

# Designação

(Naming)



Eric Teitelbaum

# Nomes (Names)

- Os nomes são usados para nos referirmos a entidades num sistema distribuído.
- Para operar sobre uma entidade, é necessário aceder através de um ponto de acesso (access point).
- Os pontos de acesso são entidades designadas através de um endereço.

# Endereços (addresses)

- Endereço é um nome utilizado para nos referirmos ao ponto de acesso de uma entidade.

O que distingue

- Exemplo: endereço de um servidor é o seu IP, Protocolo e Porta

- Uma entidade pode ter múltiplos pontos de acesso e endereços

- Exemplo: uma pessoa possui múltiplos números de telefone

- Uma entidade pode mudar os seus pontos de acesso

- Exemplo: alteração de morada

Ponto de Acesso

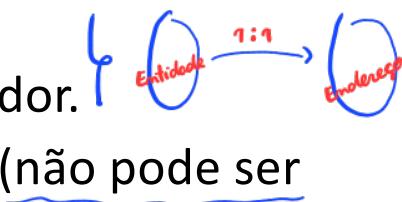
tem um ou  
mais  
endereços  
...

Podem variar ...

# Identificadores (Identifiers)

ID

- Um identificador é um nome desprovido de significado; uma *string random*.
- Os nomes puros só podem ser utilizados para comparações.
- Propriedades
  - Um identificador refere-se no máximo a uma entidade
  - Cada entidade é referida por no máximo um identificador.
  - Um identificador refere-se sempre à mesma entidade (não pode ser reutilizado)



# Sistemas de Nomes (Naming Systems)

- Um sistema de nomes mantém a relação nome-endereço e resolve nomes em endereços
  - Num Sistema distribuído, a implementação destes sistemas ocorre através de várias máquinas
- Os principais sistemas de nomes são:
  - Nomes planos (Flat Naming) *e.g., ID*
  - Nomes estruturados (Structured Naming) *e.g., /home/Documents*

# Flat Naming

- Nestes sistemas, a cada entidade é atribuído um identificador que unicamente identifica uma entidade.
  - Os identificadores são *strings* de tamanho fixo, às quais damos o nomes de **nomes planos** (flat names) ou **não estruturados**.
    - Nomes planos podem ser manipulados eficientemente por máquinas
    - Ex. endereços IP, endereços Ethernet, UUID's
- A mós (humanos) não mós  
dizem nada  
(...)
- Como mós trazem informação de onde podem ser encontrados precisamos de mecanismos...

# Broadcasting

Multicast é uma melhor solução  
♪ ♪ ♪

- Anunciar um ID, esperar uma resposta da entidade com o seu endereço actual.
  - Não escala para lá da rede local
  - Necessita que todos processos ouçam pedidos
- Exemplo:
  - ARP – Address Resolution Protocol → Data Link Layer
    - envia para a rede todo o perguntor quem tem um determinado IP e quer o MAC desse IP!!!

flat-ID → moves

Muito utilizada  
nos telemóveis

when the entity moves from A to B, it  
leaves a reference in A pointing to B

→ Problemas:

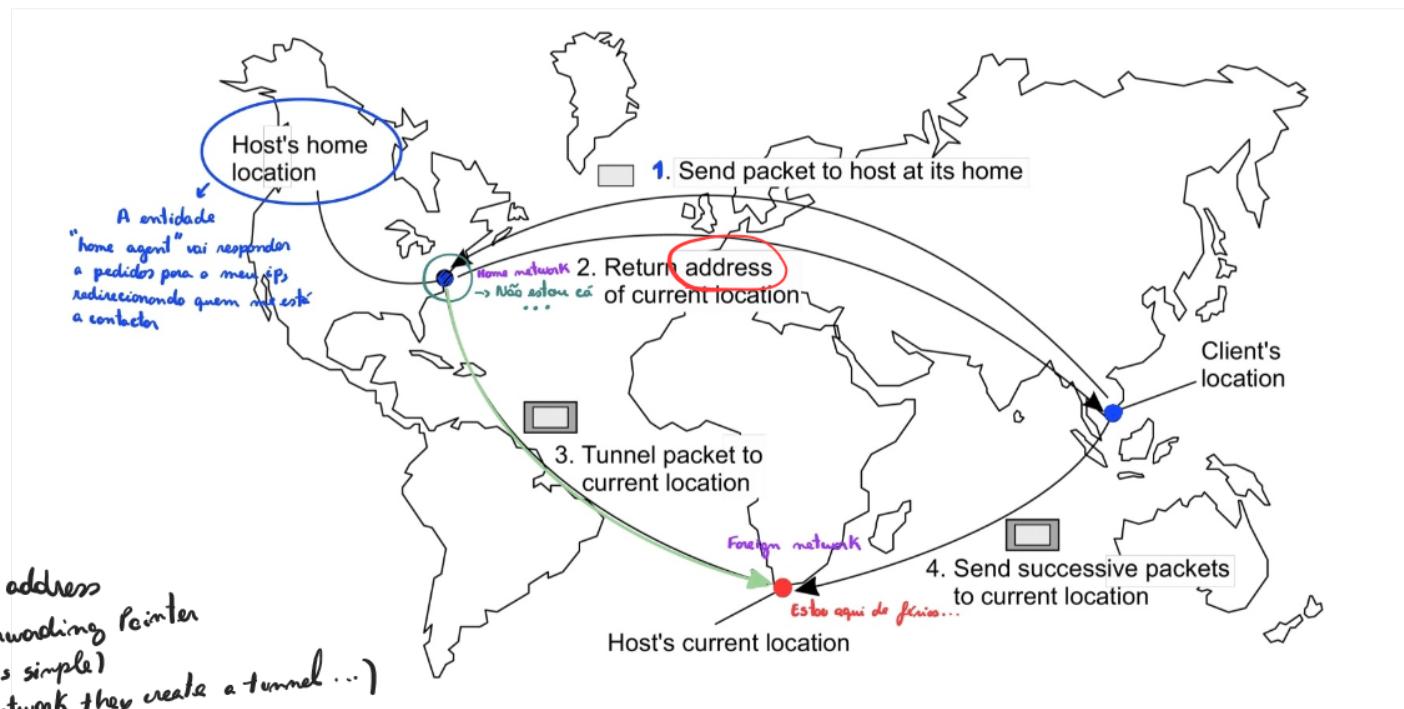
- Cadeias muito grandes
- Manter todos esses referencias
- Vulnerabilidade a falhos

