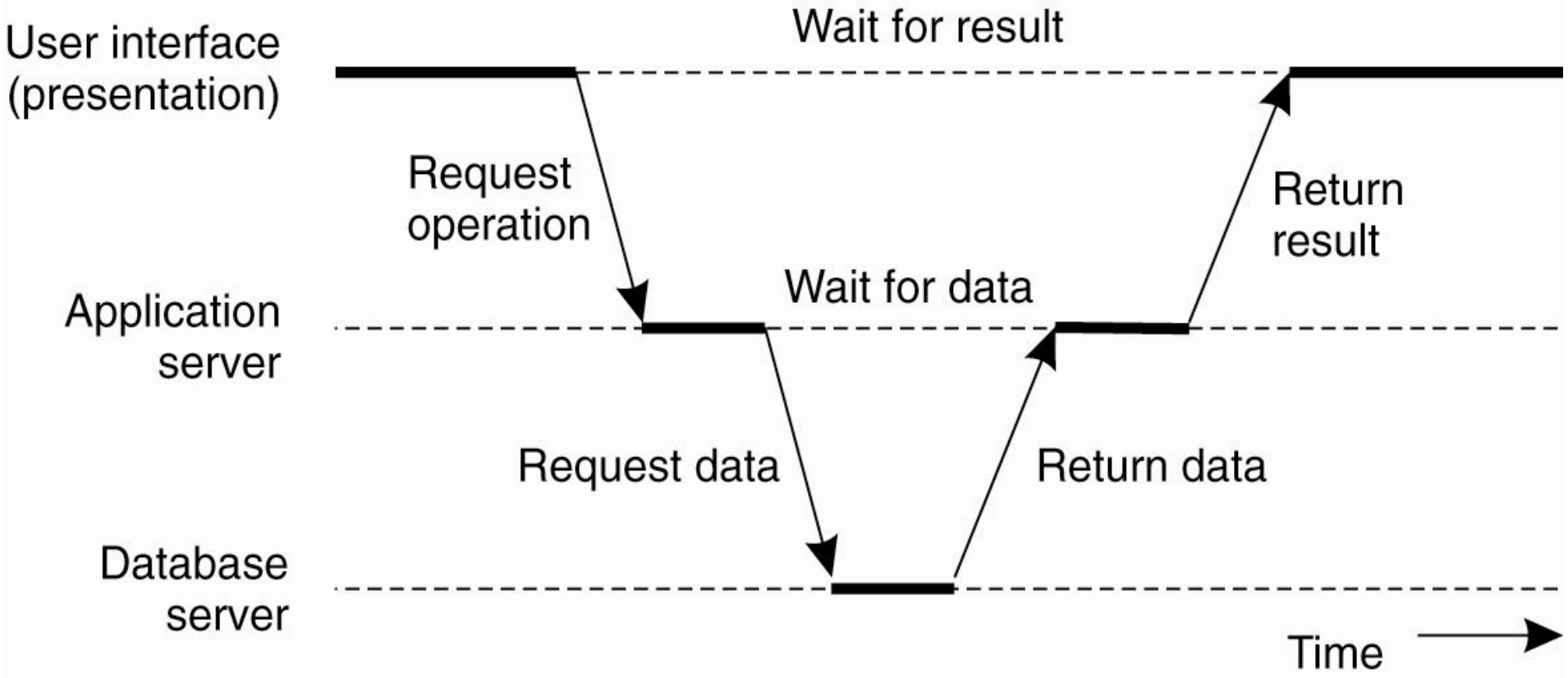


… Arquitetura em Multicamadas

- Da perspectiva de gerenciamento de sistemas, “fat clients” não são a melhor escolha, ao invés disso, “thin clients” são mais fáceis
- … custo de interfaces de usuários menos sofisticadas bem como desempenho aparente no cliente.
- … distinção entre 02 tipos de máquinas (máquina cliente e máquina servidor) é conhecido como “**two tiered architecture**”.
- … nessa lógica perdemos o possibilidade de que o servidor pode algumas vezes se comportar como cliente – abordagem esta conhecida como “**tree tiered architecture**”.

… Arquitetura em Multicamadas

- … nessa lógica perdemos o possibilidade de que o servidor pode algumas vezes se comportar como cliente – abordagem esta conhecida como "**tree tiered architecture**".



2.2.2 – Arquiteturas Descentralizadas

- Arquiteturas Multicamadas são uma consequência direta da divisão de aplicações em interface do usuário, componente de processamento e nível de dados.
- … distribuir o processamento é equivalente a organizar a aplicação cliente/servidor em arquitetura muilticamada – o que é conhecido por **distribuição vertical**.
- … quando o cliente ou o servidor são divididos em partes logicamente equivalentes e cada parte opera no seu próprio conjunto de dados, a **distribuição é horizontal**.
- … neste contexto, destacamos as arquiteturas modernas de sistemas, também conhecidas por “**peer-to-peer systems**”.

... 2.2.2 – Arquiteturas Descentralizadas

- Sistemas Peer-to-Peer podem ser classificados em:
  - Arquitetuta P2P Estruturada - nós organizados seguindo uma estrutura de distribuição de dados específica, ou seja, procedimento determinístico;
  - Arquitetura P2P não Estruturada – nós tem vizinhos escolhidos de forma aleatoria, ou seja, algoritmos randômicos são utilizados.

- Observação: … virtualmente em todos os casos estamos lidando com "**overlay networks**", ou seja, redes em que os nós são formados por processos e os "links" representam os possíveis canais de comunicação (e.g. conexões TCP).

