



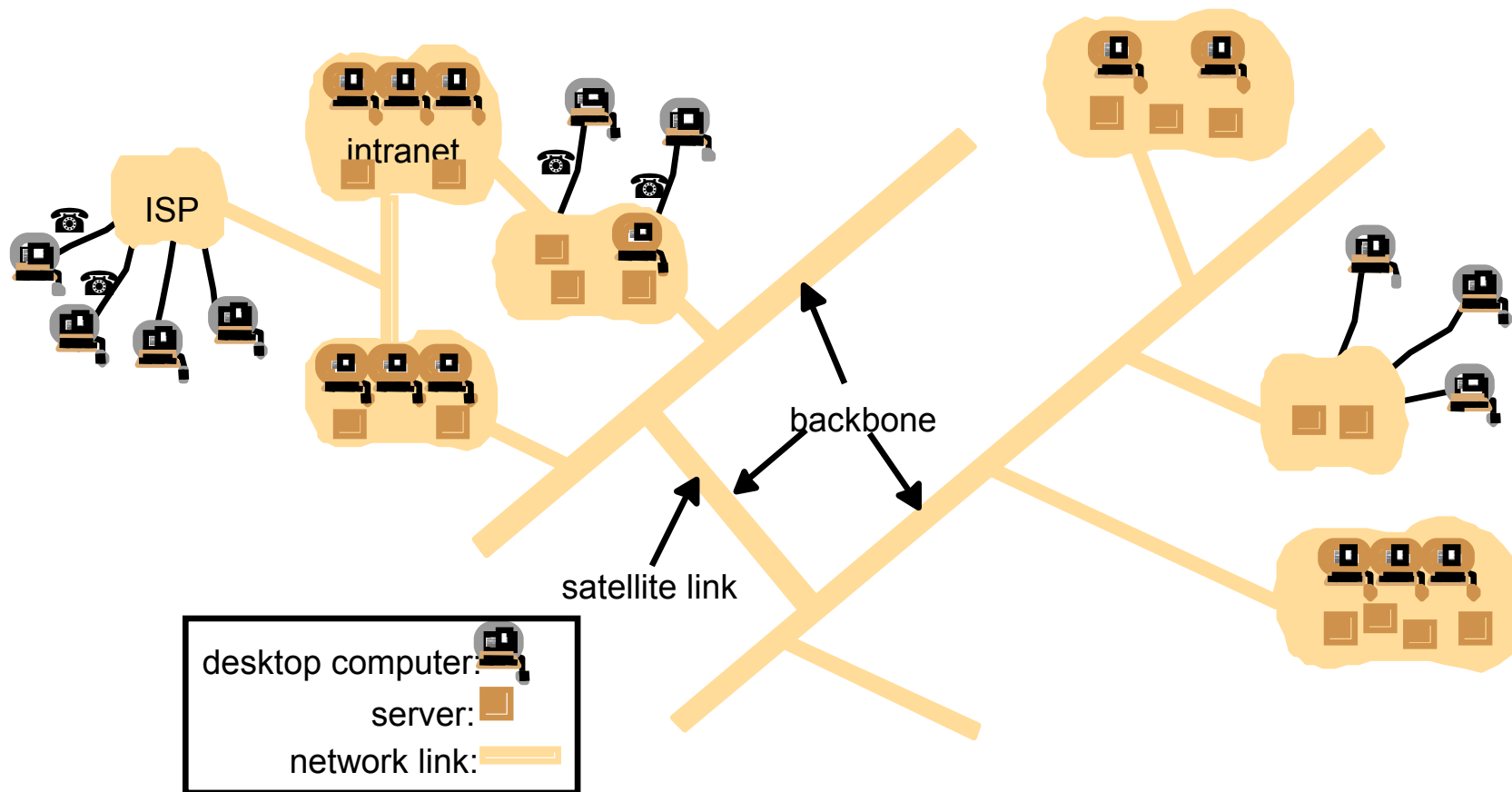
Introdução/ Arquitecturas

Computação Distribuída 2018/19

dgomes@ua.pt

1 / 31

Introdução



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)

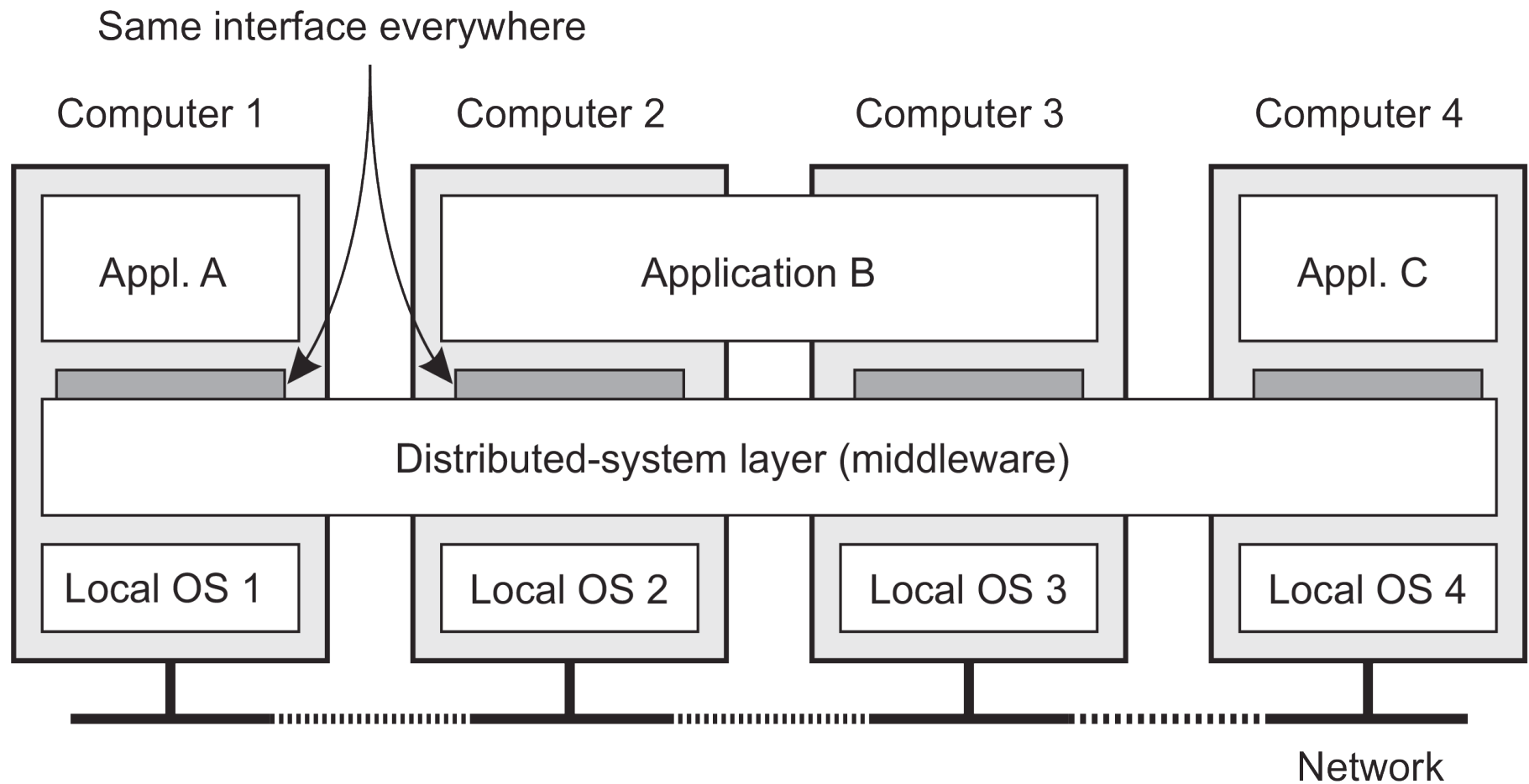
- *Um conjunto de computadores independentes que são apresentados ao utilizador como um único sistema integrado*