⇒ Short chain

④ Robust Forwarding Pointers

# Forwarding Pointers

- Quando um entidade se move, deixar uma referência à sua nova localização.
- Exemplo:
  - MobileIP



Home-based approach:

- each mobile have an IP address
- in our home we have forwarding pointer
  - (if we are in the network is simple)
  - (if we are outside of the network they create a tunnel ...)

Escondido para a aplicação

# Distributed Hash Tables (DHT)

flat-ID → names  
 O problema é sempre fazer a correspondência  
 $k \rightarrow \text{succ}(k)$   
 Para encontrar o valor...  
 $\Rightarrow$  Com Finger Tables  $\Rightarrow O(\log n)$

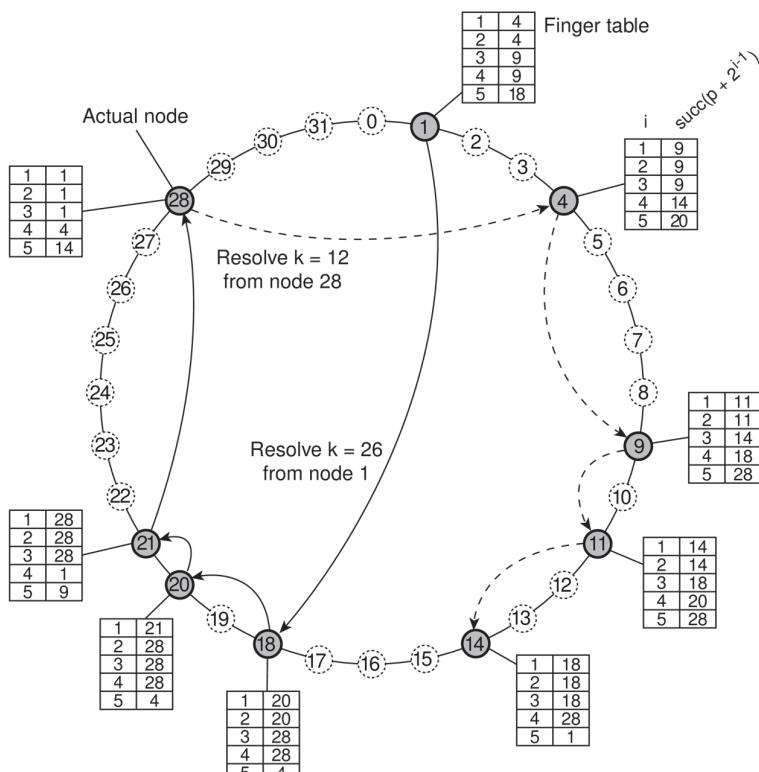
- Chord

- Cada nó tem um identificador random com m-bit's
- Cada entidade tem uma chave única com m-bit's
- A entidade com chave  $k$  fica sob a jurisdição do nó com o id menor tal que  $\text{id} \geq k$

- Cada nó p mantém uma Finger Table com máximo de m entradas
- Pesquisar a chave  $k$ , nó p encaminha o pedido para o nó com índice  $j$  que satisfaça:

$$q = FT[j] \leq k < FT[j+1]$$

↑  
 O problema é atualizar os Finger Tables...



• Como os endereços são resolvidos?

→ Percorrer a rede e com base nos id's atribuir um nó...

- No chord um request pode percorrer caminhos não ótimos....  
→ Para melhorar isso temos de criar a DHT com atenções geográficas

