"How does he ever expect to be an architect if he can't invent a new roof?"

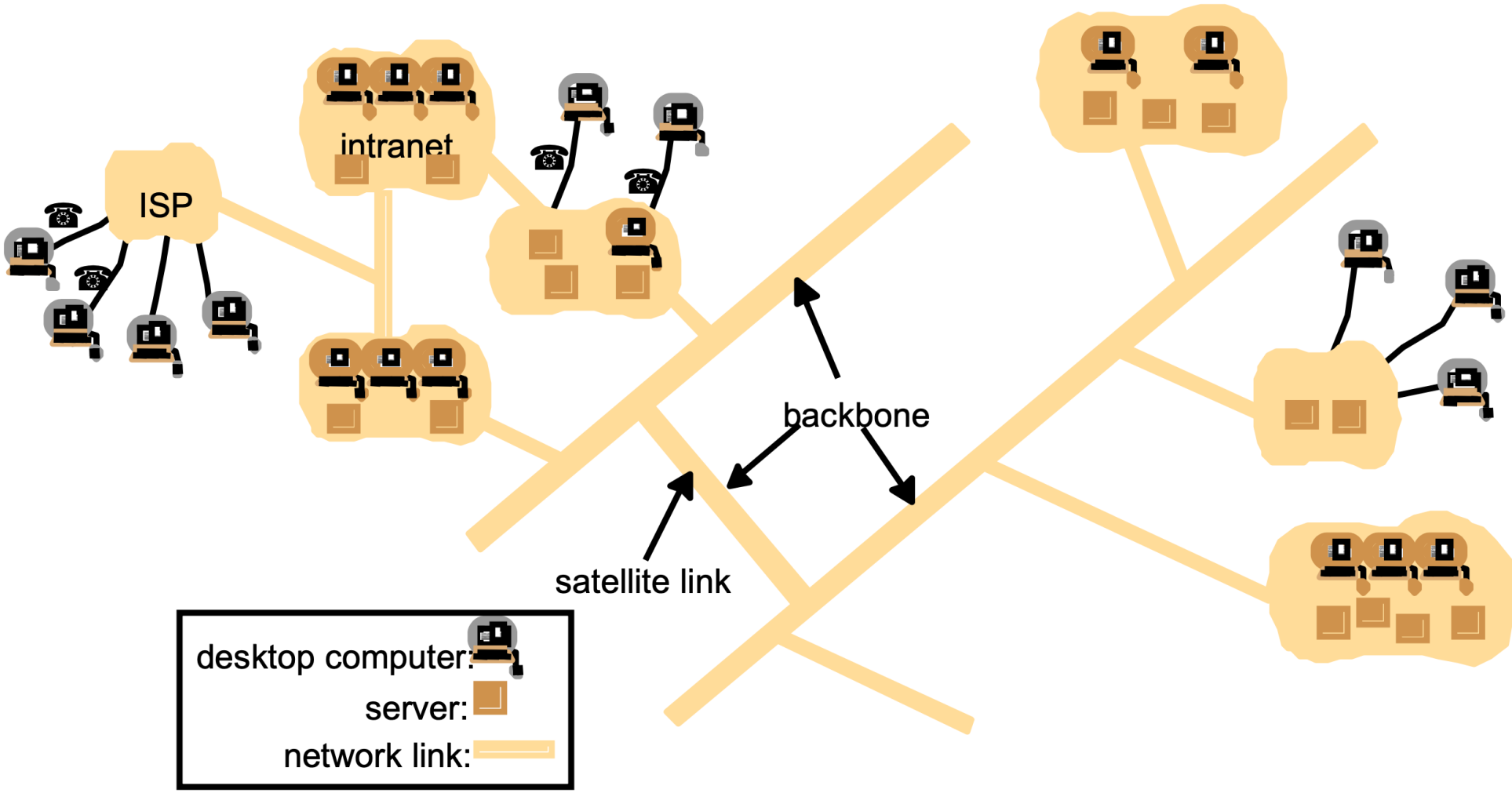
Introdução/Arquitecturas

Computação Distribuída 2023/24

dgomes@ua.pt

universidade de aveiro

Apresentação com base nos conteudos do livro "M. van Steen and A.S. Tanenbaum, Distributed Systems, 4th ed., distributed-systems.net."


ISP
intranet
backbone
satellite link
desktop computer:
server:
network link:

Sistemas Distribuídos

Definição

Sistema em que os componentes de hardware e de software, localizados numa rede de computadores, comunicam e coordenam as suas acções através da passagem de mensagens

(DS Coulouris)

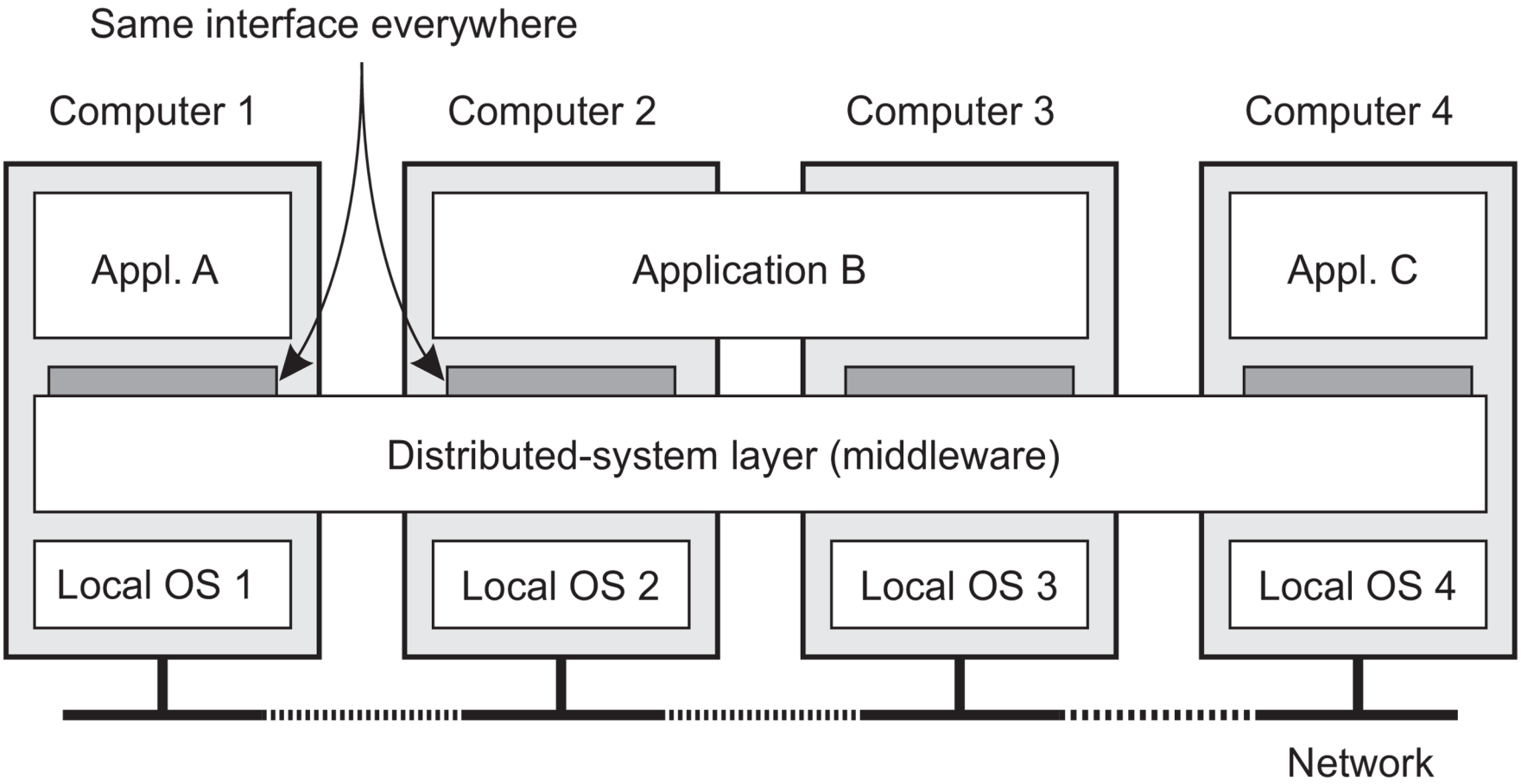
Um conjunto de computadores independentes que são apresentados ao utilizador como um único sistema integrado

(DS Tanembaum)

Caracteristicas

- Colecção de elementos de computação autónomos.
  - Relógio global
  - Gestão de membros
    - Aberto
    - Fechado
- Sistema único coerente.

