ACH2147 - Desenvolvimento de Sistemas de Informação Distribuídos

Aula 11 – Sistemas de Nomeação

Norton Trevisan Roman

2 de junho de 2022

Sistemas de Nomeação

- Nomes, Identificadores e Endereços
- Espaço de Nomes Plano
  - Soluções Simples
  - Abordagens *Home-based*
  - Tabelas de *Hash* Distribuídas
  - Abordagens Hierárquicas

Sistemas de Nomeação

- **Nomes, Identificadores e Endereços**
- Espaço de Nomes Plano
  - Soluções Simples
  - Abordagens *Home-based*
  - Tabelas de *Hash* Distribuídas
  - Abordagens Hierárquicas

Nomes, Identificadores e Endereços

Nomes e Endereços

- Usados para denotar entidades em um sistema distribuído

Nomes, Identificadores e Endereços

Nomes e Endereços

- Usados para denotar entidades em um sistema distribuído
- Para realizar operações em uma entidade, é preciso ter acesso a ela usando um ponto de acesso
  - Pontos de acessos são um tipo de entidade identificada por um endereço

Nomes, Identificadores e Endereços

Nomes e Endereços

- Usados para denotar entidades em um sistema distribuído
- Para realizar operações em uma entidade, é preciso ter acesso a ela usando um ponto de acesso
  - Pontos de acessos são um tipo de entidade identificada por um endereço
- Um endereço é um tipo de nome usado como referência ao ponto de acesso de uma entidade
  - O endereço do ponto de acesso de uma entidade é chamado também de endereço dessa entidade

Nomes, Identificadores e Endereços

Nomes e Endereços

- Uma entidade pode mudar de ponto de acesso
  - Ou um ponto de acesso pode ser associado a outra entidade
  - Ex: quando mudamos o servidor de máquina

Nomes, Identificadores e Endereços

Nomes e Endereços

- Uma entidade pode mudar de ponto de acesso
  - Ou um ponto de acesso pode ser associado a outra entidade
  - Ex: quando mudamos o servidor de máquina
- Também pode ter mais de um ponto de acesso

Nomes, Identificadores e Endereços

Nomes e Endereços

- Uma entidade pode mudar de ponto de acesso
  - Ou um ponto de acesso pode ser associado a outra entidade
  - Ex: quando mudamos o servidor de máquina
- Também pode ter mais de um ponto de acesso
- Nome independente de local
  - Nome para uma determinada entidade que é independente do endereço do ponto de acesso dessa entidade
  - Mais fácil e flexível de ser usado

Nomes, Identificadores e Endereços

Identificadores

- Nomes que possuem as seguintes propriedades:
  - Cada identificador se refere a, no máximo, uma entidade
  - Cada entidade é referenciada por, no máximo, um identificador
  - Um identificador sempre se refere à mesma entidade (nunca é reutilizado)

Nomes, Identificadores e Endereços

Mapeando nomes e identificadores a endereços

- *Name-to-address binding*
  - Estrutura mantida pelo serviço de nomes (SN)
  - Pode ser simplesmente uma tabela *(nome, endereço)*

Nomes, Identificadores e Endereços

Mapeando nomes e identificadores a endereços

- *Name-to-address binding*
  - Estrutura mantida pelo serviço de nomes (SN)
  - Pode ser simplesmente uma tabela *(nome, endereço)*
- Para SDs grandes, uma tabela centralizada não funciona
  - Podemos então decompor o nome em várias partes, fazendo com que a resolução ocorra recursivamente, a cada parte
  - Ex: www.usp.br

    SN(.) $\rightarrow$ SN(br) $\rightarrow$ SN(www.usp)

Sistemas de Nomeação

- Nomes, Identificadores e Endereços
- **Espaço de Nomes Plano**
  - **Soluções Simples**
  - Abordagens *Home-based*
  - Tabelas de *Hash* Distribuídas
  - Abordagens Hierárquicas

Espaço de Nomes Plano

Nomes Planos

- São identificadores não estruturados, usados para referenciar entidades
  - Simples arranjos de bits que, em princípio, não têm significado

Espaço de Nomes Plano

Nomes Planos

- São identificadores não estruturados, usados para referenciar entidades
  - Simples arranjos de bits que, em princípio, não têm significado
- Como localizar uma entidade dado apenas seu identificador? (Como resolver um nome plano?)

Espaço de Nomes Plano

Nomes Planos

- São identificadores não estruturados, usados para referenciar entidades
  - Simples arranjos de bits que, em princípio, não têm significado
- Como localizar uma entidade dado apenas seu identificador? (Como resolver um nome plano?)
- Duas soluções simples
  - *Broadcasting* e Ponteiros de Redirecionamento
  - Aplicáveis somente em redes locais (LAN)

E. N. Plano – Soluções Simples

Broadcasting

- Faça o *broadcast* do ID, requisitando que a entidade devolva seu endereço
  - Não escala para além de redes locais
  - Requer que todos os processos escutem e processem os pedidos de localização

E. N. Plano – Soluções Simples

Broadcasting

- Faça o *broadcast* do ID, requisitando que a entidade devolva seu endereço
  - Não escala para além de redes locais
  - Requer que todos os processos escutem e processem os pedidos de localização
- Ex: Address Resolution Protocol (ARP)
  - Para encontrar o endereço MAC associado a um endereço IP, faz o *broadcast* da consulta "quem tem esse endereço IP?"

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento

- *Forwarding pointers*

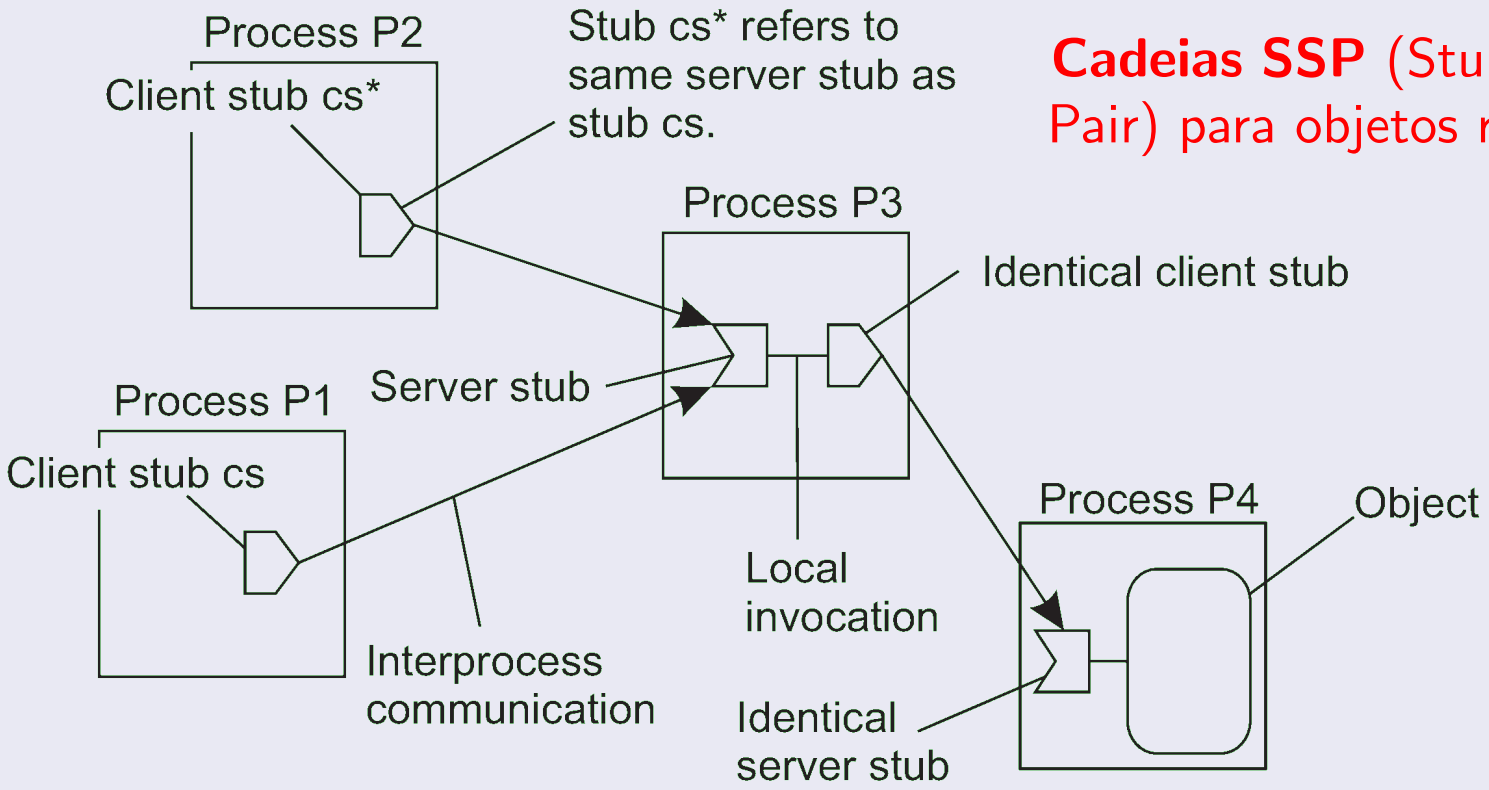
E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento

- *Forwarding pointers*
- Quando uma entidade se move, ela deixa para trás um ponteiro para sua nova localização
  - Derreferenciamento pode ser automático e invisível para o cliente, basta seguir a sequência de ponteiros
    - *Dereferencing* significa acessar o valor armazenado em um endereço guardado em um ponteiro
  - Atualize a referência do ponteiro quando o local atual for encontrado

E. N. Plano – Soluções Simples

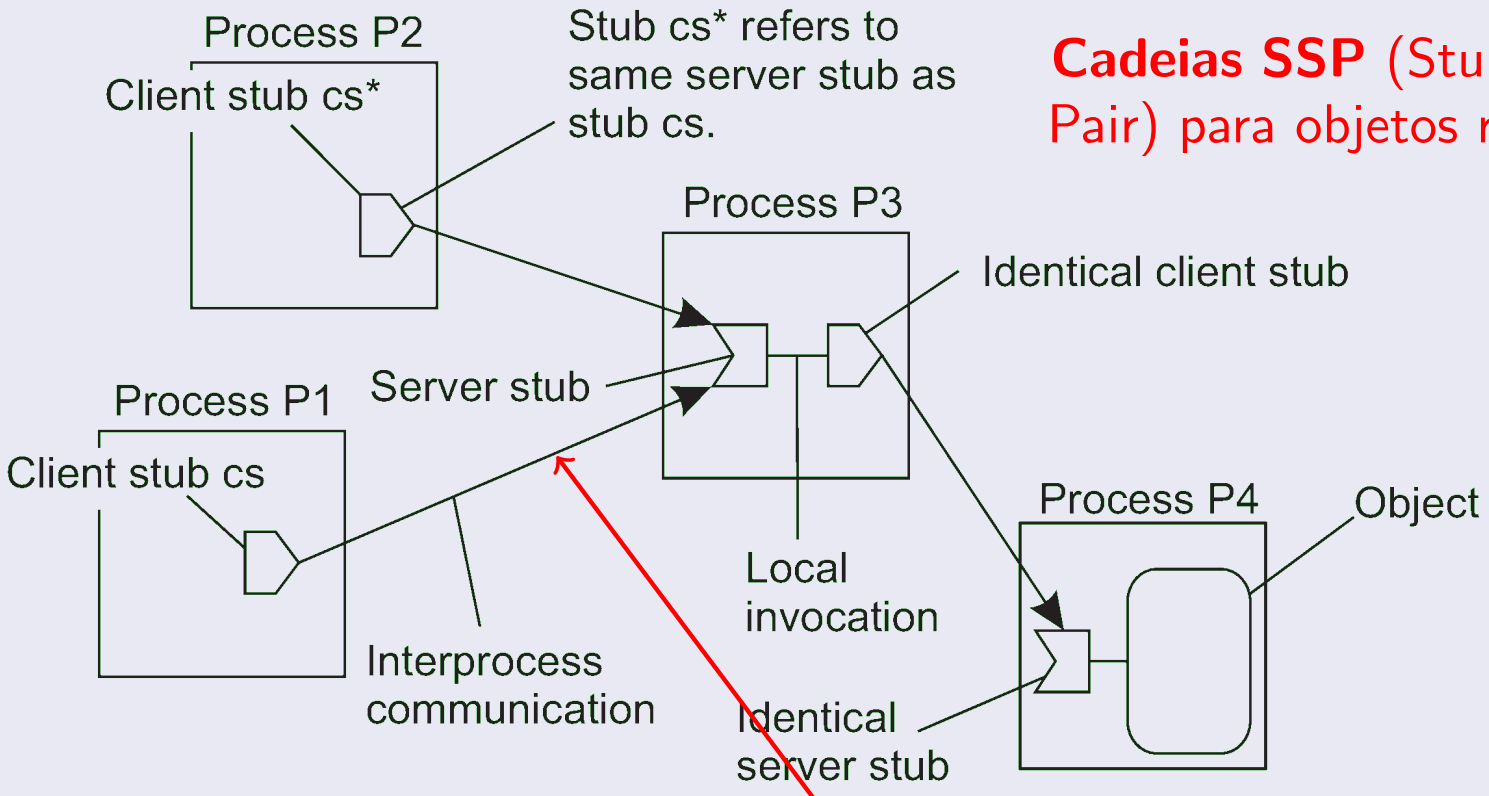
Ponteiros de Redirecionamento – Ex: SSP



Cadeias SSP (Stub-Scion Pair) para objetos remotos

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

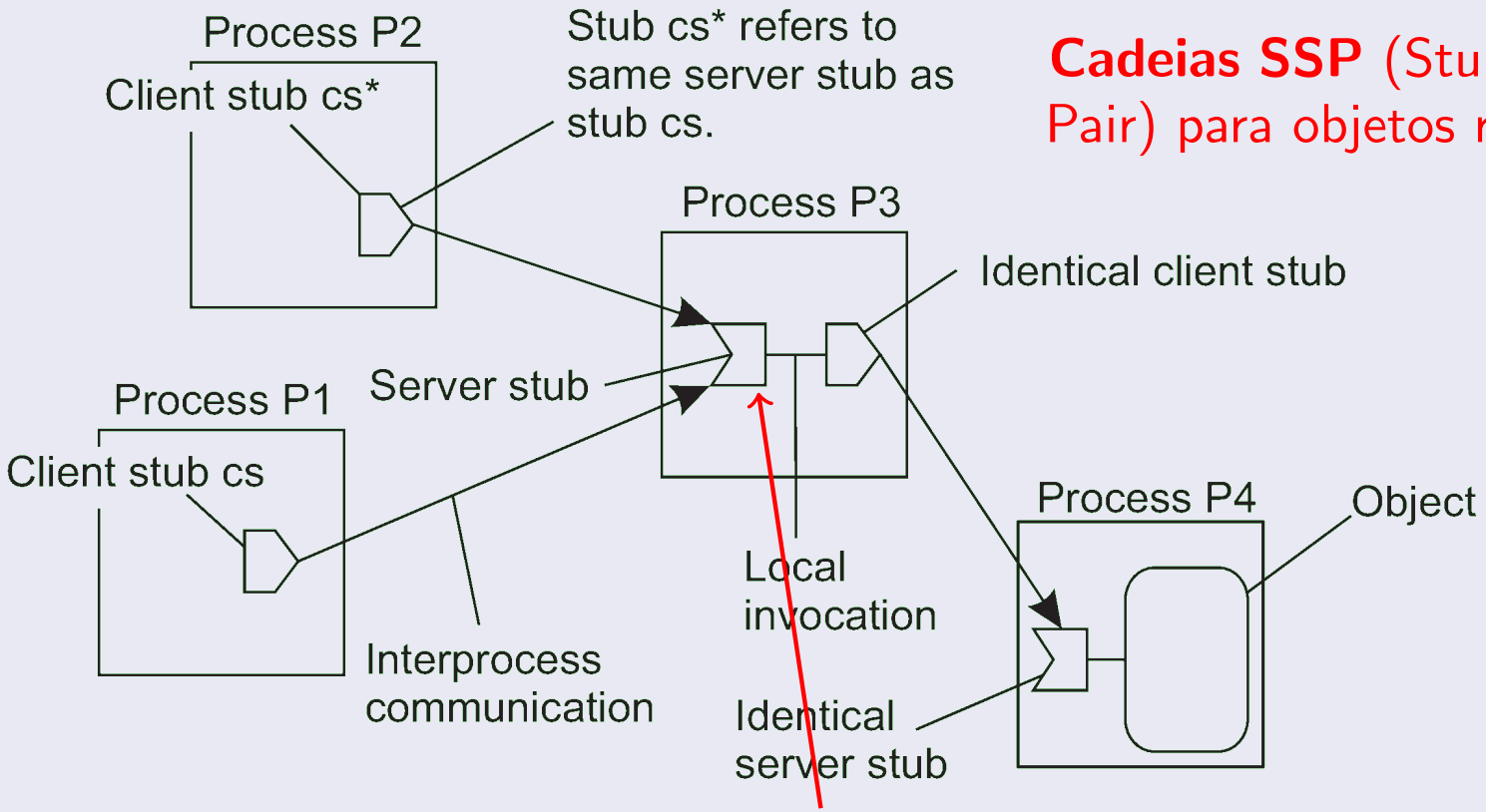


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Cada ponteiro é implementado como um par (*stub* do cliente, *stub* do servidor) (originalmente, o *stub* do servidor era chamado de scion – daí seu nome)

