Computação Distribuída

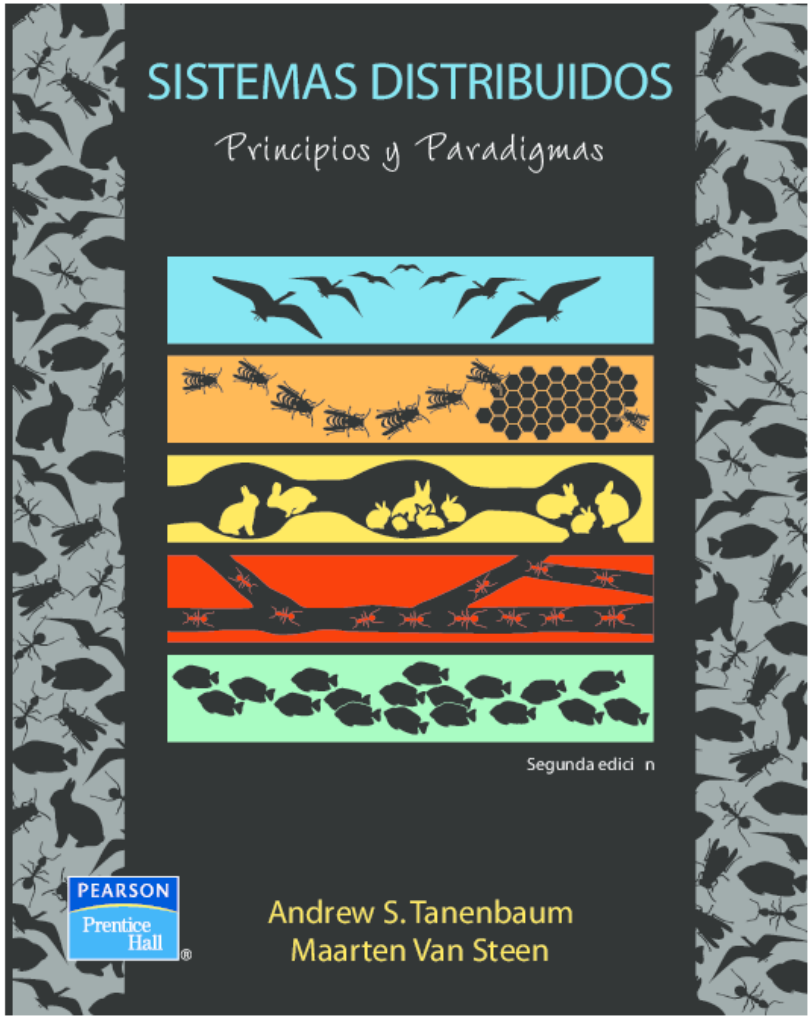
Arquiteturas de Sistemas Distribuídos

CHAPTER 2

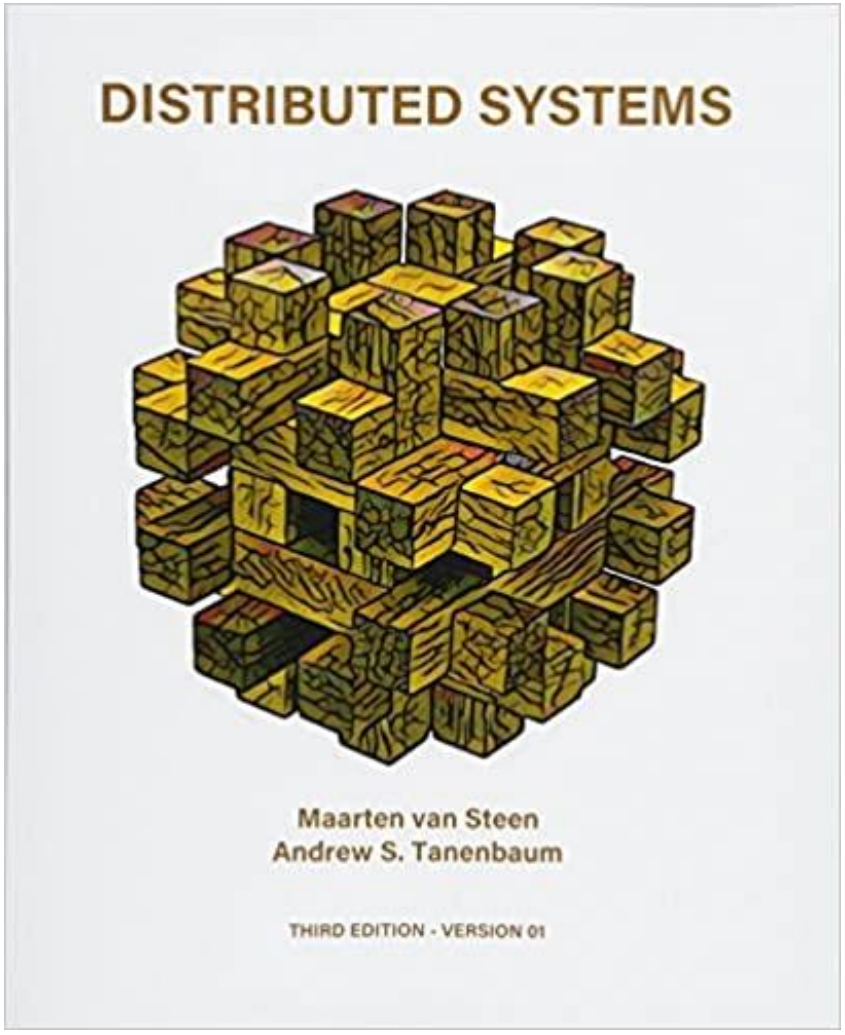
ARCHITECTURES

Vladimir Rocha (Vladi)

CMCC - Universidade Federal do ABC


SISTEMAS DISTRIBUIDOS
Principios y Paradigmas
Segunda edici n
PEARSON
Prentice
Hall
Andrew S. Tanenbaum
Maarten Van Steen

2007


DISTRIBUTED SYSTEMS
Maarten van Steen
Andrew S. Tanenbaum
THIRD EDITION - VERSION 01

2017

Disclaimer

- Estes slides foram baseados nos do professor **Emilio Francesquini** para o curso de Sistemas Distribuídos na UFABC.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.
- Estes slides foram adaptados daqueles originalmente preparados (e gentilmente cedidos) pelo professor **Daniel Cordeiro, da EACH-USP** que por sua vez foram baseados naqueles disponibilizados online pelos autores do livro "Distributed Systems", 3ª Edição em: https://www.distributed-systems.net.

Agenda

- Estilos arquiteturais
- Arquiteturas de software
- Arquiteturas versus middleware
- Sistemas distribuídos autogerenciáveis

Agenda

- Estilos arquiteturais
- Arquiteturas de software
- Arquiteturas versus middleware
- Sistemas distribuídos autogerenciáveis

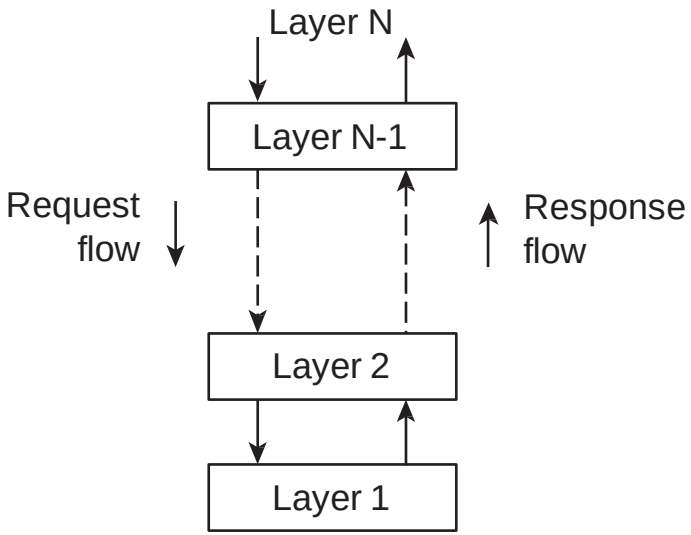
Estilos arquiteturais

1. Arquiteturas em camadas
2. Arquiteturas baseadas em objetos
3. Arquiteturas baseadas em eventos e dados
4. Arquiteturas centradas em recursos