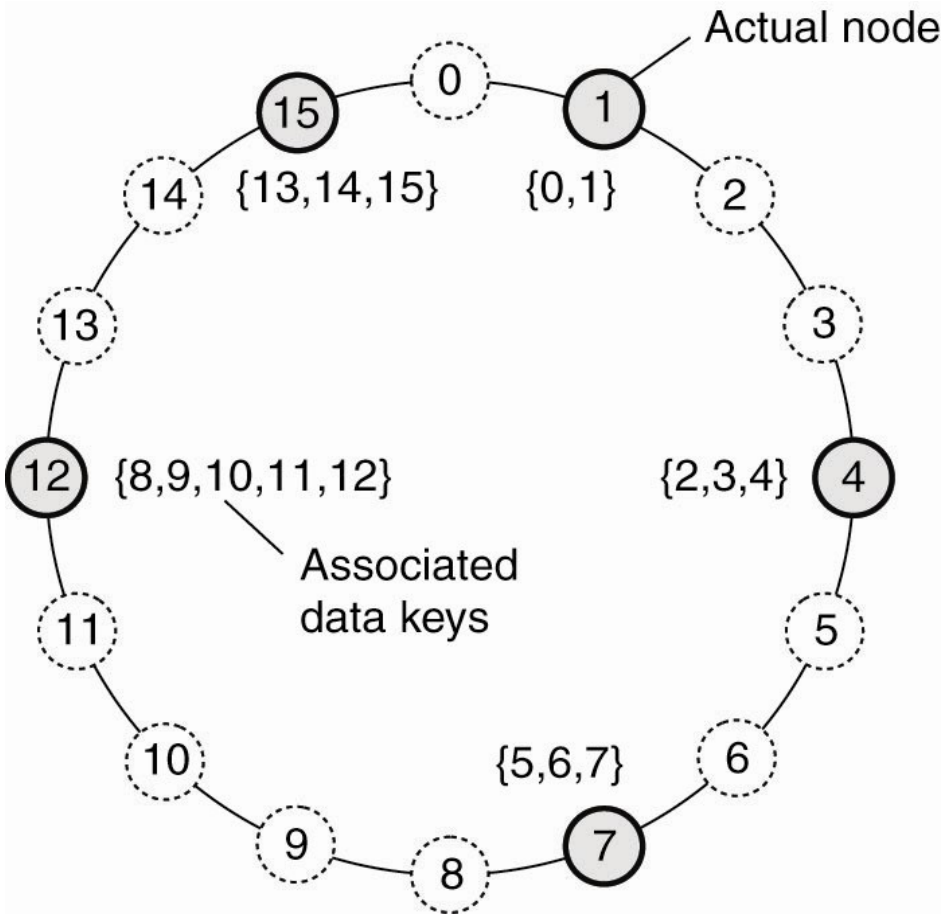
Arquitetura Peer-to-Peer Estruturada

- Idéia Básica – nós são organizados em uma rede de sobreposição estruturada como um anel lógico permitindo a distribuição de serviços nos nós bem como a sua identificação.
- "**DHT Systems**" – itens de dados são identificados por uma chave aleatória de um espaço amplo, tal como 128 ou 160 bits;
- … adicionalmente, nós dos sistema são identificados por nros. aleatórios dentro do mesmo espaço de identificadores;

- Problema em Sistema DHT – implementar um esquema eficiente e determinístico para mapear a chave do item de dado ao identificador do nó baseado em alguma métrica de distância.

… Arquitetura Peer-to-Peer Estruturada

- … sistema provê operação de “lookup(key)”;
- … requisições são roteadas para o nó associado.



… Arquitetura Peer-to-Peer Estruturada

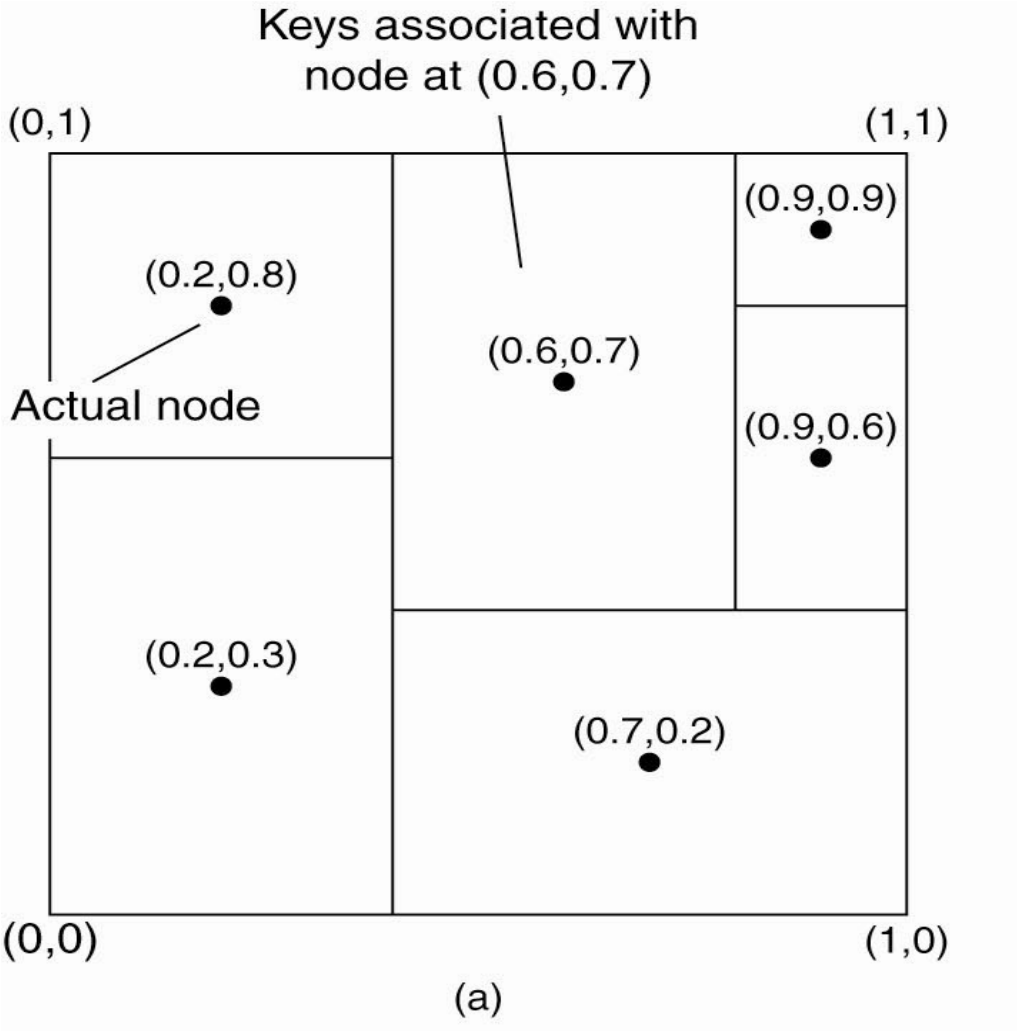
- **Chord System** – nós são organizados como um anel de modo que o item de dado com chave “k” esteja mapeado para o nó cujo menor identificador “id” seja maior ou igual a k (>= k).
- … quando um nó deseja juntar-se ao sistema, ele inicia o processo gerando um identificador aleatório “id”;
- … considerando que o espaço de identificadores é grande o bastante, a probabilidade de gerar um identificador já atribuído a algum nó atual é próximo de zero.
- nó referenciado como sucessor da chave “k” e denotado por “succ(k)” e, pode, ser obtido através da função “lookup(k)”;
- … naturalmente que este esquema possibilita a cada nó o armazenamento de seus predecessores.

… Arquitetura Peer-to-Peer Estruturada

- Outras abordagens também são baseados no DHT - “**Content Addressable Network** – CAN”
- “**Content Addressable Network**” - dispõe os nós em um espaço de dimensão “d” de tal forma que cada nó seja responsável por dados em uma região específica;
- … todo item de dado na rede endereçada por conteúdo é identificado por um único ponto no espaço bem como pelo nó responsável por aquele dado;
- … quando um nó é adicionado => região é dividida.