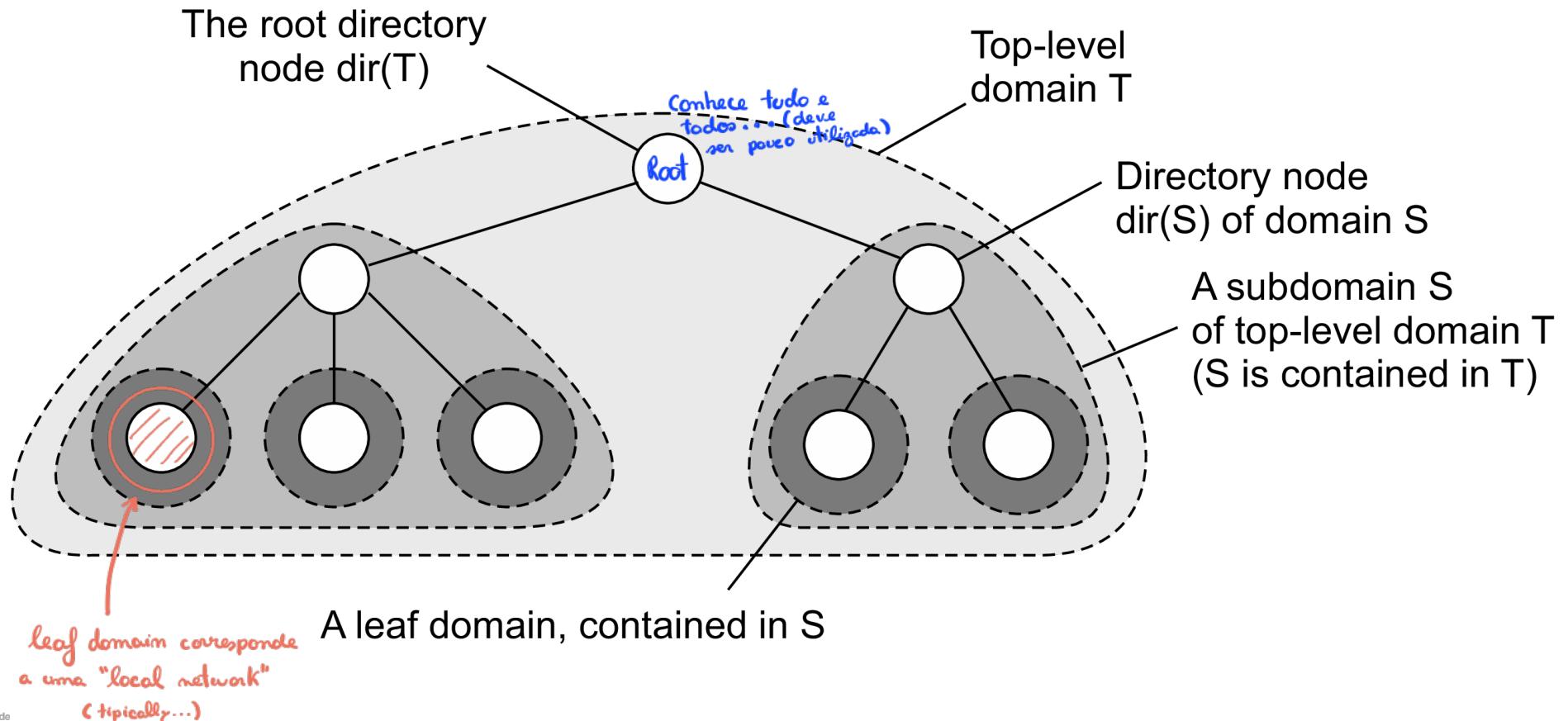
## Explorar a proximidade de rede

→ Num Sistema Distribuído organiza sempre com base em latência...

A distribuição da densidade populacional  
não é uniforme  
*(Probabilidade)*

- Atribuição de nós com base na topologia: Quando se atribui um ID a um nó, assegurar que nós perto no domínio do ID estão também perto na topologia da rede.  
*Isto no caso prático seria MUITO conveniente!!!*  
*Pode não ser geográfico... (latência)*
- Roteamento de proximidade: Manter mais do que um sucessor, e encaminhar para o mais perto.  
*Não teremos de estor SEMPRE a atualizar os FT e os Tabelos*
- Selecção do vizinho por proximidade: Quando existe uma escolha de qual será o vizinho, escolher o mais próximo.  
*⇒ Ordenar os tabelos de roteamento por latência*

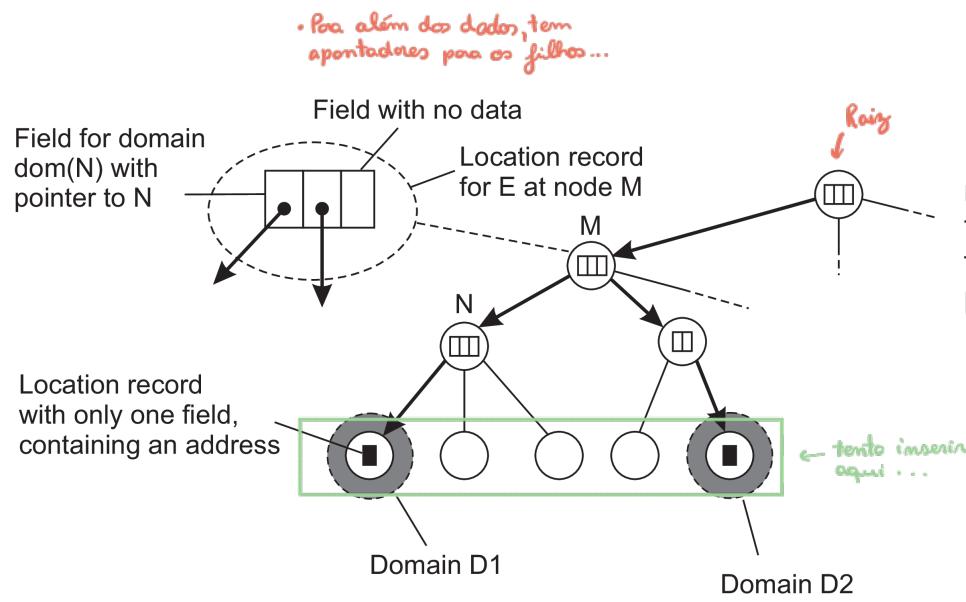
# Hierarchical Location Services (HLS)