1. Estilo arquitetural: camadas

Ideia básica

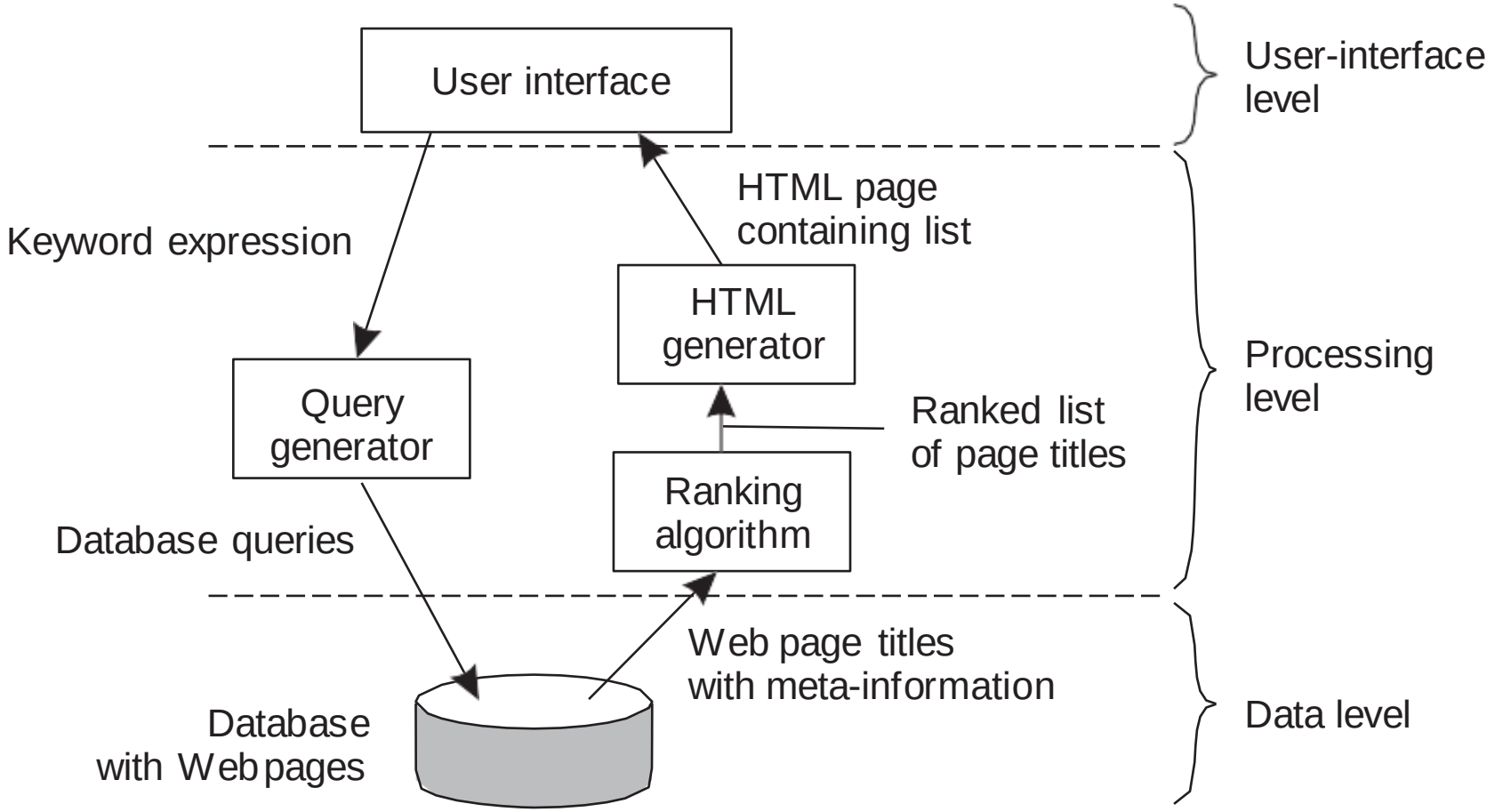
Organize em componentes logicamente diferentes e os distribua entre as máquinas disponíveis.



(a)

(a) Estilo em camadas é usado em sistemas cliente-servidor

Estilo arquitetural: exemplo de camadas

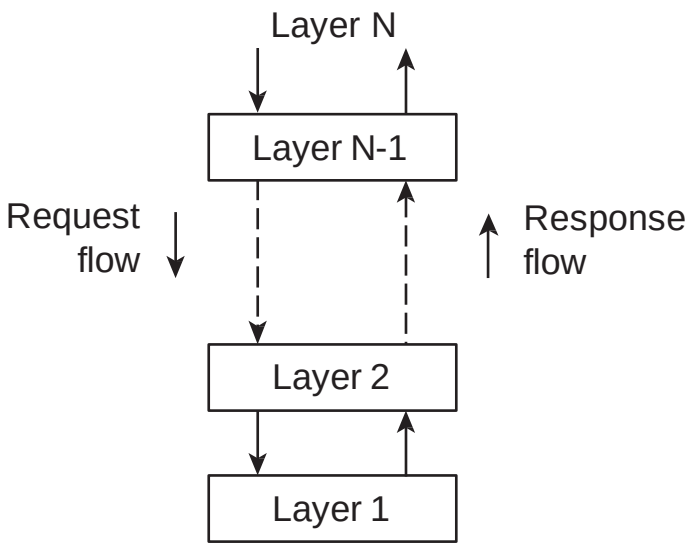


Several servers / layers

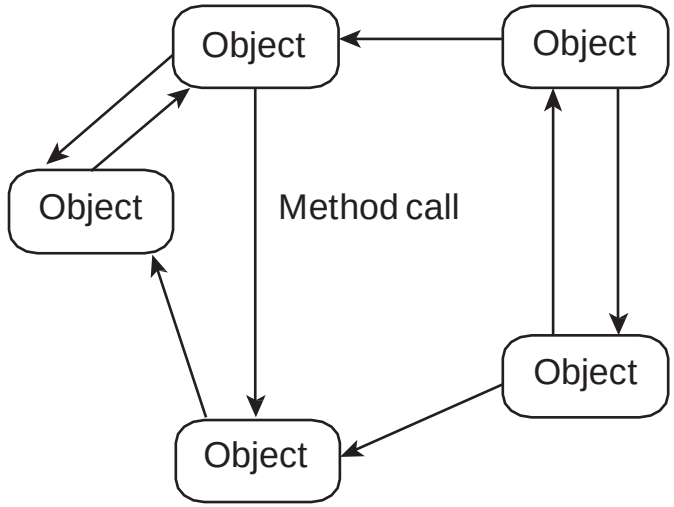
2. Estilo arquitetural: objetos

Ideia básica

Organize em componentes logicamente diferentes e os distribua entre as máquinas disponíveis.



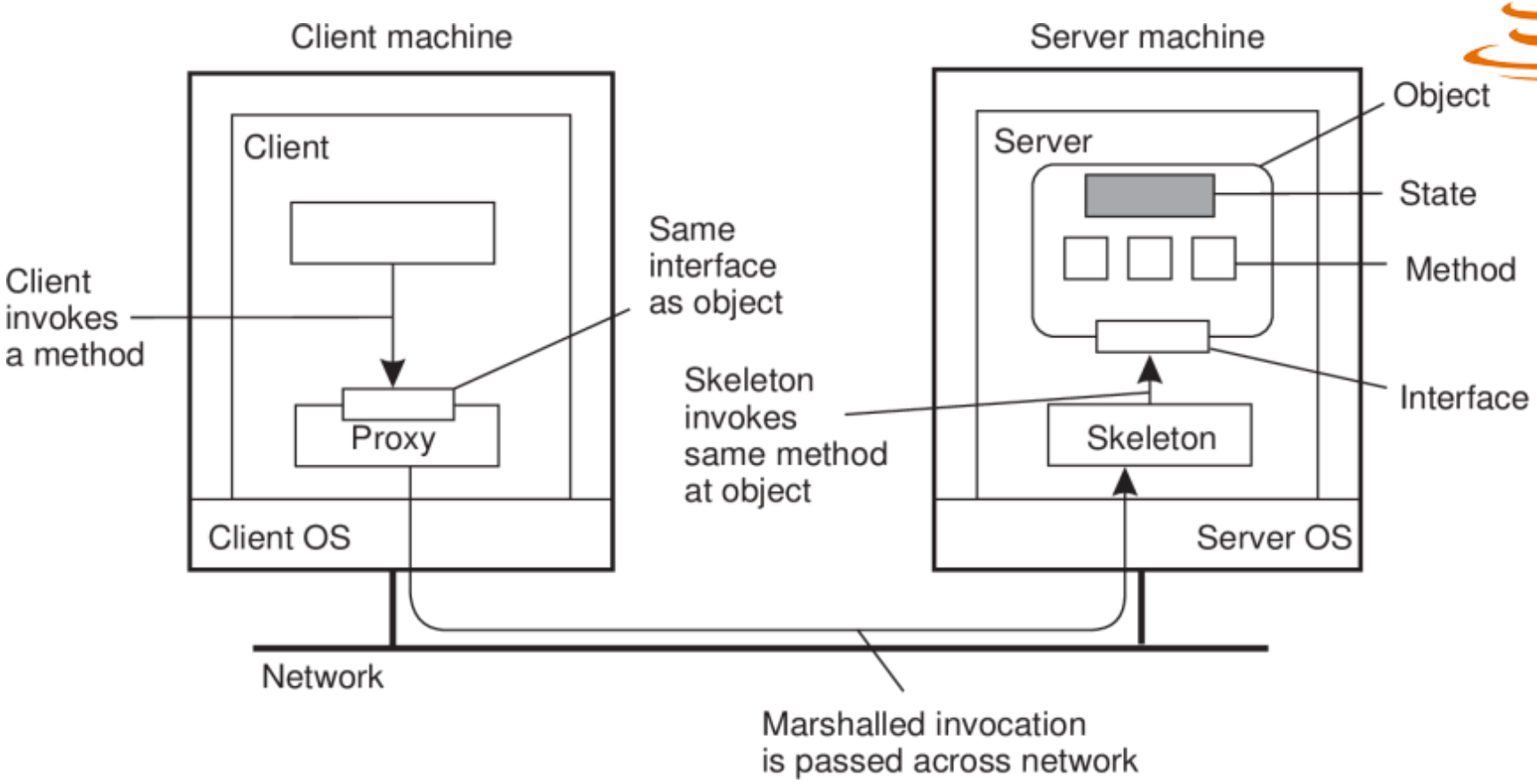
(a)



(b)

(a) Estilo em camadas é usado em sistemas cliente-servidor
(b) Estilo orientado a objetos usado em sistemas de objetos distribuídos.