Middleware


Same interface everywhere
Computer 1
Computer 2
Computer 3
Computer 4
Appl. A
Application B
Appl. C
Distributed-system layer (middleware)
Local OS 1
Local OS 2
Local OS 3
Local OS 4
Network

Middleware

Serviços disponibilizados:

- Comunicação (ex. RPC)
- Transacções
- Composição de Serviços
- Fiabilidade

Qual o objectivo?

- Suportar a partilha de recursos
- **Distribuição transparente (aplicação e utilizador)**
- Abertura
- Escalabilidade

Distribuição Transparente

Transparência	Descrição
Acesso	Esconde diferenças na representação dos dados e na forma como os mesmos são acedidos
Localização	Esconde onde um objecto está localizado
Relocalização	Esconde que um objecto pode ser movido para outra localização enquanto usado
Migração	Esconde que um objecto pode ser movido para outra localização
Replicação	Esconde que um objecto pode ser replicado
Concorrência	Esconde que um objecto pode ser partilhado entre diversos utilizadores concorrentes
Falhas	Esconde as falhas e recuperações de um objecto

Abertura

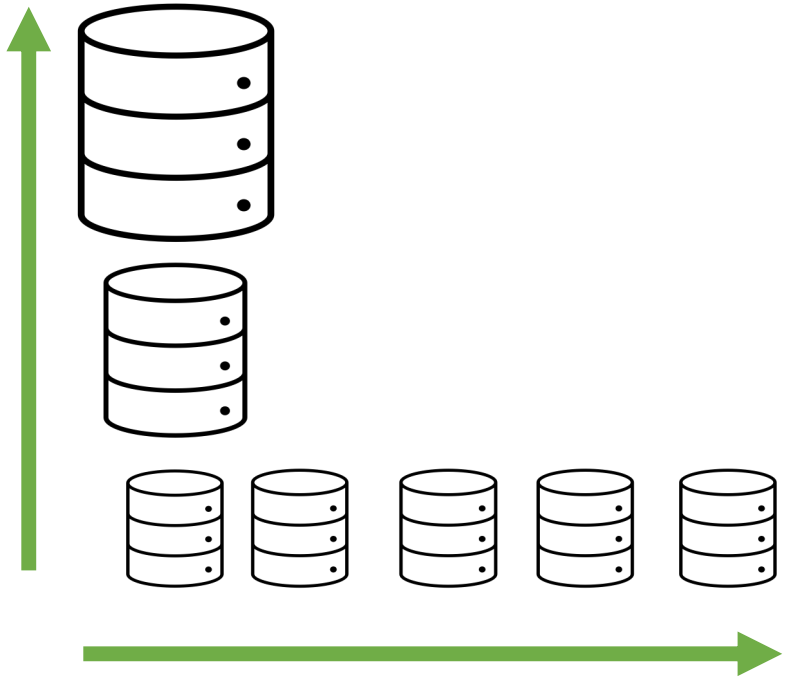
- “Being Open”
- Utilizar componentes que podem ser usados por, ou integrados, em outros sistemas
- Inter-operação
- Composição
- Extensibilidade

Escalabilidade

Dimensões

- Tamanho
- Geográfica
- Administrativa

Vertical vs Horizontal



Armadilhas (Pitfalls)