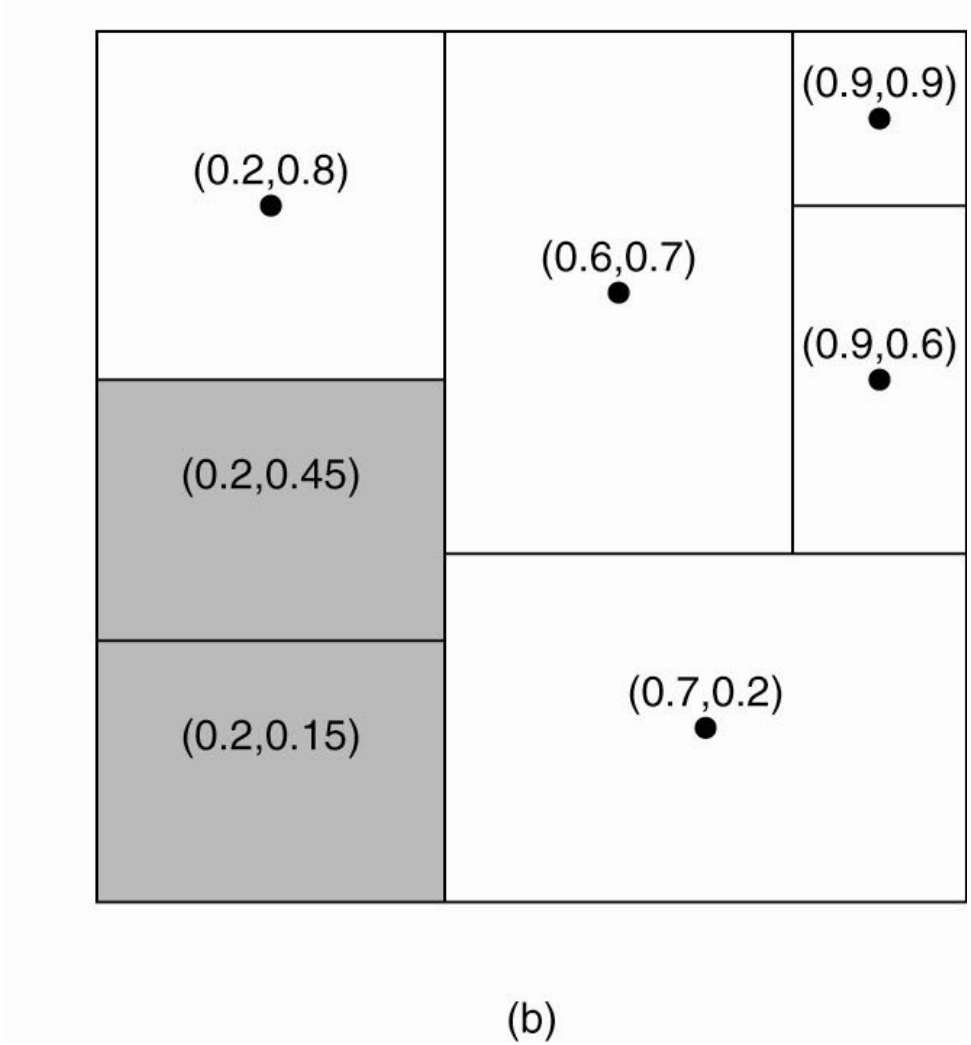
… Arquitetura Peer-to-Peer Estruturada

- … mapeamento de itens de dados em nós em uma rede.



… Arquitetura Peer-to-Peer Estruturada

- … divisão de região quando da adesão de um nó.



Arquitetura Peer-to-Peer Não Estruturada

- Arquitetura P2P Não Estruturada – cada nó mantém uma lista de “c” vizinhos, onde, idealmente cada um dos vizinhos representa uma escolha aleatória do conjunto de nós ativos;
- … se assumirmos que cada nó troca regularmente entradas da sua visão parcial, cada entrada identifica um outro nó na rede bem como quão velha/antiga é a referência para aquele nó;
- … assim há a necessidade de ao menos 02 “threads” - como detalhado nos 02 algoritmos a seguir:
  - ações para “thread” ativa;
  - ações para “thread” passiva.

… Arquitetura Peer-to-Peer Não Estruturada

Actions by active thread (periodically repeated):

```
select a peer P from the current partial view;
if PUSH_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
} else {
    send trigger to P;
}
if PULL_MODE {
    receive P's buffer;
}
construct a new partial view from the current one and P's buffer;
increment the age of every entry in the new partial view;
```

(a)

… Arquitetura Peer-to-Peer Não Estruturada

Actions by passive thread:

```
receive buffer from any process Q;
if PULL_MODE {
    mybuffer = [(MyAddress, 0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
}
construct a new partial view from the current one and P's buffer;
increment the age of every entry in the new partial view;
```

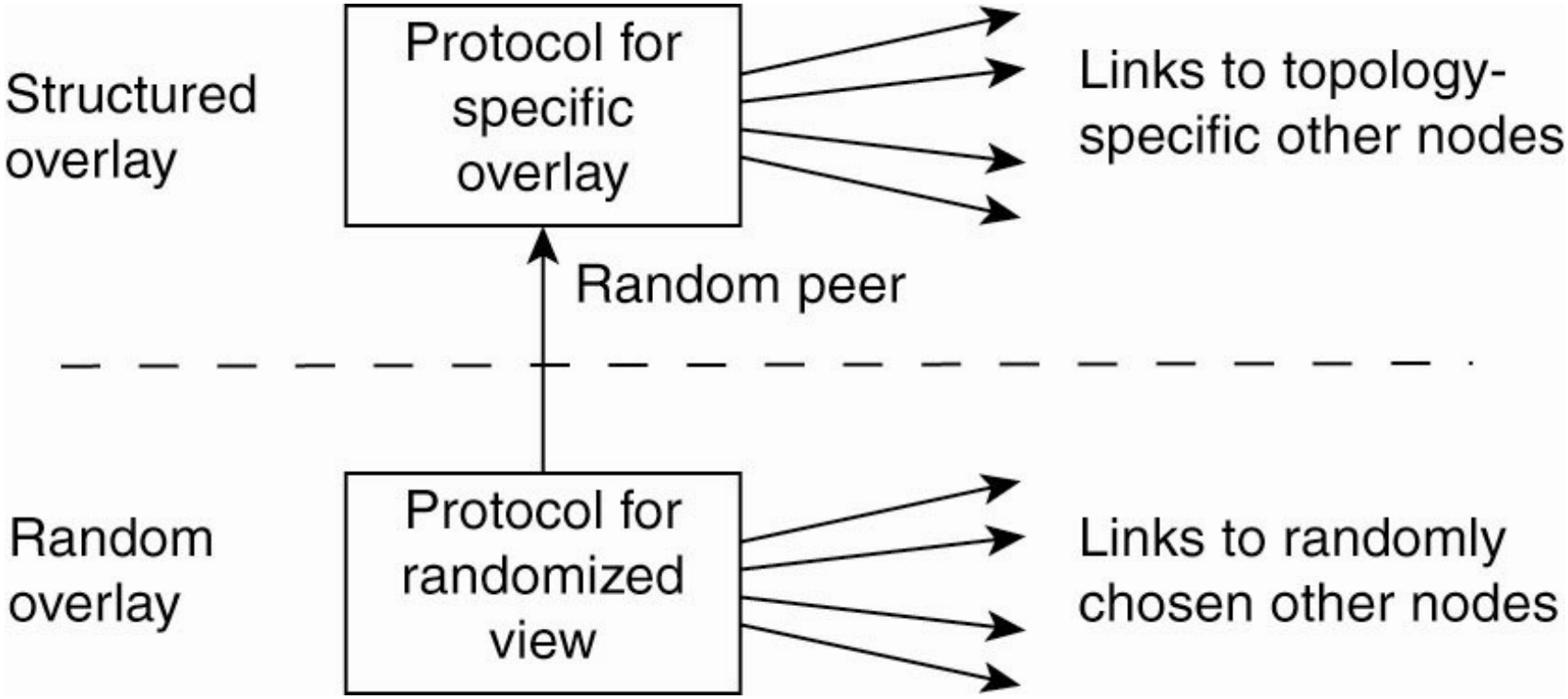
(b)