(DS Tanenbaum)

Características

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

Middleware



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

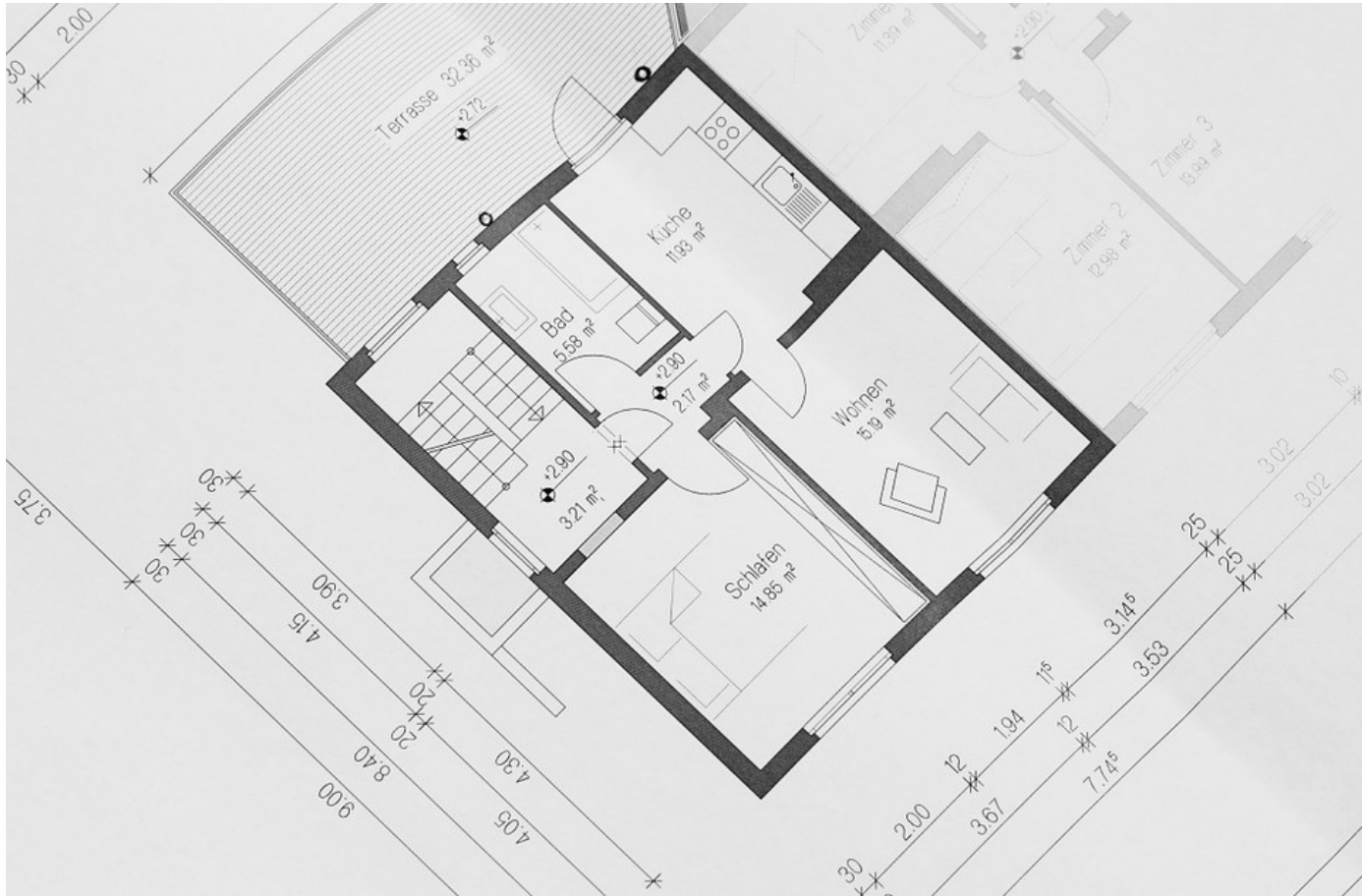
Escalabilidade

- Dimensões
 - Tamanho
 - Geográfica
 - Administrativa
- Vertical vs Horizontal

Armadilhas (Pitfalls)

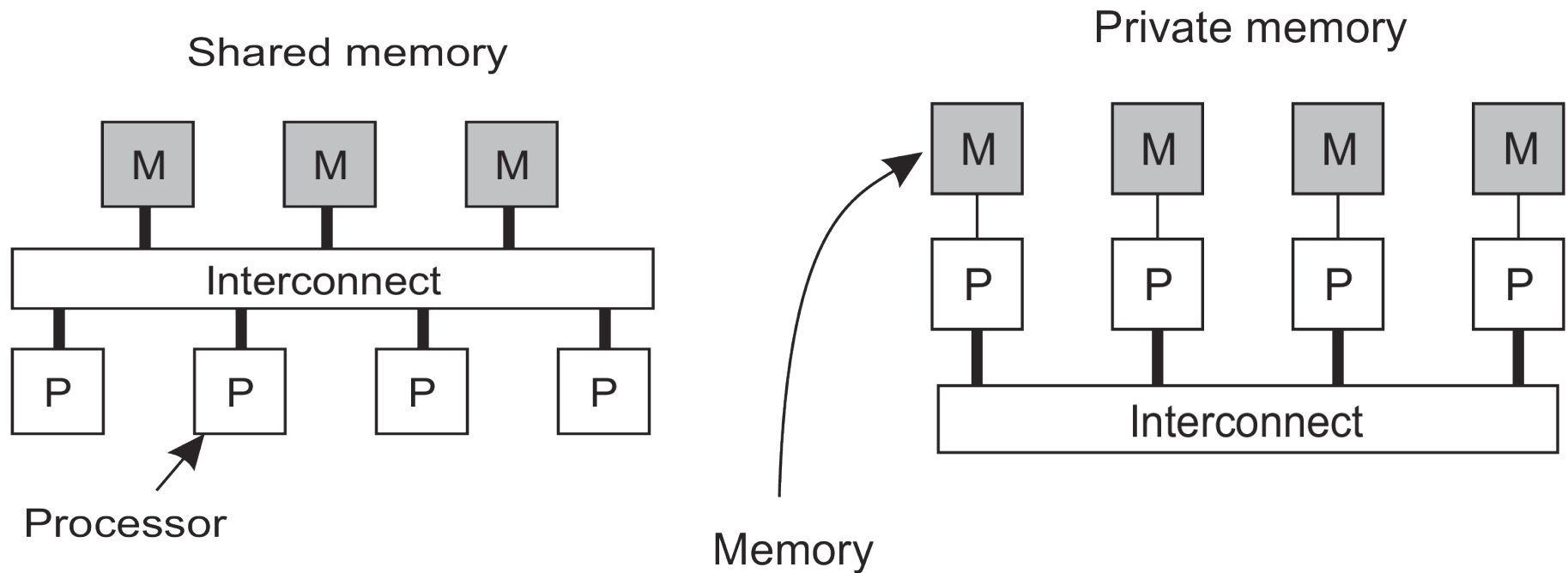
- A rede é fiável
- A rede é segura
- A rede é homogénea
- A topologia não se altera
- Latência é zero
- Largura de banda é infinita
- Custo de transporte é zero
- Existe um único administrador