A rede é fiável

A rede é segura

A rede é homogénea

A topologia não se altera

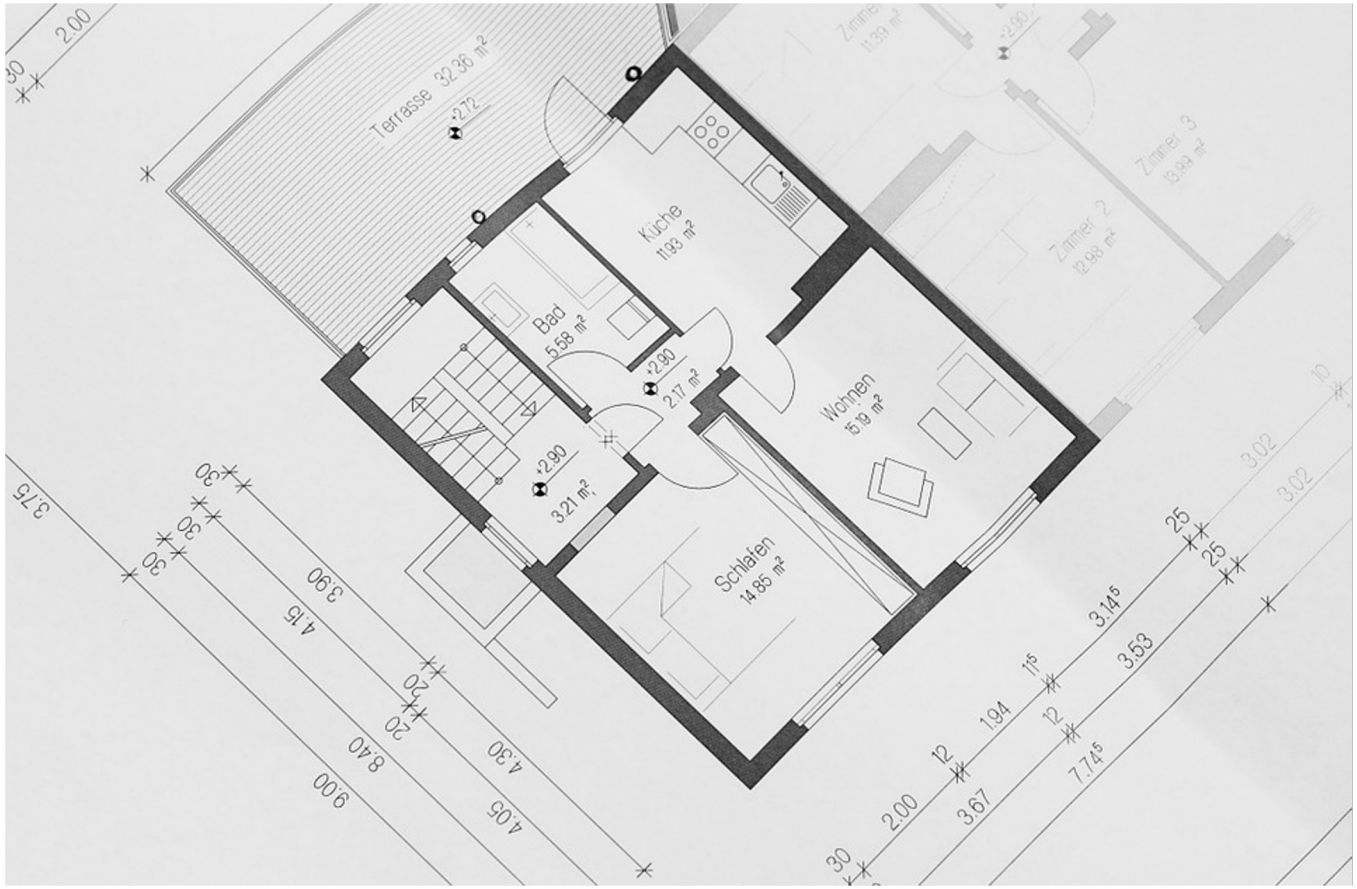
Latência é zero

Largura de banda é infinita

Custo de transporte é zero

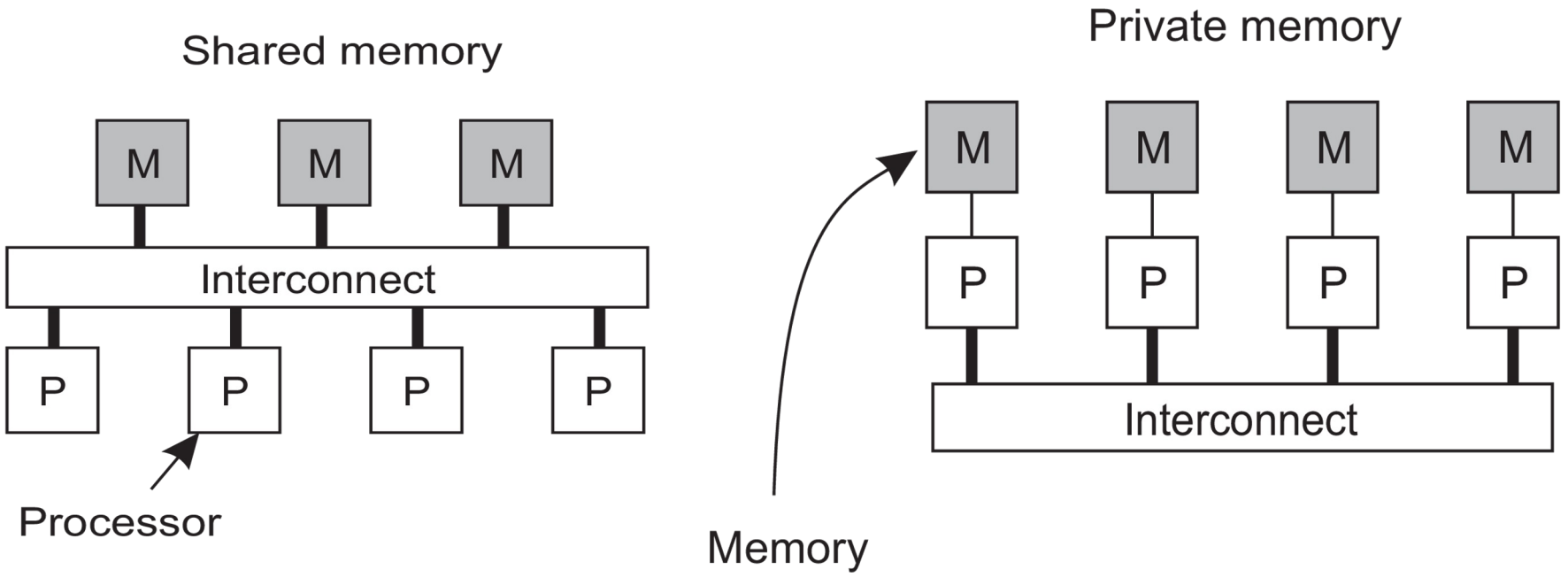
Existe um único administrador

Arquitecturas de Sistemas Distribuidos



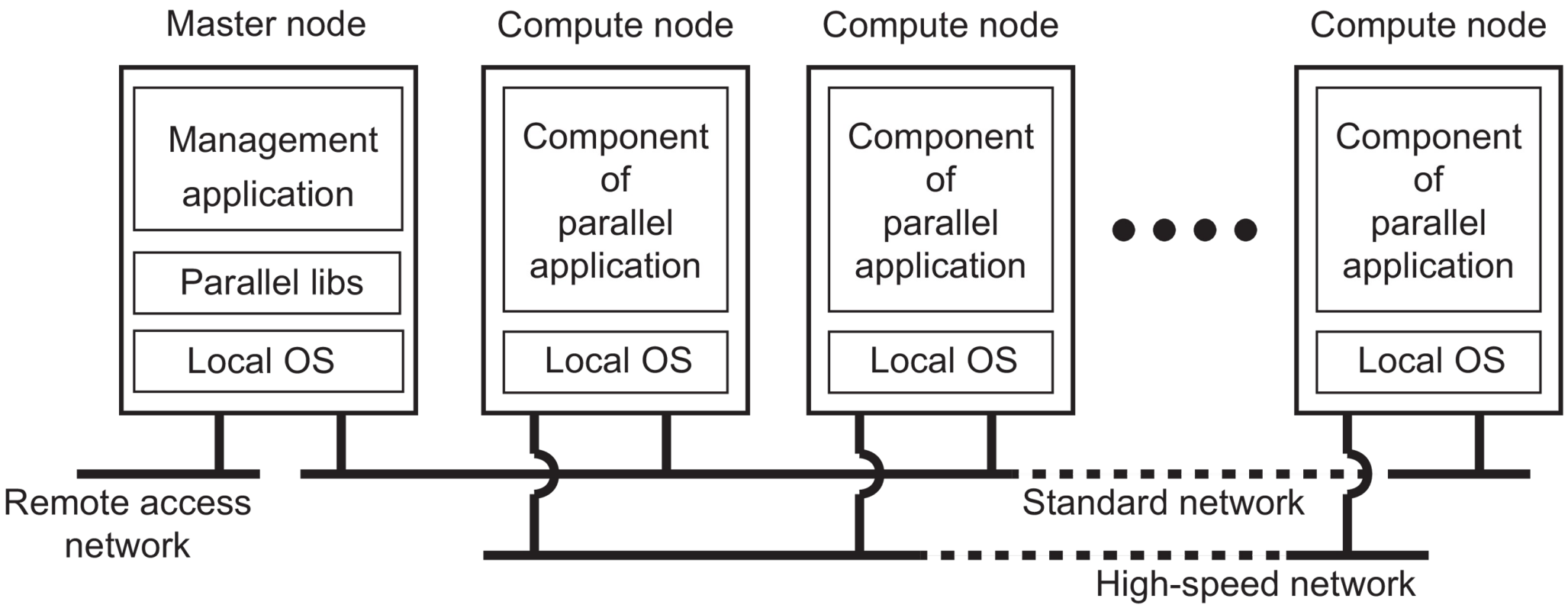
Arquitectuas de Hardware