… Arquitetura Peer-to-Peer Não Estruturada

- análise – assuma que um nó que deseja se juntar a outros nós, entre em contato com algum nó arbitrário de um lista de pontos de acesso bem conhecidos;
- … considere o ponto de acesso como um membro regular da rede de sobreposição, a menos que seja altamente disponível;
- … neste caso, pode ser mostrado que protocolos que utilizam apenas o modo “push” ou “pull” podem facilmente conduzir para a formação de redes de sobreposição disconectadas;
- conclusão – grupos de nós tornar-se-ão isolados e não conseguirão alcançar um outro nó na rede, o que é uma característica indesejável.

Topologia de Gerenciamento em Redes Overlay

- Embora possa parecer que Sistemas Peer-to-Peer Estruturados e Não Estruturados formem classes independentes, não é o caso;
- ... pois com a seleção ou troca cuidadosa de visões parciais é possível construir e manter uma topologia específica.

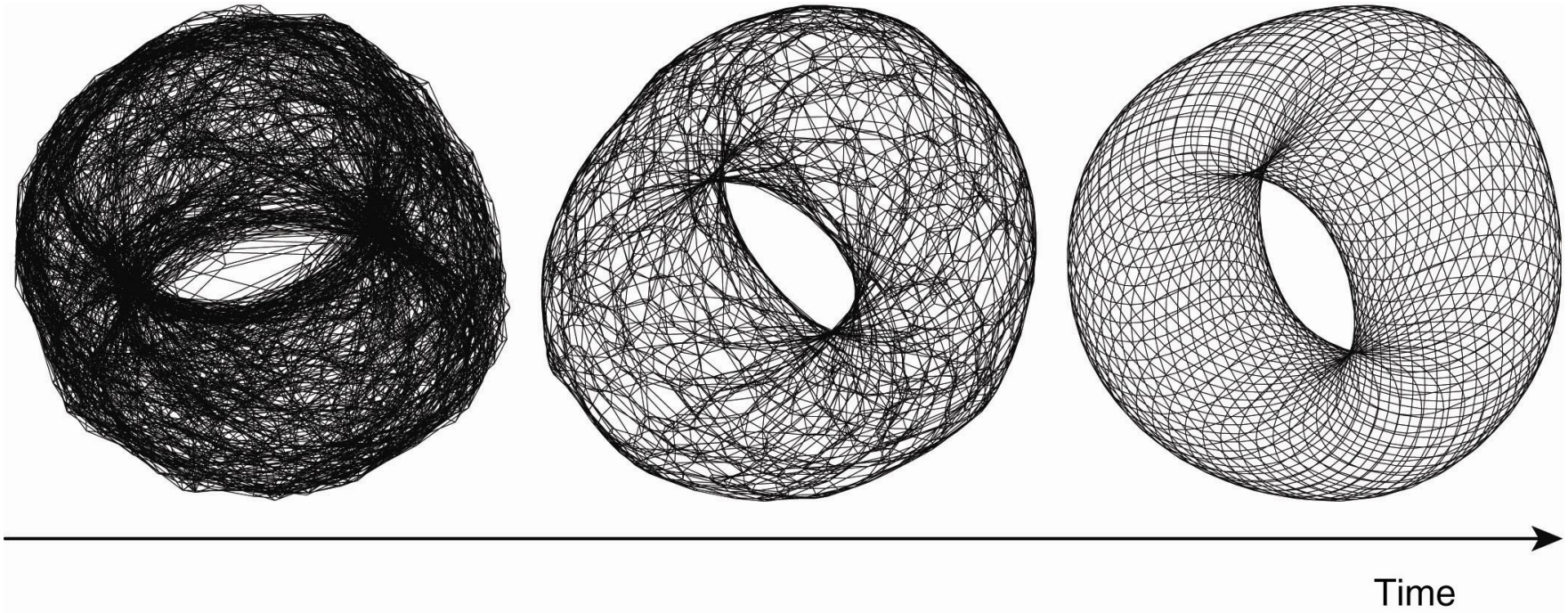


… Topologia de Gerenciamento em Redes Overlay

- Jelasity & Bagaoglu (2005): propuseram uma função de ranqueamento através da qual nós são ordenados de acordo com algum critério relativo de um dado nó;
- … e.g. considere um grade “N x N” com um nó em cada ponto da grade, onde cada nó mantém uma lista “c” de vizinho próximos;
- … distância entre um nó em (a_1, a_2) e (b_1, b_2) é definida por $d_1 + d_2$ onde $d_i = \min(N - |a_i – b_i|, |a_i – b_i|)$;
- … se a camada inferior executar periodicamente descrito anteriormente (“push mode” e “pull mode”), a topologia evolui para um toróide.

… Topologia de Gerenciamento em Redes Overlay

- …rede de sobreposição específica usando 02 camadas de sistemas peer-to-peer não estruturados.