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

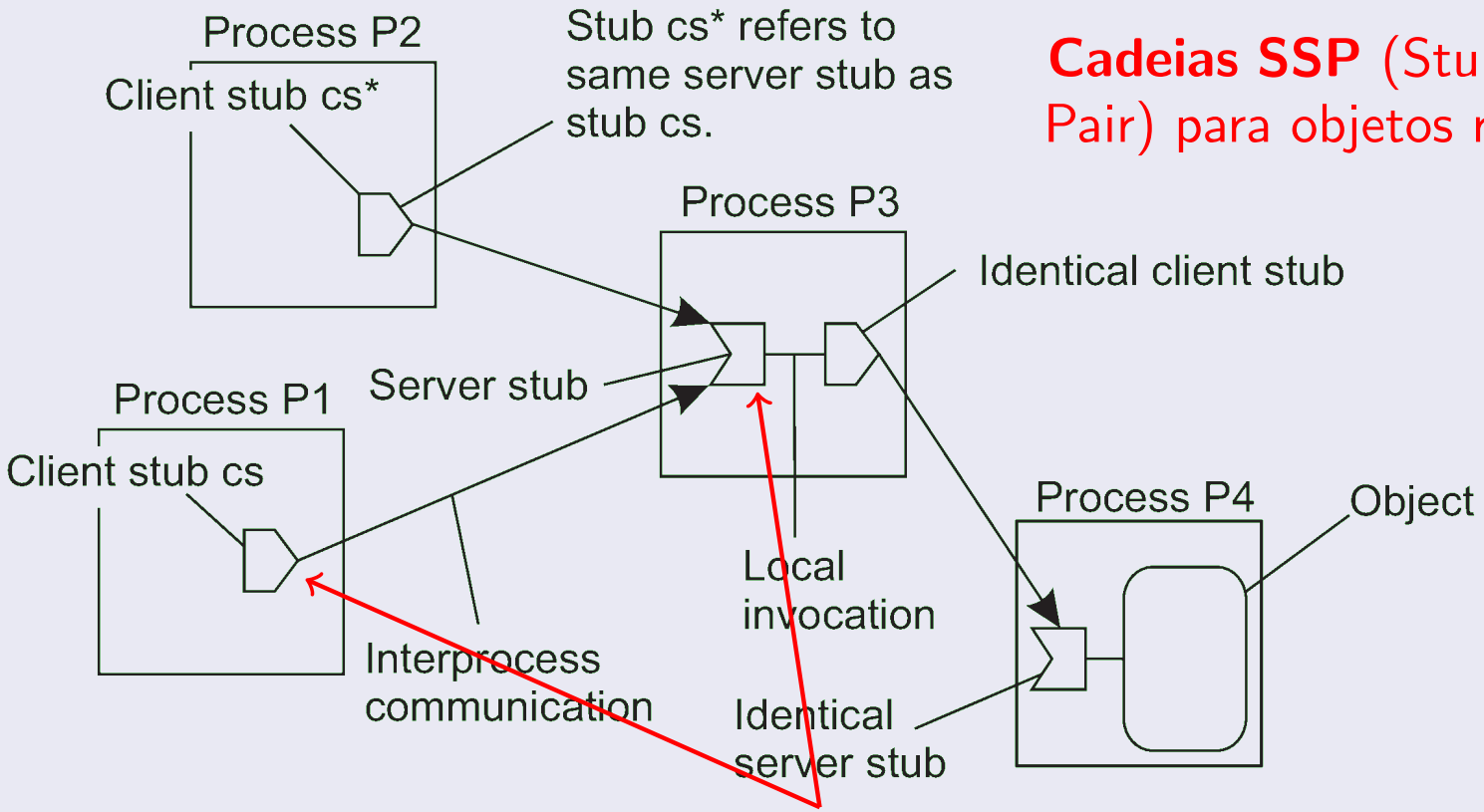


Cadeias SSP (Stub-Scion Pair) para objetos remotos

O *stub* do servidor contém ou uma referência local ao objeto, ou uma referência local a um stub de cliente para aquele objeto

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

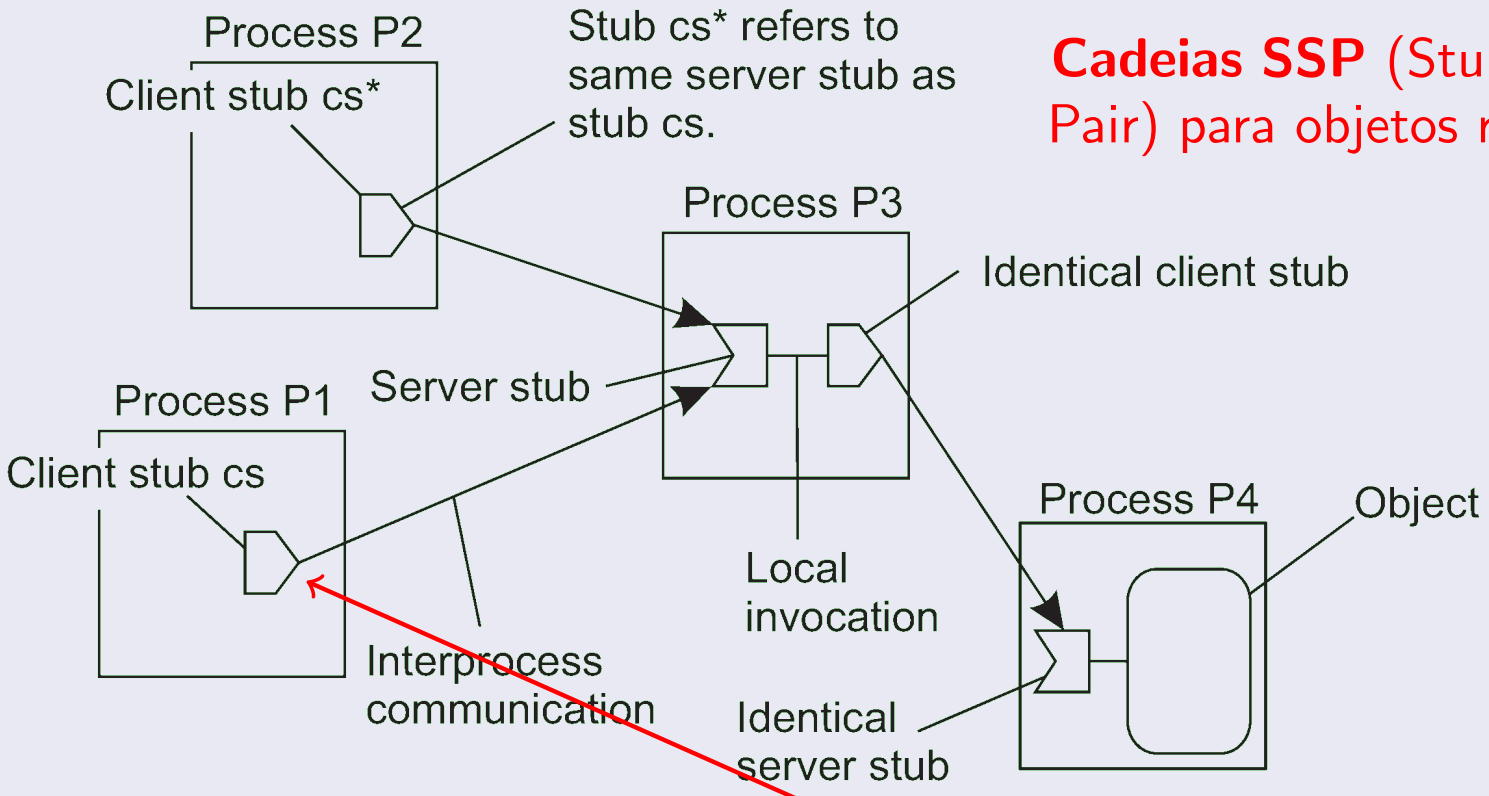


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Quando um objeto se move, deixa para trás um *stub* de cliente em seu lugar, instalando um *stub* de servidor, que se refere ao *stub* do cliente, em sua nova localização

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

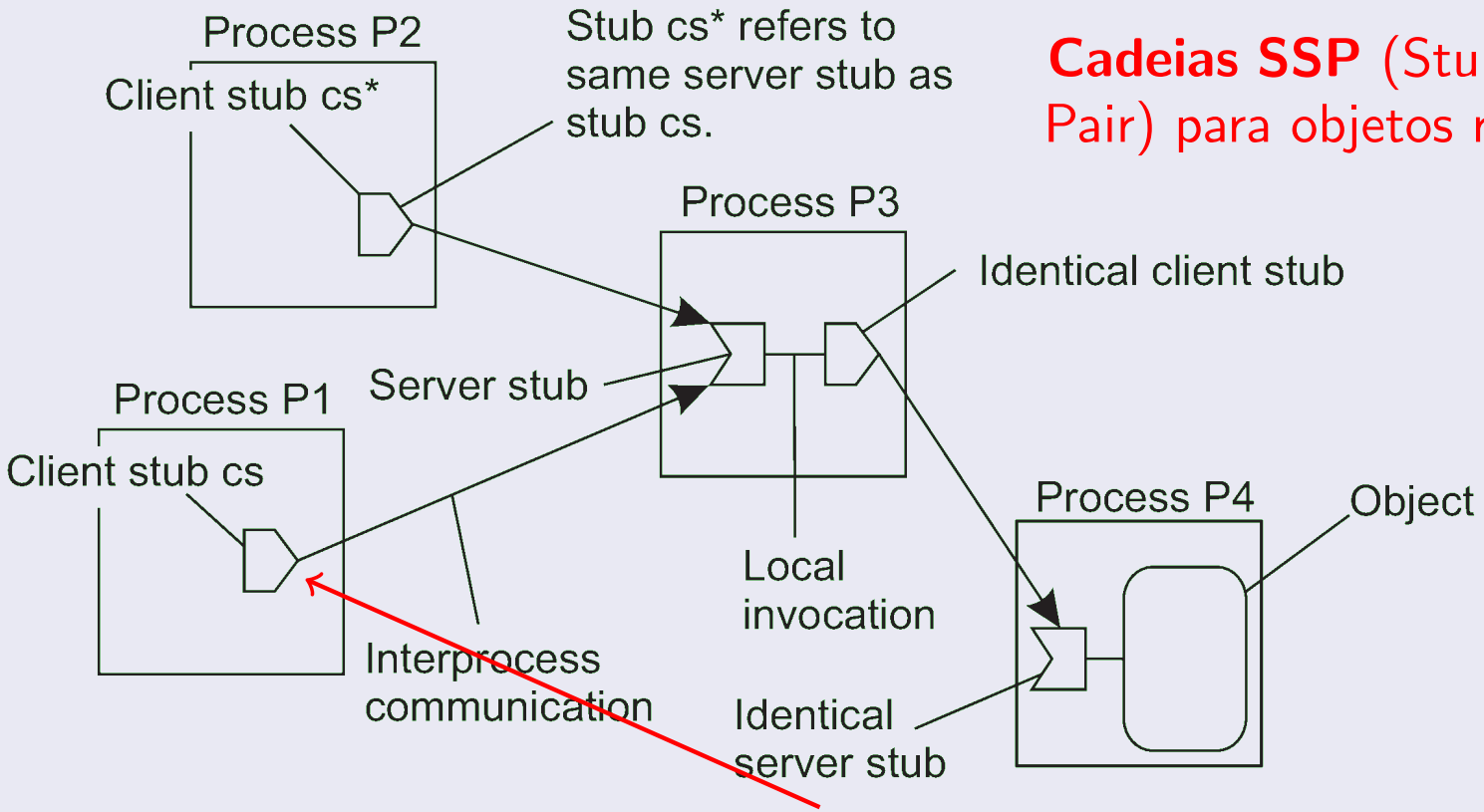


Cadeias SSP (Stub-Scion Pair) para objetos remotos

A migração é completamente transparente para o cliente – ele só vê o *stub* do cliente para o objeto

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

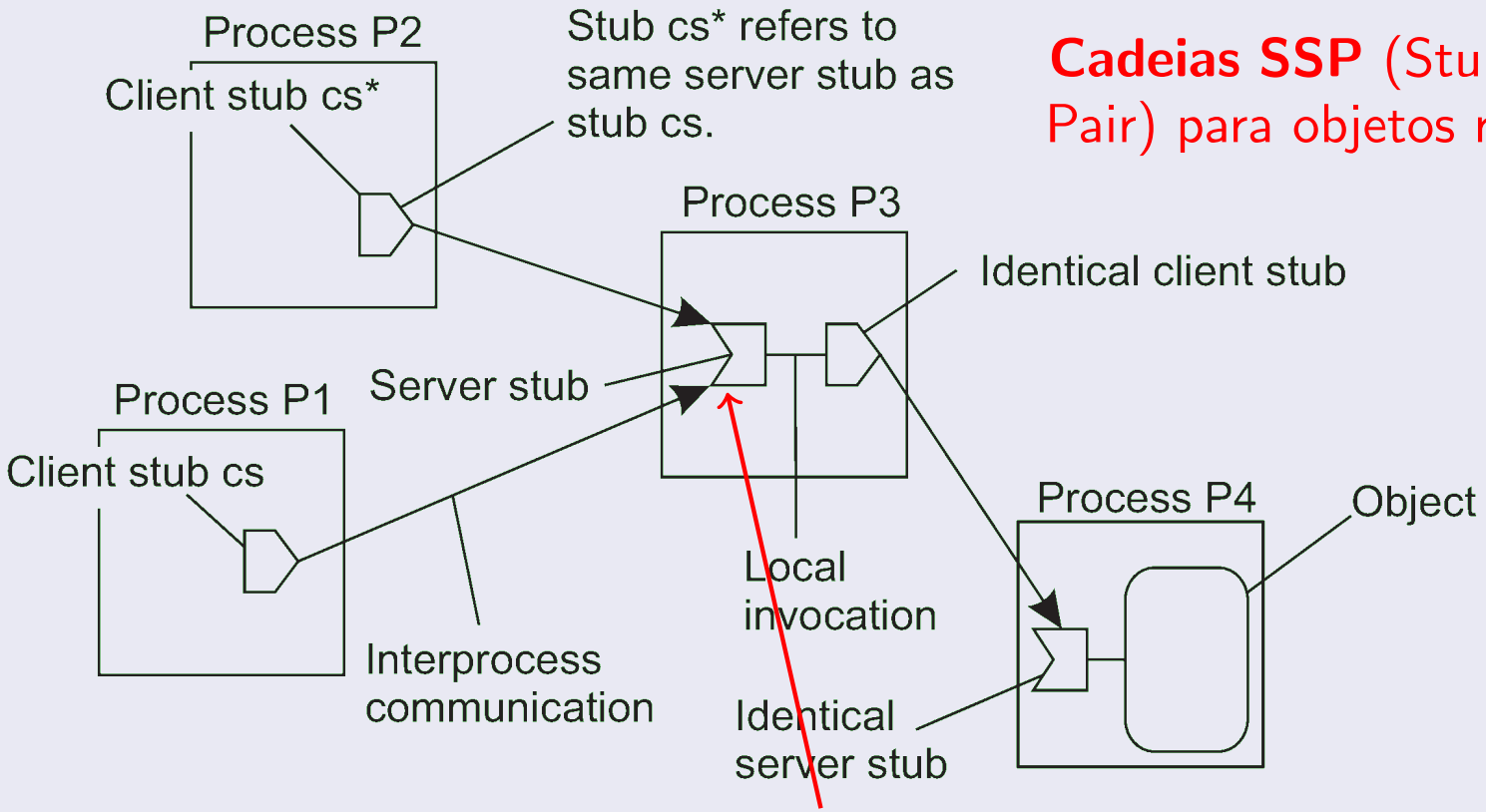


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Qualquer requisição do cliente é repassada ao longo da cadeia até o objeto