100



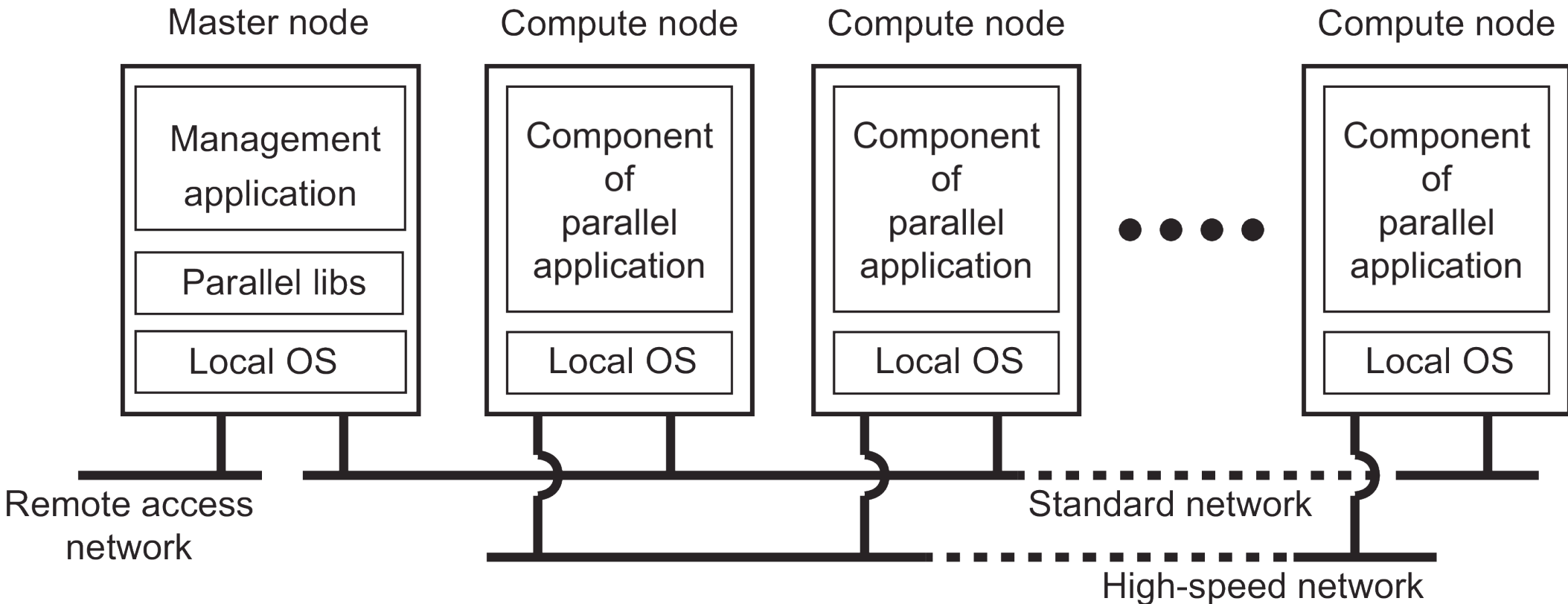
Arquitecturas de Hardware

Multiprocessador e multicore VS multicomputador



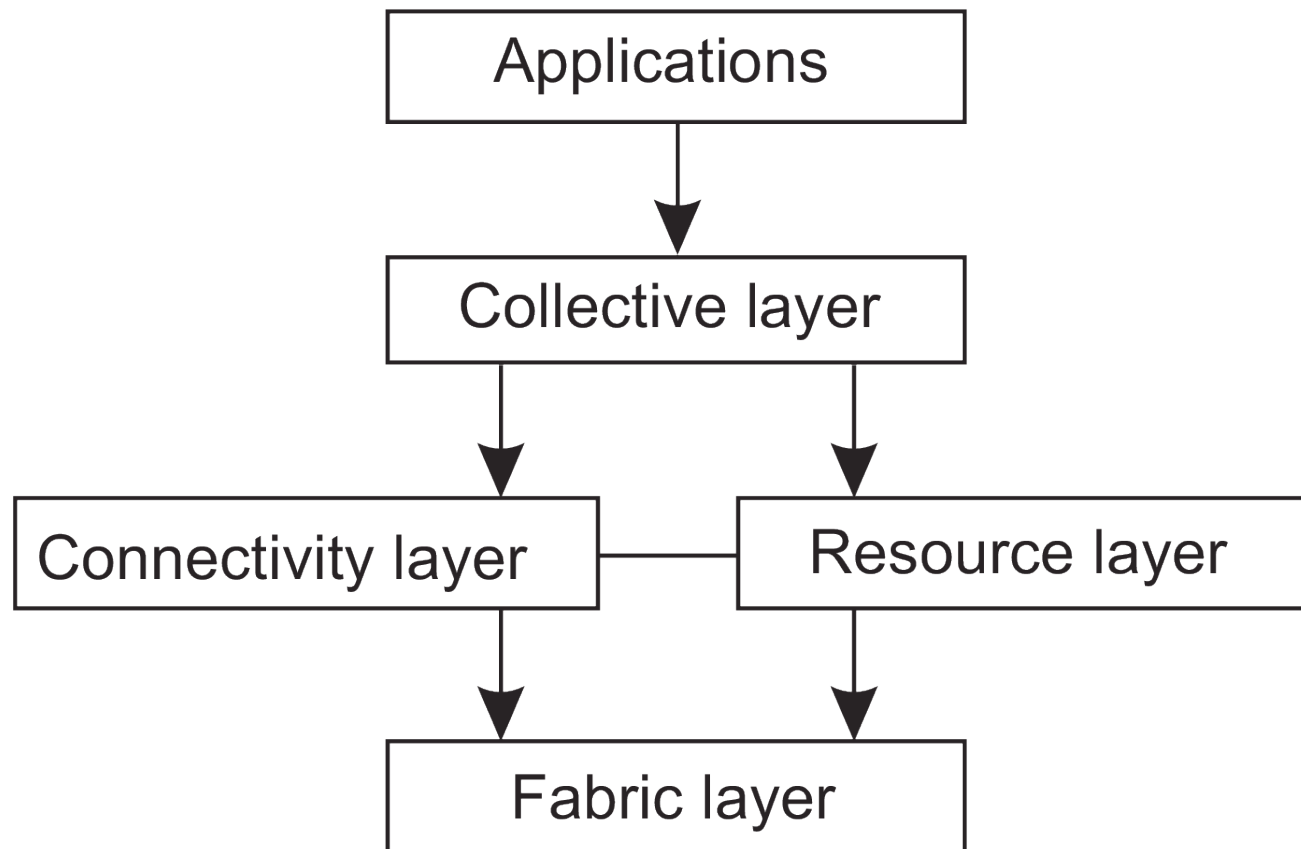
Arquitecturas

- Cluster Computing



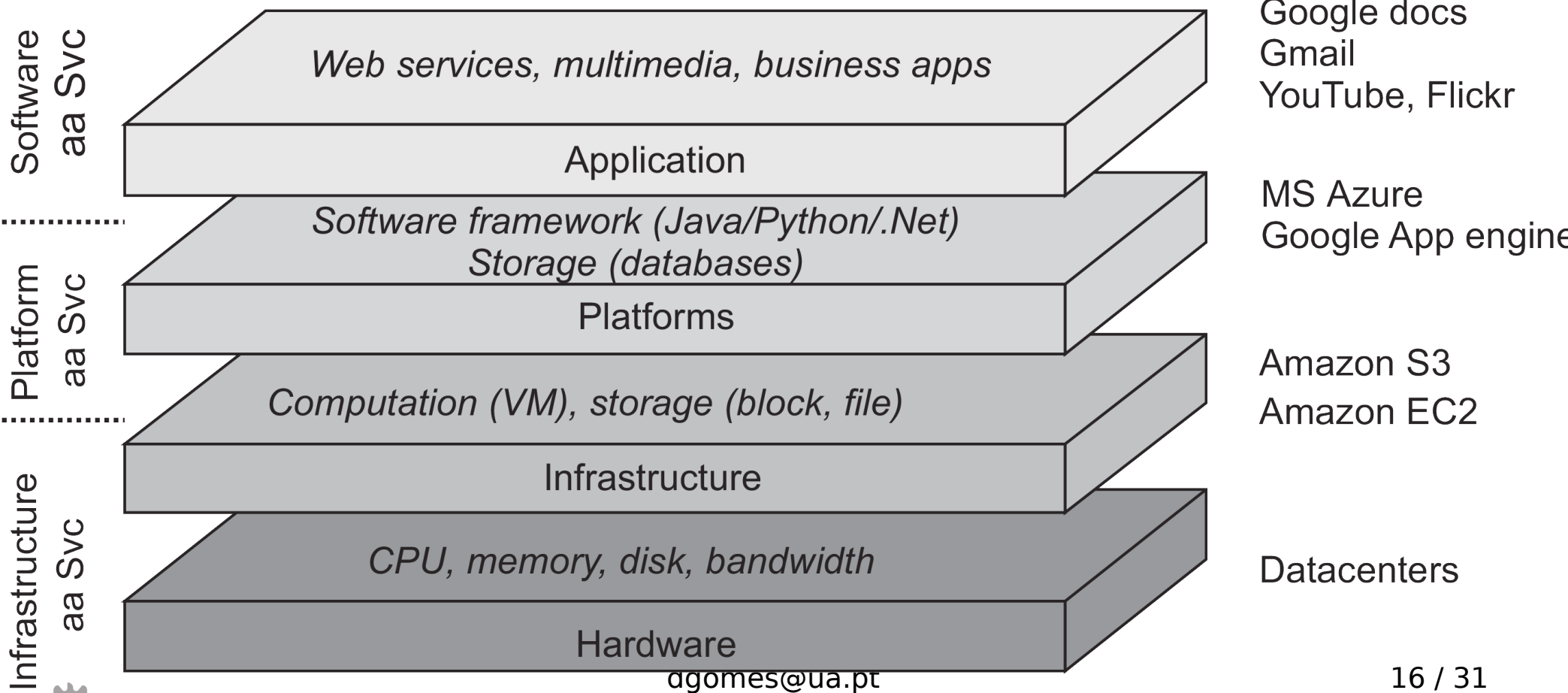