Estilo arquitetural: exemplo de objetos distrib. (RMI)



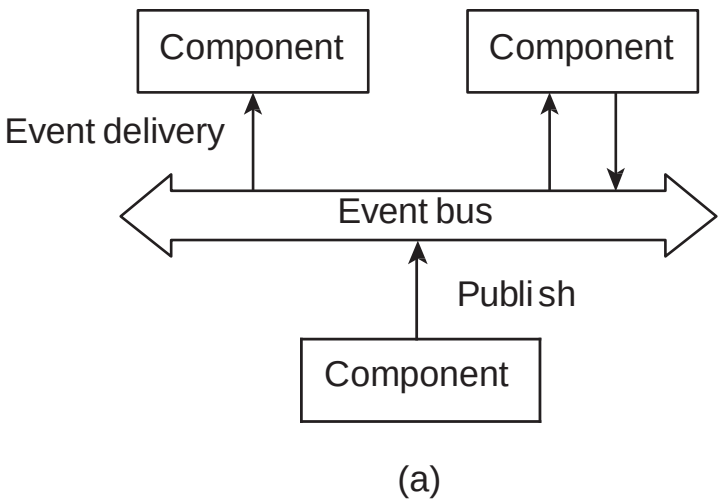
Encapsulamento

Os objetos ficam distribuídos pelo sistema (servidores). Apesar do usuário fazer chamadas que são equivalentes a chamadas locais, elas podem estar sendo feitas em objetos remotos.

3. Estilo arquitetural: eventos e dados

Ideia básica

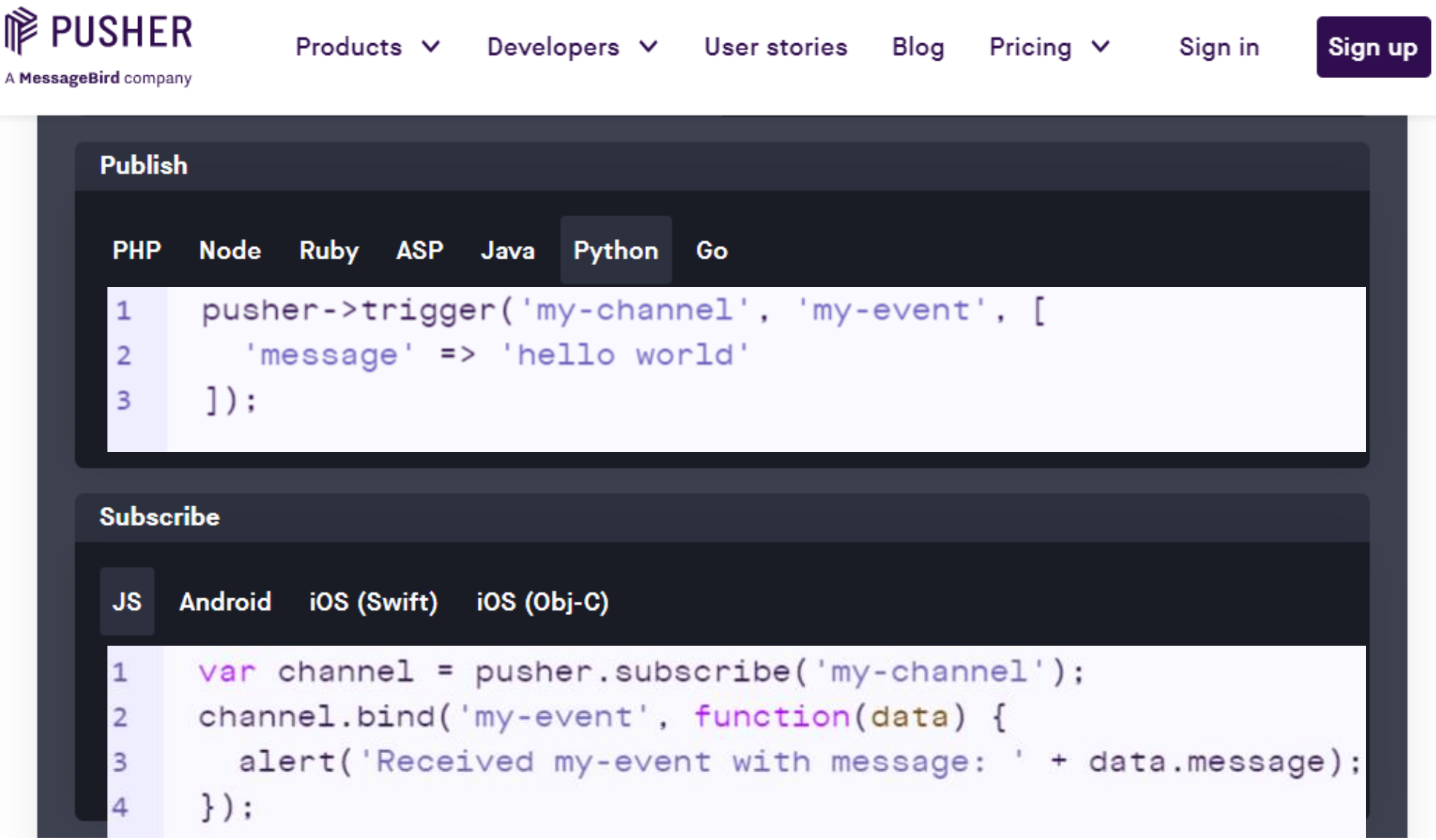
Desacoplar processos na referência (anônimos)



(a)

(a) Publish/subscribe [desacoplado na referência]

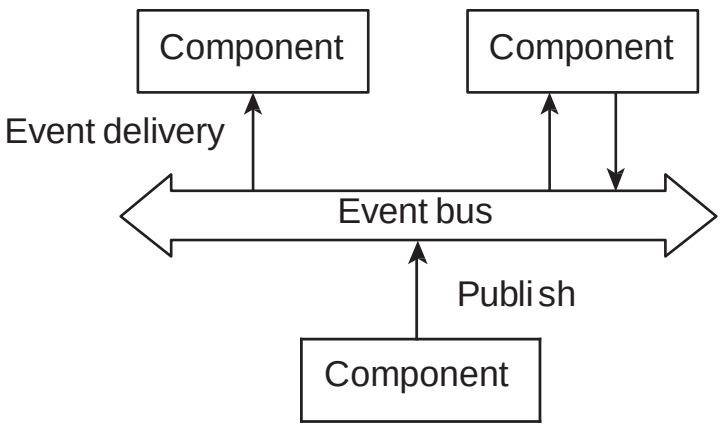
Estilo arquitetural: exemplo eventos (publish/subscribe)



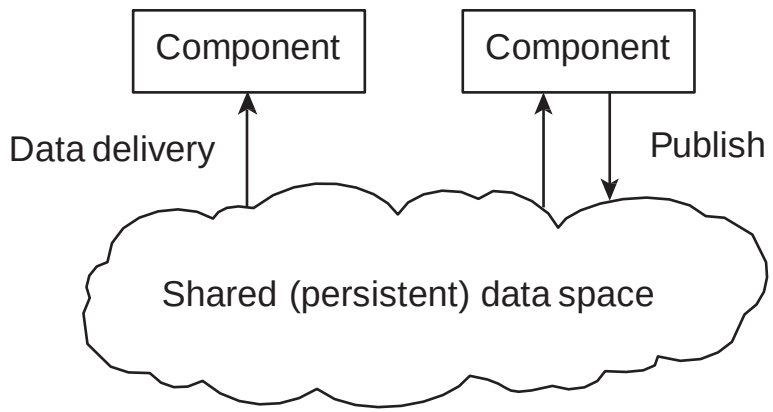
3. Estilo arquitetural: eventos e dados

Ideia básica

Desacoplar processos na referência (anônimos) e tempo (assíncronos)



(a)

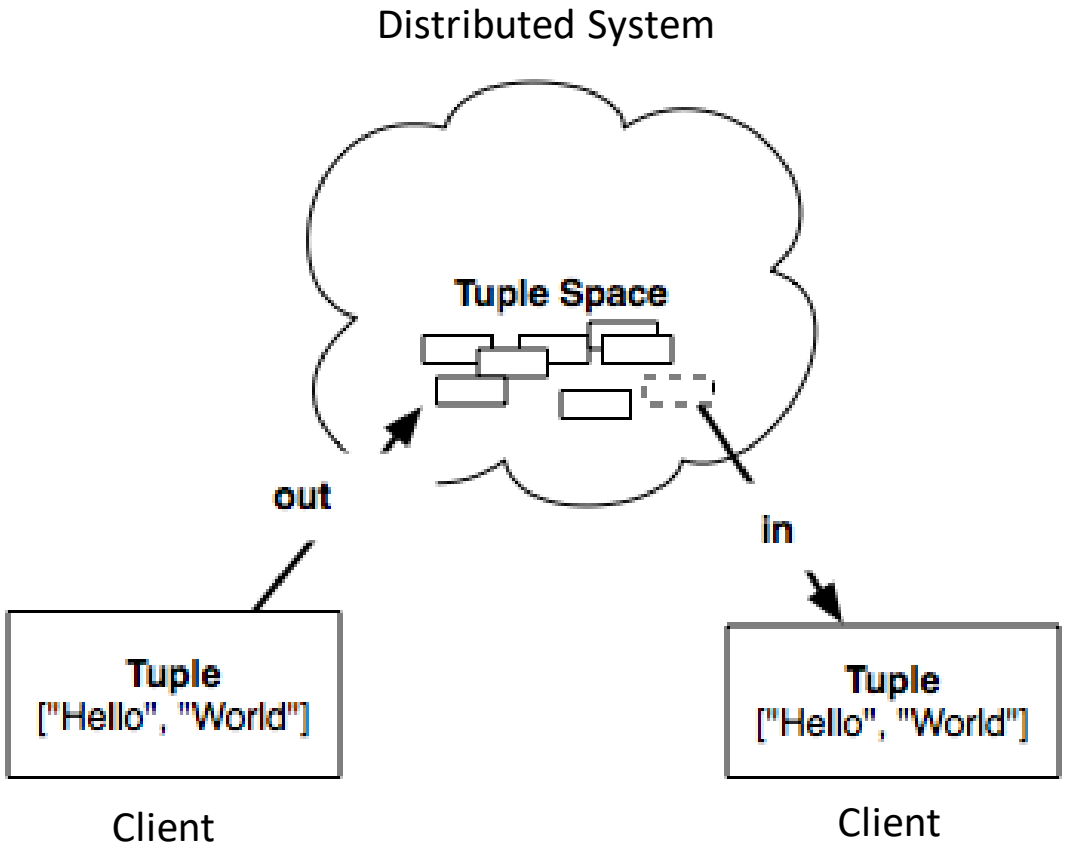


(b)

(a) Publish/subscribe [desacoplado na referência]
(b) Espaço de dados compartilhados [desacoplado na ref. e tempo]

Estilo arquitetural: exemplo de dados compartilhados

GIGASPACES
innovate with confidence



https://www.druby.org/sidruby/6-1-introducing-linda-and-rinda.html

4. Estilo arquitetural: recursos

Vê um sistema distribuído como uma coleção de recursos gerenciados individualmente por componentes. Recursos podem ser adicionados, removidos, recuperados e modificados por aplicações remotas - REST [Fielding 2000].