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

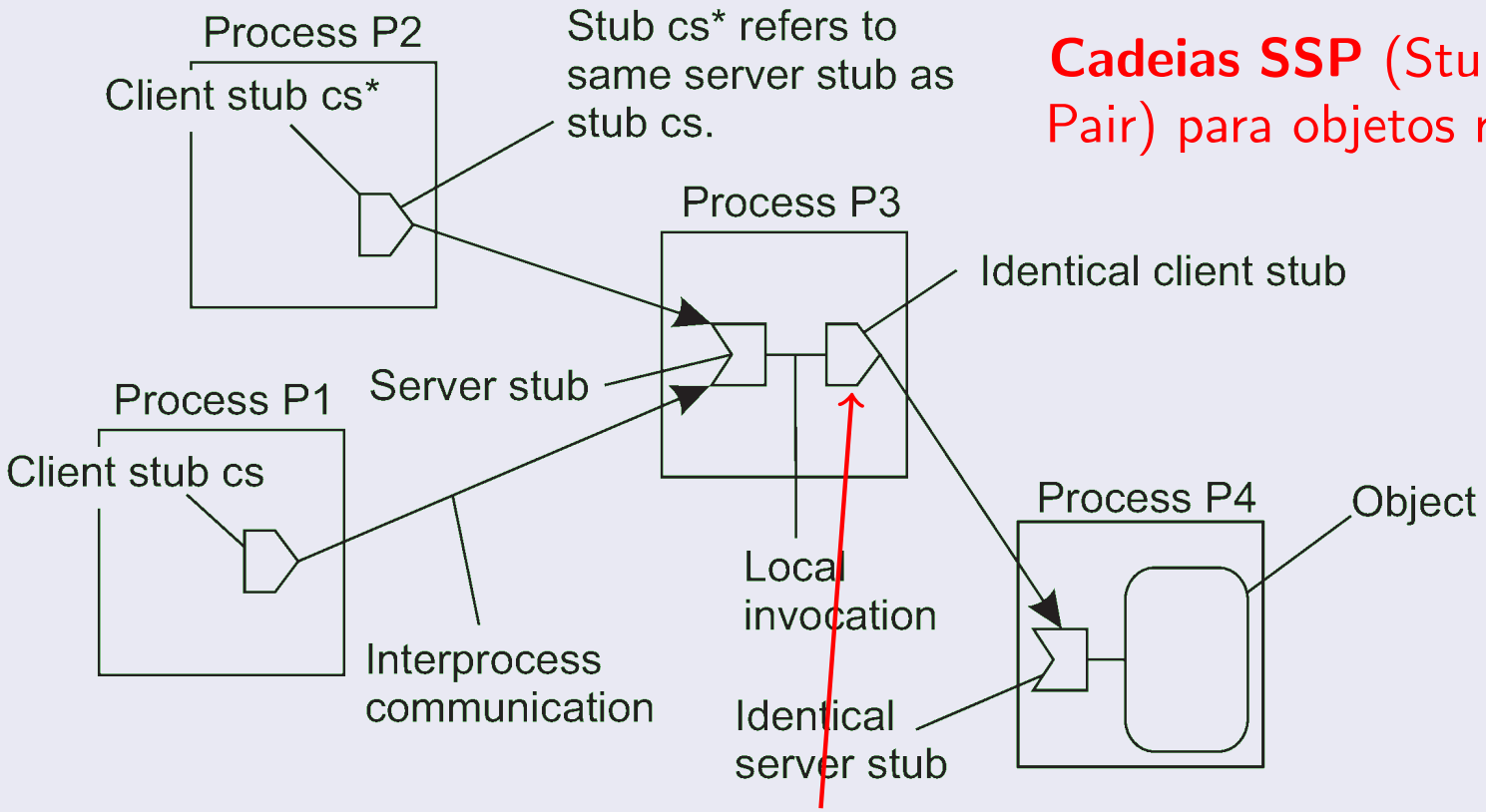


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Qualquer requisição do cliente é repassada ao longo da cadeia até o objeto

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

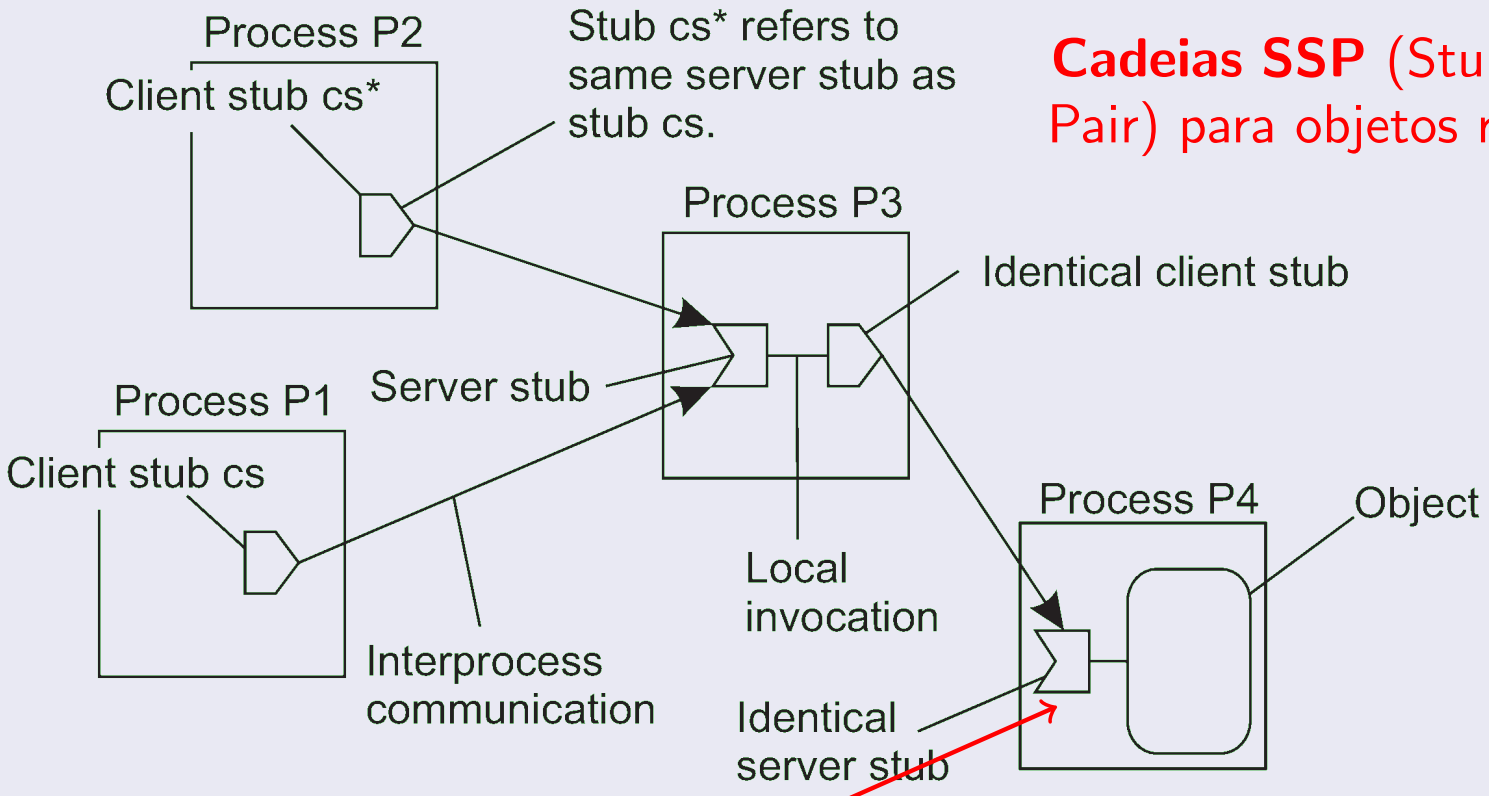


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Qualquer requisição do cliente é repassada ao longo da cadeia até o objeto

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

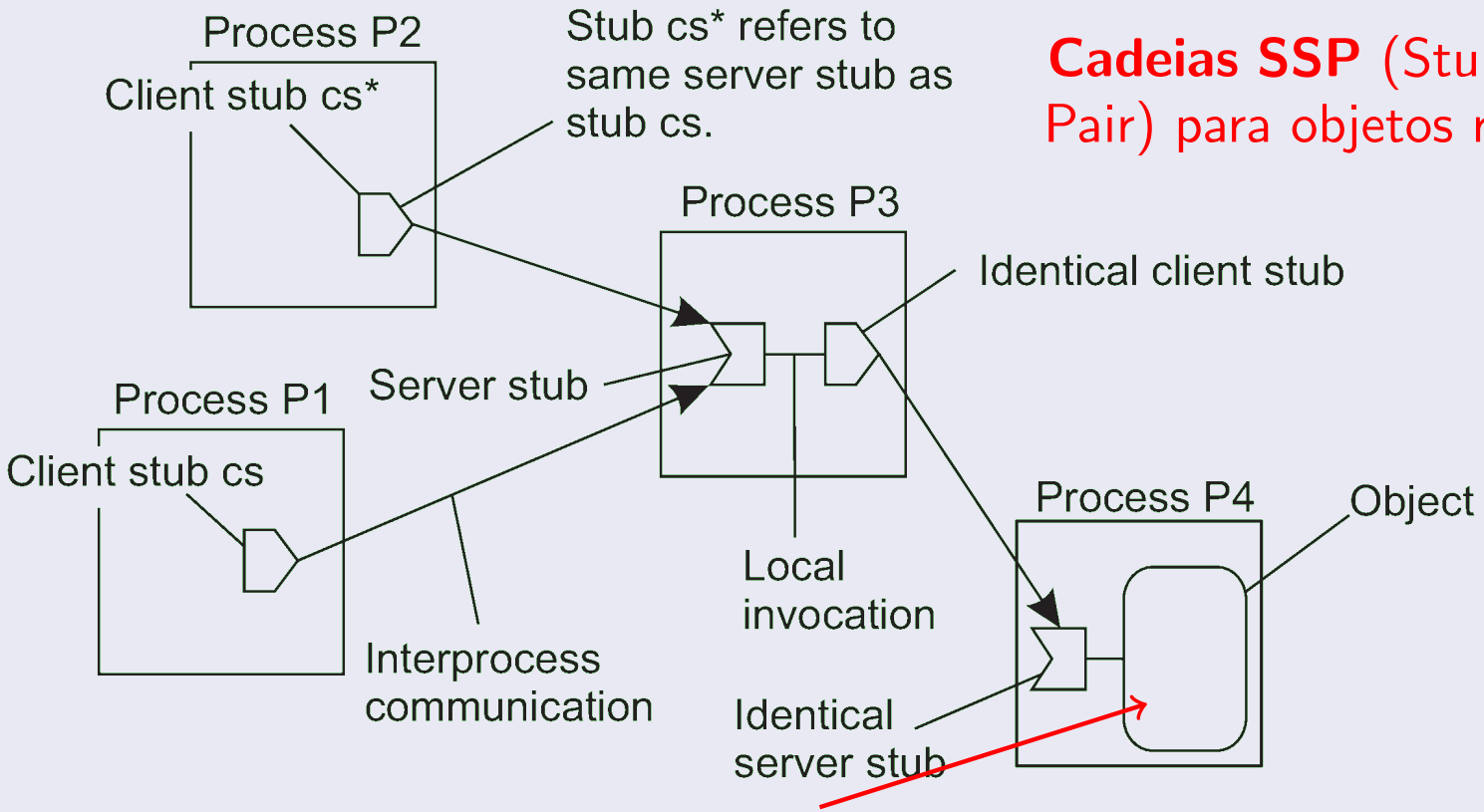


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Qualquer requisição do cliente é repassada ao longo da cadeia até o objeto