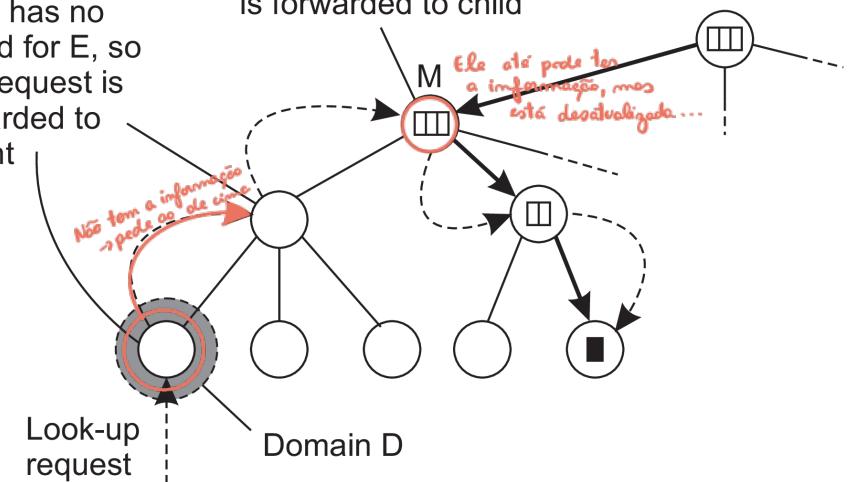
# HLS: Organização da árvore e Pesquisa

→ Faço TUDO nos folhos...