Arquitecturas

- Grid Computing



Arquitecturas

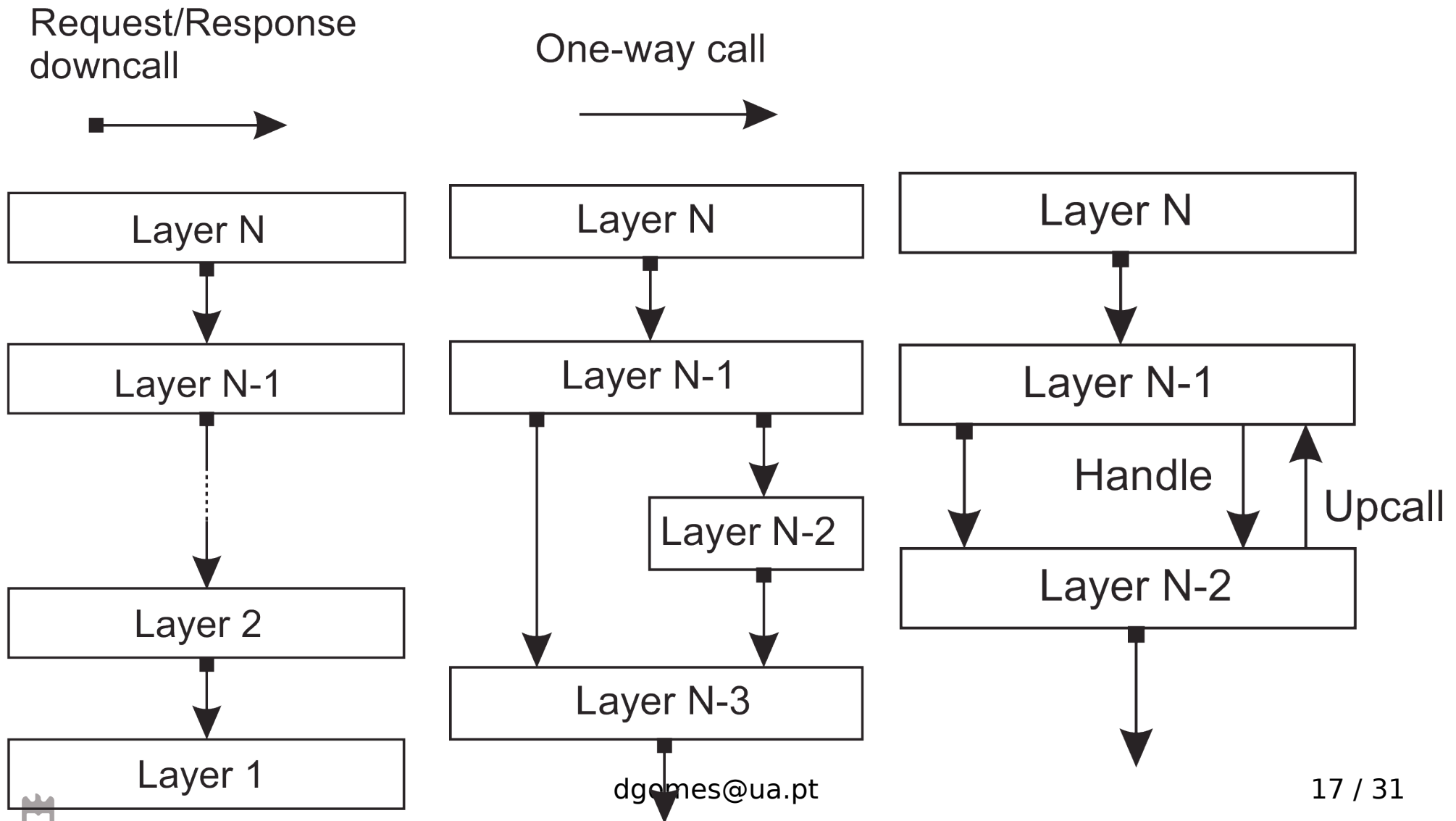
- Cloud Computing



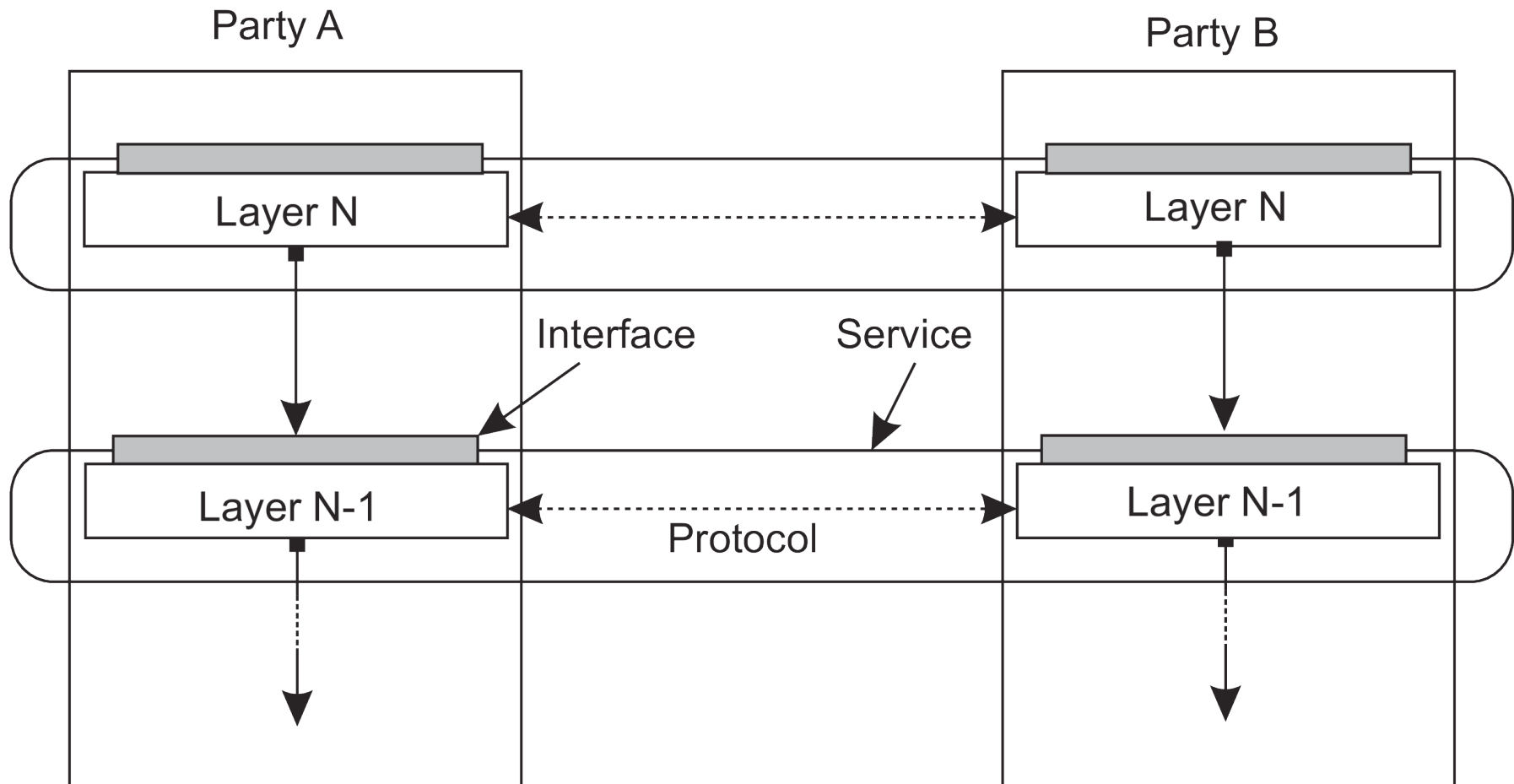
dgomes@ua.pt



Arquitetura por camadas



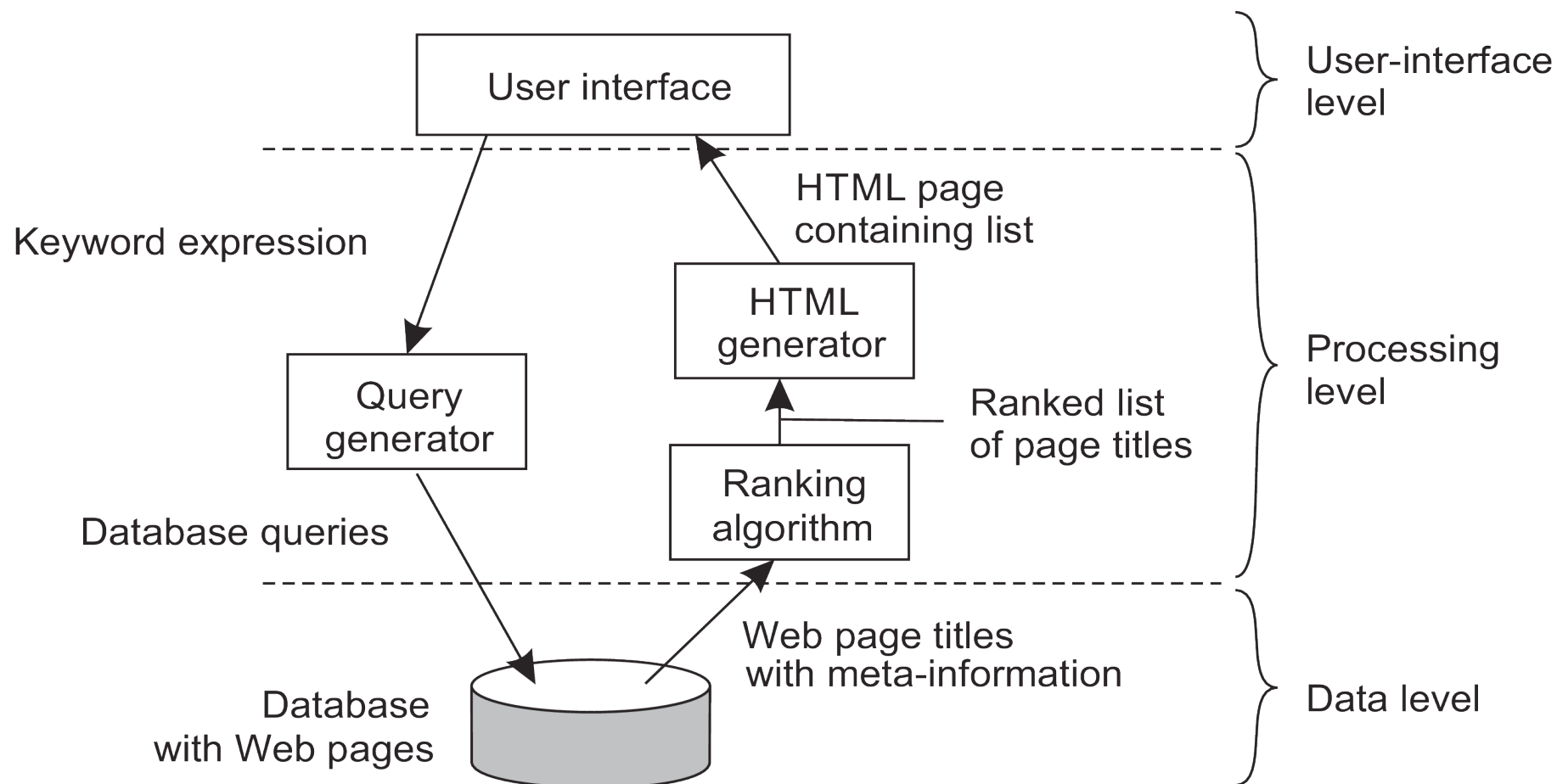