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento – Ex: SSP

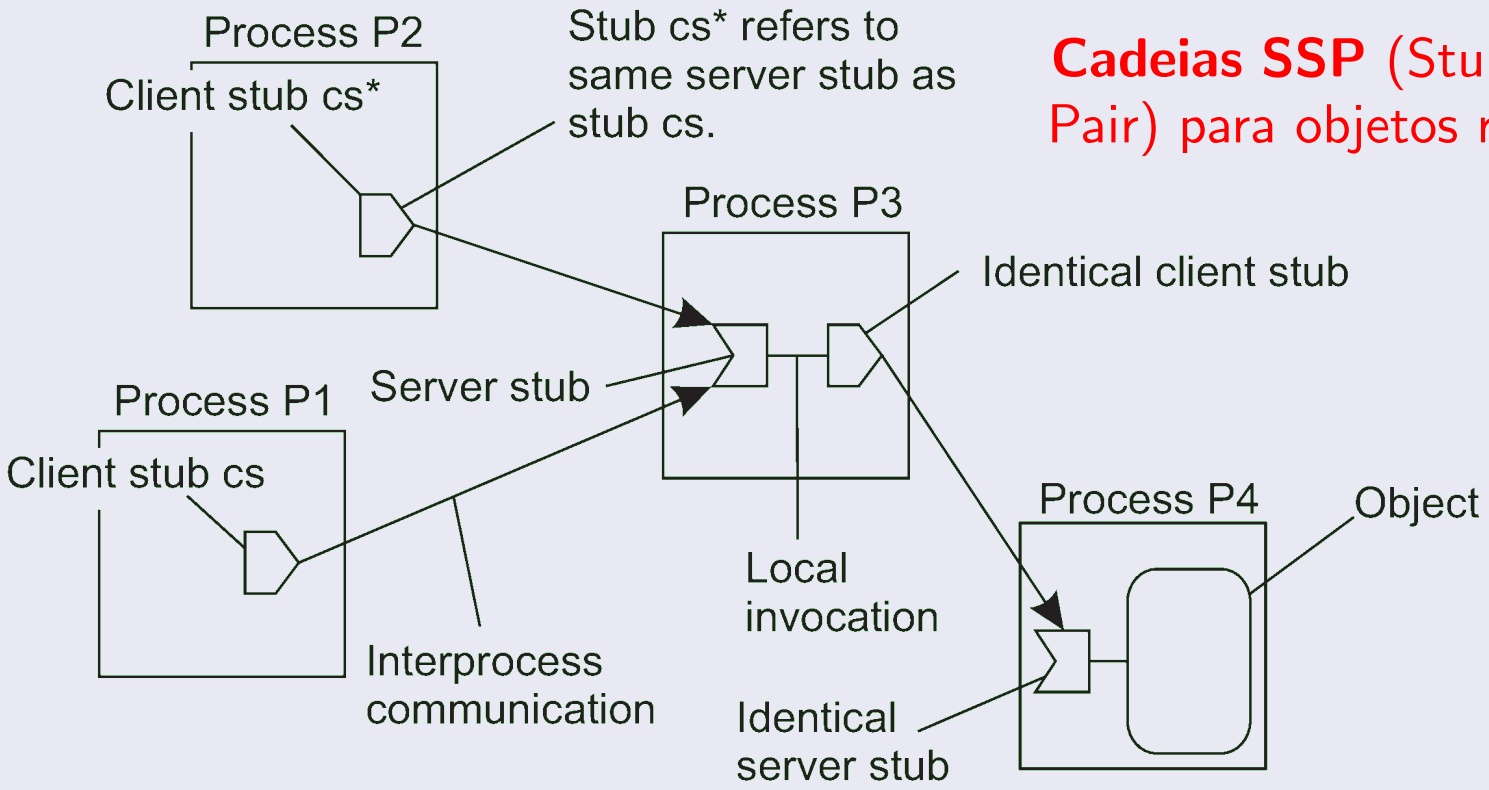


Cadeias SSP (Stub-Scion Pair) para objetos remotos

Qualquer requisição do cliente é repassada ao longo da cadeia até o objeto

E. N. Plano – Soluções Simples

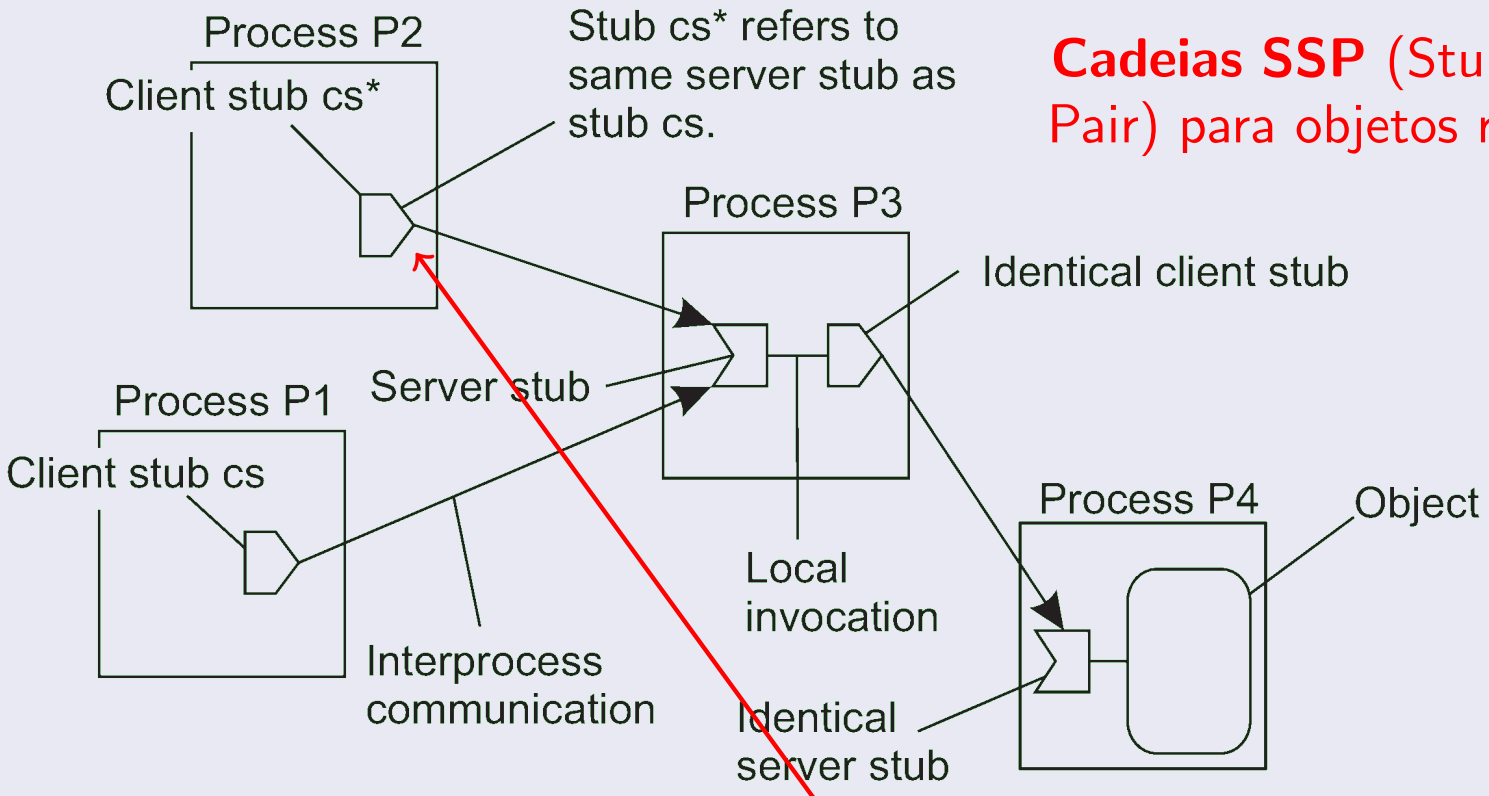
Ponteiros de Redirecionamento – Ex: SSP



Cadeias SSP (Stub-Scion Pair) para objetos remotos

E como podemos fazer para que o processo P1 passe uma **referência** ao objeto para P2?

Ponteiros de Redirecionamento – Ex: SSP



Cadeias SSP (Stub-Scion Pair) para objetos remotos

P1 instala uma cópia do *stub* do cliente em P2, se referindo ao mesmo *stub* do servidor

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento

- Problemas de escalabilidade geográfica (que podem requerer um mecanismo separado para redução da sequência)
  - Maior latência de rede devido ao processo de derreferenciamento

E. N. Plano – Soluções Simples

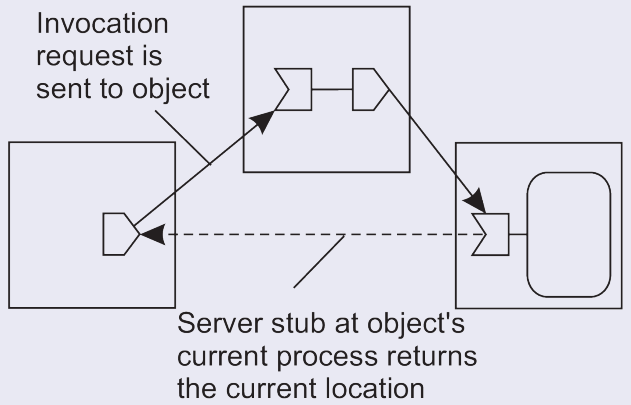
Ponteiros de Redirecionamento

- Problemas de escalabilidade geográfica (que podem requerer um mecanismo separado para redução da sequência)
  - Maior latência de rede devido ao processo de derreferenciamento
- Sequencias longas não são tolerantes a falhas
  - Todos os locais intermediários da sequência têm que manter o redirecionamento
  - Vulnerável a *links* quebrados, por conta de um servidor caído ou fora do alcance

E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento: SSP

- Sequências Longas: crie atalhos na cadeia

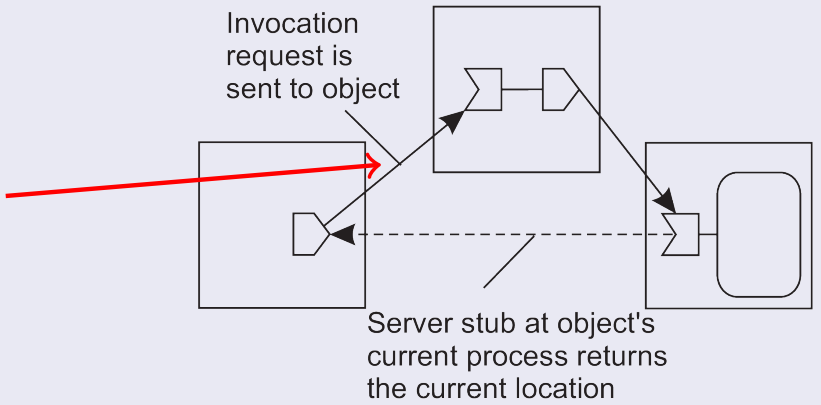


E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento: SSP

- Sequências Longas: crie atalhos na cadeia

Invocações a um objeto contêm a identificação do *stub* do cliente de onde foram iniciadas

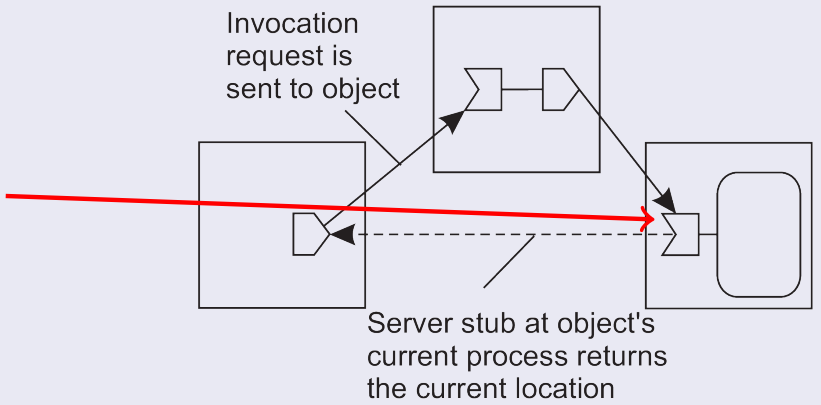


E. N. Plano – Soluções Simples

Ponteiros de Redirecionamento: SSP

- Sequências Longas: crie atalhos na cadeia

Quando a invocação chega ao objeto, uma resposta é enviada de volta ao *stub* do cliente inicial

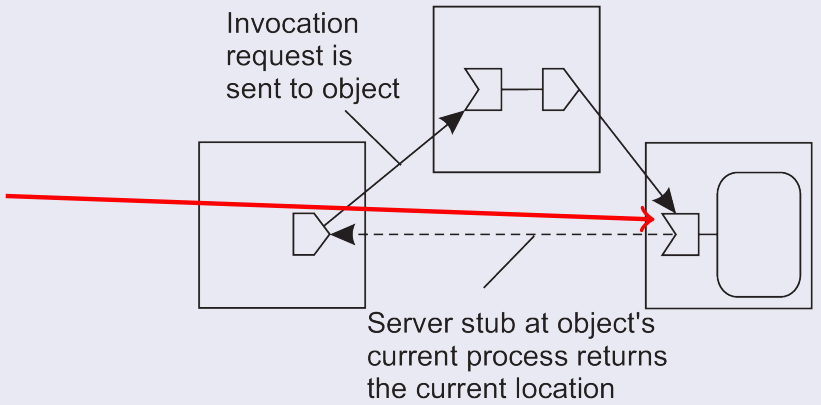


E. N. Plano – Soluções Simples

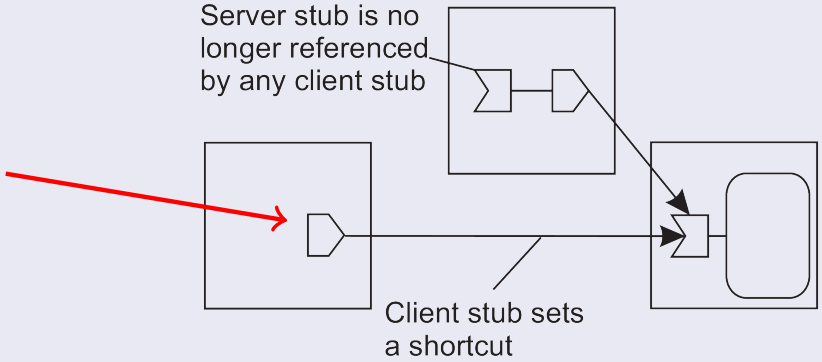
Ponteiros de Redirecionamento: SSP

- Sequências Longas: crie atalhos na cadeia

Quando a invocação chega ao objeto, uma resposta é enviada de volta ao *stub* do cliente inicial



O *stub* do cliente atualiza então seu registro, apontando para o *stub* da localização atual do objeto

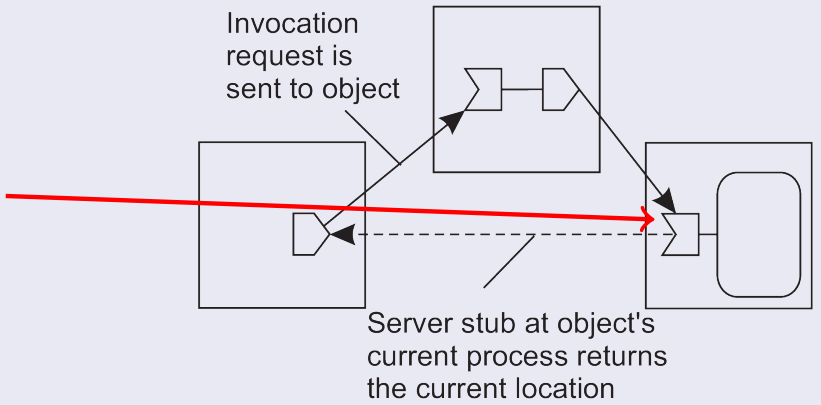


E. N. Plano – Soluções Simples

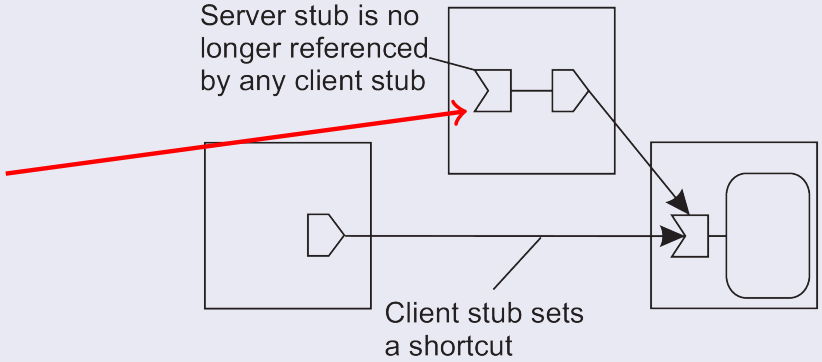
Ponteiros de Redirecionamento: SSP

- Sequências Longas: crie atalhos na cadeia

Quando a invocação chega ao objeto, uma resposta é enviada de volta ao *stub* do cliente inicial



Quando um *stub* de servidor não é mais referenciado por nenhum cliente, ele pode ser removido



Sistemas de Nomeação

- Nomes, Identificadores e Endereços
- **Espaço de Nomes Plano**
  - Soluções Simples
  - **Abordagens Home-based**
  - Tabelas de *Hash* Distribuídas
  - Abordagens Hierárquicas

E. N. Plano – Abordagens *Home-based*

Home Location

- Local responsável por rastrear a localização atual de uma entidade
  - Frequentemente o lugar onde a entidade foi criada

E. N. Plano – Abordagens *Home-based*

Home Location

- Local responsável por rastrear a localização atual de uma entidade
  - Frequentemente o lugar onde a entidade foi criada