Bottom - Up

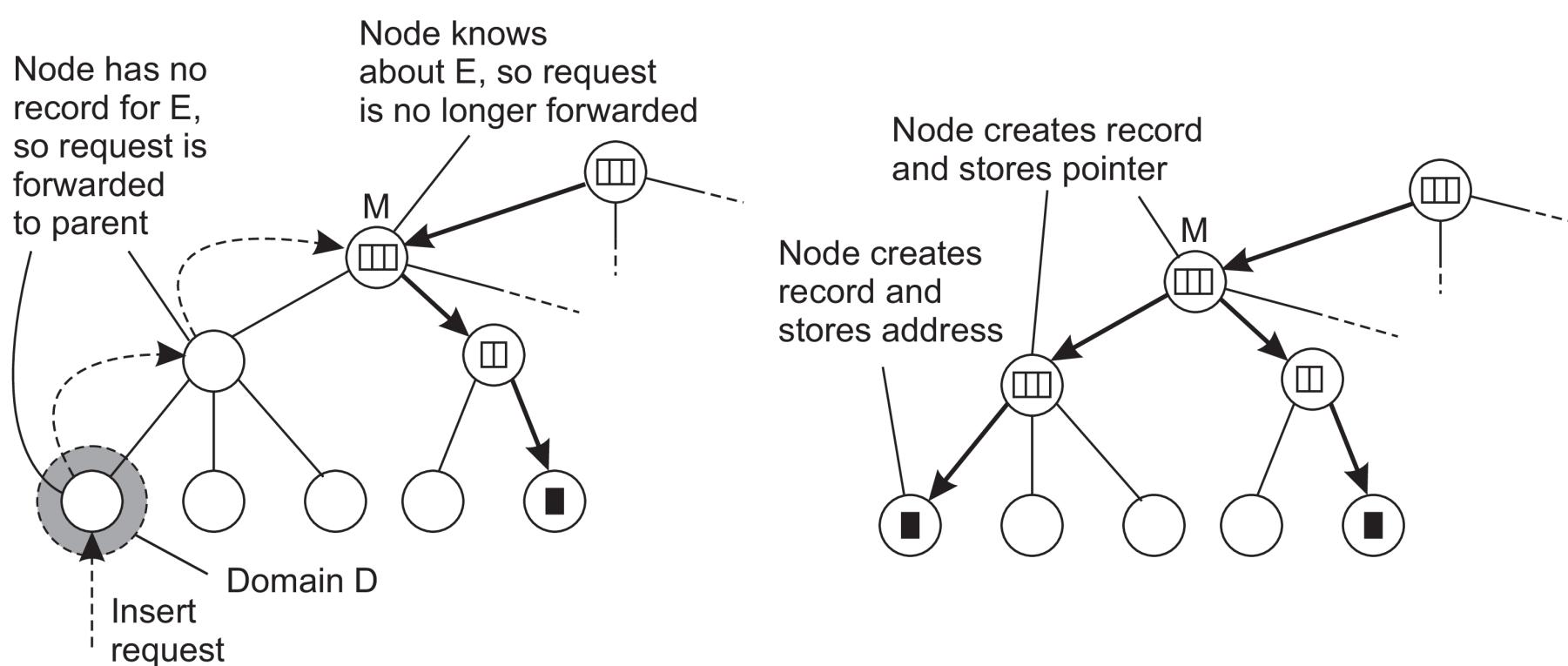


Node has no record for E, so that request is forwarded to parent



Aqui, só pesquisas a partir dos folhos

# HLS: Inserção



- File name
- Hosting name

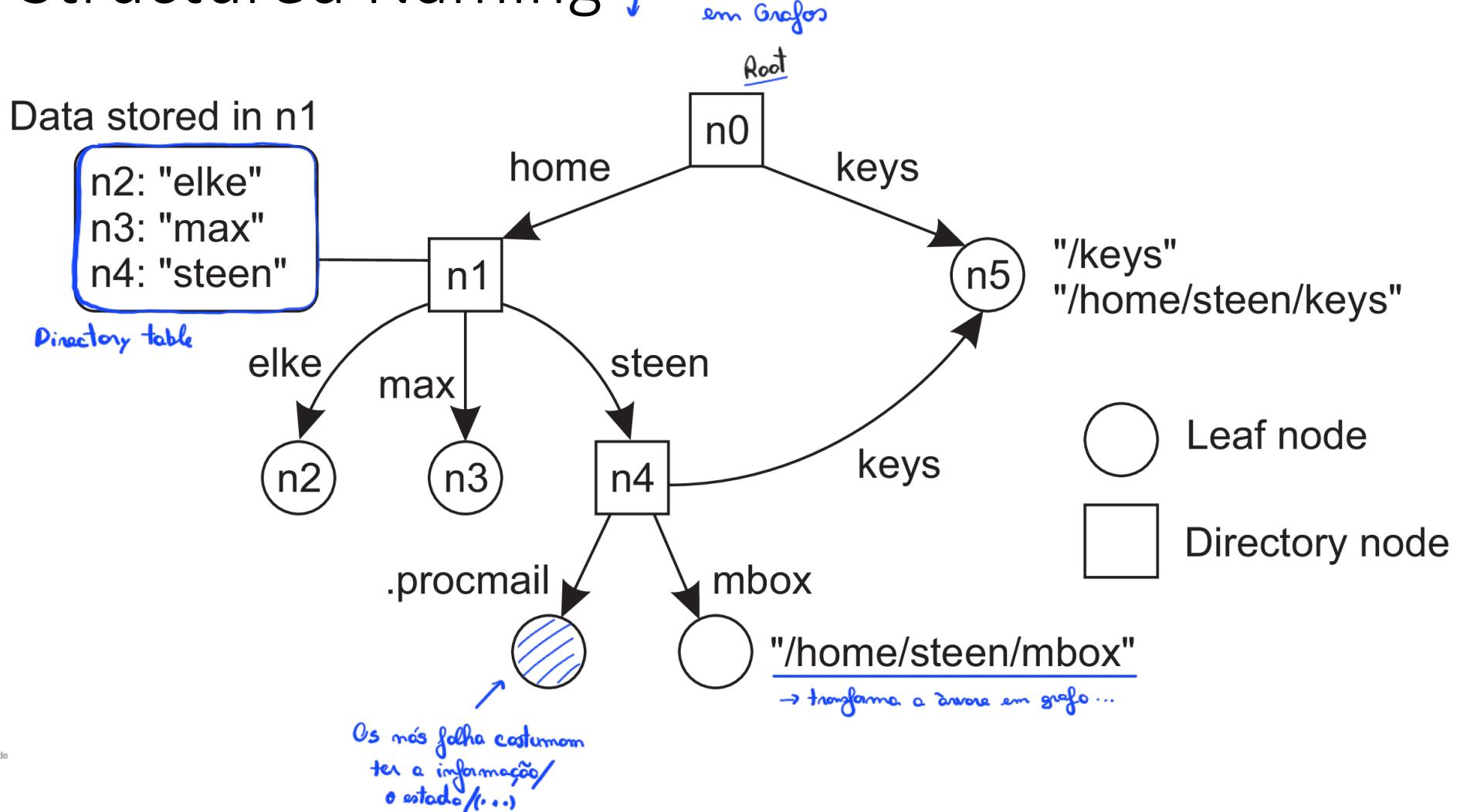
# Nomes estruturados (Structured Naming)

→  Fácil para humanos

- Os nomes planos são dificeis de fixar para os humanos.
- Os nomes estruturados são compostos por diversas partes, estas são human-friendly
  - Ex: nomes de ficheiros, nomes de servidores na internet, nomes DNS...

/home/Documentos/ ...

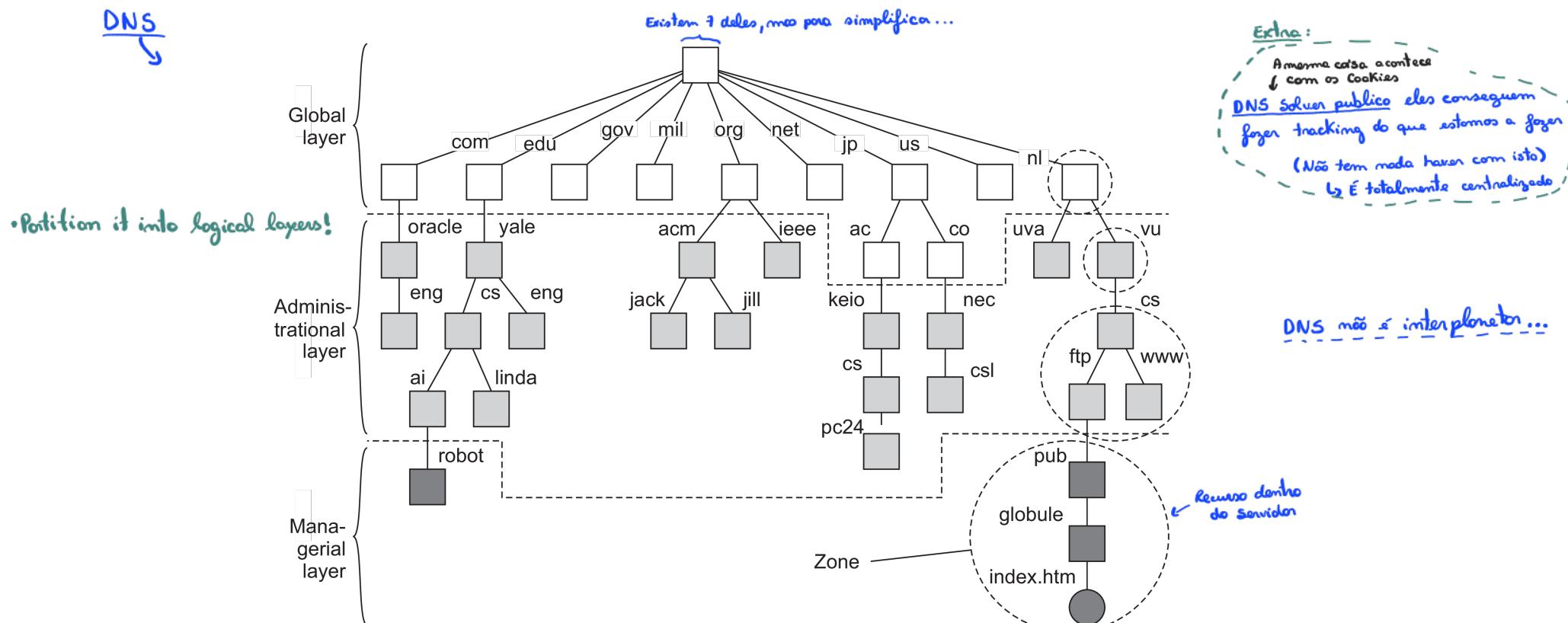
# Structured Naming



"Coração" do Naming Service



# Implementação de um Name Space



# Requisitos de um Name Space

→ Nacional (.pt/.es/...)  
 → Manager (.ua/.ist/...)