1. Recursos são identificados usando um único esquema de nomeação
2. Todos os serviços oferecem a mesma interface
3. Mensagens enviadas de ou para um serviço são auto-descritivas
4. Após a execução de uma operação em um serviço, o componente esquece tudo sobre quem chamou a operação

Operações básicas

Operação	Descrição
PUT	Cria um novo recurso
GET	Recupera o estado de um recurso usando um tipo de representação
DELETE	Apaga um recurso
POST	Modifica um recurso ao transferir um novo estado

Estilo arquitetural: exemplo de recursos

Essência

Objetos (arquivos) são armazenados em buckets (diretórios).
Operações em ObjectName em BucketName requerem o seguinte identificador:

http://BucketName.s3.amazonaws.com/ObjectName

Todas as operações são realizadas com requisições HTTP:

- Criar um bucket/objeto: PUT + URI
- Listar objetos: GET em um nome de bucket
- Ler um objeto: GET em uma URI completa

Agenda

- Estilos arquiteturais
- Arquiteturas de software
- Arquiteturas versus middleware
- Sistemas distribuídos autogerenciáveis

Arquiteturas

- Arquiteturas centralizadas e multicamadas (multidivididas)
- Arquiteturas decentralizadas
- Arquiteturas híbridas

Arquiteturas

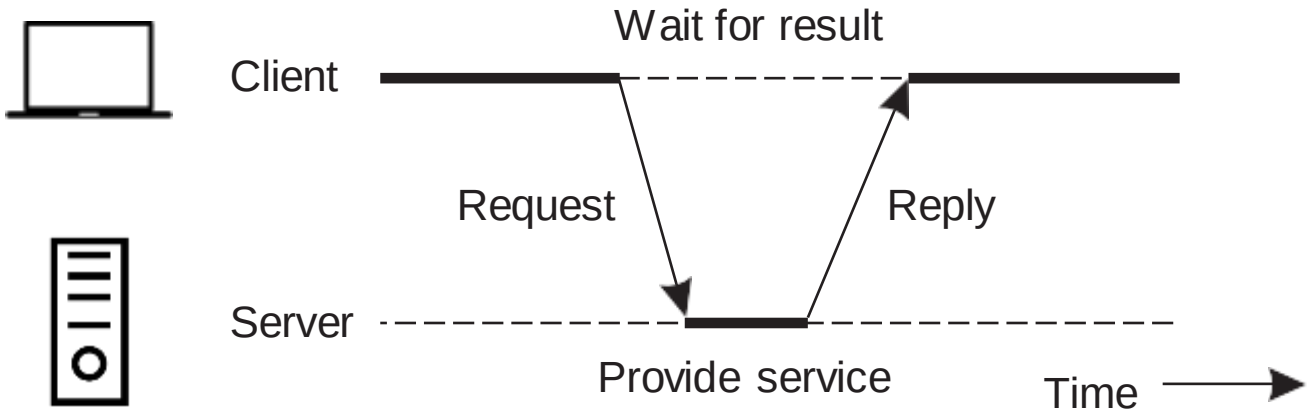
- Arquiteturas centralizadas e multicamadas (multidivididas)
- Arquiteturas decentralizadas
- Arquiteturas híbridas

Arquiteturas centralizadas



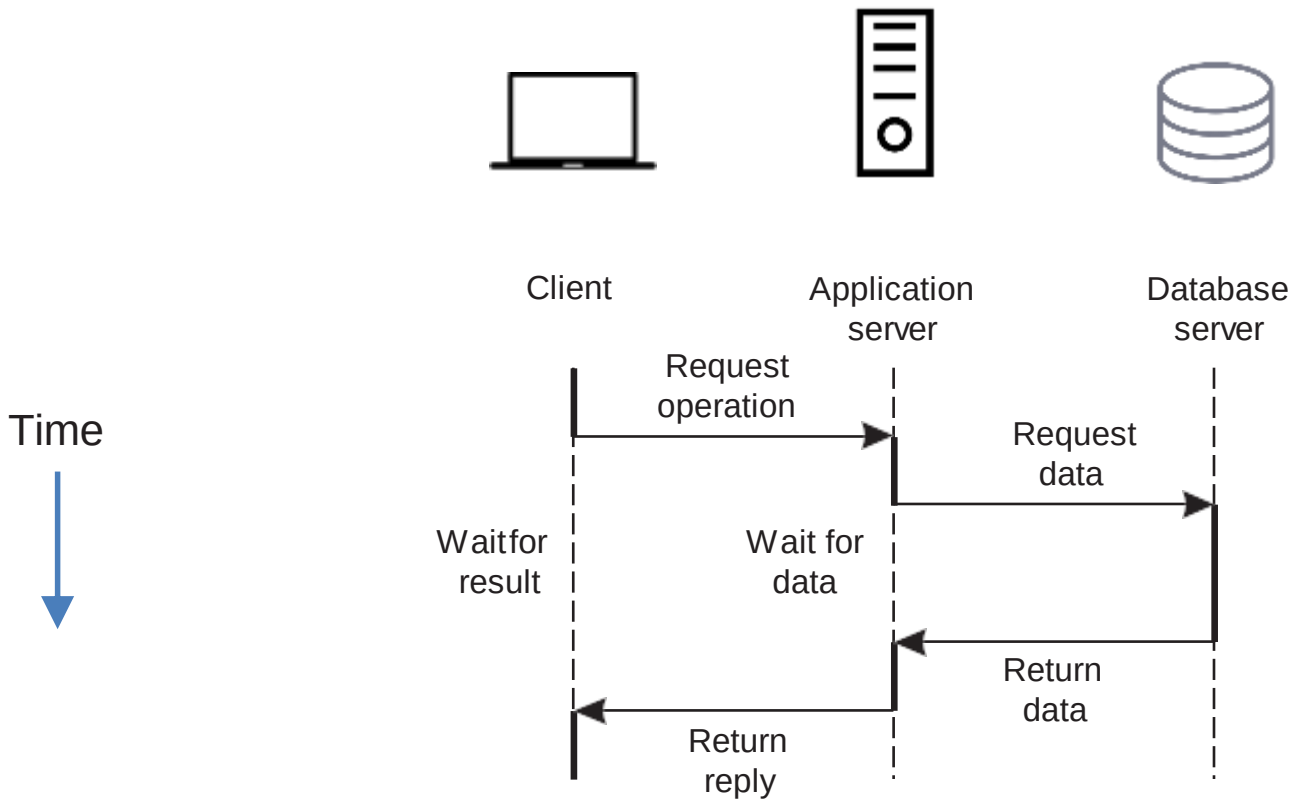
Características do modelo Cliente–Servidor [desde os anos 70]

- Existem processos que oferecem serviços (servidores)
- Existem processos que usam esses serviços (clientes)
- Clientes e servidores podem estar em máquinas diferentes
- Clientes seguem um modelo requisição/resposta ao usar os serviços



Arquiteturas centralizadas

Sendo Cliente e Servidor ao mesmo tempo



Arquiteturas multicamada

Visão tradicional em três camadas lógicas

- A camada de apresentação (user interface) contém o necessário para a aplicação poder interagir com o usuário
- A camada de negócio (application) contém as funções de uma aplicação
- A camada de dados (database) contém os dados que o cliente quer manipular através dos componentes da aplicação

Observação

Estas camadas são encontradas em muitos sistemas de informação distribuídos, que usam tecnologias de bancos de dados tradicionais e suas aplicações auxiliares.