Abordagem usada como mecanismo de emergência para Ponteiros de Redirecionamento

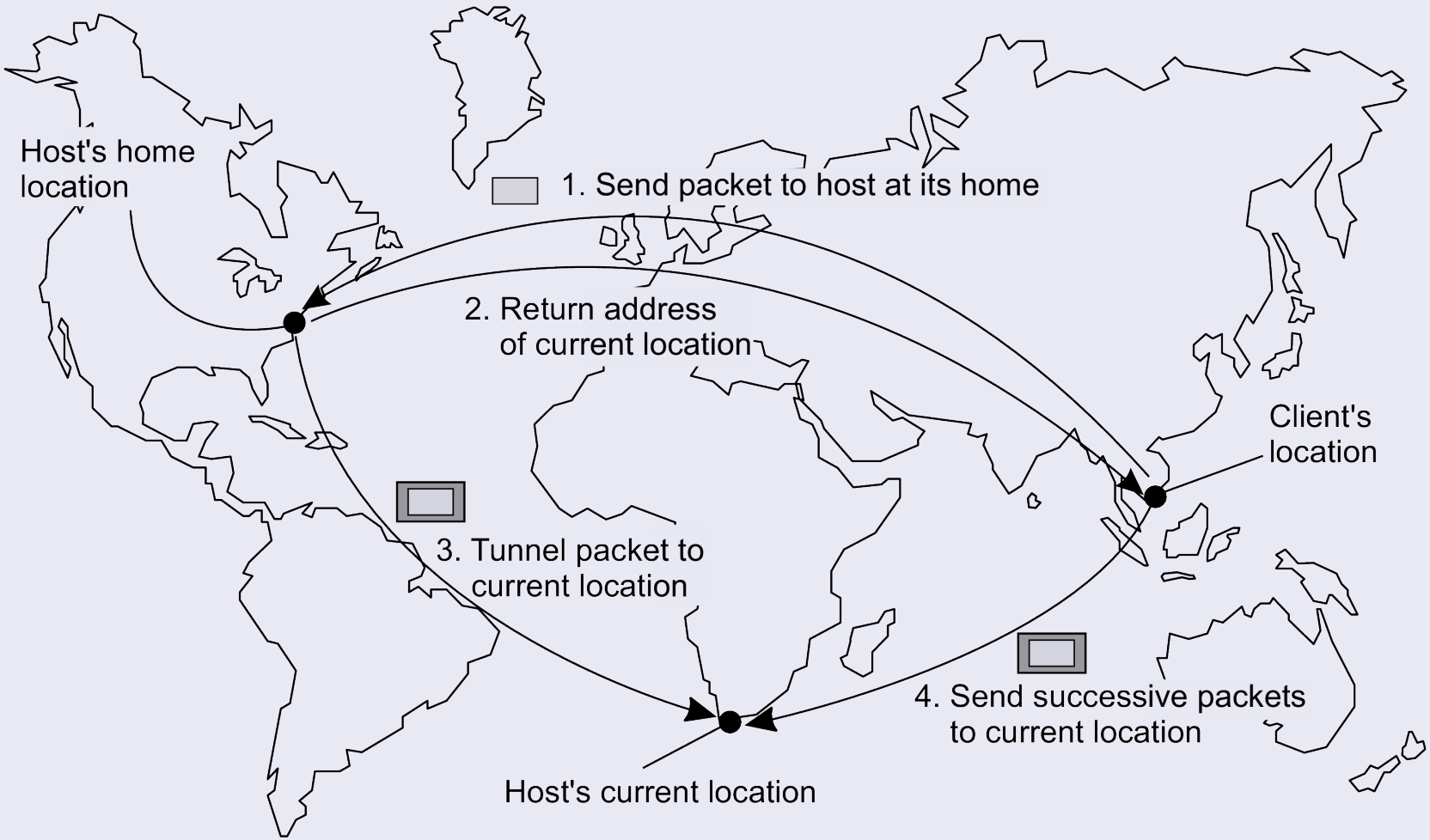
E. N. Plano – Abordagens *Home-based*

Funcionamento

- Faça com que um local fixo (sua "residência") sempre saiba onde a entidade está:
  - O endereço de residência é registrado em um serviço de nomes
  - A residência mantém um registro do endereço externo da entidade
  - O cliente primeiro contacta o endereço de residência e depois continua para seu endereço externo