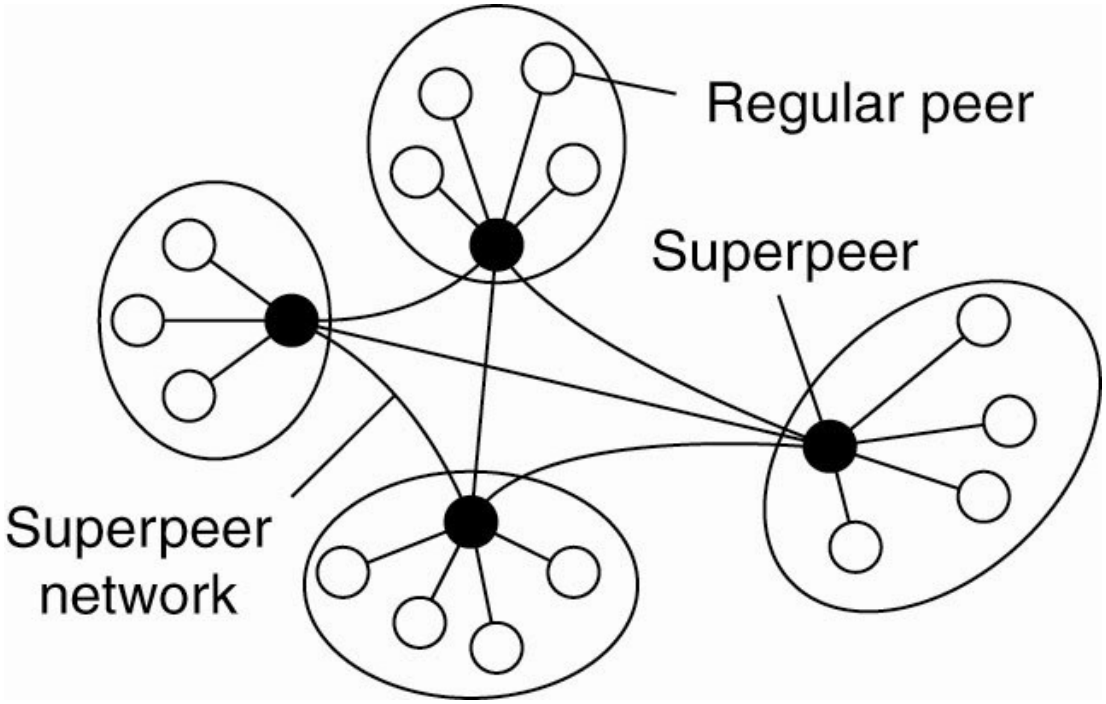


... Topologia de Gerenciamento em Redes Overlay

- É persceptível que a localização de itens de dados pode se tornar um problema a medida que as redes não estruturadas crescem;
- … pois não há um meio determinístico de pesquisa, o que por outro lado pressupõe que a técnica a ser utilizada é a técnica de inundação de requisições.
- Há várias maneiras nas quais a Inundação de requisições pode ser repressada, mas uma alternativa é a utilização de nós especiais que mantém um índice de itens de dados.
- … tais estruturas que mantém um índice ou agem como um interceptador são geralmente referenciadas como “superpeers”.

... Topologia de Gerenciamento em Redes Overlay

- …organização hierárquica de nós dentro da rede de super nós, onde um nó regular está conectado como cliente a um super nó.
- … em muitos casos, a relação cliente – super nó é fixa, ou seja, quando um nó regular se junta a rede, ele se conectar a um super nó e permanece conectado até deixar a rede.



... Topologia de Gerenciamento em Redes Overlay

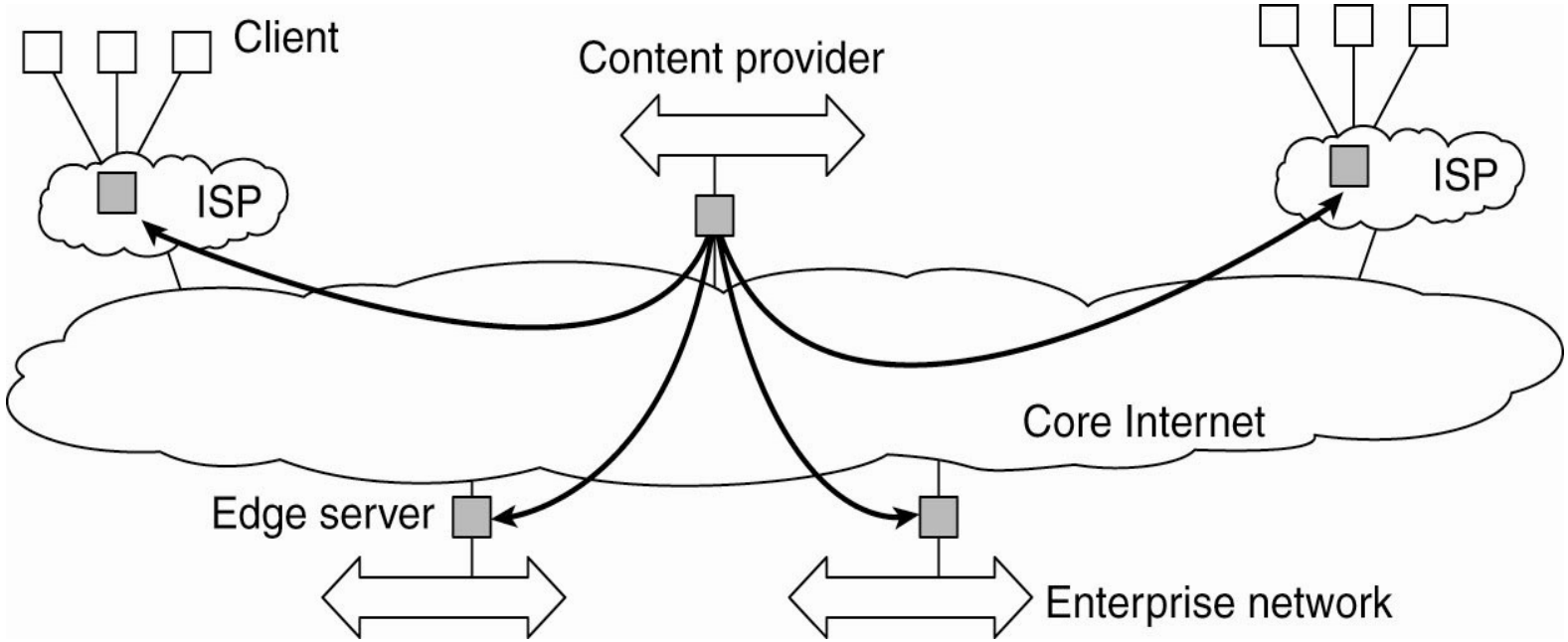
- Como já discutido, redes ponto a ponto oferecem meios flexíveis para nós se juntarem ou saírem da rede.
- … entretanto, com redes de super nós um novo problema aparece, ou seja, como selecionar nós elegíveis para se tornarem super nós ?
- … este problema está fortemente relacionado com o problema de eleição de lider a ser discutido no Cap. 06.

2.2.3 – Arquiteturas Híbridas

- Muitos Sistemas Distribuídos combinam aspectos arquiteturais e são assim referenciados como Sistemas de Arquiteturas Híbridas;
- … uma importante classe de sistemas distribuídos organizados de acordo com uma arquitetura híbrida é formada pelos sistemas de servidores de borda ou “edge-server systems”.
- “**Edge-Server Systems**” - sistemas são implantados na rede onde servidores estão localizados na borda da rede.
  - … esta borda é formada pelo limite da rede corporativa e a Rede Internet (e.g.) como previsto pelo ISP (Internet Service Provider);
  - … também quando usuários finais se conectam a Internet através do seu ISP, ou seja, o ISP pode ser considerado como residente de borda da rede.

… 2.2.3 – Arquiteturas Híbridas

- "**Edge-Server Systems**" - sistemas são implantados na rede onde servidores estão localizados na borda da rede.
  - … esta borda é formada pelo limite da rede corporativa e a Rede Internet (e.g.) como previsto pelo ISP (Internet Service Provider).