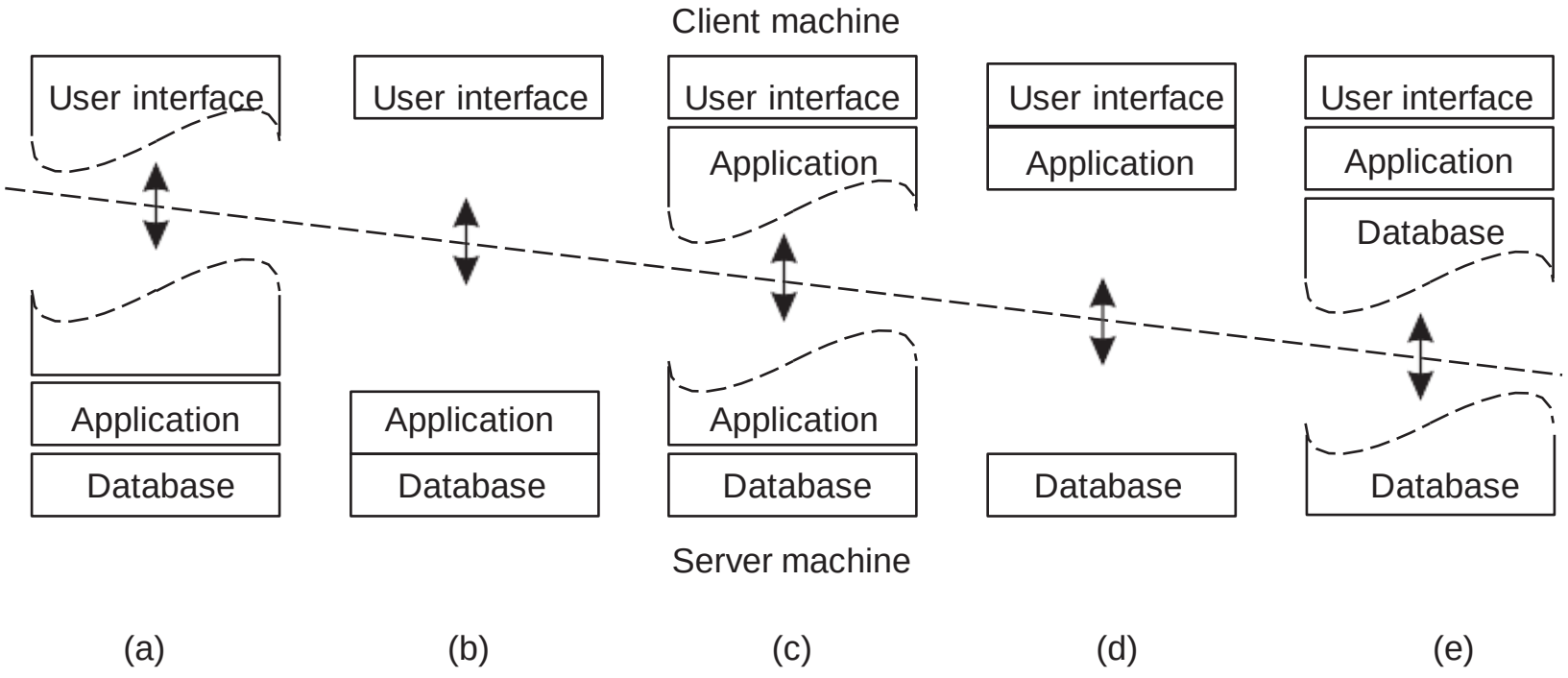
Arquiteturas multicamada

uma camada física: configurações de terminal burro/mainframe

duas camadas físicas: configuração cliente–servidor único.

três camadas físicas: cada camada em uma máquina separada

Configurações tradicionais em 2 camadas físicas + 3 lógicas:



Arquiteturas

- Arquiteturas centralizadas e multicamadas (multidivididas)
- Arquiteturas decentralizadas
- Arquiteturas híbridas

Arquiteturas decentralizadas

P2P estruturado os nós são organizados seguindo uma estrutura de dados distribuída específica

amazon DynamoDB

P2P não-estruturado os nós selecionam aleatoriamente seus vizinhos

P2P híbrido alguns nós são designados, de forma organizada, para executar funções especiais

Nota:

Praticamente todos os casos são exemplos de redes overlay: dados são roteados usando conexões definidas pelos nós

Arquiteturas decentralizadas

P2P estruturado os nós são organizados seguindo uma estrutura de dados distribuída específica

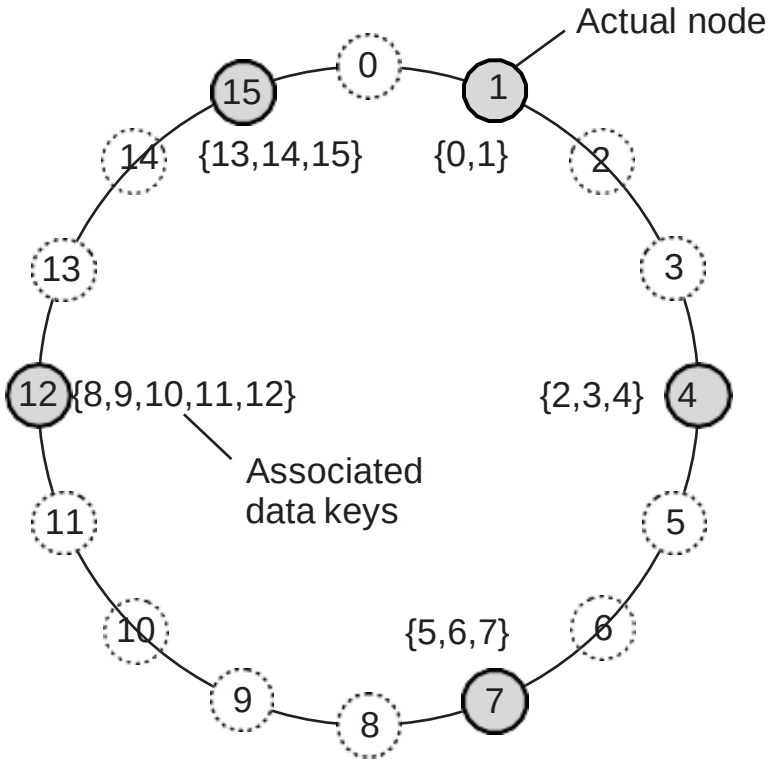
amazon DynamoDB

[Amazon 2012]

Sistemas P2P estruturados

Ideia básica

Organizar os nós em uma rede overlay estruturada, tal como um anel lógico, e fazer com que alguns nós se tornem responsáveis por alguns serviços baseado unicamente em seus Ids [Stoica 2001].



Nota

O sistema provê uma operação LOOKUP(key) que irá fazer o roteamento de uma requisição até o nó correspondente usando as conexões lógicas.

Arquiteturas decentralizadas

P2P estruturado os nós são organizados seguindo uma estrutura de dados distribuída específica

P2P não-estruturado os nós selecionam aleatoriamente seus vizinhos

[NullSoft 2000]

[2003]

P2P híbrido alguns nós são designados, de forma organizada, a executar funções especiais

Sistemas P2P não-estruturados

Observação

Muitos sistemas P2P não-estruturados tentam manter um grafo aleatório.

Princípio básico

Cada nó deve contactar um outro nó selecionado aleatoriamente:

- Cada participante mantém uma visão parcial da rede, consistindo de c outros nós (denominados vizinhos)
- Cada nó P seleciona periodicamente um nó Q de sua visão parcial
- P e Q trocam informação e participantes de suas respectivas visões parciais

Sistemas P2P não-estruturados

Observação

Muitos sistemas P2P não-estruturados tentam manter um grafo aleatório.

Princípio básico

Cada nó deve contactar um outro nó selecionado aleatoriamente:

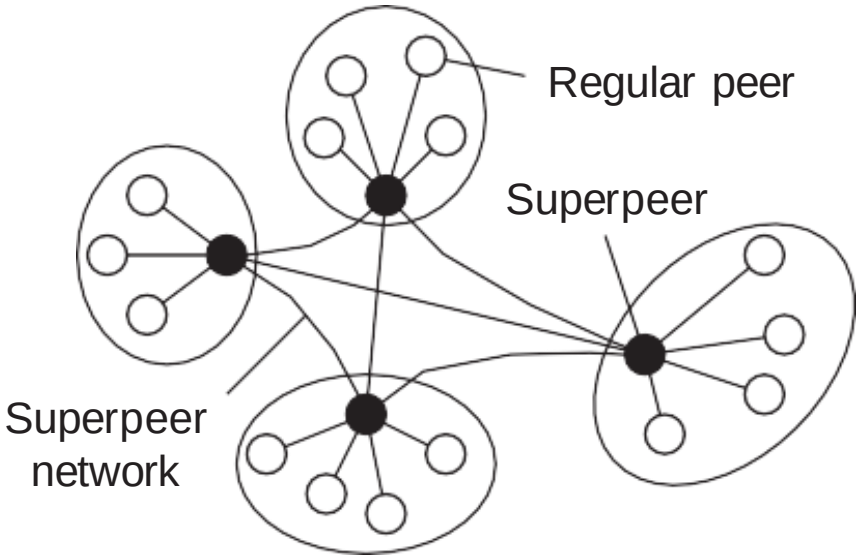
- Cada participante mantém uma visão parcial da rede, consistindo de c outros nós (denominados vizinhos)
- Cada nó P seleciona periodicamente um nó Q de sua visão parcial
- P e Q trocam informação e participantes de suas respectivas visões parciais