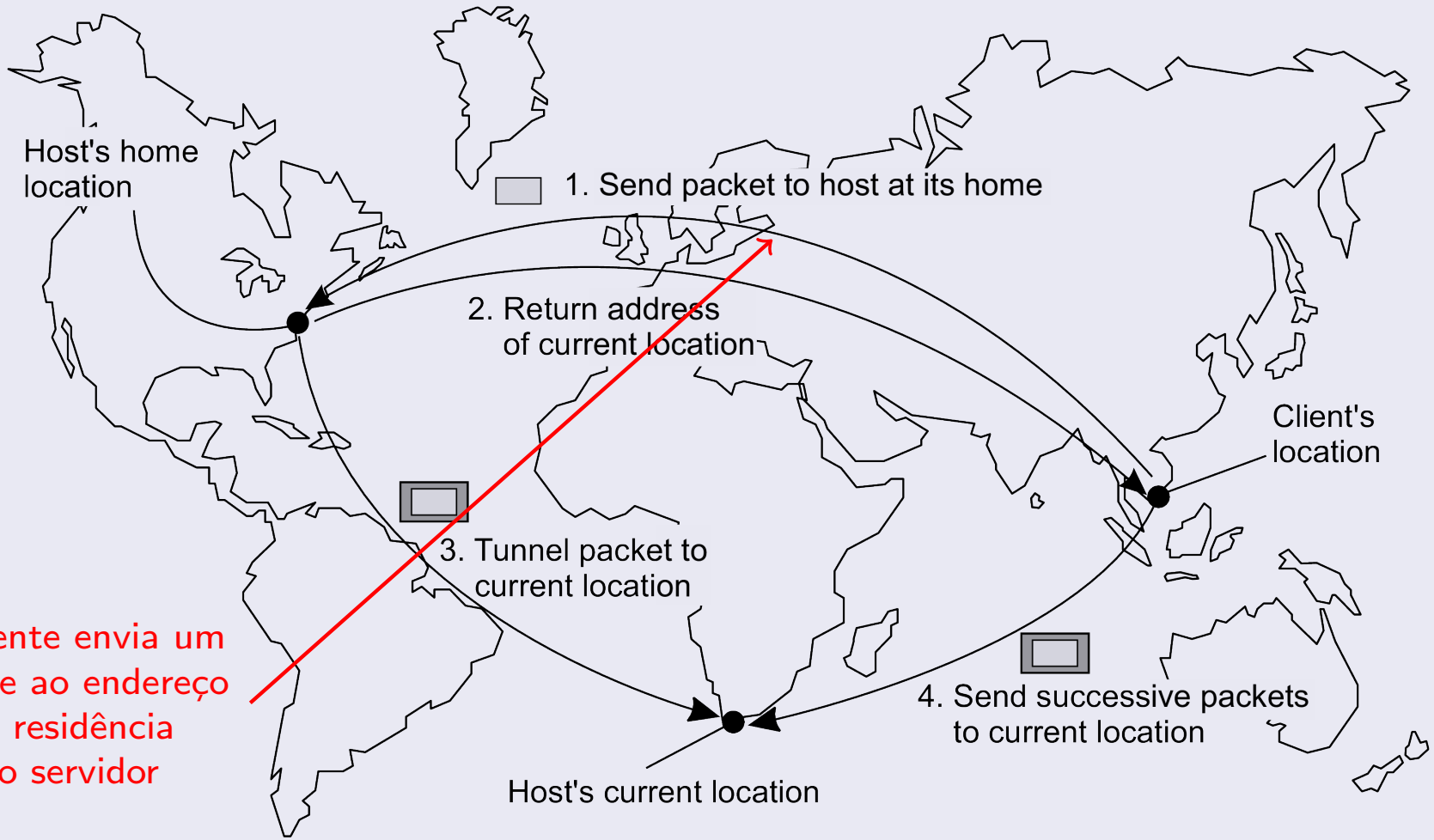
E. N. Plano – Abordagens *Home-based*

Ex: IP móvel



E. N. Plano – Abordagens *Home-based*

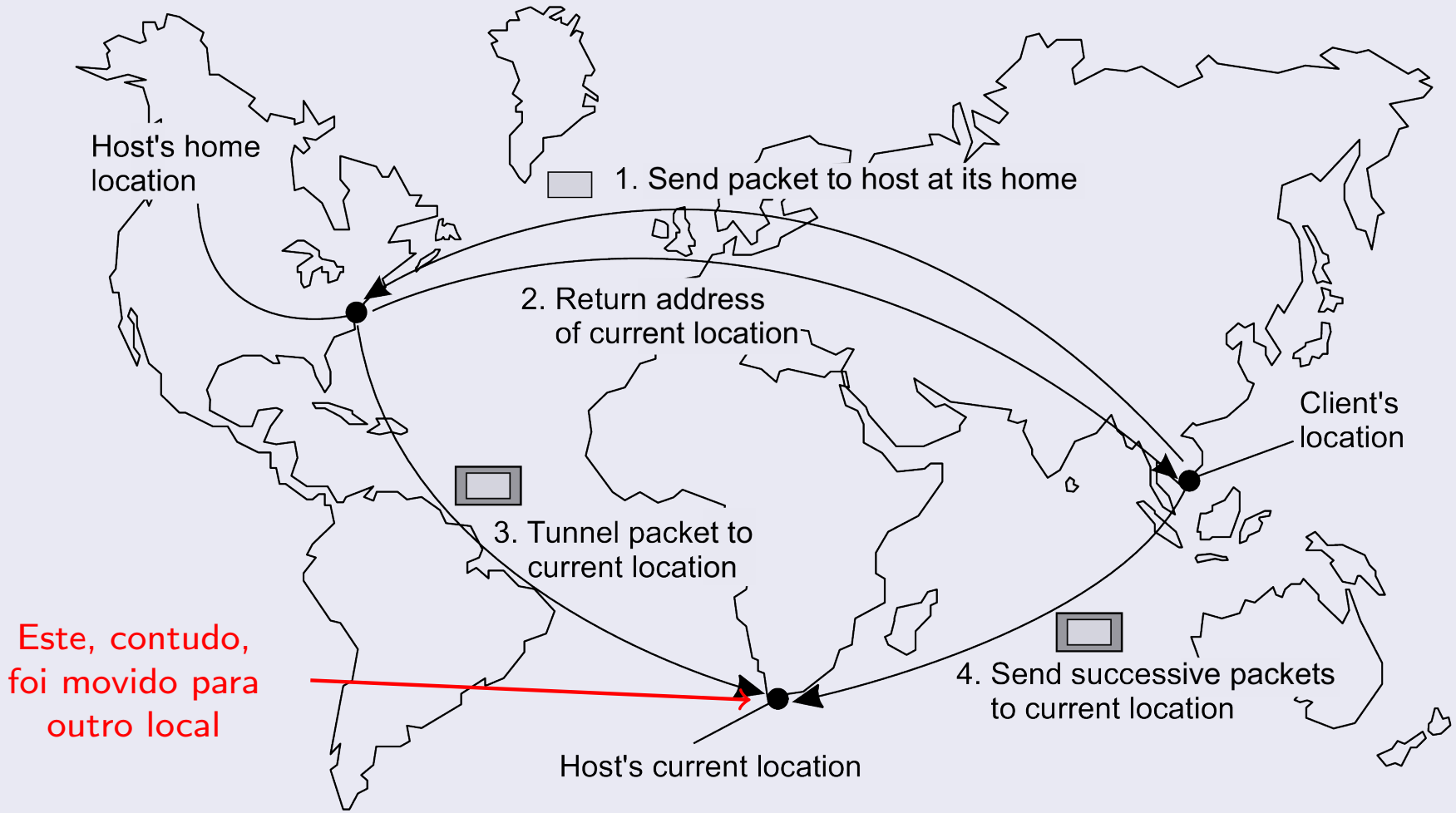
Ex: IP móvel



O cliente envia um pacote ao endereço de residência do servidor

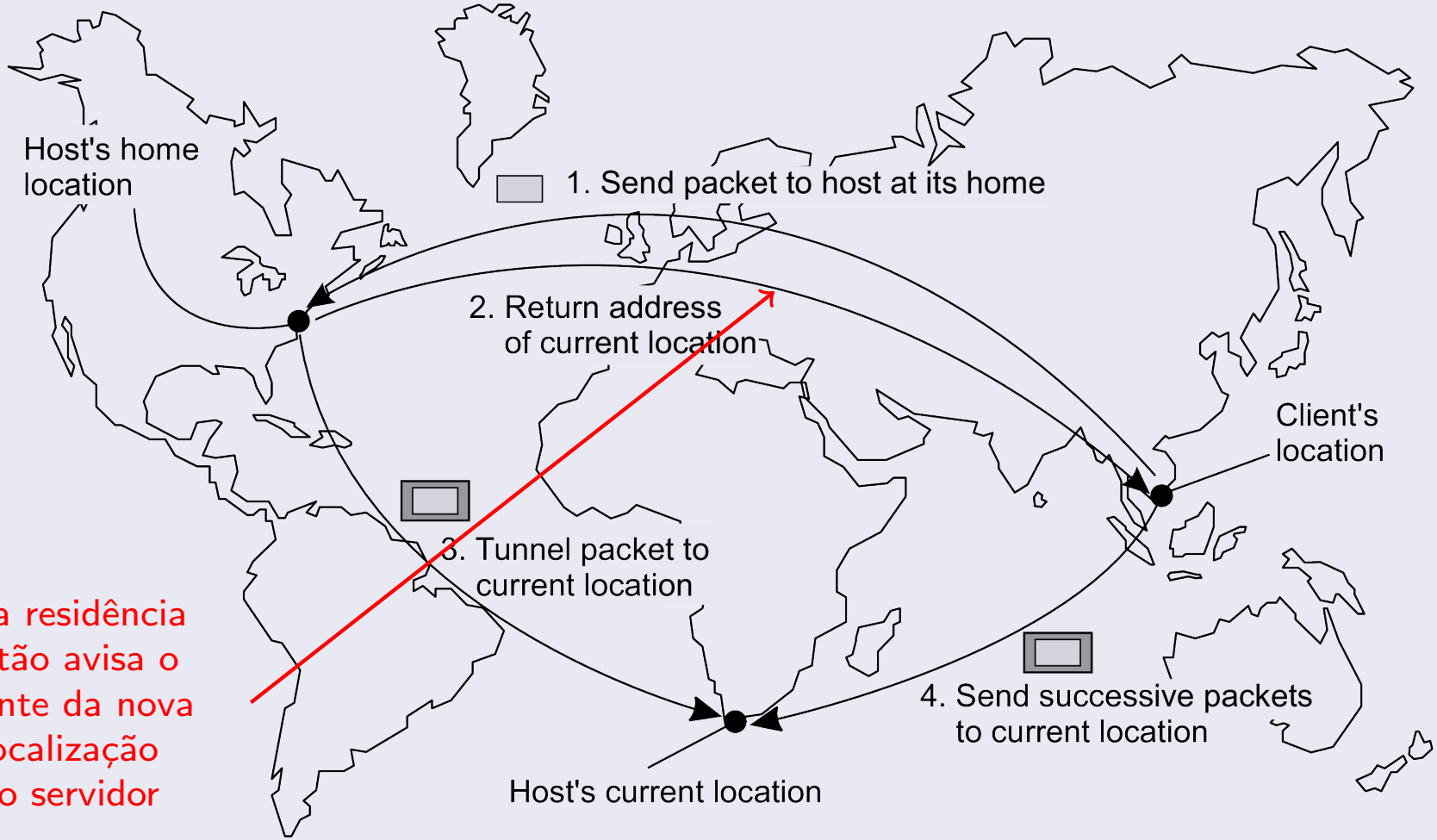
E. N. Plano – Abordagens *Home-based*

Ex: IP móvel



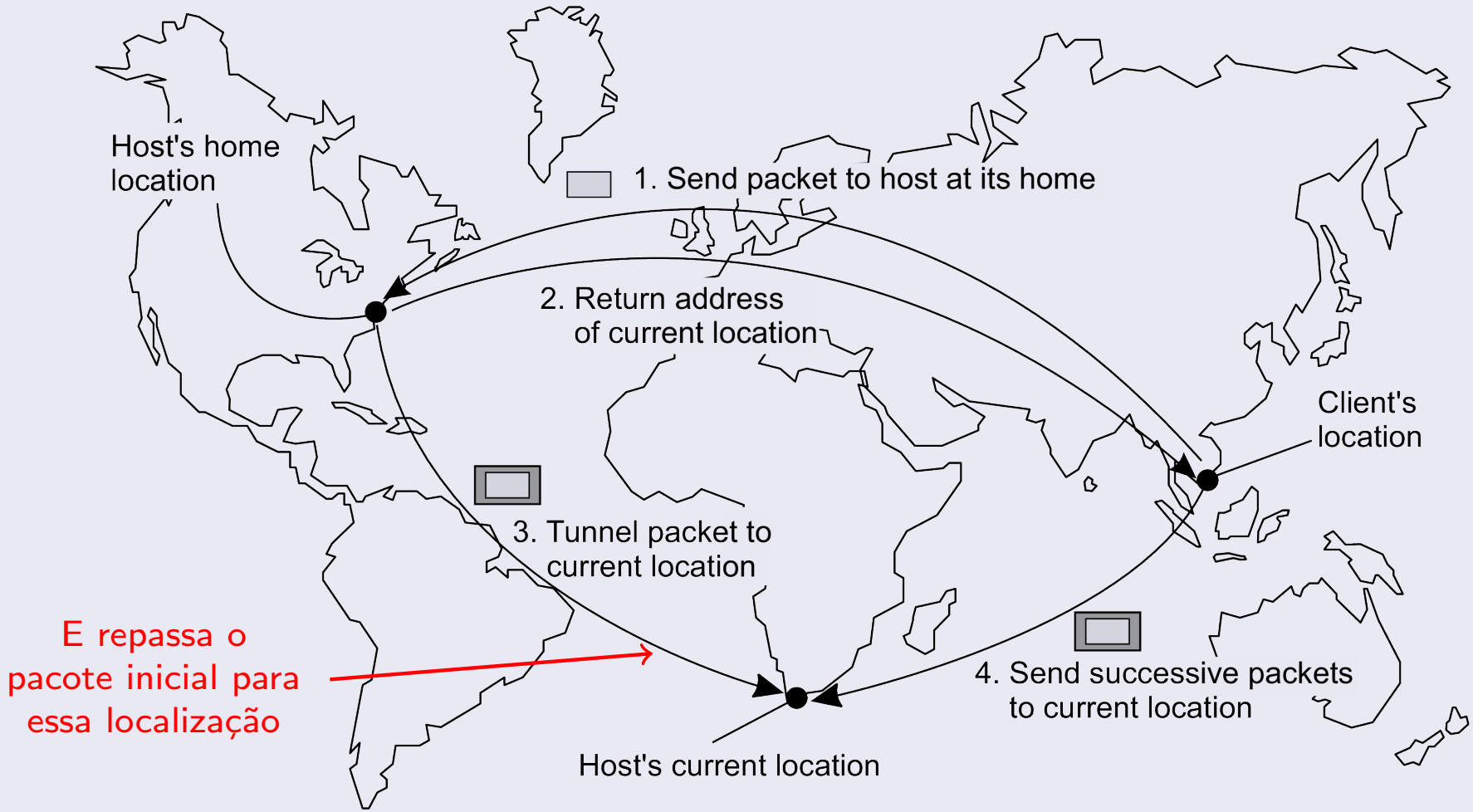
E. N. Plano – Abordagens *Home-based*

Ex: IP móvel



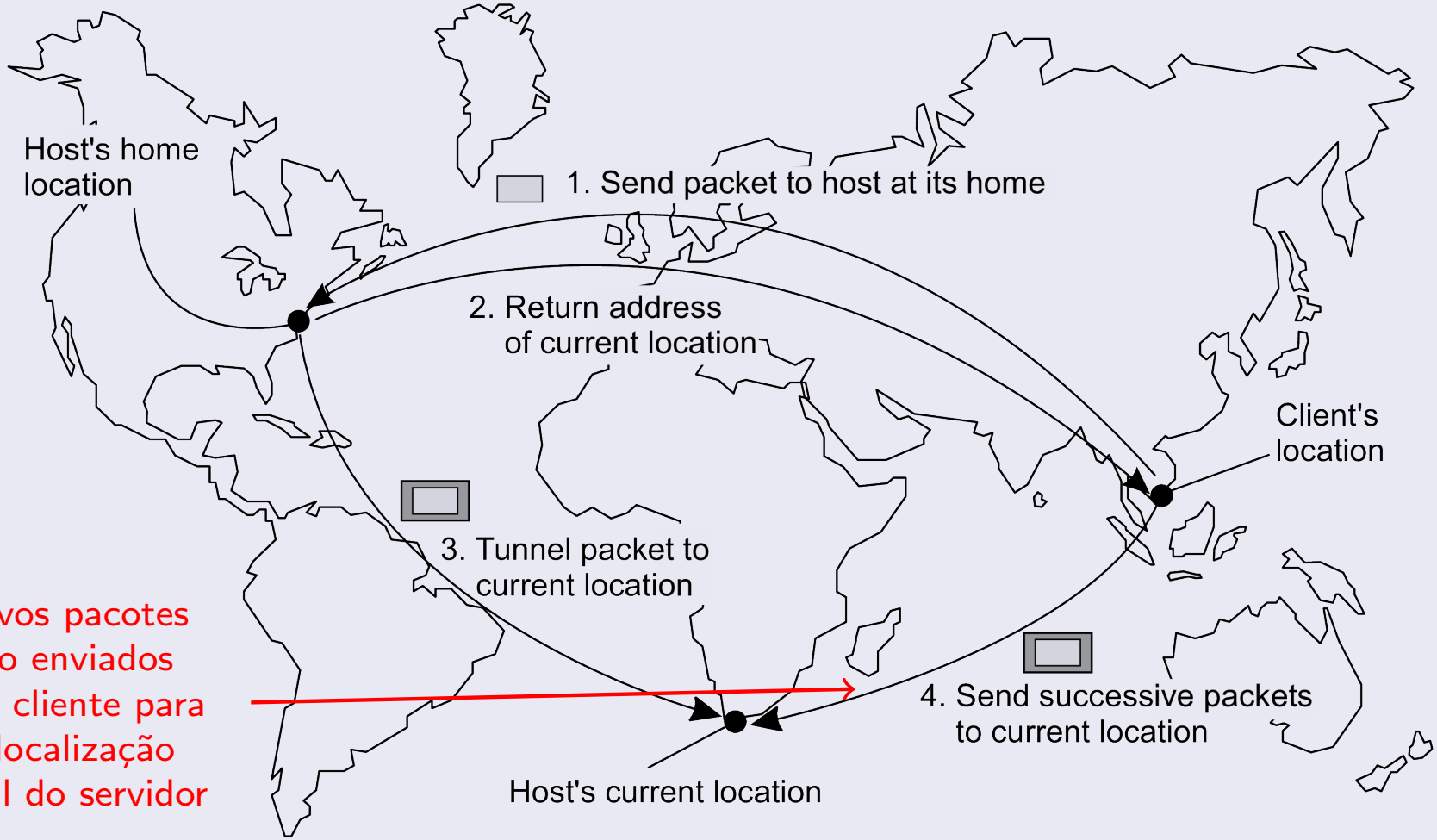
E. N. Plano – Abordagens *Home-based*

Ex: IP móvel



E. N. Plano – Abordagens *Home-based*

Ex: IP móvel



E. N. Plano – Abordagens *Home-based*

Problemas

- O endereço de residência deve existir enquanto a entidade existir

E. N. Plano – Abordagens *Home-based*

Problemas

- O endereço de residência deve existir enquanto a entidade existir
- O endereço é fixo
  - Carga desnecessária se a entidade se mover permanentemente para outro lugar
  - Mudança permanente pode ser tratada com outro nível de nomeação (DNS)

E. N. Plano – Abordagens *Home-based*

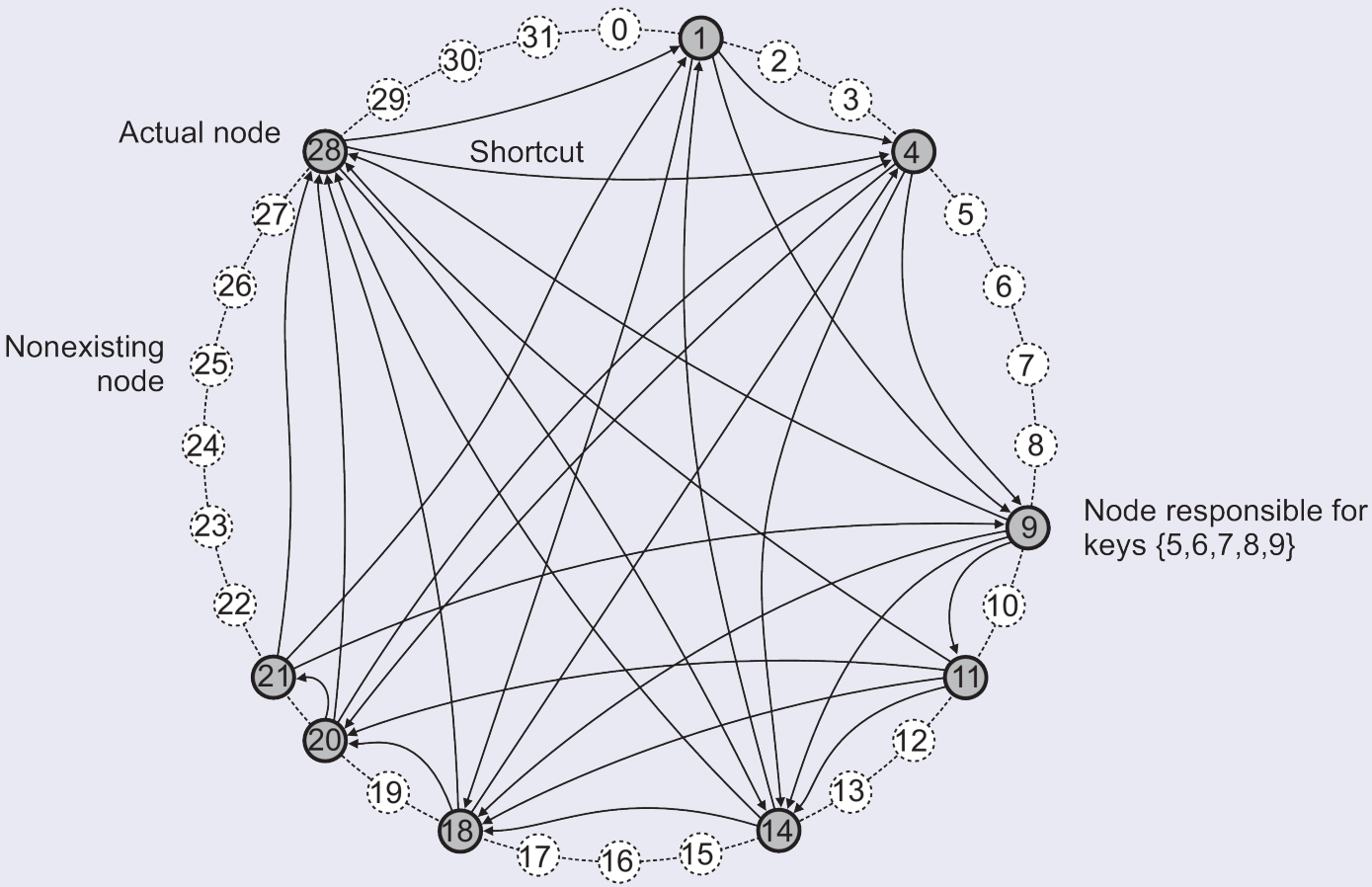
Problemas

- O endereço de residência deve existir enquanto a entidade existir
- O endereço é fixo
  - Carga desnecessária se a entidade se mover permanentemente para outro lugar
  - Mudança permanente pode ser tratada com outro nível de nomeação (DNS)
- Escalabilidade geográfica ruim
  - A entidade pode estar próxima ao cliente

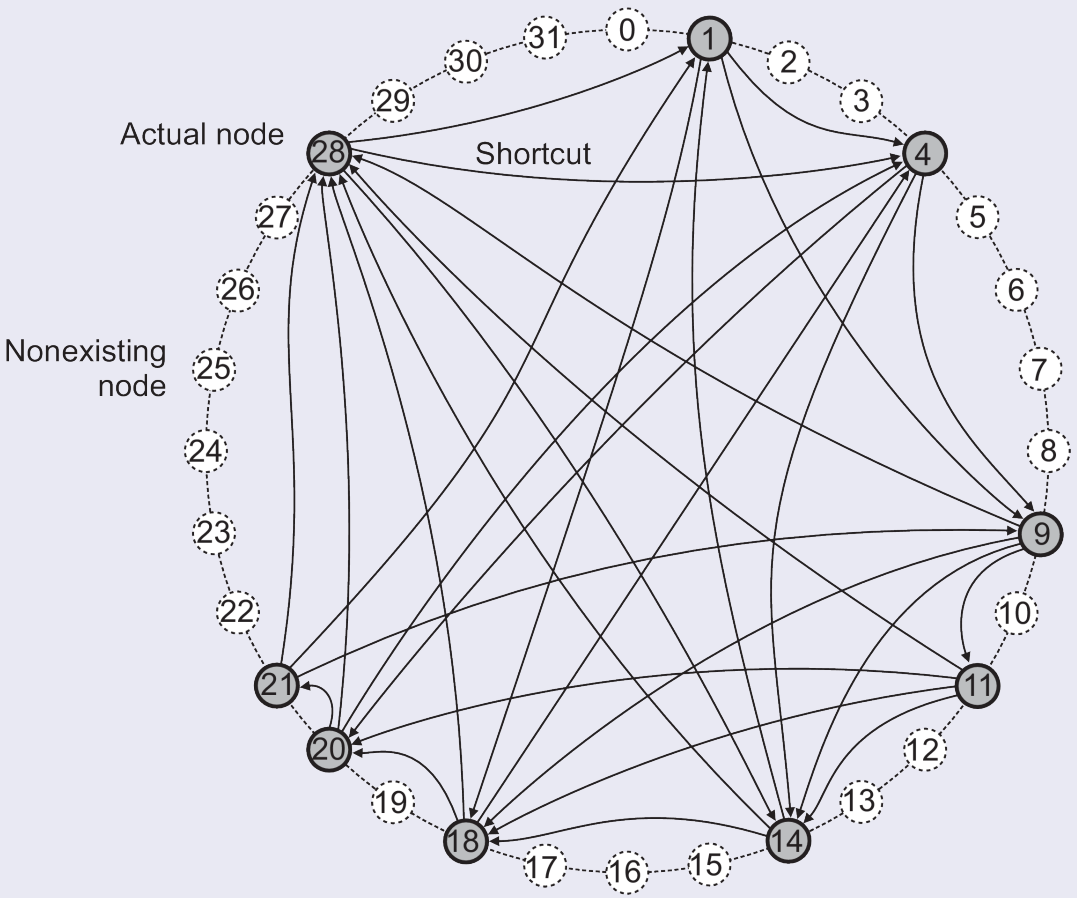
Sistemas de Nomeação

- Nomes, Identificadores e Endereços
- **Espaço de Nomes Plano**
  - Soluções Simples
  - Abordagens *Home-based*
  - **Tabelas de Hash Distribuídas**
  - Abordagens Hierárquicas