Multiprocessador e multicore VS multicomputador



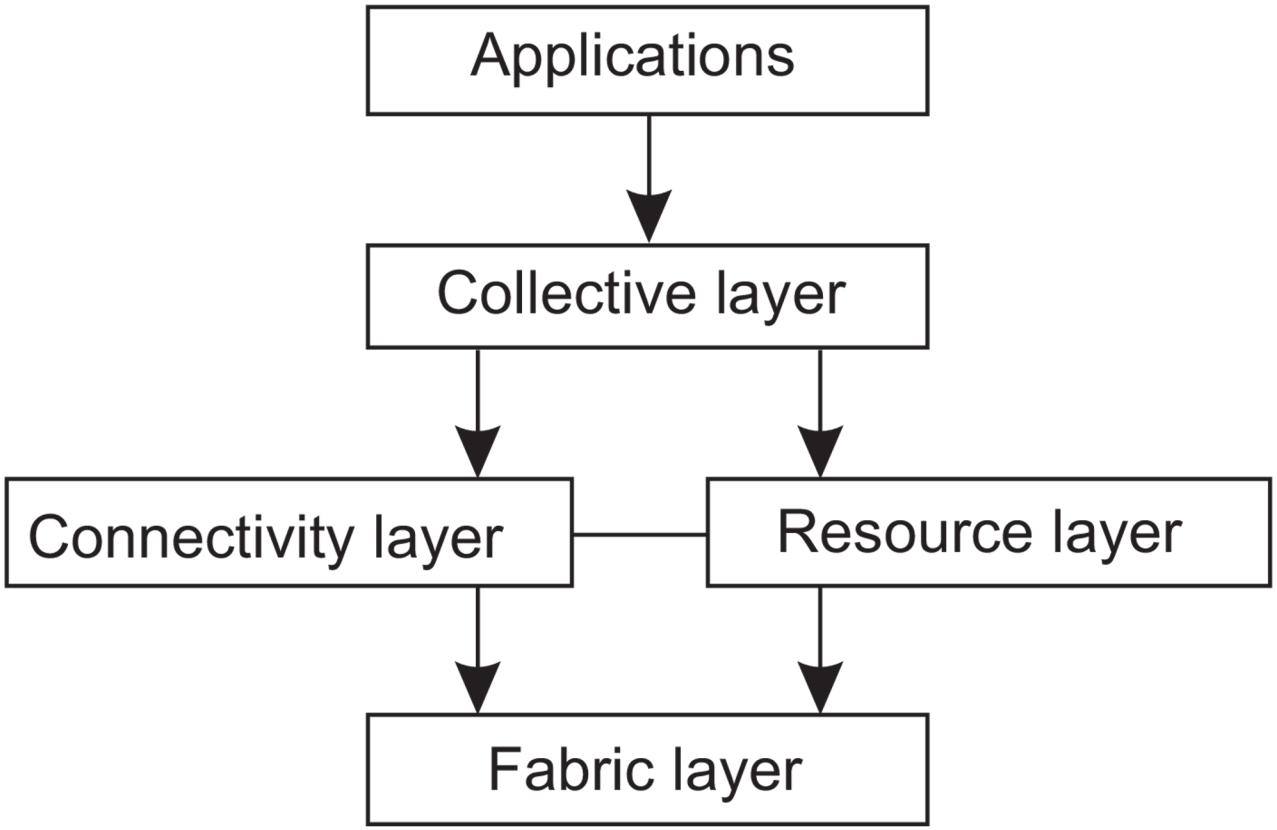
Arquitecturas

Cluster Computing



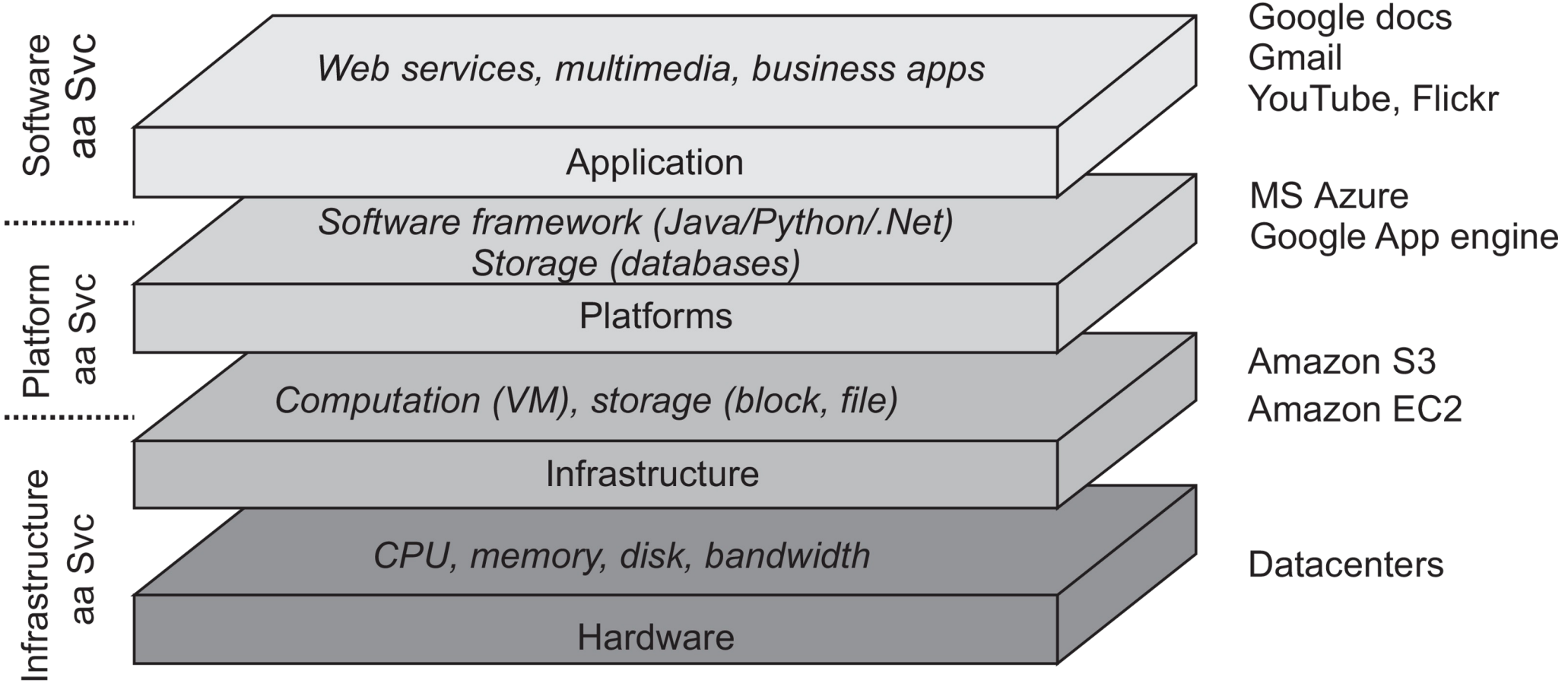
Arquitecturas

Grid Computing

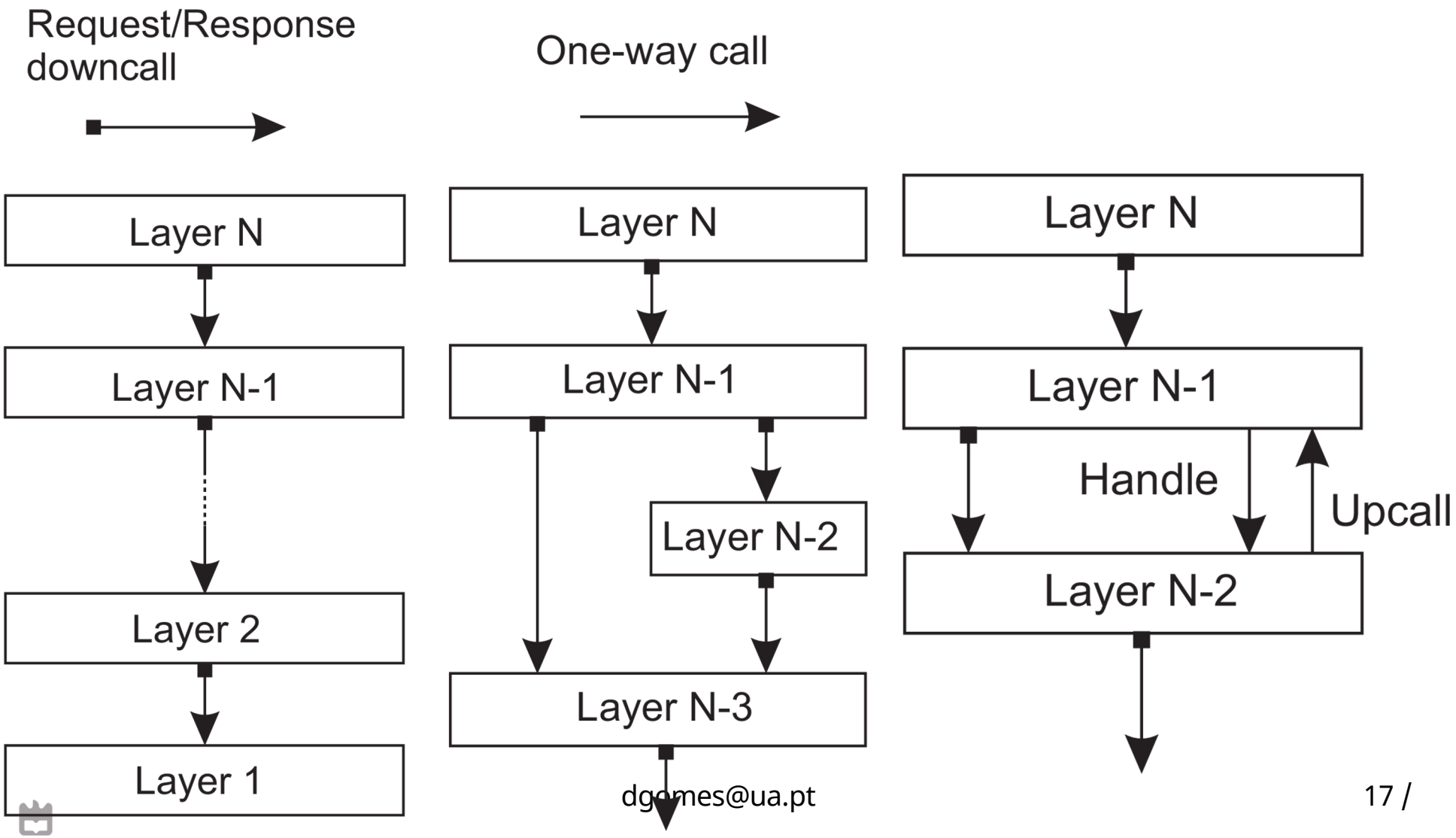


Arquitecturas

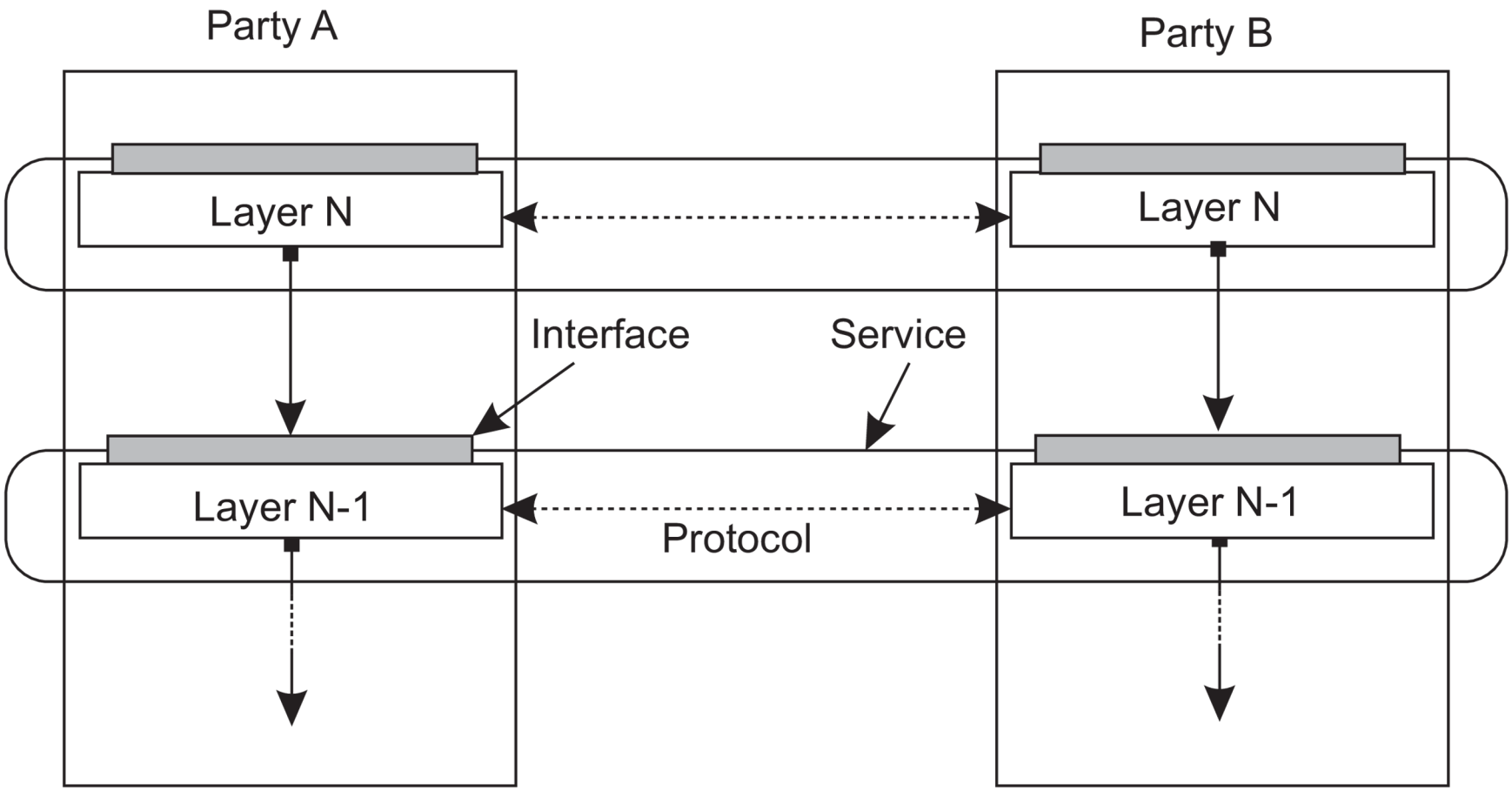
Cloud Computing



Arquitectura por camadas