Protocolo, serviço, interface



Camadas Aplicacionais

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

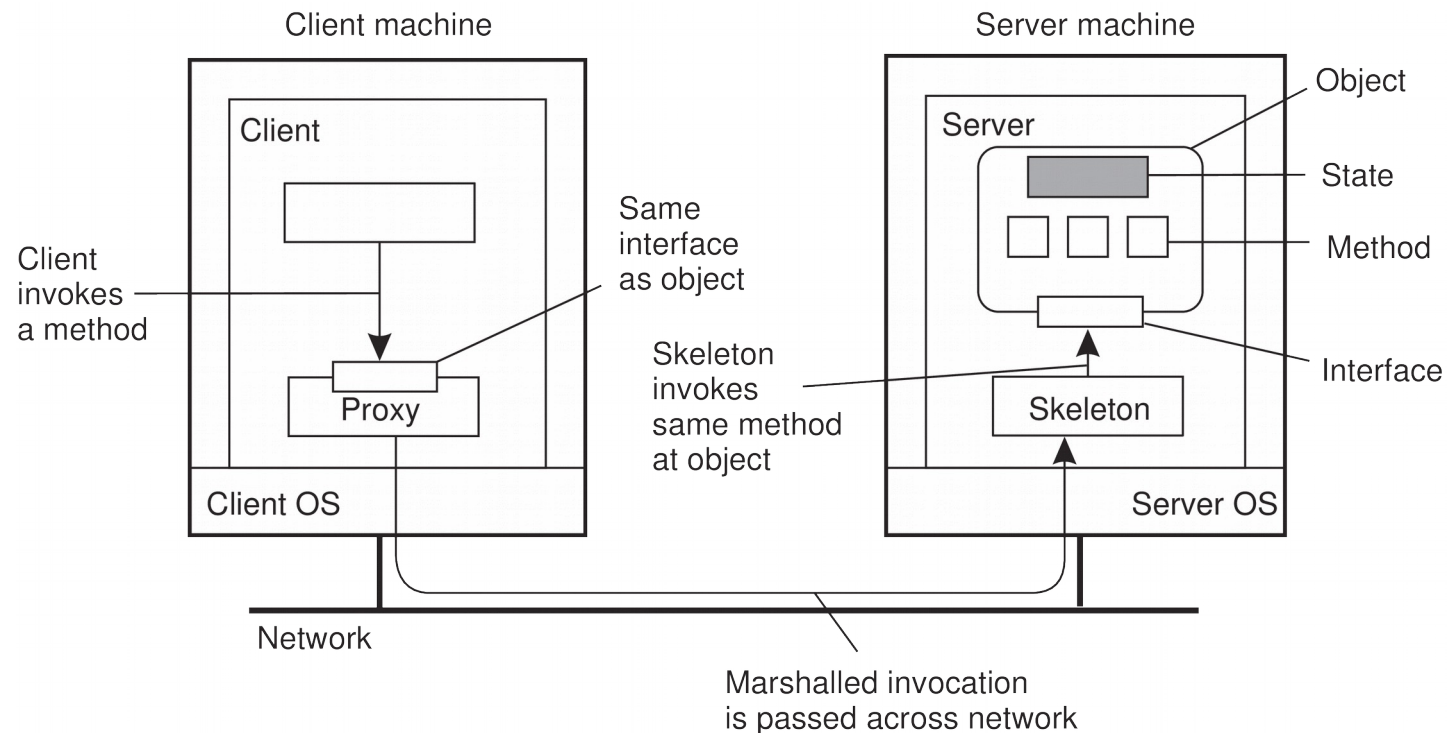
Exemplo: Motor de Pesquisa



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 