Chord – Mecanismo Geral

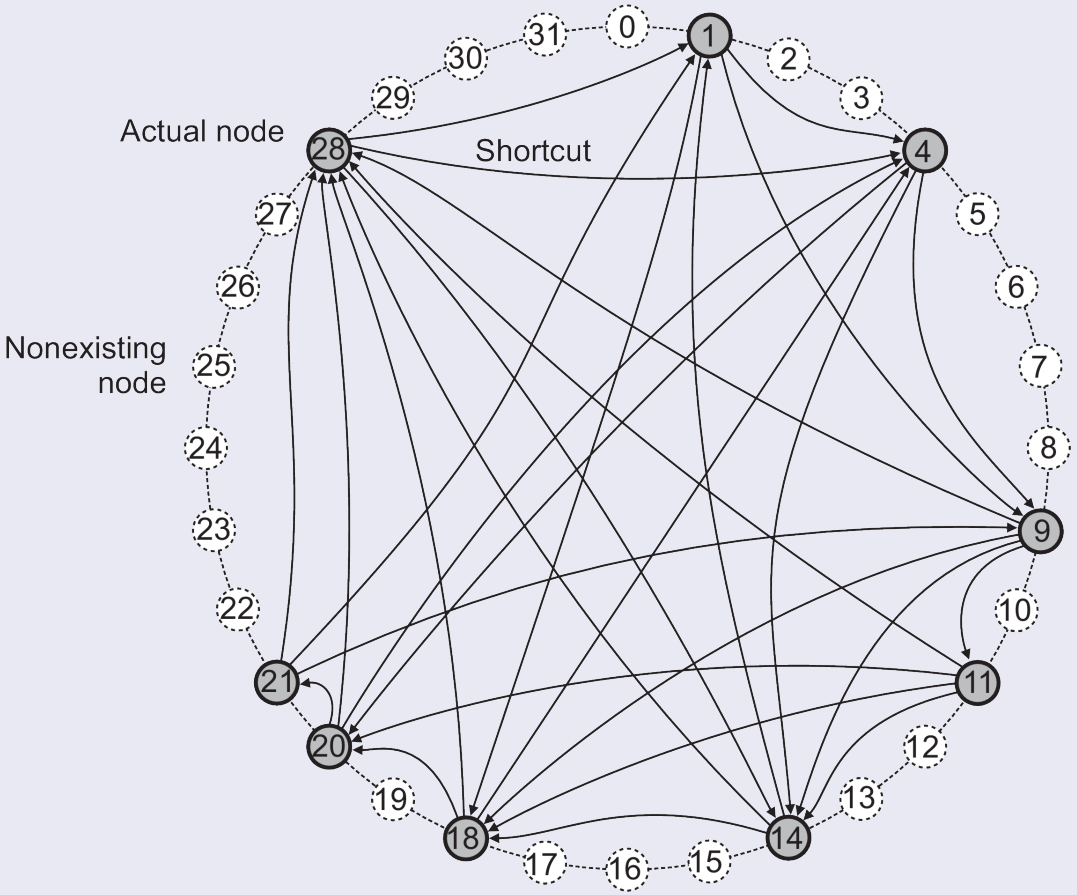


Chord – Mecanismo Geral



Sistema peer-to-peer estruturado, em que os nós são organizados na forma de um anel lógico

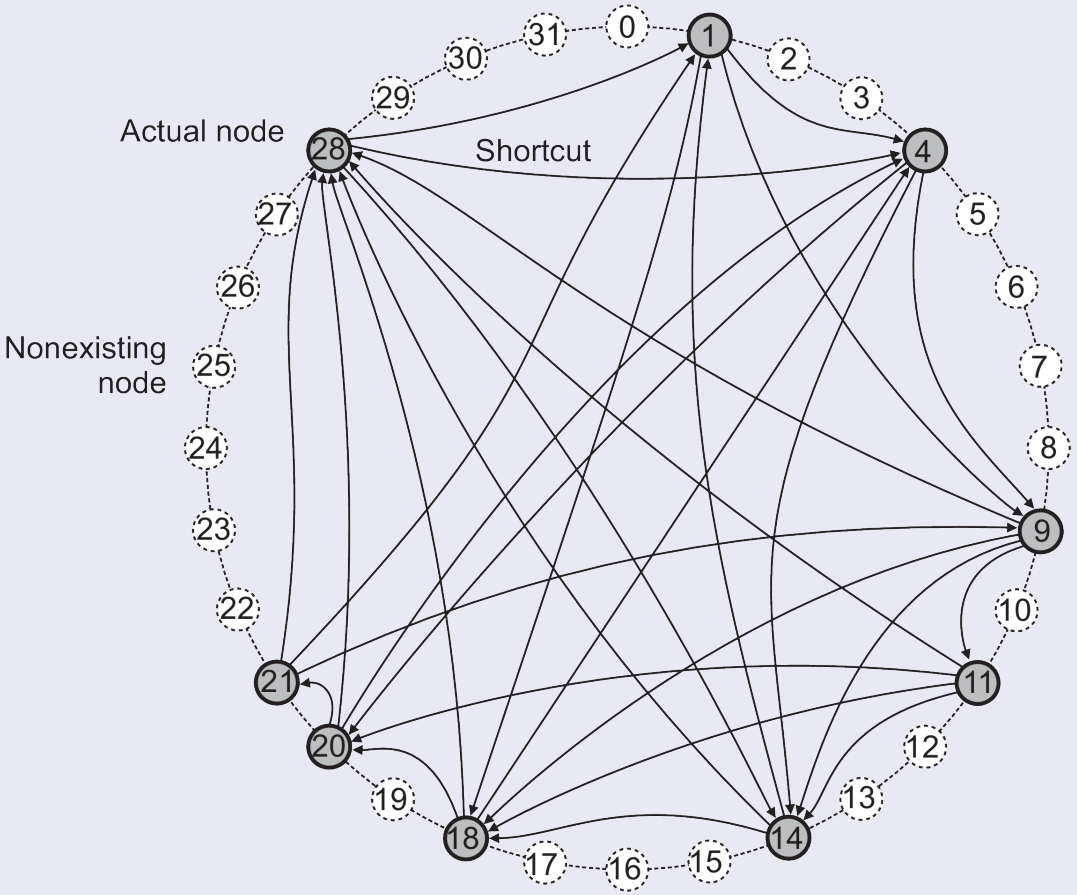
Chord – Mecanismo Geral



Sistema peer-to-peer estruturado, em que os nós são organizados na forma de um anel lógico

A cada nó é atribuído um identificador aleatório de *m*-bits

Chord – Mecanismo Geral

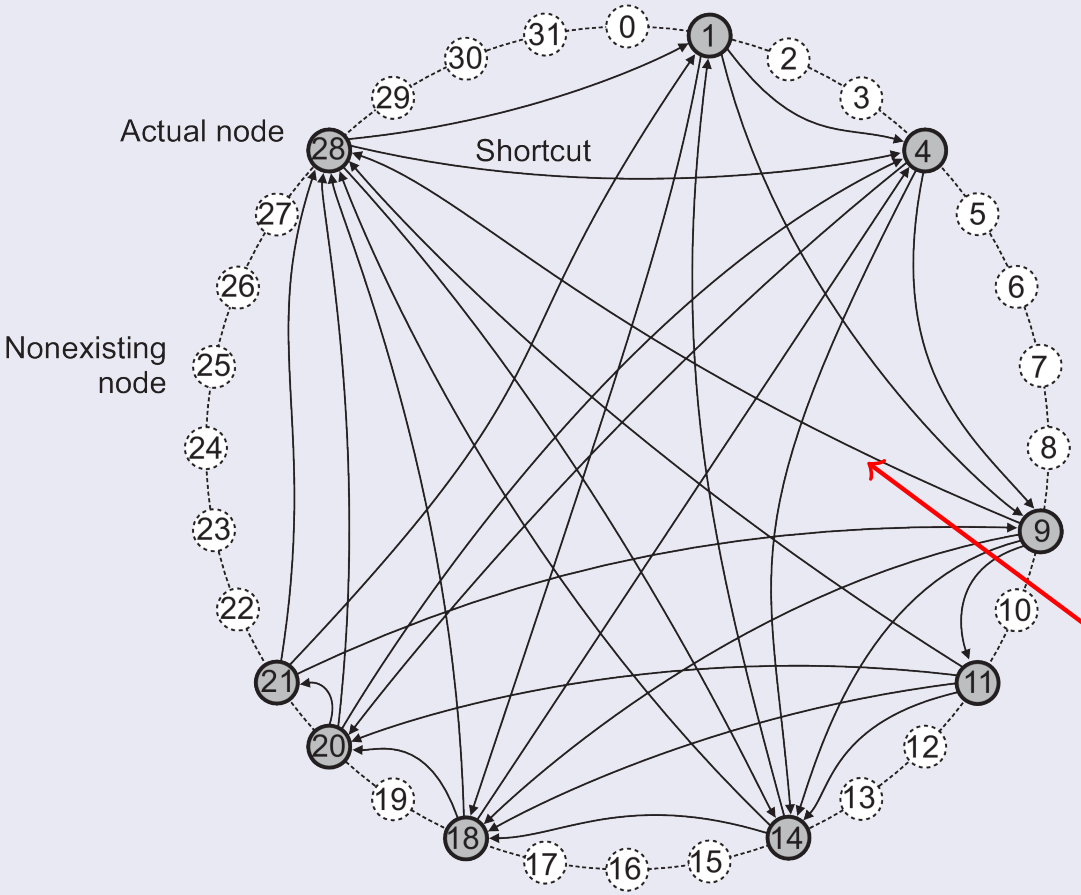


Sistema peer-to-peer estruturado, em que os nós são organizados na forma de um anel lógico

A cada nó é atribuído um identificador aleatório de *m*-bits

A cada entidade é atribuída uma única chave de *m*-bits

Chord – Mecanismo Geral



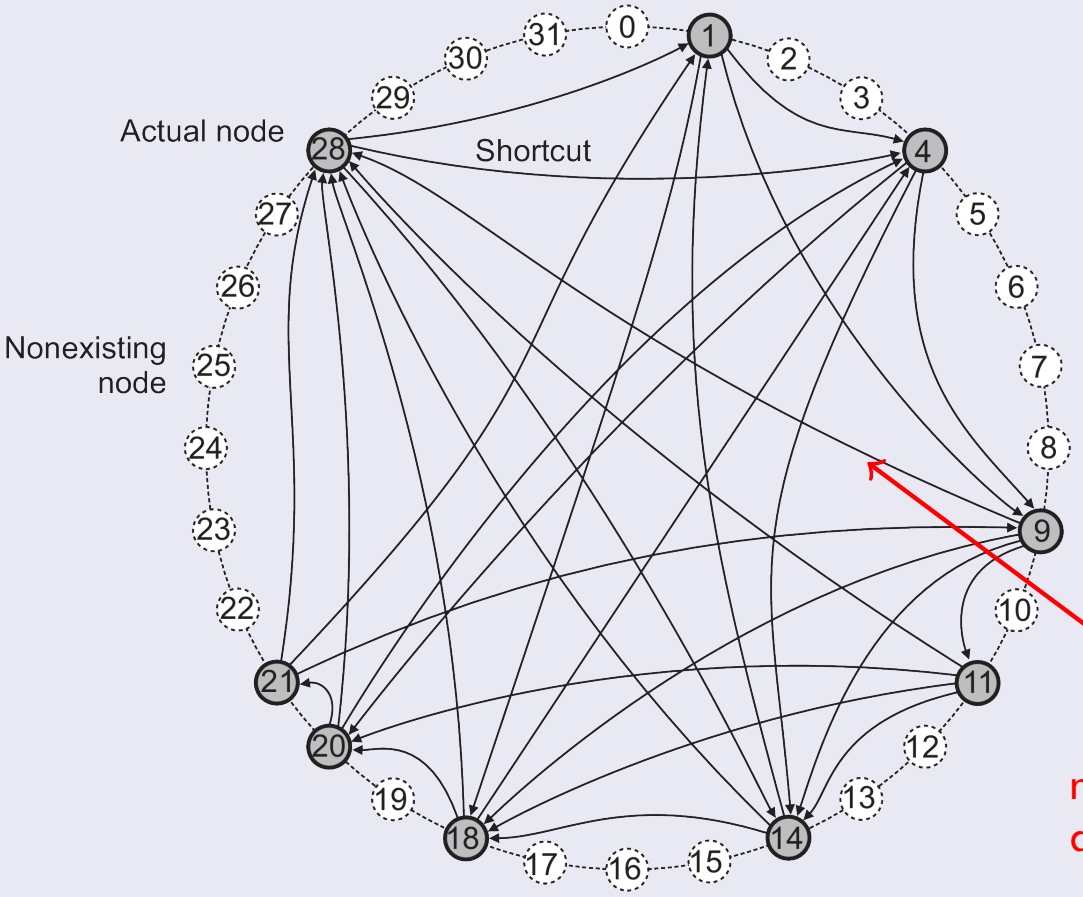
Sistema peer-to-peer estruturado, em que os nós são organizados na forma de um anel lógico

A cada nó é atribuído um identificador aleatório de *m*-bits

A cada entidade é atribuída uma única chave de *m*-bits

O anel é estendido com vários links de atalho para outros nós

Chord – Mecanismo Geral



Sistema peer-to-peer estruturado, em que os nós são organizados na forma de um anel lógico

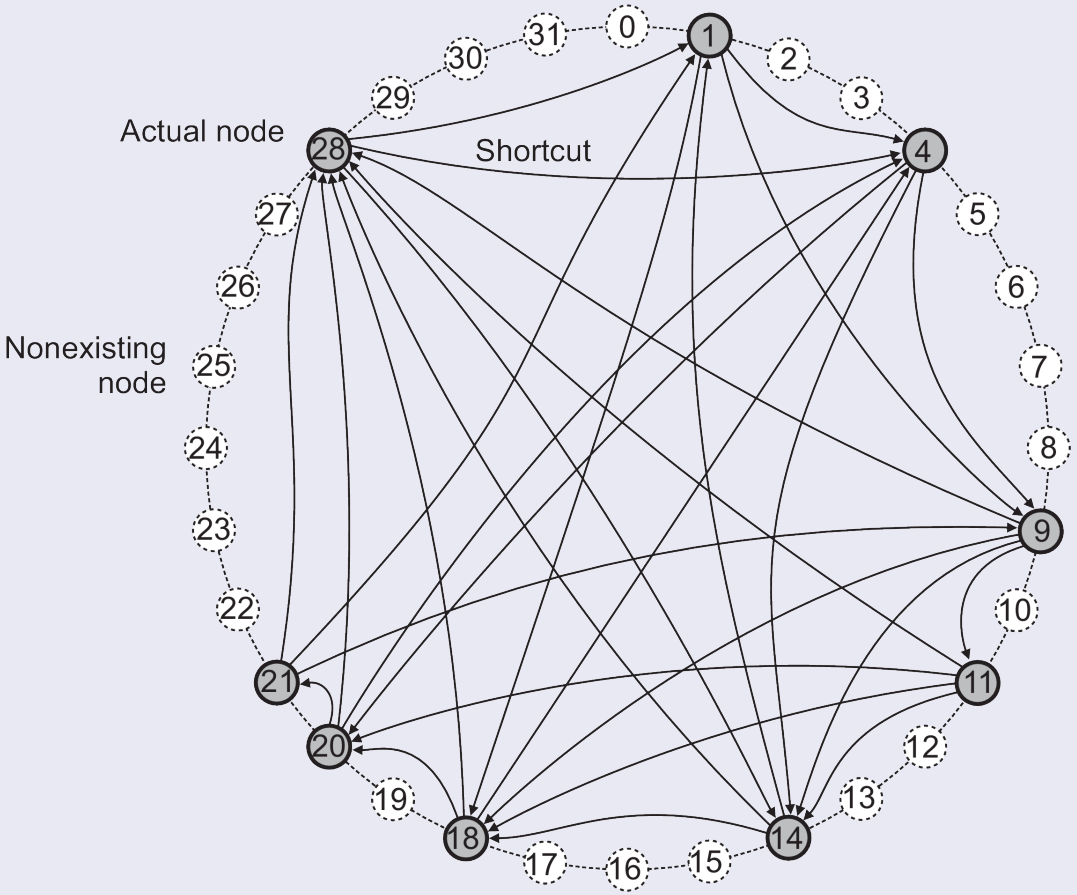
A cada nó é atribuído um identificador aleatório de *m*-bits