Gossip

Sistemas P2P não-estruturados: Superpeers

Observação

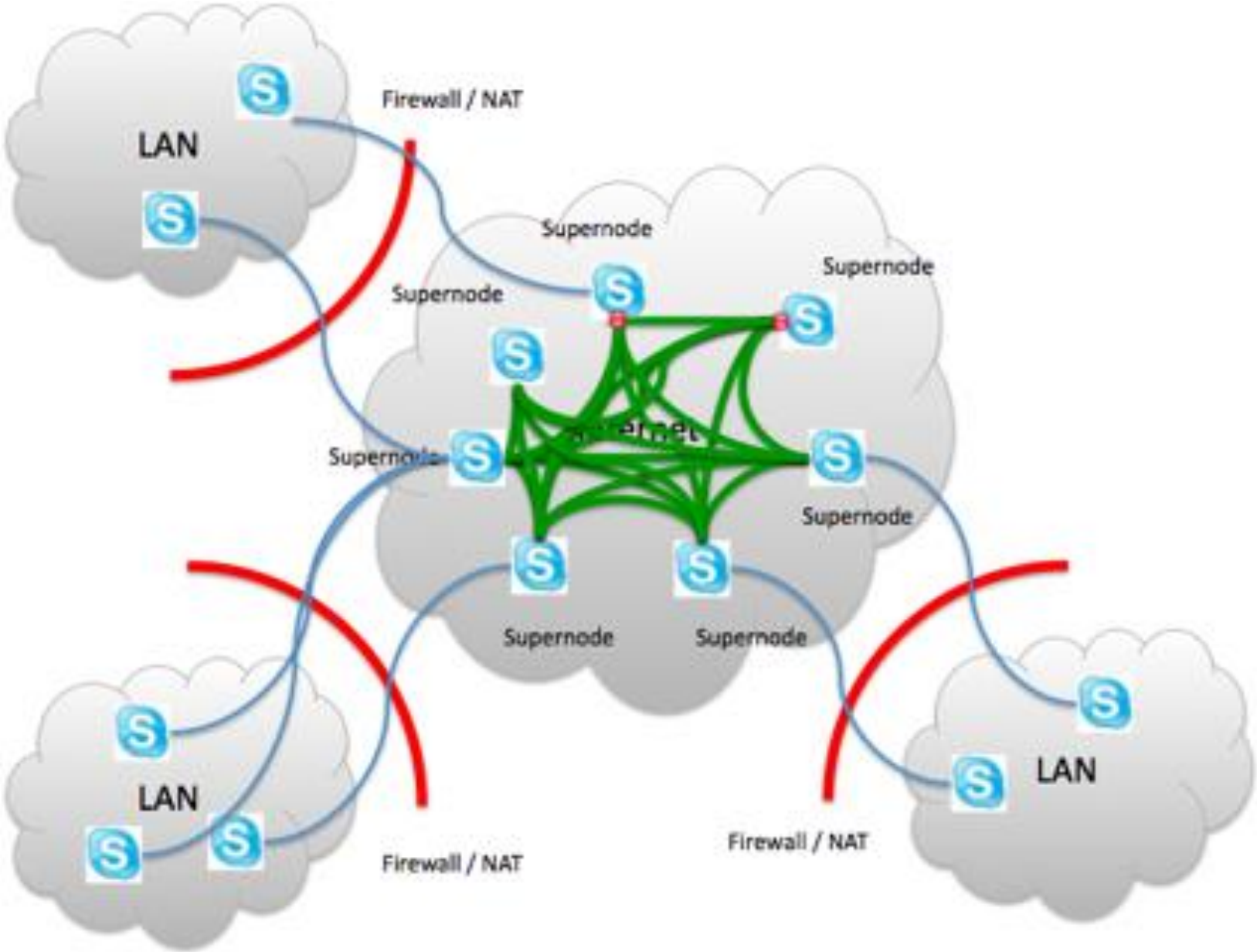
Às vezes, selecionar alguns nós para realizar algum trabalho específico pode ser útil.



Exemplos:

- Peers para manter um índice (para buscas)
- Peers para monitorar o estado da rede
- Peers capazes de configurar conexões

Princípio de operação do Skype: A quer contactar B



https://www.disruptivetelephony.com/2010/12/understanding-todays-skype-outage-explaining-supernodes.html

Arquiteturas decentralizadas

P2P estruturado os nós são organizados seguindo uma estrutura de dados distribuída específica

P2P não-estruturado os nós selecionam aleatoriamente seus vizinhos

P2P híbrido alguns nós são designados, de forma organizada, a executar funções especiais

Cliente-Servidor + P2P

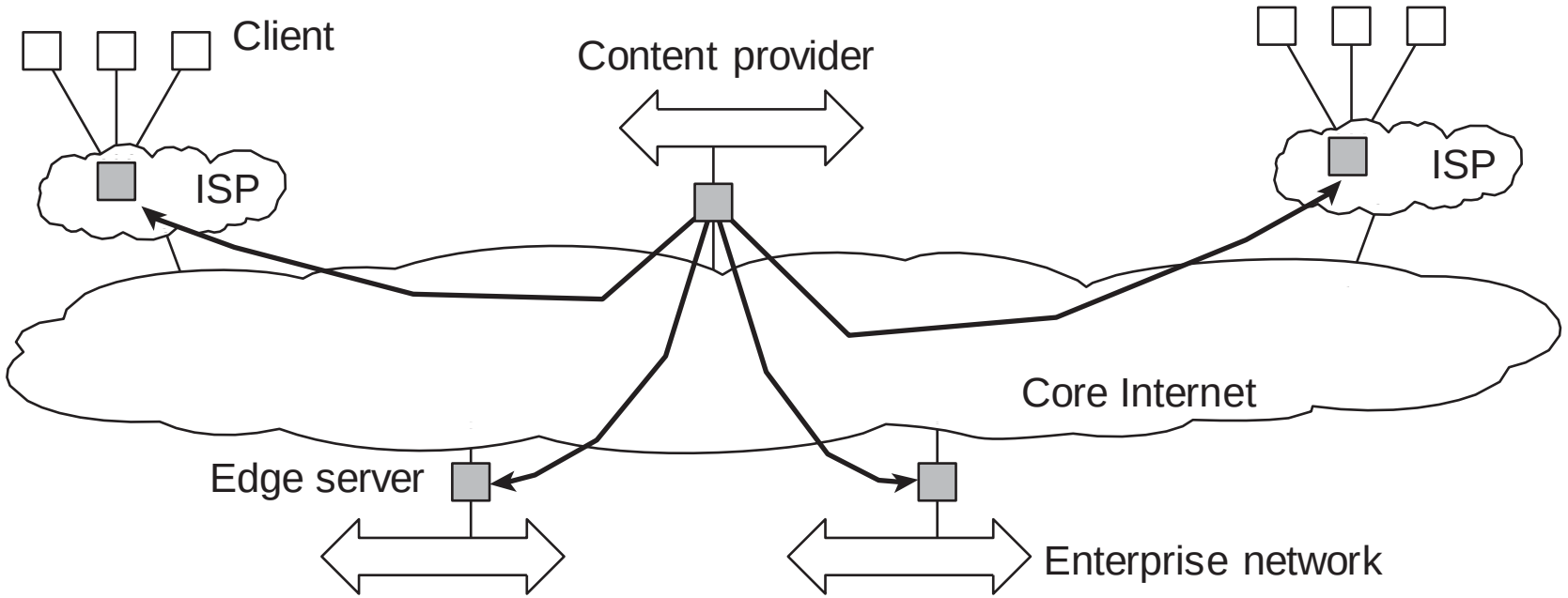
[Amazon 2012]

[2015]

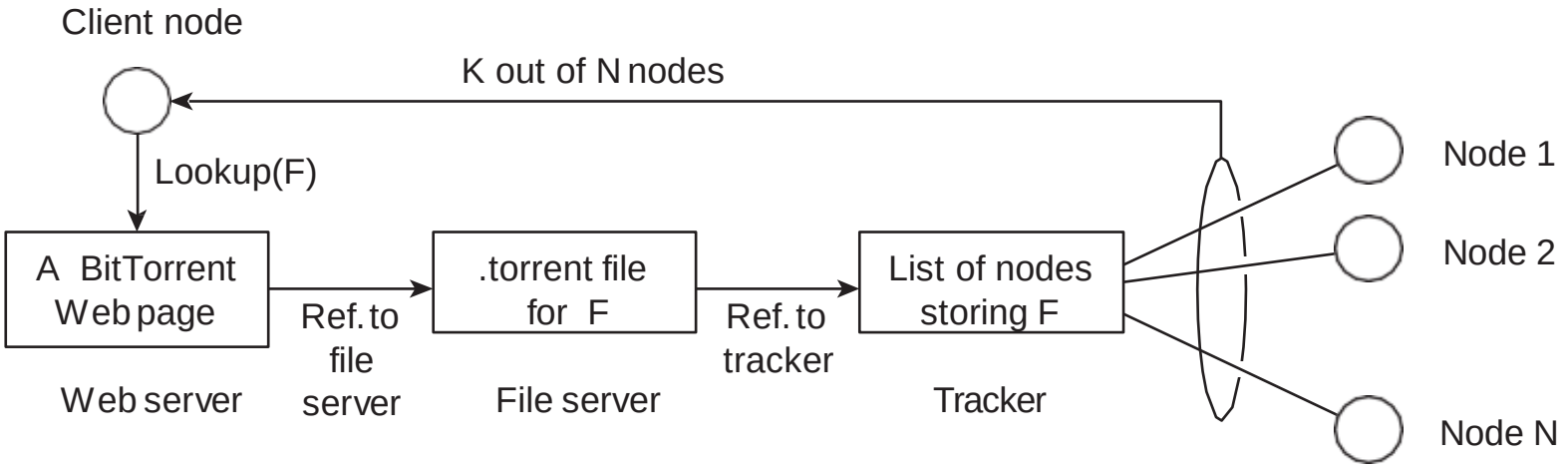
Arquiteturas híbridas: C–S com P2P – CDNs

Exemplo:

Arquiteturas de servidores de borda (*edge-servers*), utilizados com frequência como Content Delivery Networks (redes de distribuição de conteúdo).



Arquiteturas híbridas: C–S com P2P – BitTorrent



Ideia básica

Assim que um nó identifica de onde o arquivo será baixado, ele se junta a uma swarm (multidão) de pessoas que, em paralelo, receberão pedaços do arquivo da fonte e redistribuirão esses pedaços entre os outros [Cohen 2001].