… 2.2.3 – Arquiteturas Híbridas

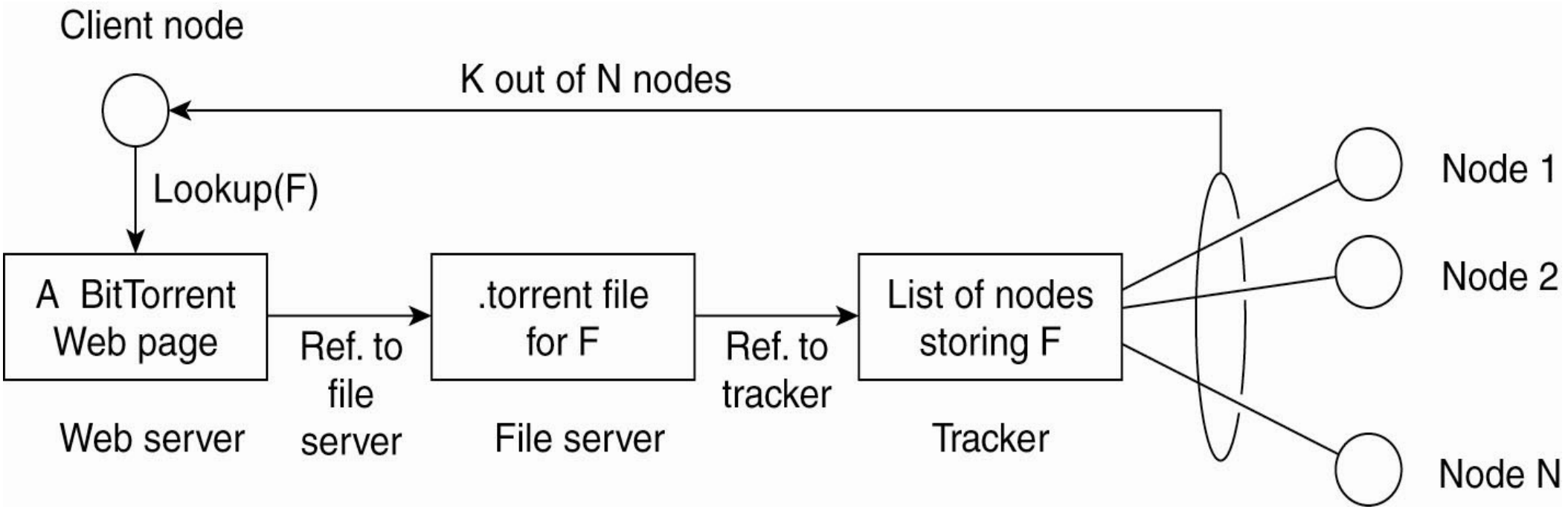
- Obs. – coleção de servidores de borda podem ser usados para otimizar distribuição de aplicações e conteúdo;
- … ou seja, o modelo básico é o de que para um organização específica, cada servidor de borda se comporte como um servidor origem do qual todo o conteúdo se origina.
- e.g. … como exemplo de servidores de borda, enquadramos as soluções distribuídos baseadas na Web

... 2.2.3 – Arquiteturas Híbridas

- "**Collaborative Distributed Systems**" - uma vez que o nó tenha se juntado ao sistema, ele pode utilizar um esquema totalmente descentralizado para colaboração.
- e.g., para maior perscepção, considere o BitTorrent – sistema de "download" de arquivo ponto-a-ponto que combina soluções centralizadas e descentralizadas;
- **idéia básica** – quando um usuário busca por um arquivo, ele descarrega pedaços - "chunks" do arquivo de um outro usuário até que os pedaços descarregados - "chunks downloaded" possam ser juntados para produzir o arquivo original;
- … um importante aspecto de projeto presente nesta proposta é a garantia de colaboração durante todo o processo.

... 2.2.3 – Arquiteturas Híbridas

- BitTorrent – sistema de “download” de arquivo ponto-a-ponto que combina soluções centralizadas e descentralizadas.



... 2.2.3 – Arquiteturas Híbridas

- … para efetuar o “download” de um arquivo, usuário acessa um diretório global, que é um dos “web sites” bem conhecidos;
- … este diretório contém referências para o que são denominados arquivos “.torrent” - que contém informações necessários para o “download” do referido arquivo – arquivo específico;
- … “.torrent” referencia o que é conhecido por “tracker”, ou seja, um servidor que mantém precisamente quais nós ativos contém pedaços - “chunks” do arquivo requisitado;
- … um nó ativo é por exemplo que está correntemente efetuando o “downloading” um outro arquivo.
- Obs. - geralmente são diferentes “trackers”, embora tenha-se um “tracker” por arquivo ou por uma coleção de arquivos.

... 2.2.3 – Arquiteturas Híbridas

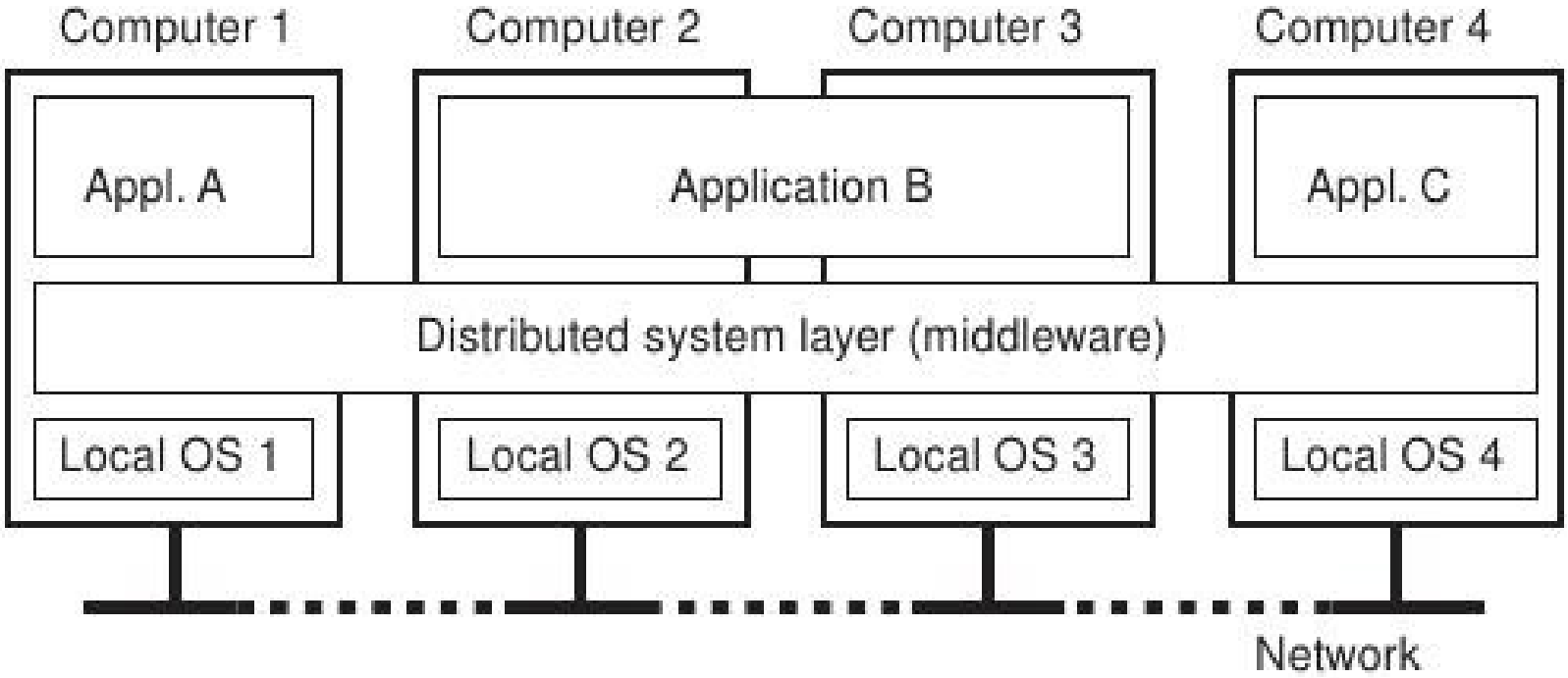
- … um vez identificado de onde os “chunks” podem ser descarregados, o nó que solicitou o “download” é efetivamente ativado;
- … neste ponto ele será forçado a ajudar outros, p.ex., oferecendo “chunks” do arquivo que já tenham sido descarregados para outros usuários que ainda não tenham estes pedaços.
- … este comportamento vem de uma regra muito simples: se nó P percebe que o nó Q está recebendo mais do que está fornecendo, P pode decidir decrementar a taxa na qual envia dados para Q;
- … este esquema funciona bem desde que P tenha algo para descarregar do nó Q.
- Por esta razão, nós são referenciados por muitos outros nós, permitindo que tenham condições para negociar dados.