Era essa, ver:

ROOT DNS → Preciso com o...  
 → nslookup

Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few → Poucos ...	Many	Vast numbers
Responsiveness to lookups	Seconds In this is noisy domain...	Miliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None → Não temos ONS em cesa...
Is cliente-side caching applied?	Yes	Yes	Sometimes

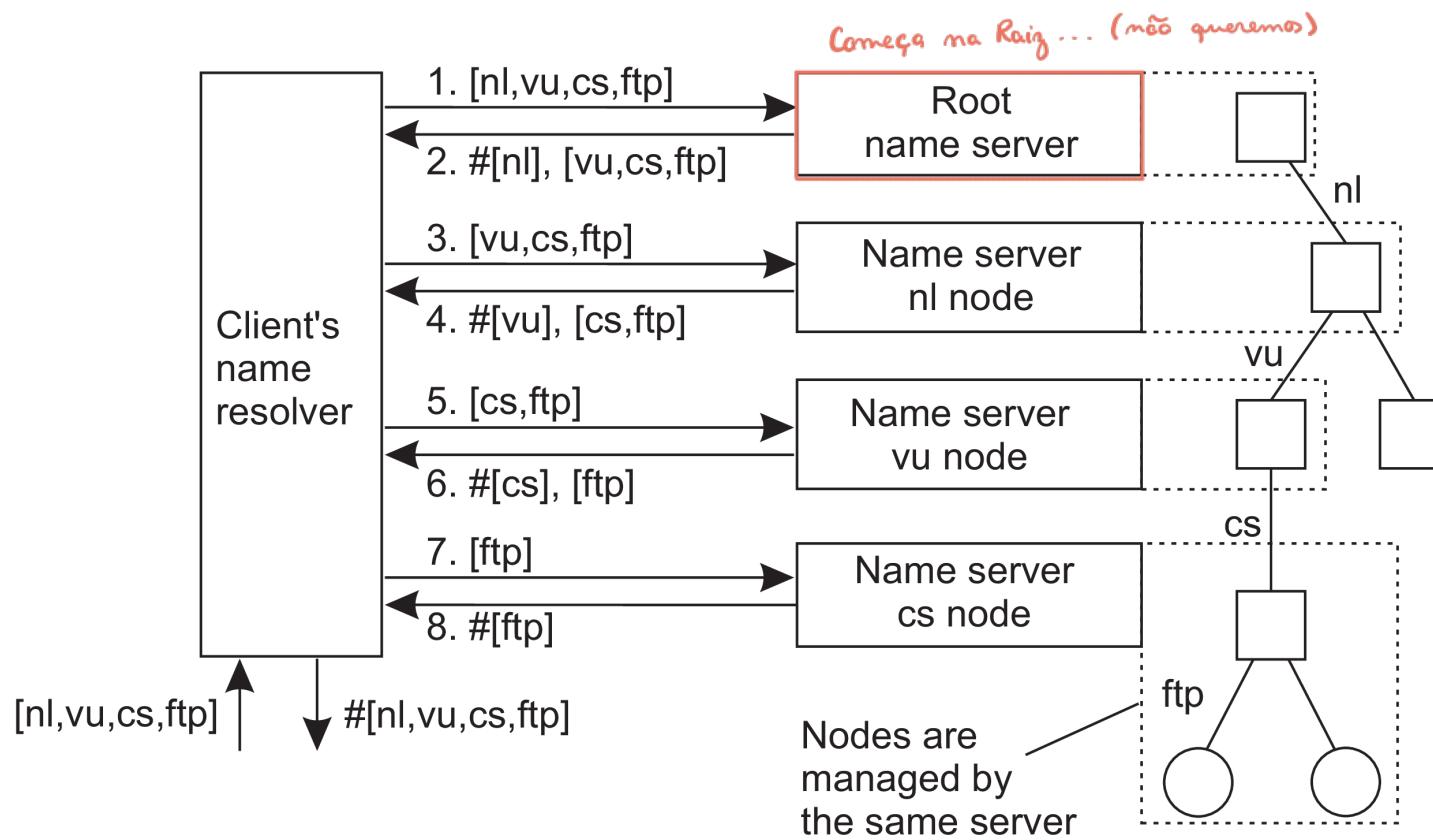
editor... →

São os layers mais difíceis de implementar

Problemas:

- Replicação e caching
  - Performance e disponibilidade
- { Pode criar problemas ... de consistência}