Agenda

- Estilos arquiteturais
- Arquiteturas de software
- Arquiteturas versus middleware
- Sistemas distribuídos autogerenciáveis

Arquiteturas versus Middleware

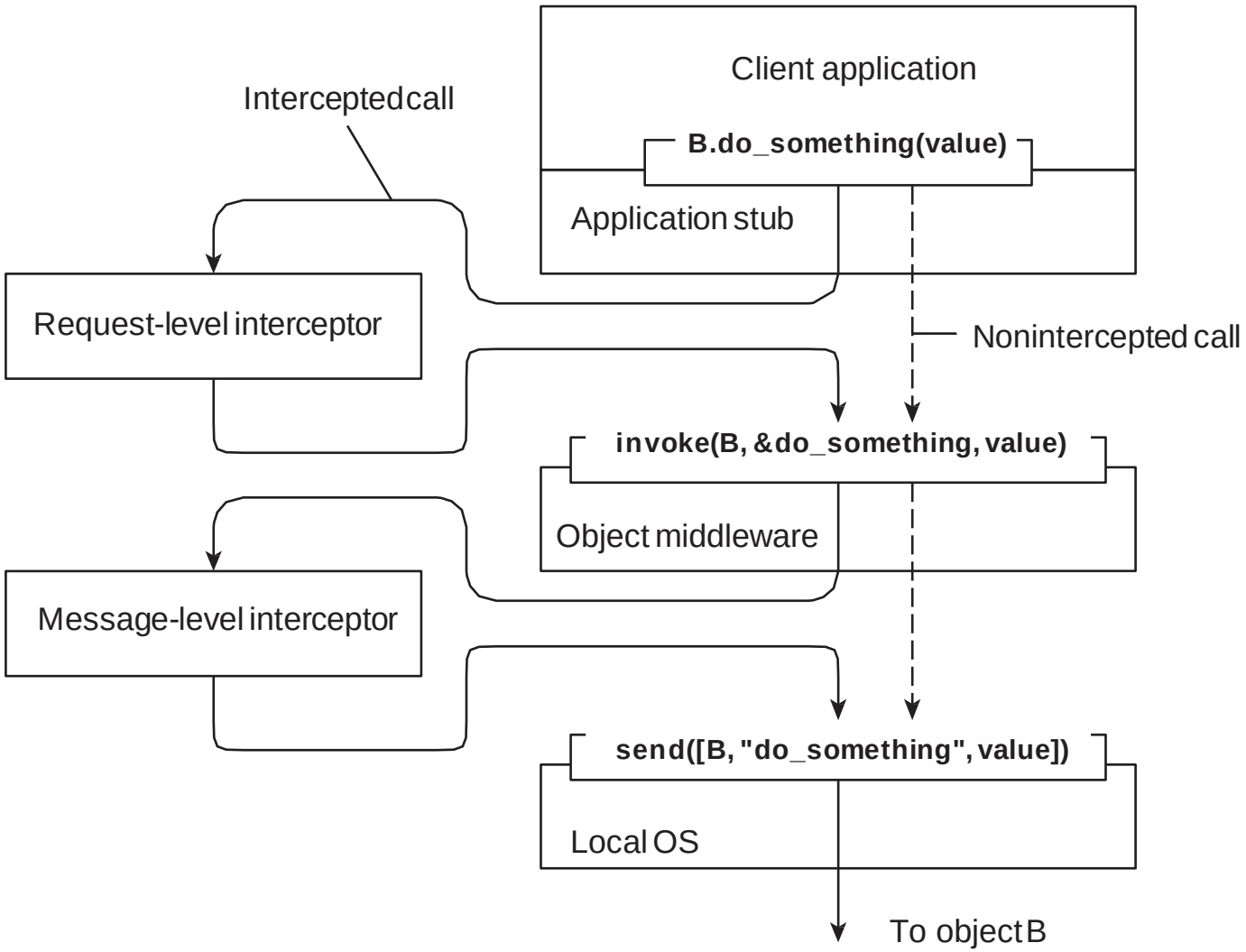
Problema

Em muitos casos, arquiteturas/sistemas distribuídos são desenvolvidos de acordo com um estilo arquitetural específico. O estilo escolhido pode não ser o melhor em todos os casos ⇒ é necessário adaptar o comportamento da arquitetura usando um middleware (dinamicamente).

Interceptors

Interceptam o fluxo de controle normal quando um objeto remoto for invocado.

Interceptors



Conceitos adquiridos

- Arquitetura em camadas, objetos, eventos e recursos.
- Acoplamento na referência e no tempo.
- Arquitetura cliente-servidor.
- Arquitetura de 3 camadas.
- P2P (Peer-to-Peer).
- Interceptor.

Agenda

- Estilos arquiteturais
- Arquiteturas de software
- Arquiteturas versus middleware
- Sistemas distribuídos autogerenciáveis

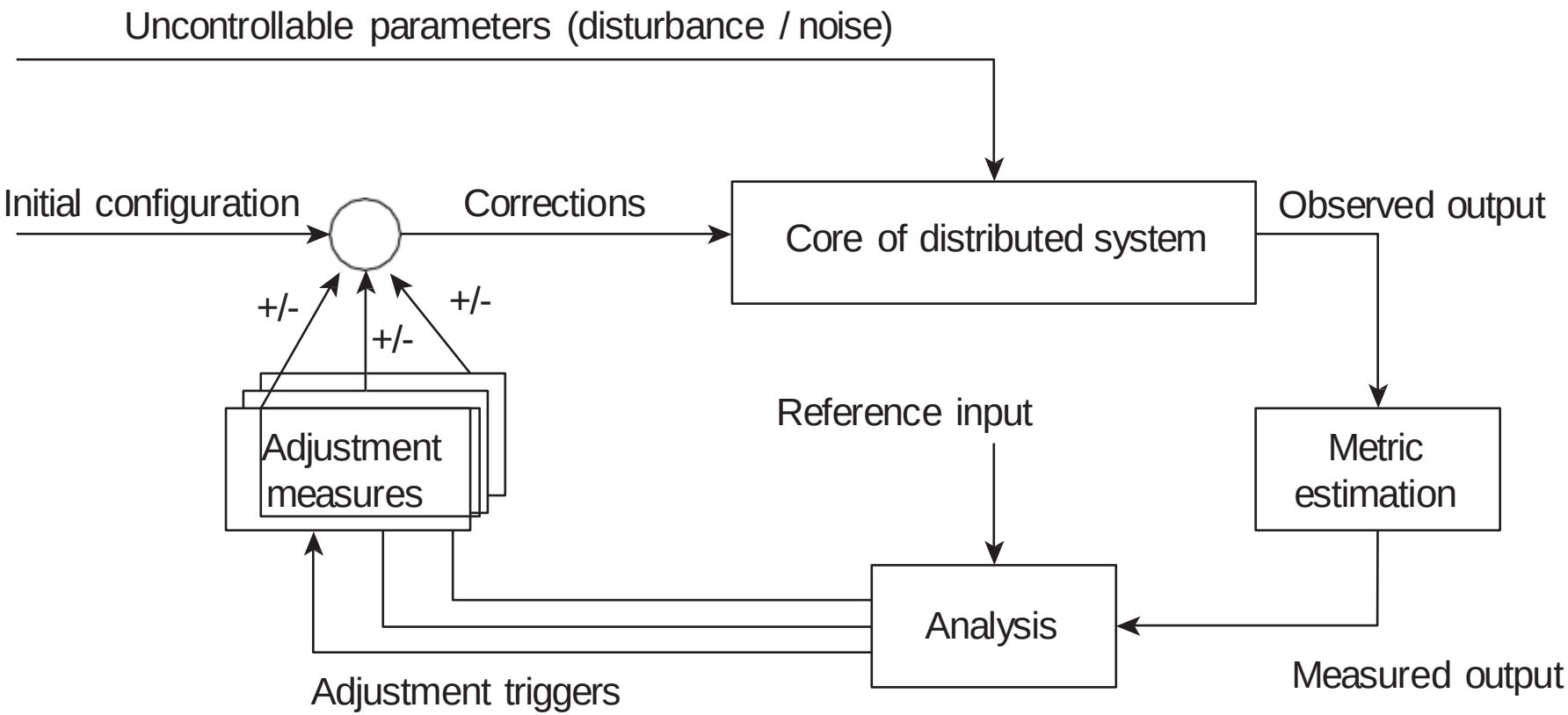
Sistemas distribuídos autogerenciáveis

Sistemas e arquiteturas de software que consideram a adaptação automática do seu comportamento:

- Autoconfiguração
- Autogerenciamento
- Autocura
- Auto-otimização
- Auto-*

Modelo de regulação por feedback

Em muitos casos, sistemas auto-* são organizados como um sistema de regulação por feedback



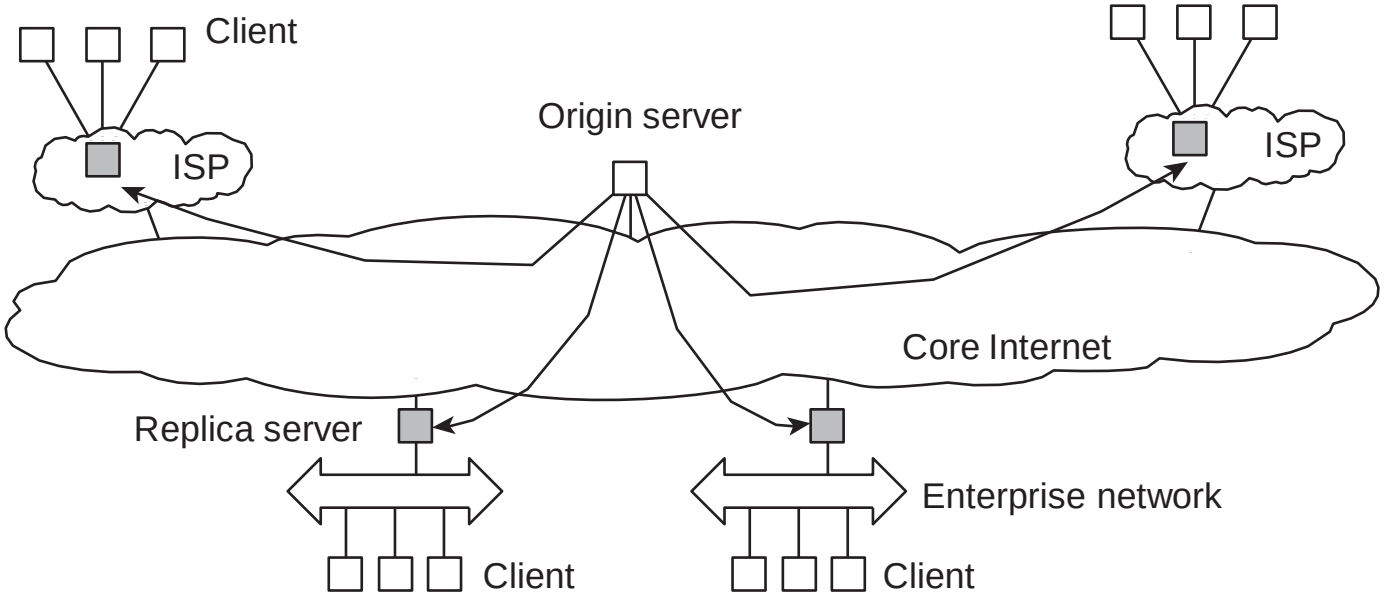
Modelo de regulação por feedback

Globule

CDN colaborativo que analisa as requisições dos navegadores para decidir onde por as réplicas do conteúdo Web. As decisões são baseadas por um modelo de custo geral:

$$Minimize\ cost = (w_1 \times m_1) + (w_2 \times m_2) + \cdots + (w_n \times m_n)$$