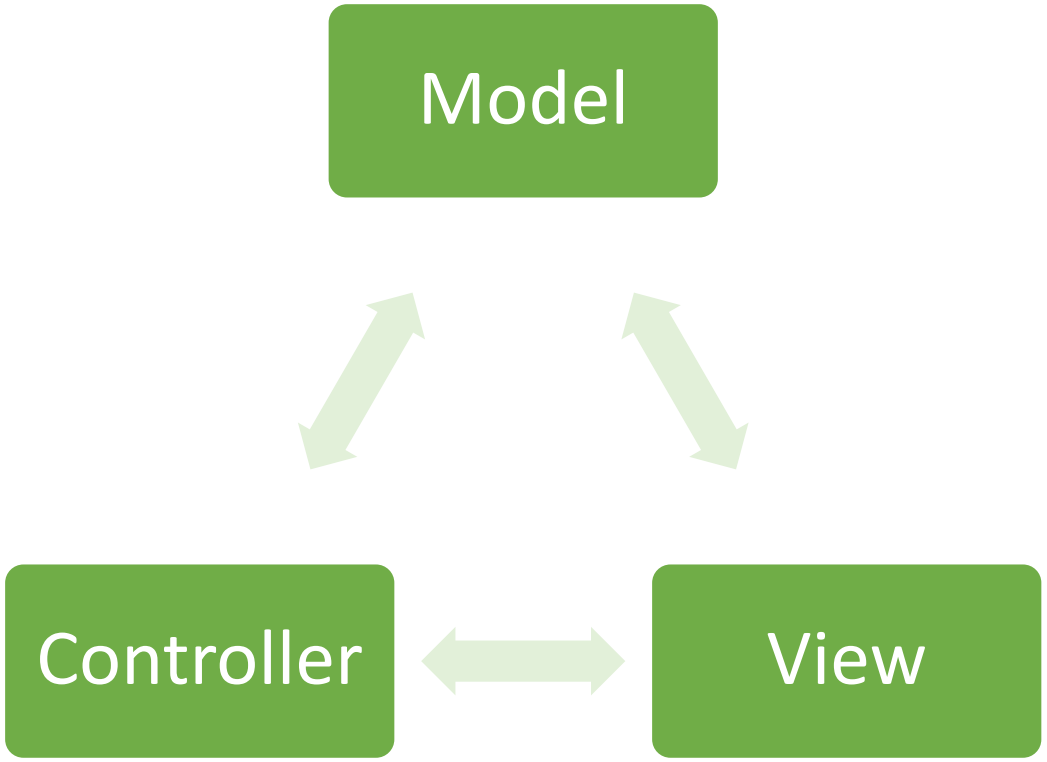
Protocolo, serviço, interface



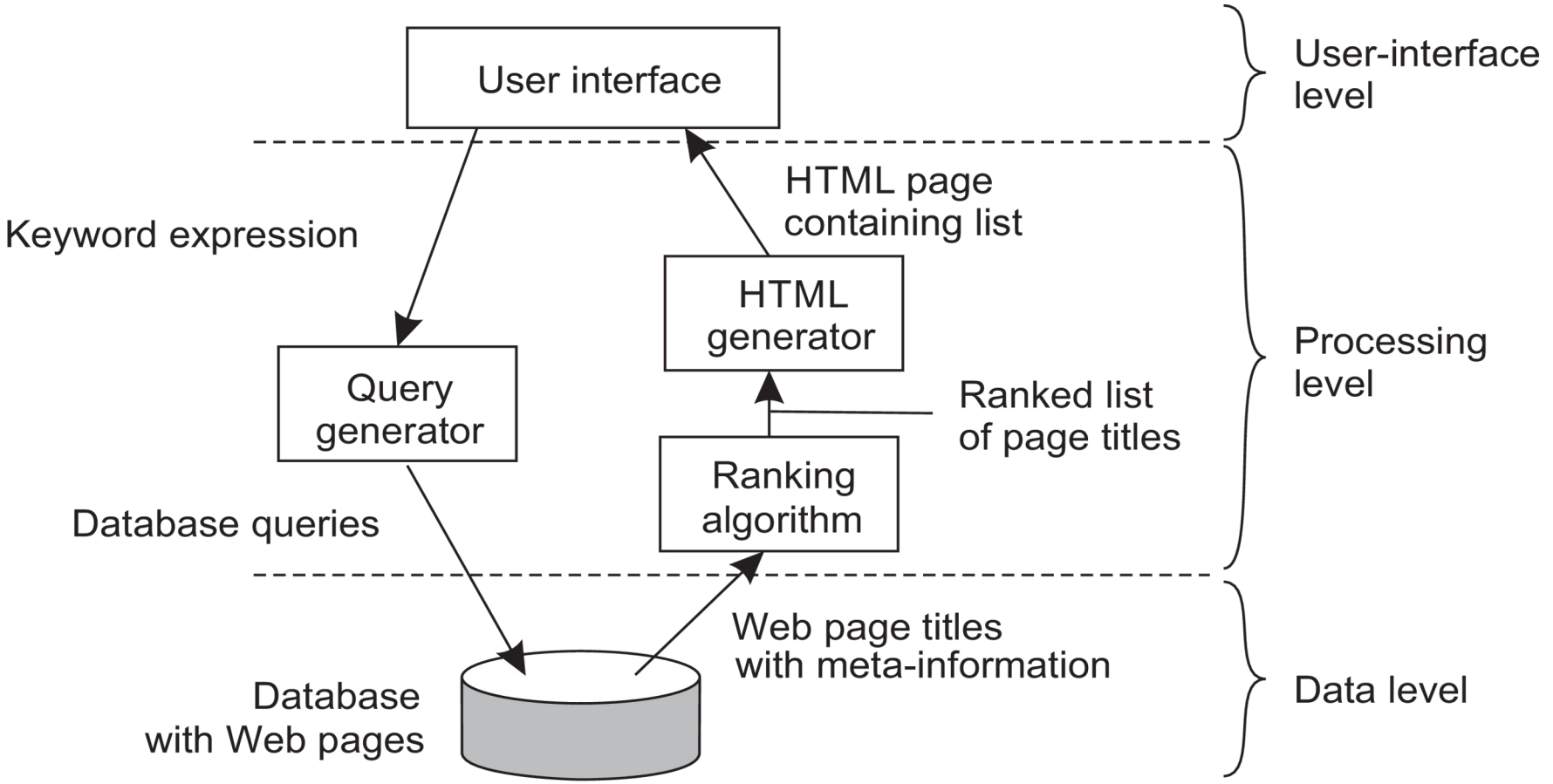
Camadas Aplicacionais

- Visão tradicional de 3-camadas
  - Camada de Dados
  - Camada de Interface
  - Camada de Processamento

Em inglês: **MVC** – Model-View-Controller



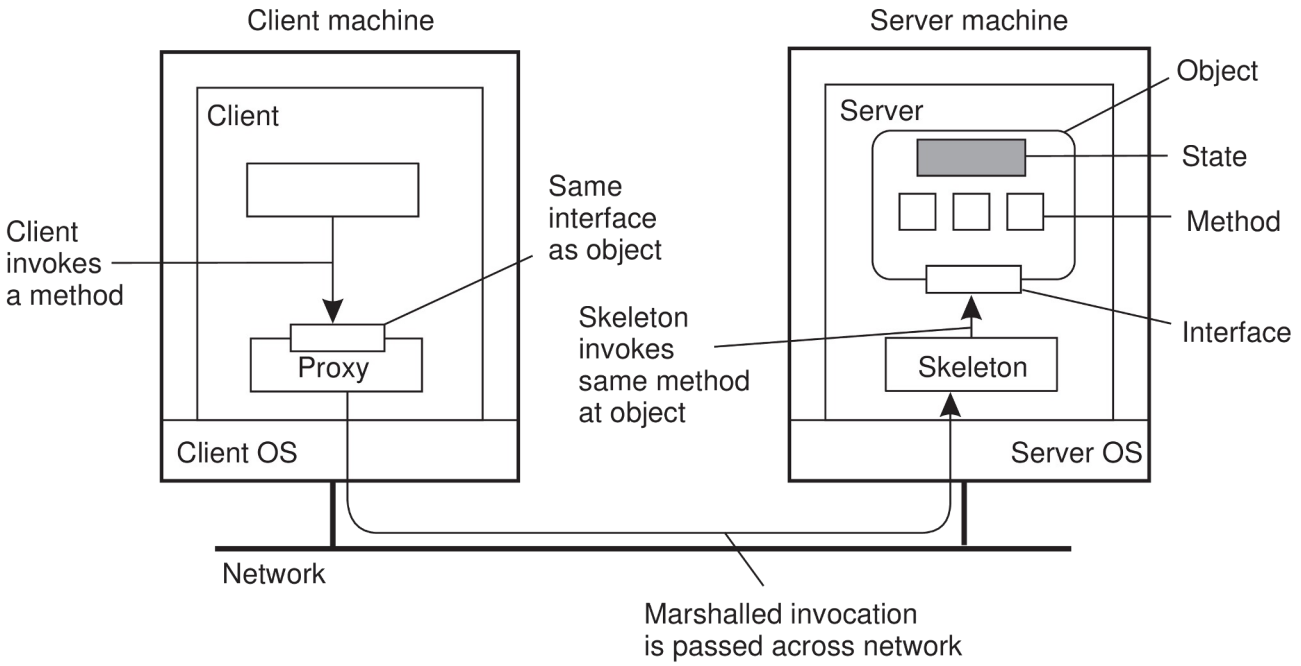
Exemplo: Motor de Pesquisa



Arquitecturas baseadas em objectos e orientadas a serviço

Arquitecturas baseadas em objectos:

- encapsulamento natural dos dados
- ***Interface*** do objecto esconde detalhes de implementação
- Quando cliente faz ***bind*** a um objecto distribuído, é carregada uma implementação da interface do objecto – ***proxy***
- Do lado do servidor surge uma entidade complementar o ***skeleton***