... 2.2.3 – Arquiteturas Híbridas

- "**Globule**" - rede de distribuição de conteúdo colaborativa na qual usuários ou organizações voluntariamente disponibilizam servidores "web" melhorados que são capazes de colaborar para a replicação de páginas "web";
- … na forma mais simples um servidor contempla:
  - componente para redirecionar requisições de cliente p/ outros servidores;
  - componente para analisar padrões de acesso;
  - componente para gerenciar a replicação de páginas Web.
- … mantém um componente centralizado ("broker" - corretor), responsável por registrar os servidores e fazer com que os mesmos conheçam uns aos outros.

2.3 – Arquiteturas vs Middleware

- **middleware** – “software” que oferece serviços para as aplicações tendo por base serviços disponíveis no sistema operacional;
- … middleware não é parte do sistema operacional, ou do gerenciamento do sistema ou nem mesmo das aplicações.



… 2.3 – Arquiteturas vs Middleware

- … desenvolver “middleware” segundo uma padrão arquitetural traz o benefício de tornar as aplicações distribuídas simples, mas por outro lado, o desempenho pode ser comprometido em relação ao que o desenvolvedor tinha em mente.
- e.g. … inicialmente na Especificação CORBA (Common Object Request Broker Architecture), objetos somente podiam ser invocados por clientes remotos;
- ... mais tarde, após identificar que uma única abordagem de interação não era suficiente, permitiu-se a inclusão de outros padrões de interação, como troca de messagens.

… 2.3 – Arquiteturas vs Middleware

- … embora o “middleware” seja o meio para prover transparência de distribuição, percebe-se que soluções específicas devem ser adaptáveis aos requisitos da aplicação;
- **solução** – disponibilizar várias versões do “middleware” de modo que cada versão seja adaptada para uma classe específica.
- **alternativa** – uma abordagem mais interessante é permitir que o “middleware” seja fácil de ser configurado, adaptado e personalizado como exigido por uma aplicação.
- **conclusão** – sistemas vem sendo desenvolvidos com separação mais rigorosa entre políticas e mecanismos que possibilitem que o comportamento do “middleware” seja modificado.

2.3.1 – Interceptadores

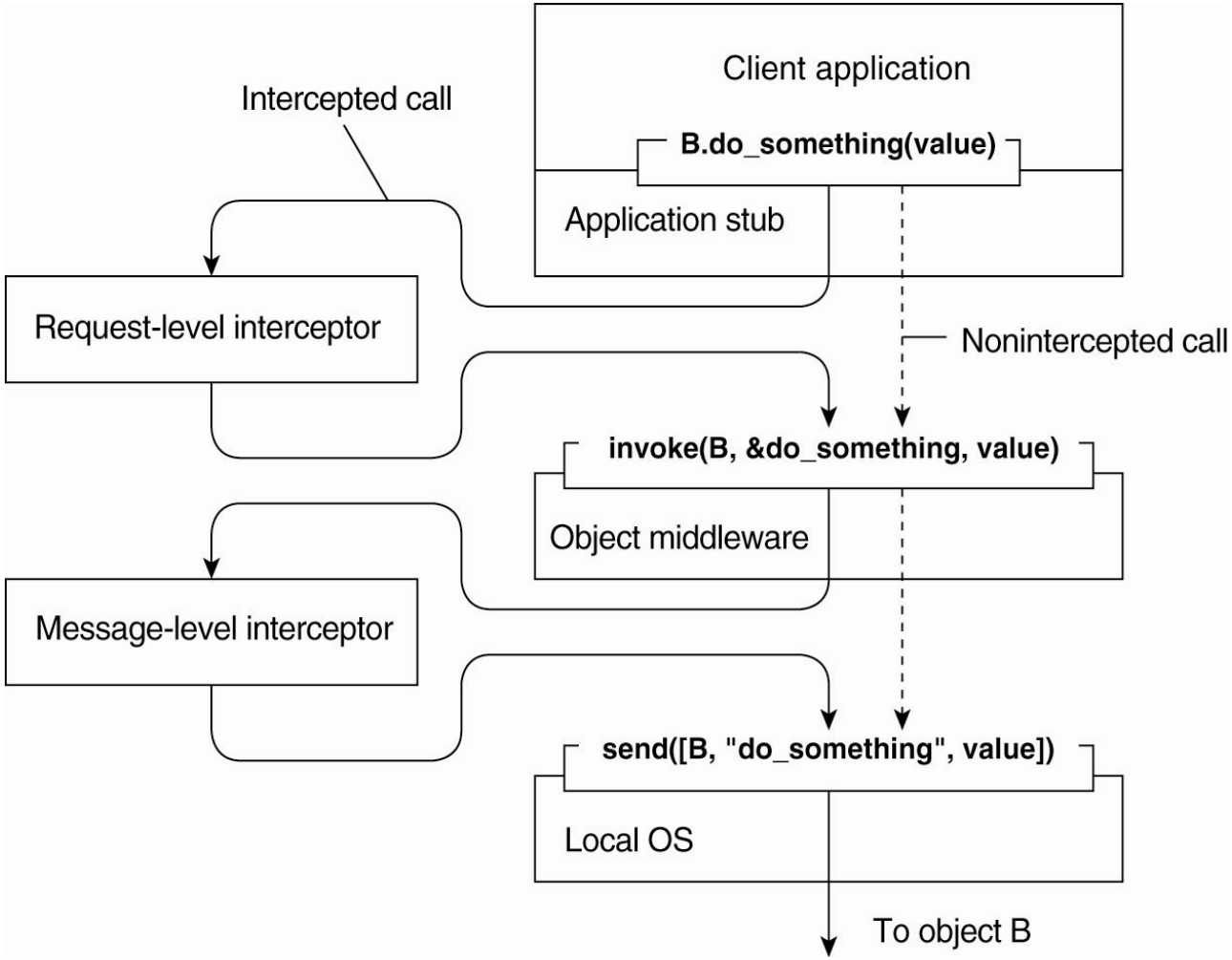
- “interceptor” - construção em software que intercepta um fluxo usual de controle e aciona um outro código para ser executado, normalmente uma aplicação específica.
- … decisão de implementar um interceptor – que exige esforço substancial – não é clara quando comparada com simplicidade e aplicabilidade restrita de uma versão de “middleware”.
- … em muito casos, ter funcionalidade de interceptação ainda que limitadas irá melhorar o gerenciamento do software bem como do sistema distribuído como um todo.
- Para tornar o problema mais concreto, considere a invocação de um método do objeto “B” por um objeto “A”.