Modelo de regulação por feedback

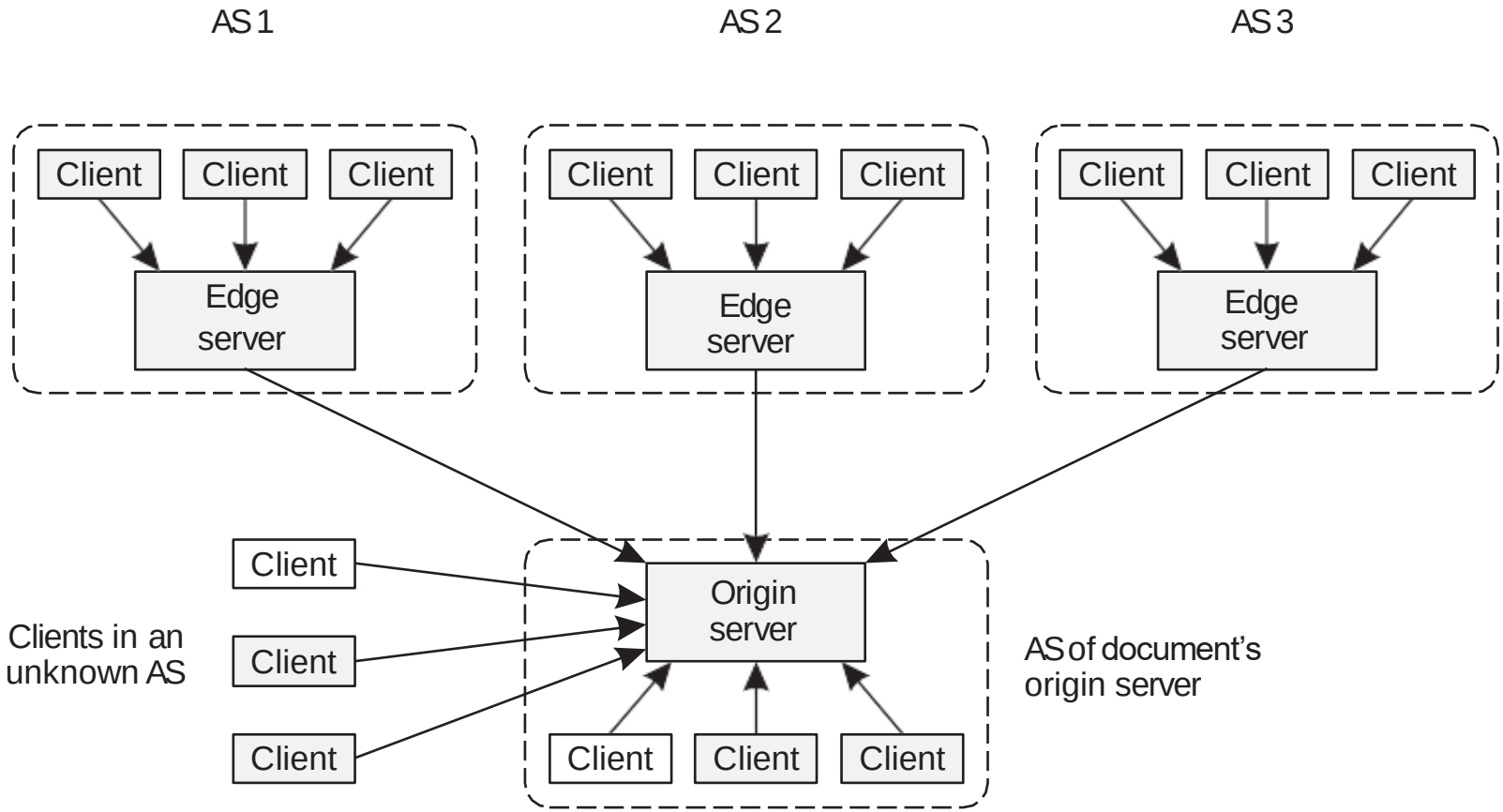


O servidor de origem (Origin server) do Globule coleta as requisições e realiza a análise what-if do que teria ocorrido se a página *P* tivesse estado em algum servidor de ponta S. Diversas estratégias são avaliadas, escolhendo a melhor delas.

Modelo de regulação por feedback

Research question

Faz sentido aplicar um estratégia diferente para cada página Web ou é melhor aplicar uma estratégia única para todas elas?



Modelo de regulação por feedback

Estratégias de Cache

Abbr.	Name	Description
NR	No replication	Sem replicação ou cache. Os clientes repassam seus *requests* diretamente ao origin server.

Modelo de regulação por feedback

Estratégias de Cache

Abbr.	Name	Description
NR	No replication	Sem replicação ou cache. Os clientes repassam seus *requests* diretamente ao origin server.
CV	Verification	Edge servers faz cache de documentos. A cada subsequente request, o origin server é contatado para revalidação.

Modelo de regulação por feedback

Estratégias de Cache

Abbr.	Name	Description
NR	No replication	Sem replicação ou cache. Os clientes repassam seus *requests* diretamente ao origin server.
CV	Verification	Edge servers faz cache de documentos. A cada subsequente request, o origin server é contatado para revalidação.
CLV	Limited validity	Edge servers faz cache de documentos. O documento tem associado um tempo de expiração e é removido do cache quando fica inválido.

Modelo de regulação por feedback

Estratégias de Cache

Abbr.	Name	Description
NR	No replication	Sem replicação ou cache. Os clientes repassam seus *requests* diretamente ao origin server.
CV	Verification	Edge servers faz cache de documentos. A cada subsequente request, o origin server é contatado para revalidação.
CLV	Limited validity	Edge servers faz cache de documentos. O documento tem associado um tempo de expiração e é removido do cache quando fica inválido.
CDV	Delayed verification	Edge servers faz cache de documentos. O documento tem associado um tempo de expiração e contata o origin server para revalidação.

Modelo de regulação por feedback

Estratégias de Replicação

Abbr.	Name	Description
SI	Server invalidation	Edge servers faz cache de documentos, mas o origin server invalida as cópias quando o documento é atualizado.

Modelo de regulação por feedback

Estratégias de Replicação

Abbr.	Name	Description
SI	Server invalidation	Edge servers faz cache de documentos, mas o origin server invalida as cópias quando o documento é atualizado.
SU*x*	Server updates	O origin server mantém cópias nos *x* edge servers mais relevantes; *x* = 10, 25, 50

Modelo de regulação por feedback

Estratégias de Replicação

Abbr.	Name	Description
SI	Server invalidation	Edge servers faz cache de documentos, mas o origin server invalida as cópias quando o documento é atualizado.
SU*x*	Server updates	O origin server mantém cópias nos *x* edge servers mais relevantes; *x* = 10, 25, 50
SU50 + CLV	Hybrid SU50 & CLV	O origin server mantém cópias nos 50 edge servers mais relevantes; outros edge servers usam a estratégia de cache CLV.

Modelo de regulação por feedback

Estratégias de Replicação

Abbr.	Name	Description
SI	Server invalidation	Edge servers faz cache de documentos, mas o origin server invalida as cópias quando o documento é atualizado.
SU*x*	Server updates	O origin server mantém cópias nos *x* edge servers mais relevantes; *x* = 10, 25, 50
SU50 + CLV	Hybrid SU50 & CLV	O origin server mantém cópias nos 50 edge servers mais relevantes; outros edge servers usam a estratégia de cache CLV.
SU50 + CDV	Hybrid SU50 & CDV	O origin server mantém cópias nos 50 edge servers mais relevantes; outros edge servers usam a estratégia de cache CDV.

Modelo de regulação por feedback

Turnaround time (TaT) and bandwidth (BW) in relative measures; stale documents as fraction of total requested documents.

Strategy	Site 1			Site 2		
	TaT	Stale docs	BW	TaT	Stale docs	BW
NR	203	**0**	118	183	**0**	115
CV	227	**0**	113	190	**0**	**100**
CLV	182	0.0061	113	142	0.0060	**100**
CDV	182	0.0059	113	142	0.0057	**100**
SI	182	**0**	113	141	**0**	**100**
SU10	128	**0**	**100**	160	**0**	114
SU25	114	**0**	123	132	**0**	119
SU50	102	**0**	165	114	**0**	132
SU50+CLV	**100**	0.0011	165	**100**	0.0019	125
SU50+CDV	**100**	0.0011	165	**100**	0.0017	125

Conclusão: Não há uma estratégia única