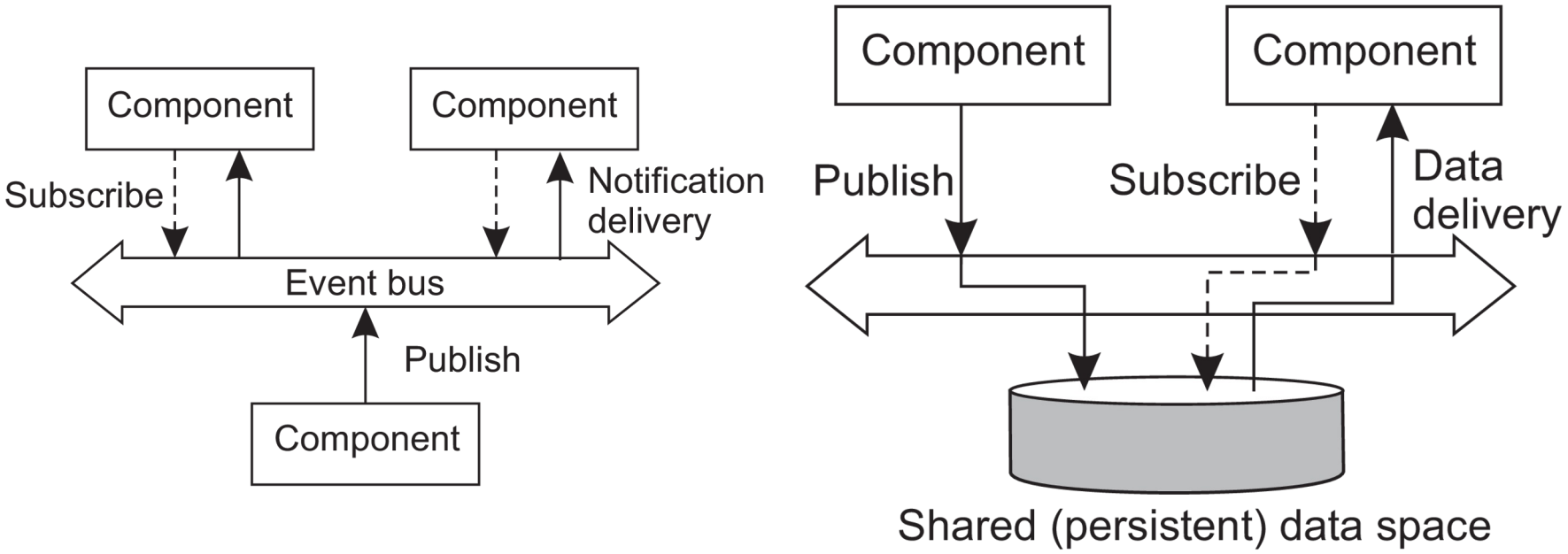


Arquitecturas baseadas em Recursos

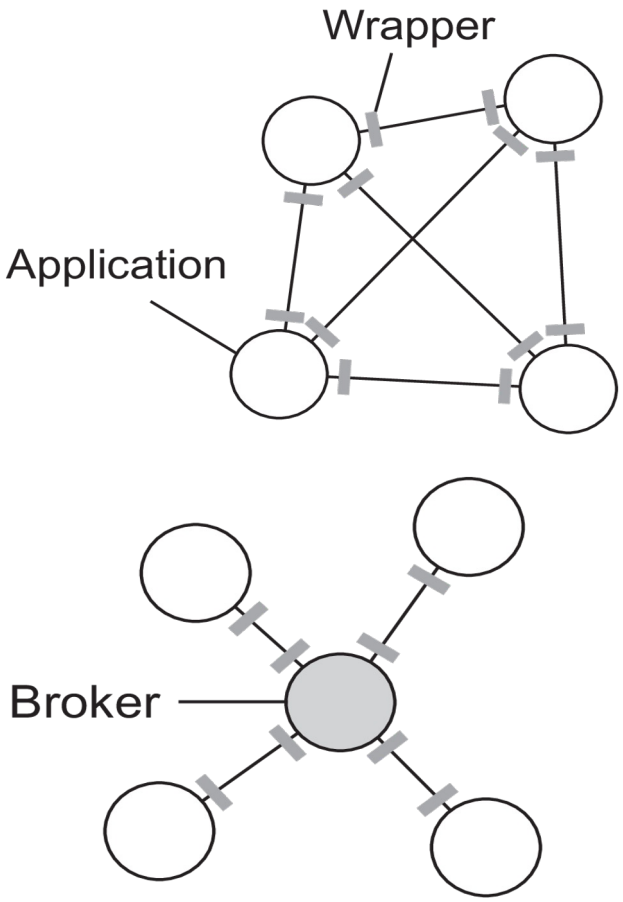
- Recursos são identificados por um esquema de identificadores único.
- Todos serviços oferecem mesma interface
- Todas mensagens são autocontidas
- Após executar uma operação o componente esquece tudo sobre quem o chamou

Operação	Descrição
POST	Cria um recurso
GET	Acede ao estado de um recurso
DELETE	Destrói um recurso
PUT	Altera um recurso, substituindo o estado

Arquitecturas *Publish-Subscribe*



Organizações de Middleware



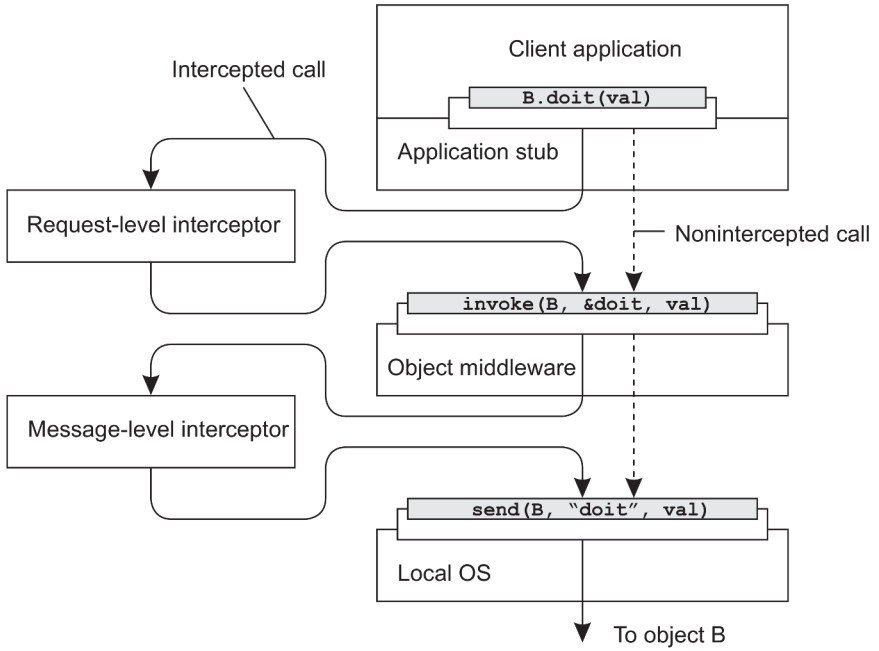
Wrapper ou **Adapter**

Por forma a evitar explosão no número de wrappers:

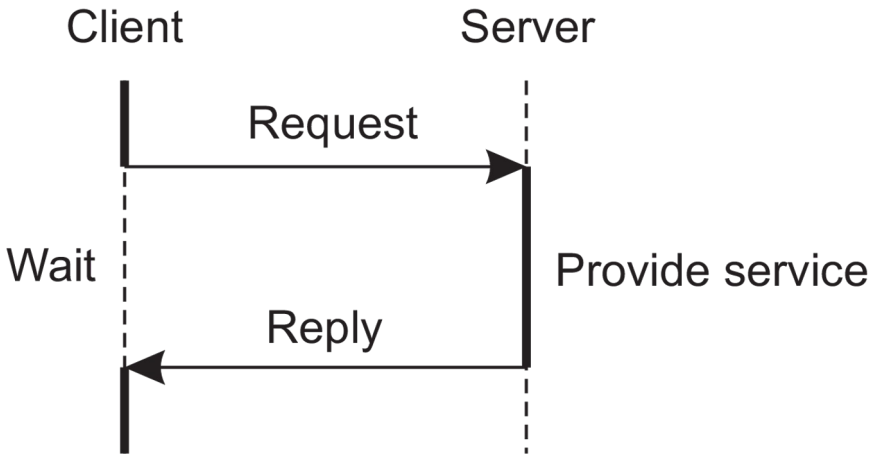
- **Broker** componente central
- Interface única com cada componente

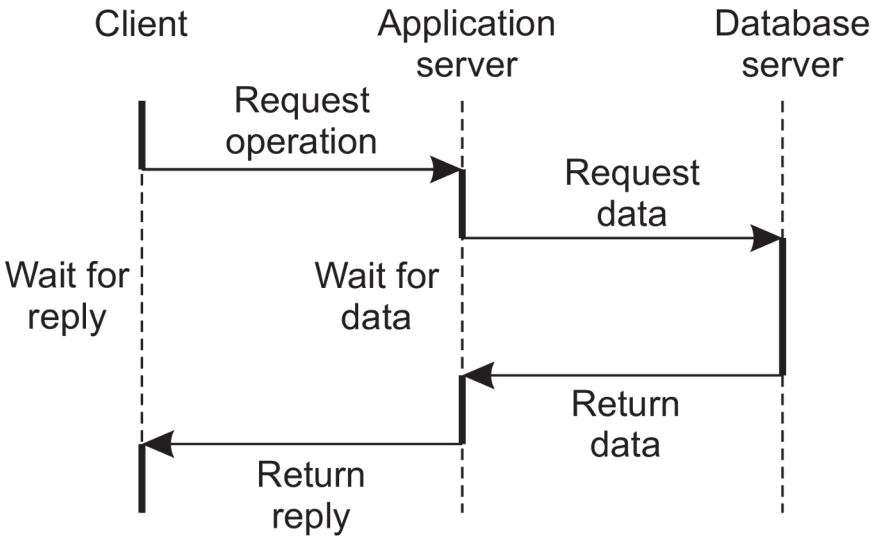
Organizações de Middleware

- **Interceptors**
- Ao objecto A é oferecido um interface local
- A chamada de A é transformada num objecto genérico pelo middleware disponível na máquina A
- Finalmente o objecto genérico é transformado numa mensagem enviada pela rede