distribuido, é carregada uma implementação da interface do objecto – **proxy**
- Do lado do servidor surge uma entidade complementar o **skeleton**

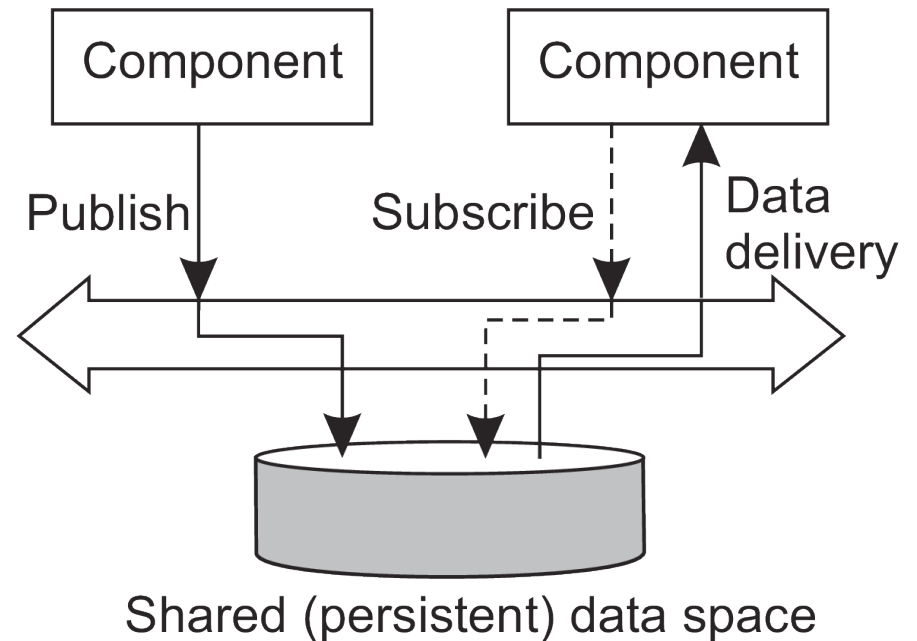
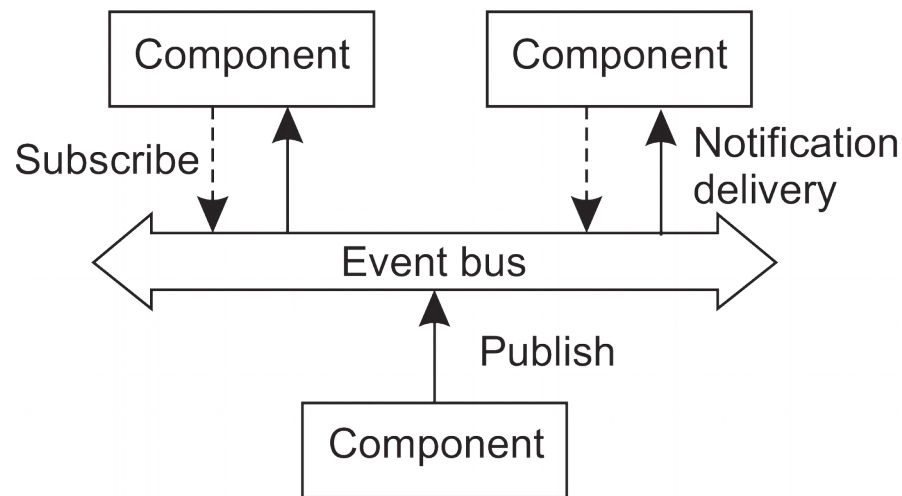