# Resolução de nomes iterativa

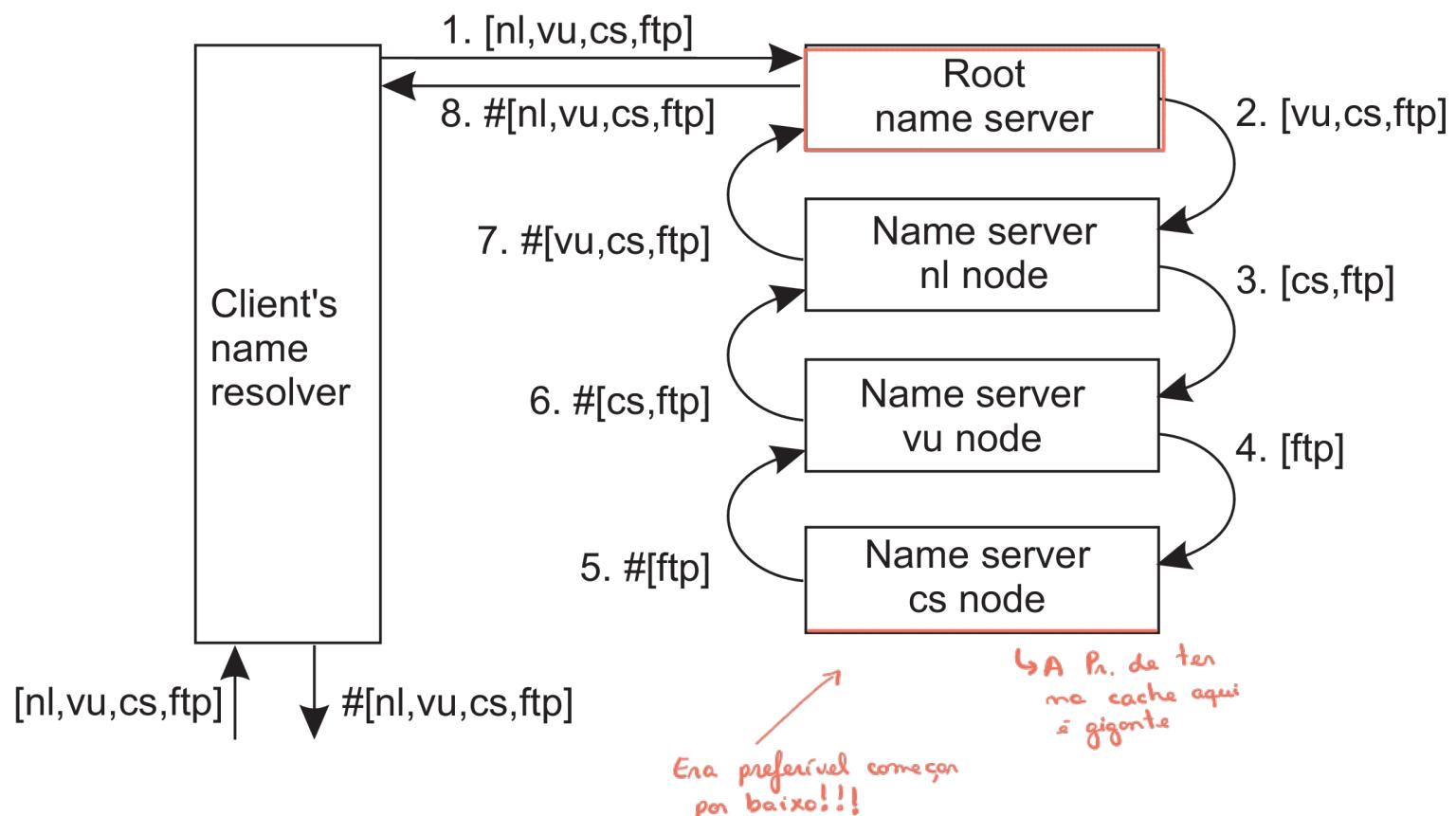


# Resolução de nomes recursiva

Vantagens:  
• Caching  
• Speed

Aqui colocamos a computação do endo dos servidores

No layer Global  
Costumom  
Superior  
openos iterativa...  
...



# Internet Domain Name System (DNS)

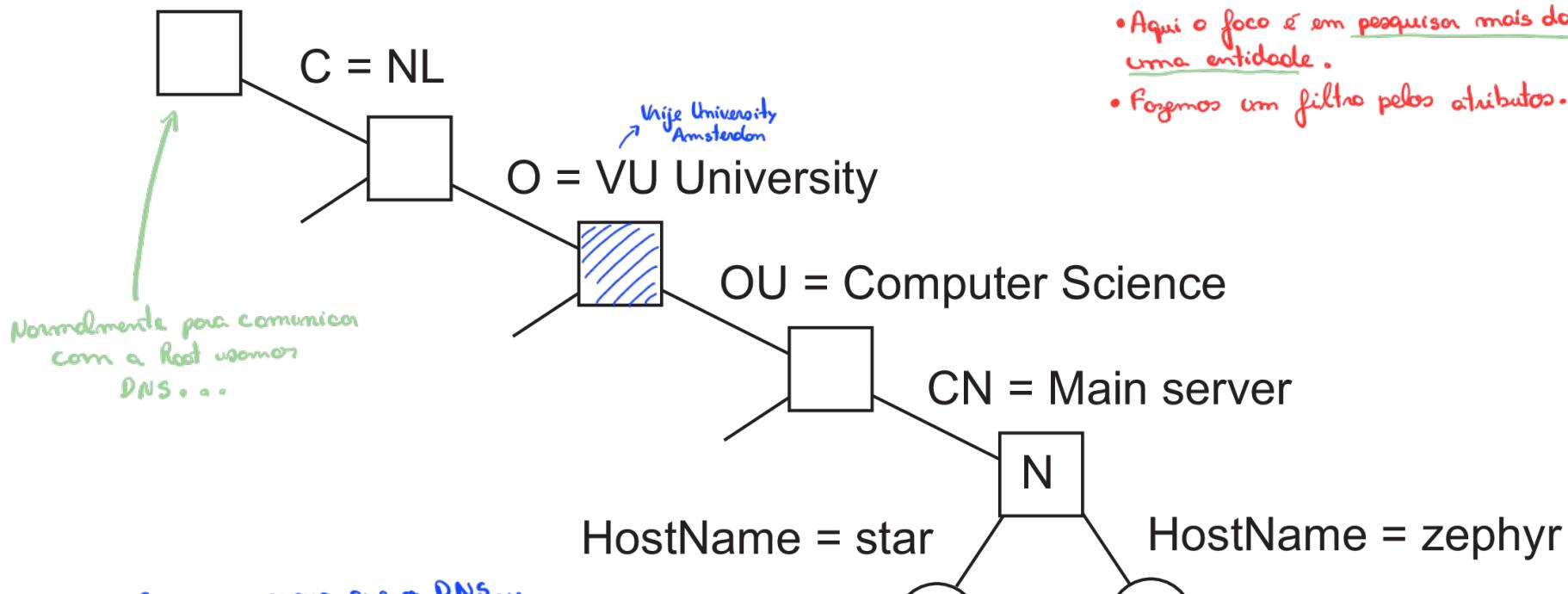
		<i>DNS transforma nomes em serviços!</i>
SOA	Zone	Informação sobre a zona representada
A	Host	Endereço IP do host que o nó representa
MX	Domain	Servidor de eMail para o nó
SRV	Domain	Servidor que lida com um dado serviço
NS	Zone	Servidor de nomes para a zona representada
CNAME	Node	Apontador simbólico
PTR	Host	Nome canónico de um host
HINFO	Host	Informação sobre o host
TXT	Any Kind	Qualquer Informação considerada útil