Arquitecturas de Sistema
(Organizações centralizadas)





Cliente/Servidor

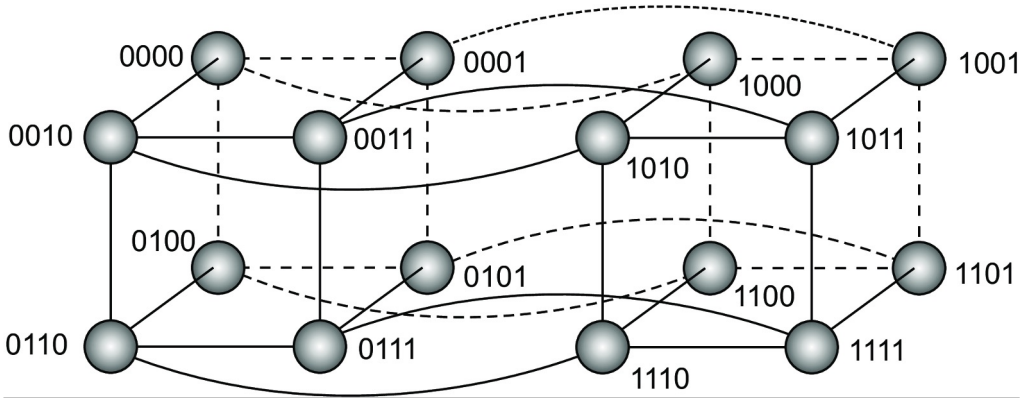
```
# server.py
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.bind(('', 8888)) # bind to port on this machine
s.listen()
(conn, addr) = s.accept() # returns new socket and addr.
while True: # forever
    data = conn.recv(1024) # receive data from client
    if not data: break # stop if client stopped
    conn.send(b"*" + data) # return sent data plus and "*"
conn.close()
```

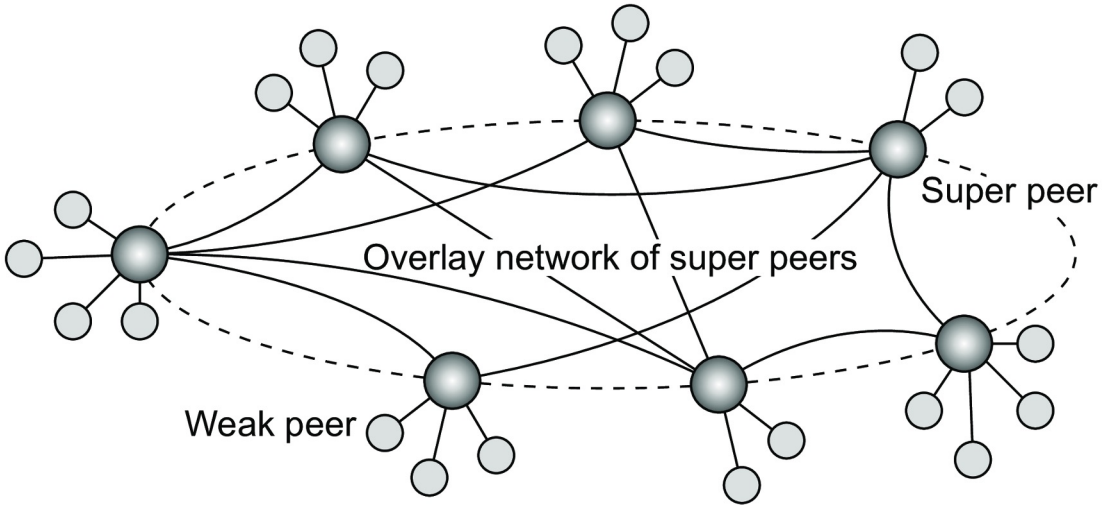
Cliente/Servidor

```
# cliente.py
from socket import *
s = socket(AF_INET, SOCK_STREAM)
s.connect(('127.0.0.1', 8888)) # connect to server (block until accepted)
s.send(b"Hello, world") # send some data
data = s.recv(1024) # receive the response
print(data) # print the result
s.close() # close the connection
```

Arquitecturas de Sistema (Organizações descentralizadas)

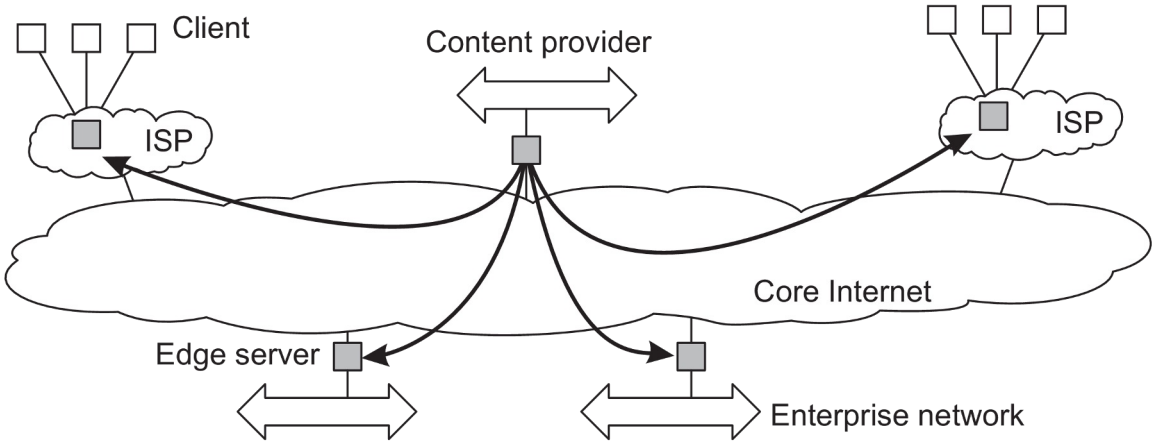
- Peer-to-peer
  - Estruturado
  - Não Estruturado
    - Flooding
    - Random walks
  - Hierárquico

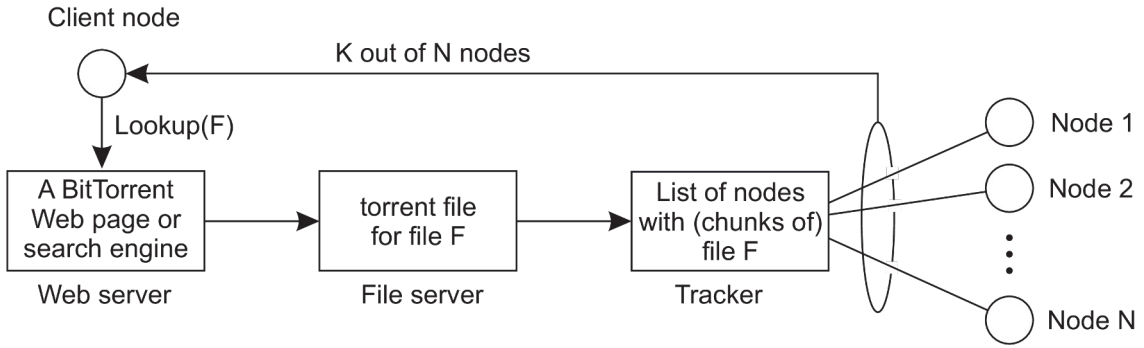




Arquitecturas de Sistema (Organizações hibridas)

- Sistemas Edge-server
  - CDN's
- Sistemas distribuição colaborativa
  - BitTorrent





universidade de aveiro

Imagens retiradas de

http://distributed-systems.net