Arquitecturas baseadas em Recursos

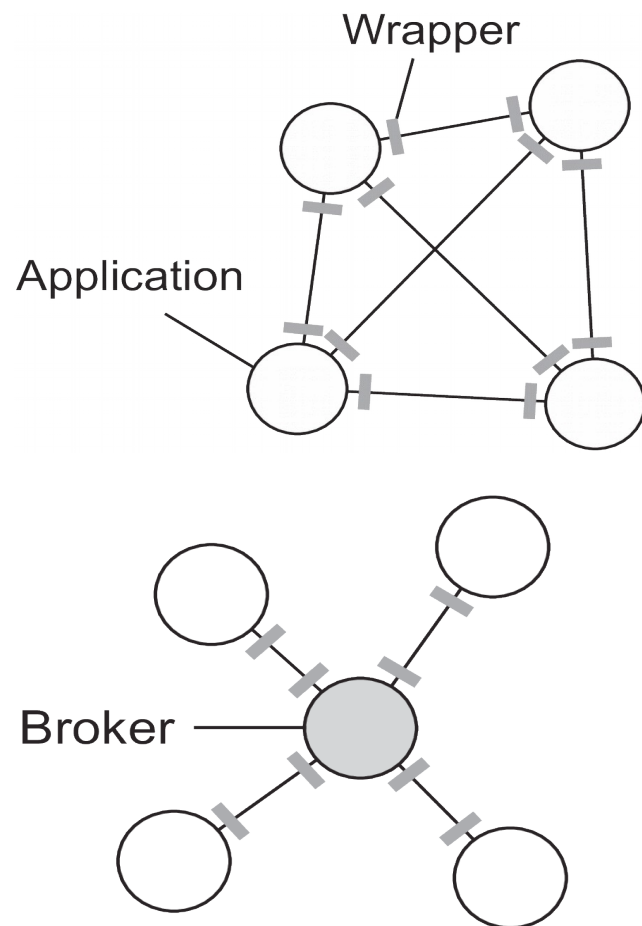
- Recursos são identificados por um esquema de identificadores único.
- Todos serviços oferecem mesma interface
- Todas mensagens são auto-contidas
- Após executar uma operação o component esquece tudo sobre quem o chamou

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

Arquitecturas *Publish-Subscribe*



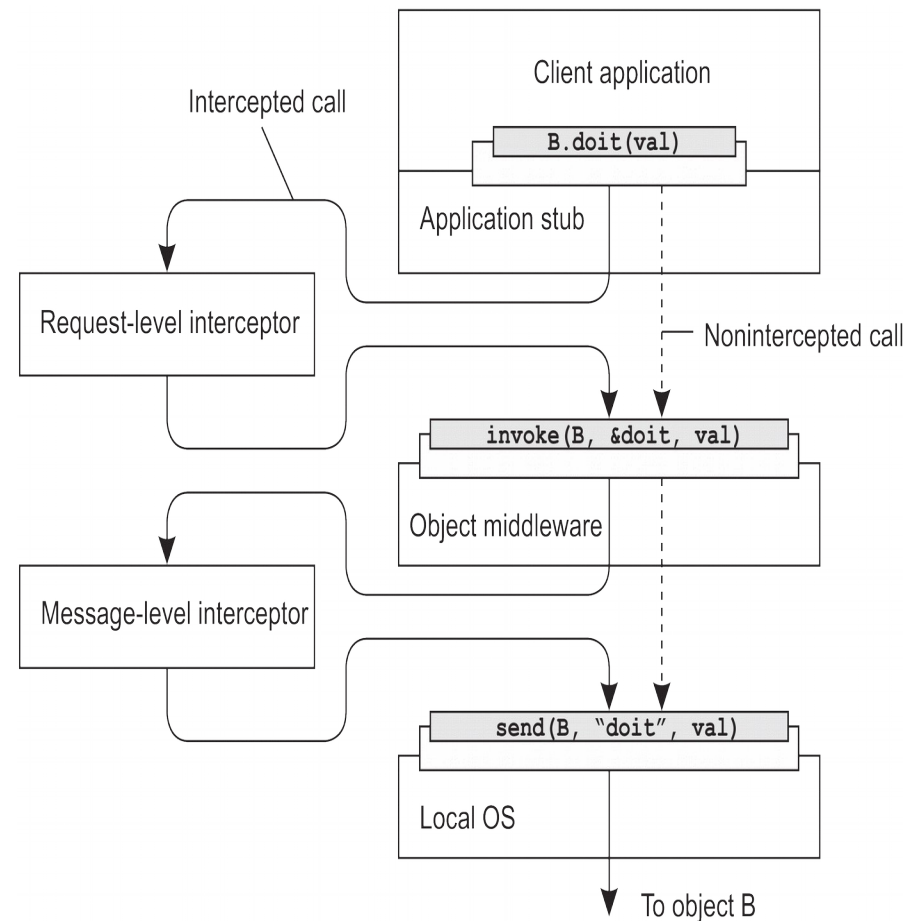
Organizações de *Middleware*



- **Wrapper** or **Adapter**
- Por forma a evitar explosão no número de *wrapper*:
 - **Broker** componente central
 - Interface única com cada componente

Organizações de *Middleware*

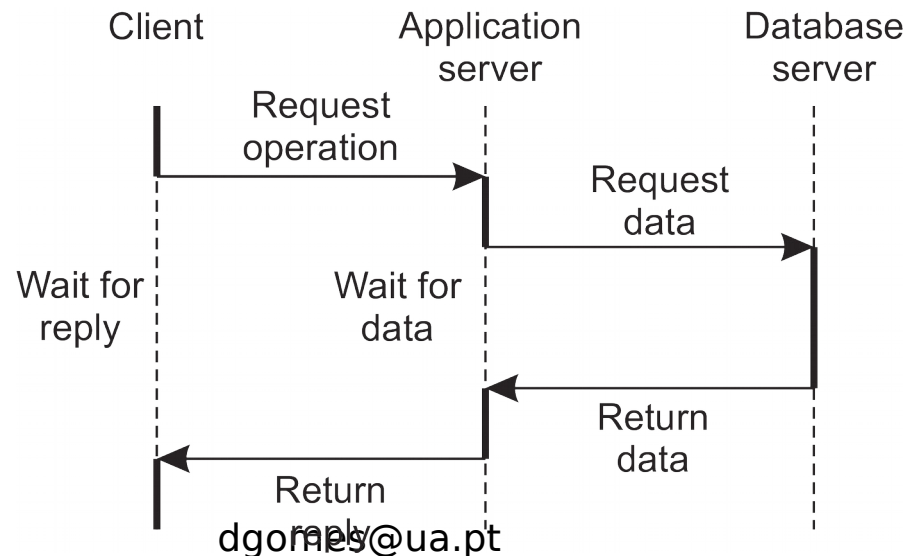
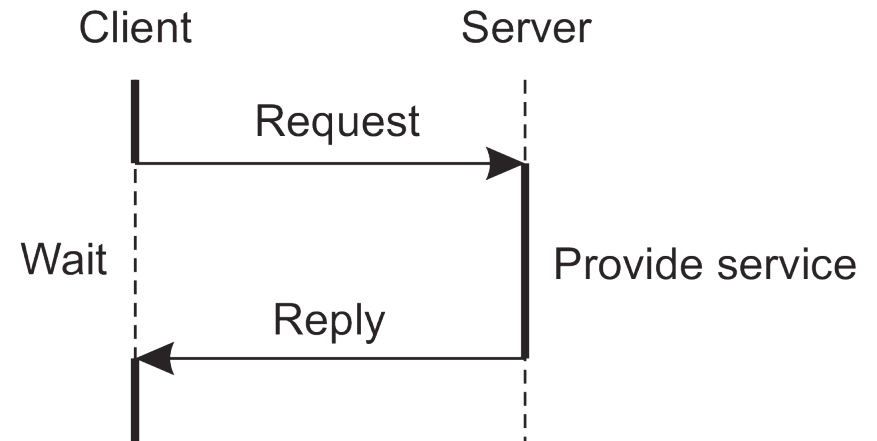
- **Interceptors**
- Ao objecto A é oferecido um interface local
- A chamada de A é transformada num objecto genérico pelo middleware disponível na máquina A
- Finalmente o objecto genérico é transformado numa mensagem enviada pela rede