2.3.1 – Interceptadores

- e.g., objeto “A” pode invocar um método do objeto “B”, ainda que “B” esteja em um outra máquina, ou seja, a invocação de objeto remoto se dá em 03 passos:
  - objeto “A” disponibiliza uma interface local que é exatamente a mesma que a interface disponibilizada pelo objeto “B”, assim “A” simplesmente invoca o método disponível naquela interface;
  - chamada de “A” é transformada em um invocação genérica de objeto, possivelmente através de uma interface de invocação de objeto genérica oferecida pelo “middleware” na máquina onde “A” reside;
  - finalmente, a invocação genérica de objeto é transformada em uma mensagem que é enviada pela interface da camada de transporte do sistema operacional de rede da máquina onde “A” reside.

… 2.3.1 – Interceptadores

- "**interceptor**" - construção em software que intercepta um fluxo usual de controle e aciona um outro código para ser executado.



… 2.3.1 – Interceptadores

- … após o primeiro passo, a chamada “B.do_something(value)” é transformada em uma chamada genérica tal como “invoke(B, &do_something, value)” com a referência do método B bem como os parâmetros da chamada do método;
- … considere o cenário que o objeto “B” tenha sido replicado, neste caso cada réplica deveria ser invocada.
- … este é um ponto onde o interceptador pode ajuda !!
- “Intercptor” irá chamar “invoke(B, &do_something, value)” para cada uma das réplicas sem que “A” saiba da existência das réplicas de “B” ou mesmo tenha que gerenciá-las;
- … somente o “interceptor” no nível de requisição que foi adicionado ao “middleware” necessita saber de “B” e “ B' ”.

… 2.3.1 – Interceptadores

- … ao sair do “middleware” a chamada para o objeto remoto deve ser enviada para a rede, ou seja, interface de mensagem do sistema operacional de rede local deve ser invocada;
- … neste nível, o “interceptor” no nível de mensagem pode auxiliar a transferência de invocação para o objeto alvo.
- e.g., parâmetro “value” corresponde a um array grande de dados, assim, é mais inteligente fragmentá-lo em partes menores e remontá-las no destino.
- … middleware não está a par desta fragmentação, mas o “interceptor” no nível de mensagem se encarrega de forma transparente tratar a parte final da comunicação com o SO.

2.3.2 – Software Adaptativo

- “interceptor” - constituem-se no meio para adaptar o “middleware”.
- ... esta necessidade vem do fato de que o ambiente no qual aplicações distribuídas são executadas mudam continuamente;
- ... assim, muitos projetistas de “middleware” passaram a considerar a construção de “software adaptativos”.
- McKinley et al. (2004) propõe três técnicas básicas para adaptação de software:
  - separação de responsabilidades;
  - computação reflexiva;
  - projeto baseado em componentes.

... 2.3.2 – Software Adaptativo

- ... separação de responsabilidades – está relacionado ao modo tradicional de modularização de sistemas;
  - ... ou seja, separa as partes que implementam funcionalidades de outras partes que são especializadas em outras funcionalidades tais como: confiabilidade; desempenho, segurança, etc.
- computação refletiva – habilidade de um programa inspecionar a si mesmo e, se necessário, adaptar-se;
  - … normalmente contemplada na linguagem de programação.
- projeto baseado em componentes – oferece suporte à adaptação através da composição, permitindo que um sistema seja configurado estaticamente emtempo de concepção do projeto ou dinamicamente em tempo de execução.

2.3.3 – Discussão

- … Arquitetura de Software para Sistemas Distribuídos, também referenciado com “middleware” são volumosas e complexas.
- ... não são raros os conflitos entre requisitos funcionais das aplicações distribuídas e do “middleware” sobre o qual estas aplicações repousam, e.g., transparência;
- … tais requisitos conflitantes em função da generalidade ou especialização como resultado tem resultado em soluções de “middlewares” altamente flexíveis.
- … preço no entanto é a complexidade da solução.

… 2.3.3 – Discussão

- Alguns argumentam que é necessário focar na simplicidade, um modo mais simples de construir “middleware” por componentes e independência da aplicação;
- … nenhuma das técnicas propostas tem recebido massivamente adeptos e nem mesmo tem sido aplicadas com sucesso em sistemas de larga escala.
- ... o mais forte e, certamente válido, argumento para suportar certas propriedades como p.ex. adaptação de software é o fato de que muitos sistemas distribuídos não podem ser desligados !

… 2.3.3 – Discussão

- Alguns argumentam que é necessário focar na simplicidade, um modo mais simples de construir “middleware” por componentes e independência da aplicação;
- … nenhuma das técnicas propostas tem recebido massivamente adeptos e nem mesmo tem sido aplicadas com sucesso em sistemas de larga escala.
- suposição – necessidade de software adaptativo no sentido de que o software deve permitir mudanças em algumas situações como consequência das mudanças do ambiente;
- … e.g.; falha de hardware, ataque de segurança, drenagem de energia, ou seja, em casos de influências ambientais passíveis de serem antecipadas pelo software.

… 2.3.3 – Discussão

- O mais forte e, certamente válido, argumento para suportar certas propriedades como p.ex. adaptação de software é o fato de que muitos sistemas distribuídos não podem ser desligados !
- … esta restrição invoca soluções para substituir e atualizar componentes em operação, mas não deixa claro qual das soluções propostas são as melhores para combater o problema.
- O que permanece é que sistemas distribuídos devem ser capazes de reagir a mudanças no seus ambientes, p.ex., mudando a política de alocação de recursos;
- … todos os componentes de software para abilitar tal adaptação estarão em seus lugares bem como os algoritmos contidos nestes componentes e que ditam o seu comportamento.