Faz outros coisas... → SOA  
 Tem o IP... → A  
 Escalabilidade → MX  
 Generalização do MX, para todo o tipo de serviços → SRV  
 Servidores para os domínios em baixo... → NS  
 Endereço do servidor que processa o Email mail.ua.pt → NS  
 Quem eu sou? (protocolo DNS está sempre a usar em coisa...) → TXT  
 → Tudo... → Informações de autenticação...

## Attribute-based naming - LDAP

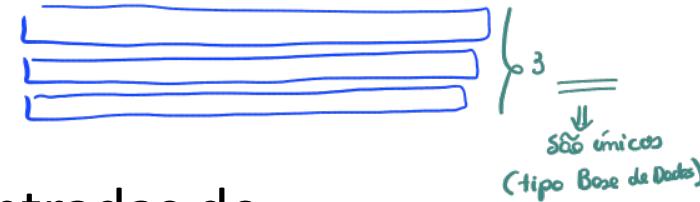
Lightweight Directory Access Protocol

→ Hierárquico em grafo!



• Quase o mesmo que o DNS...  
→ mas aqui temos mais formas de procurar ...

# LDAP



- **Directory Information Base**: coleção de todas as entradas de diretórios num serviço LDAP  
(DIB)
- Cada *record* é designado univocamente pela sequência de atributos de nome (**Relative Distinguished Name**), para que possa ser pesquisado.  
→ Identificador Hierárquico
- **Directory Information Tree**: grafo de nomes de um diretório se serviço LDAP, cada nó representa uma entrada de diretório.  
(DIT)

# Exemplo pesquisa

- search("(C=NL)(O=VU University)(OU=\*)(CN=Main server)")

- Microsoft Active Directory

- Implements LDAP + extensions
- Forest of LDAP domains
  - Index by Global Catalog
  - Root of an LDAP tree is the domain controller
    - Published in DNS with appropriate SRV record

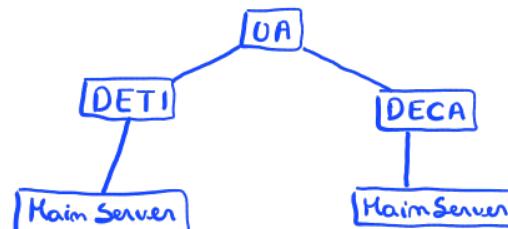
Poderia por o \*  
aqui ....

é usado para procura  
em larga escala  
...

Tem mais do que uma árvore...  
(Escalabilidade)

Isto é apenas para  
uma única árvore!

• Nos precisamos  
dos endereços  
de cada árvore  
(usamos DNS)



PT/UA/DETI/Main server

Endereço de  
todos os  
Main Servers //=> Outro exemplo seria  
encontrar os alunos  
de um determinado  
departamento ....

LDAP é bom  
para este tipo de  
pesquisas ...

# Sumário

- Nomes permitem-nos referir a entidades
  - Pontos de Acesso*
  - Entidades*
- 3 tipos de nomes: endereços (address), identificadores (identifier), nomes comuns (human-friendly name)
- Resolução de nomes em endereços:
  - Broadcast/Multicast (e.g.: ARP, mdns)<sup>multicast</sup>
  - Forwarding Points (e.g.: mobileIP)
  - Home Anchors (e.g.: mobileIP)
  - P2P estruturado + protocolo de routing
  - Arvore de pesquisa hierárquica

(Chord, Bit-Torrent)

Pequisa: Top-Down, Bottom-Up, qualquer nó...

⇒ Nem tudo são árvore ...

O Chord pode  
ser alterado para  
aceder a qualquer  
nó ...

Hierarchical  
Location  
Services

# Sumário

- Nomes estruturados são facilmente organizados em grafos
  - Resolução de nomes é o processo de atravessar o grafo
  - Aproximações hierárquicas permitem ao grafo tornar-se uma árvore
- 
- Sistemas de nomes baseados em atributos permitem maior flexibilidade à custa de maior complexidade no algoritmo de resolução.  
    + Complexidade ...
- LDAP → Pesquisa para todos os enraizes
- 