Arquitecturas de Sistema

(Organizações centralizadas)

- Cliente/
Servidor



Cliente/Servidor

- Servidor

```
from socket import *  
s = socket(AF_INET, SOCK_STREAM)  
(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(str(data)+"*")    # return sent data plus an "*"   
conn.close()                    # close the connection
```

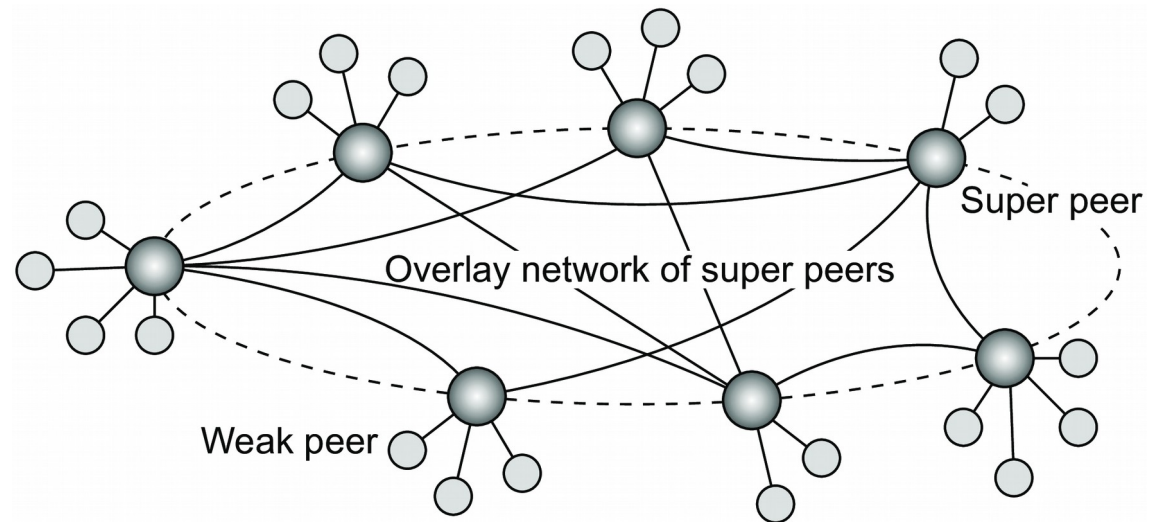
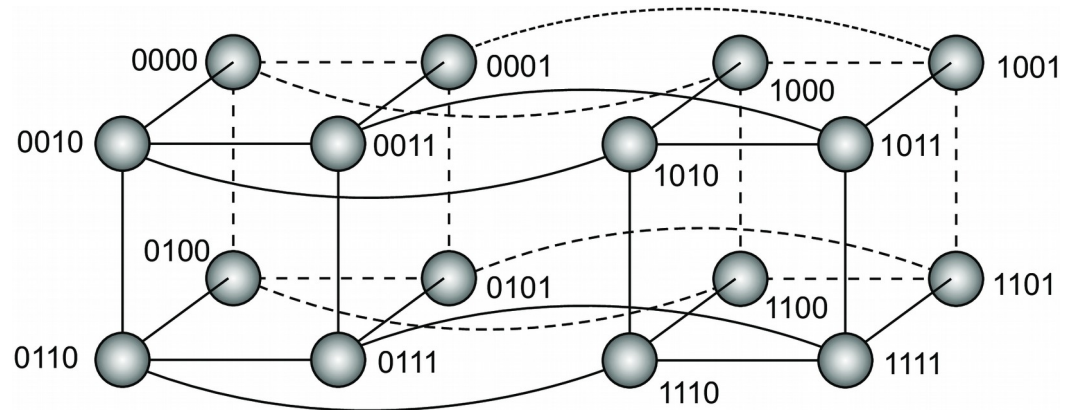

Cliente/Servidor

- Cliente

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

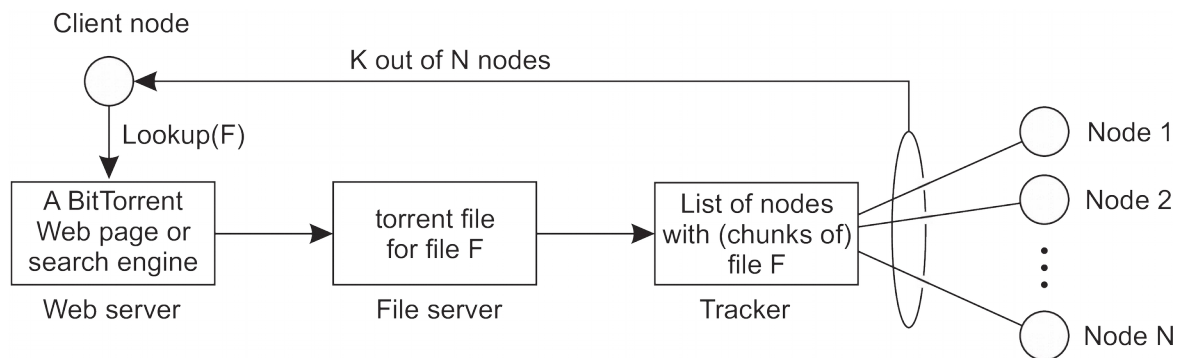
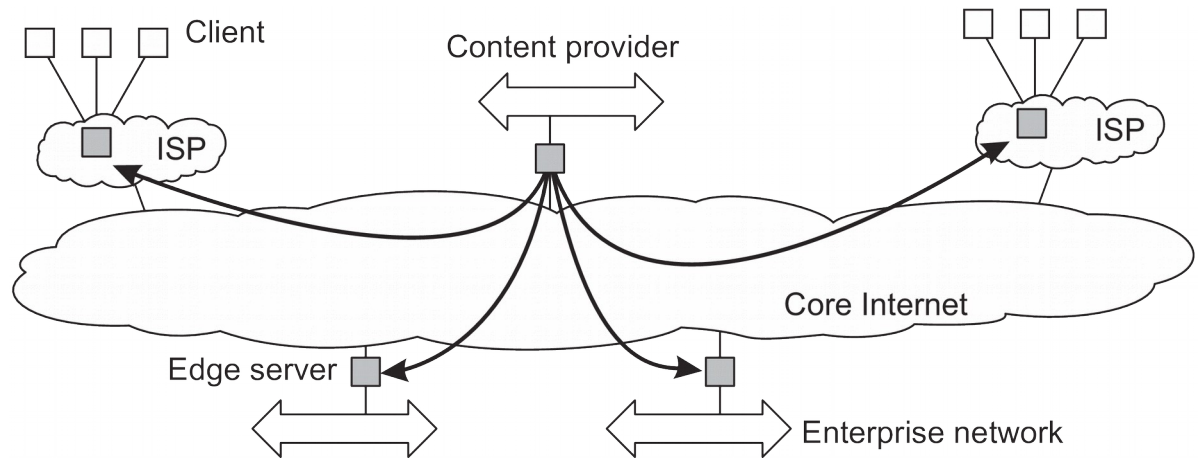

Arquitecturas de Sistema (Organizações descentralizadas)


- *Peer-to-peer*
 - Estruturado
 - Não Estruturado
 - *Flooding*
 - *Random walks*
 - Hierarquico



Arquitecturas de Sistema (Organizações híbridas)

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





Imagens retiradas de
<http://distributed-systems.net>