A cada entidade é atribuída uma única chave de *m*-bits

O anel é estendido com vários links de atalho para outros nós

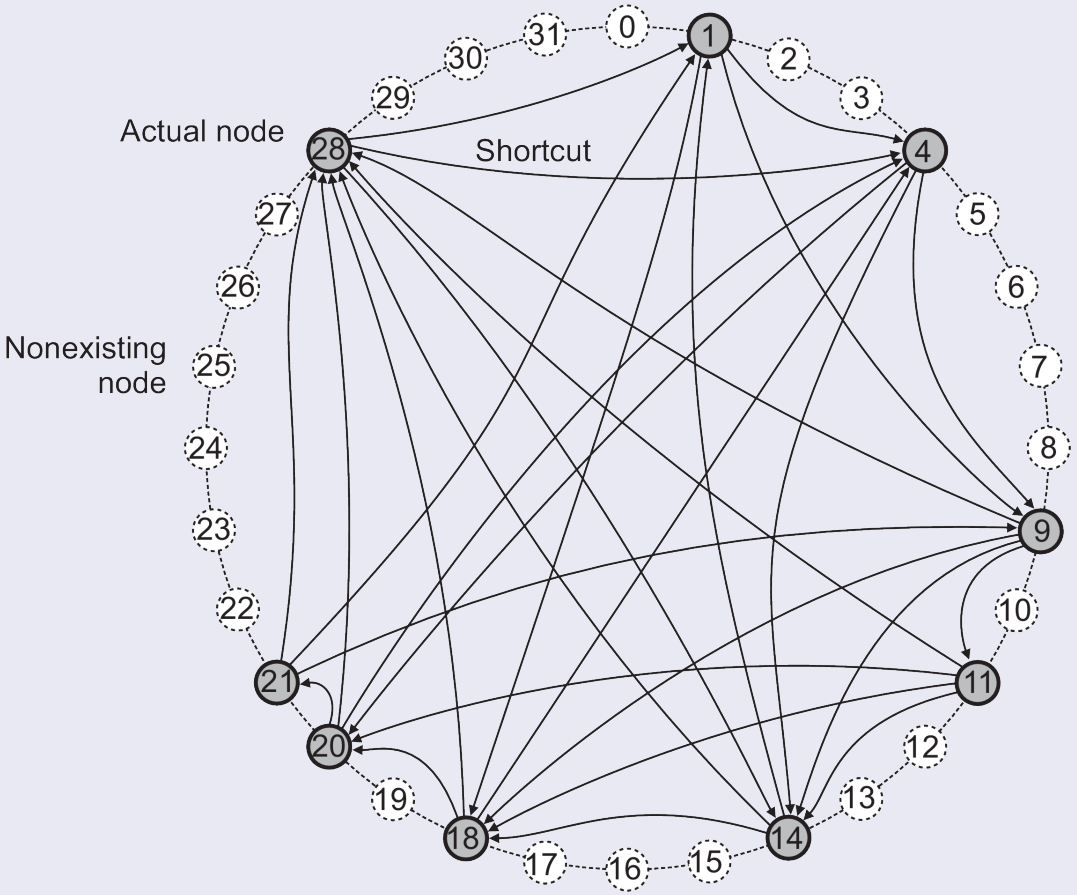
Construído de modo que o comprimento do menor caminho entre qualquer par de nós é da ordem de O(log N), onde N é o número total de nós

Chord – Mecanismo Geral



Entidades com chave k são armazenadas no nó com o menor $id \geq k$ (seu sucessor $suc(k)$)

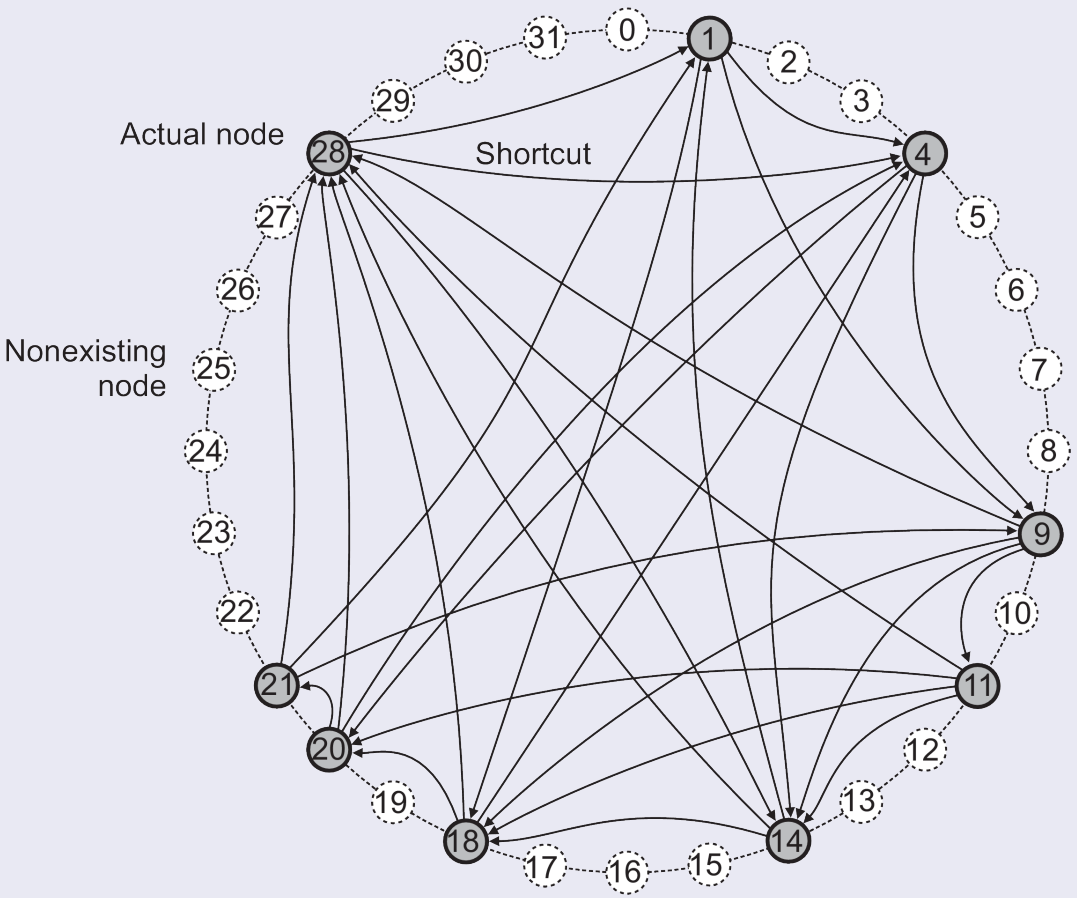
Chord – Mecanismo Geral



Entidades com chave k são armazenadas no nó com o menor $id \geq k$ (seu sucessor $suc(k)$)

No exemplo, $m = 5$ bits e a rede contém 9 nós: {1, 4, 9, 11, 14, 18, 20, 21, 28}

Chord – Mecanismo Geral

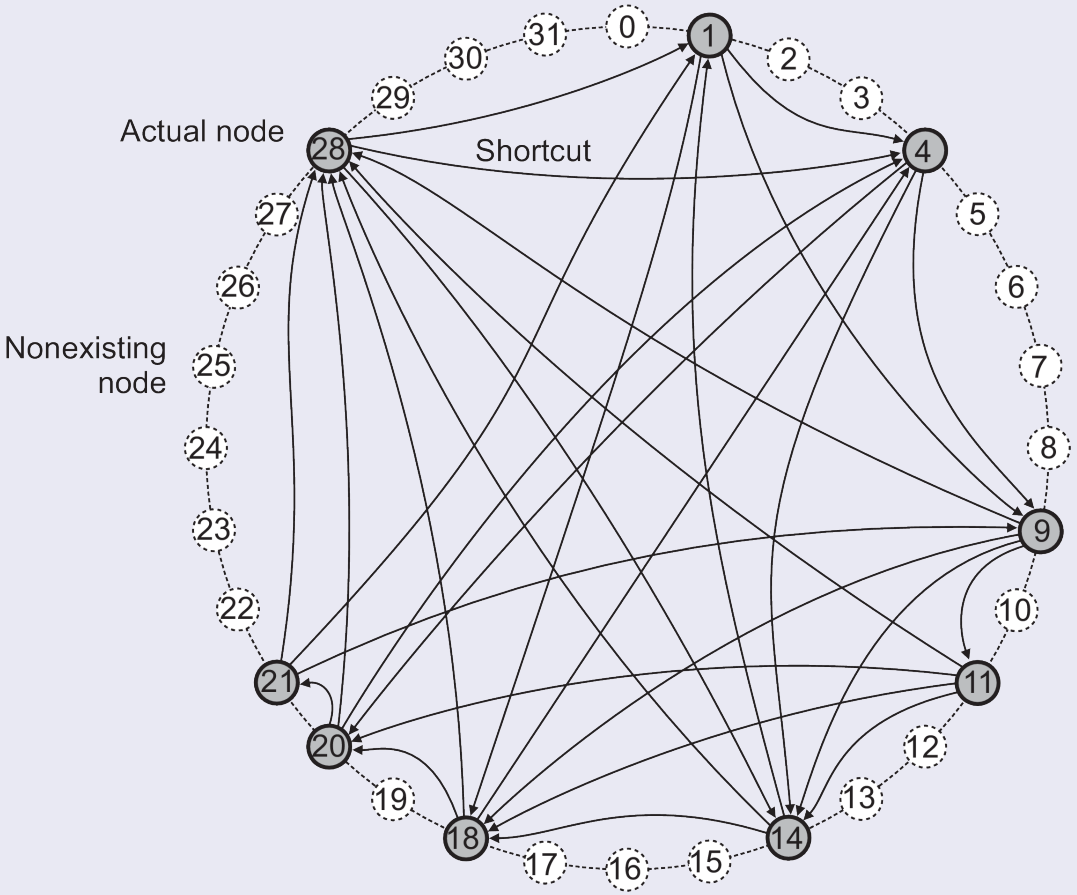


Entidades com chave k são armazenadas no nó com o menor $id \geq k$ (seu sucessor $suc(k)$)

No exemplo, $m = 5$ bits e a rede contém 9 nós: $\{1, 4, 9, 11, 14, 18, 20, 21, 28\}$

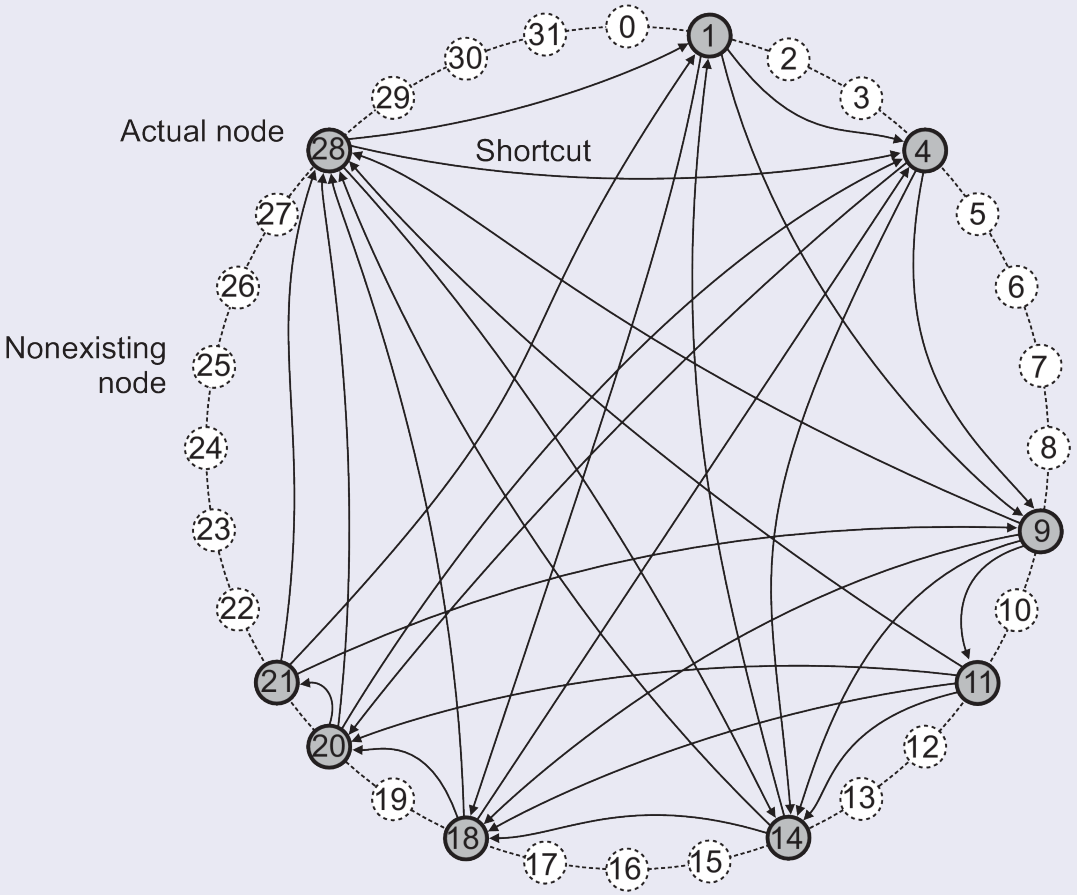
Entidades com $k = 5, 6, 7, 8, 9$ são armazenadas no nó 9, seu sucessor

Chord – Mecanismo Geral



E como fazemos para buscar uma entidade (uma chave) nesse modelo?

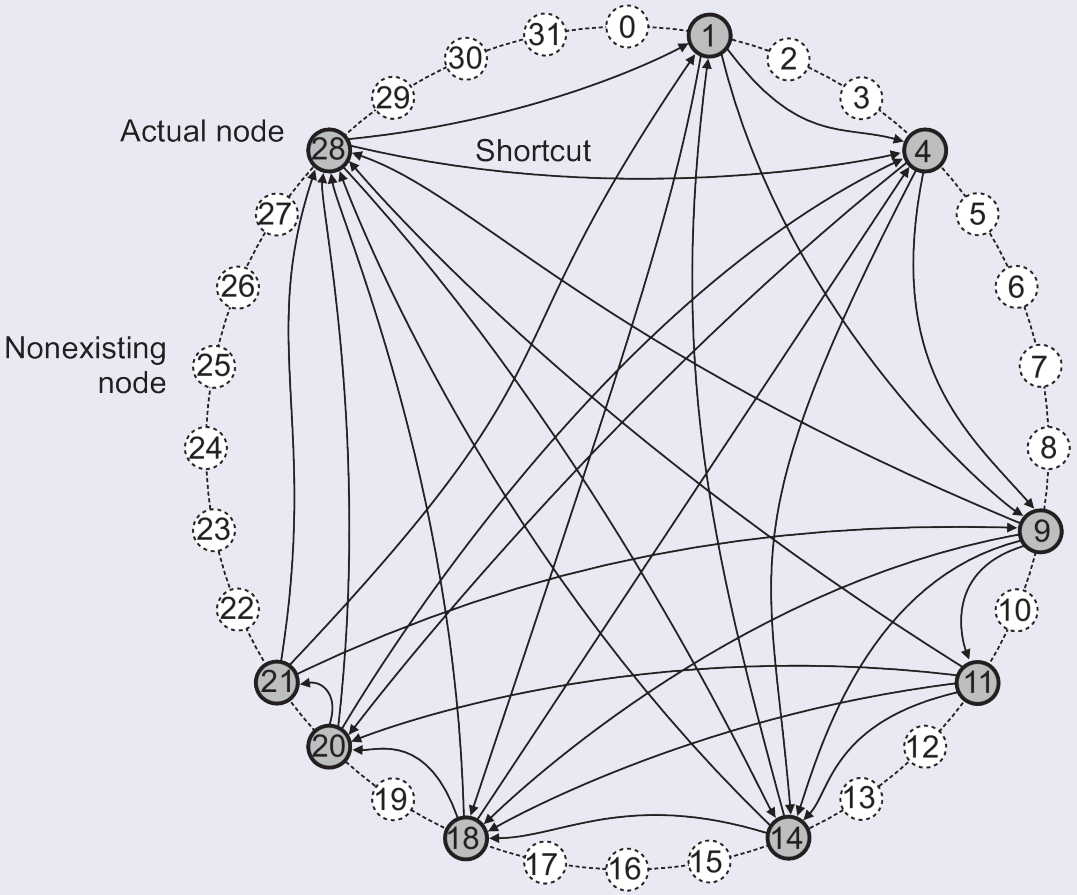
Chord – Mecanismo Geral



E como fazemos para buscar uma entidade (uma chave) nesse modelo?

Faça com que cada nó mantenha o registro de seus vizinhos e faça uma busca linear ao longo do anel (**solução ruim**)

Chord – Mecanismo Geral



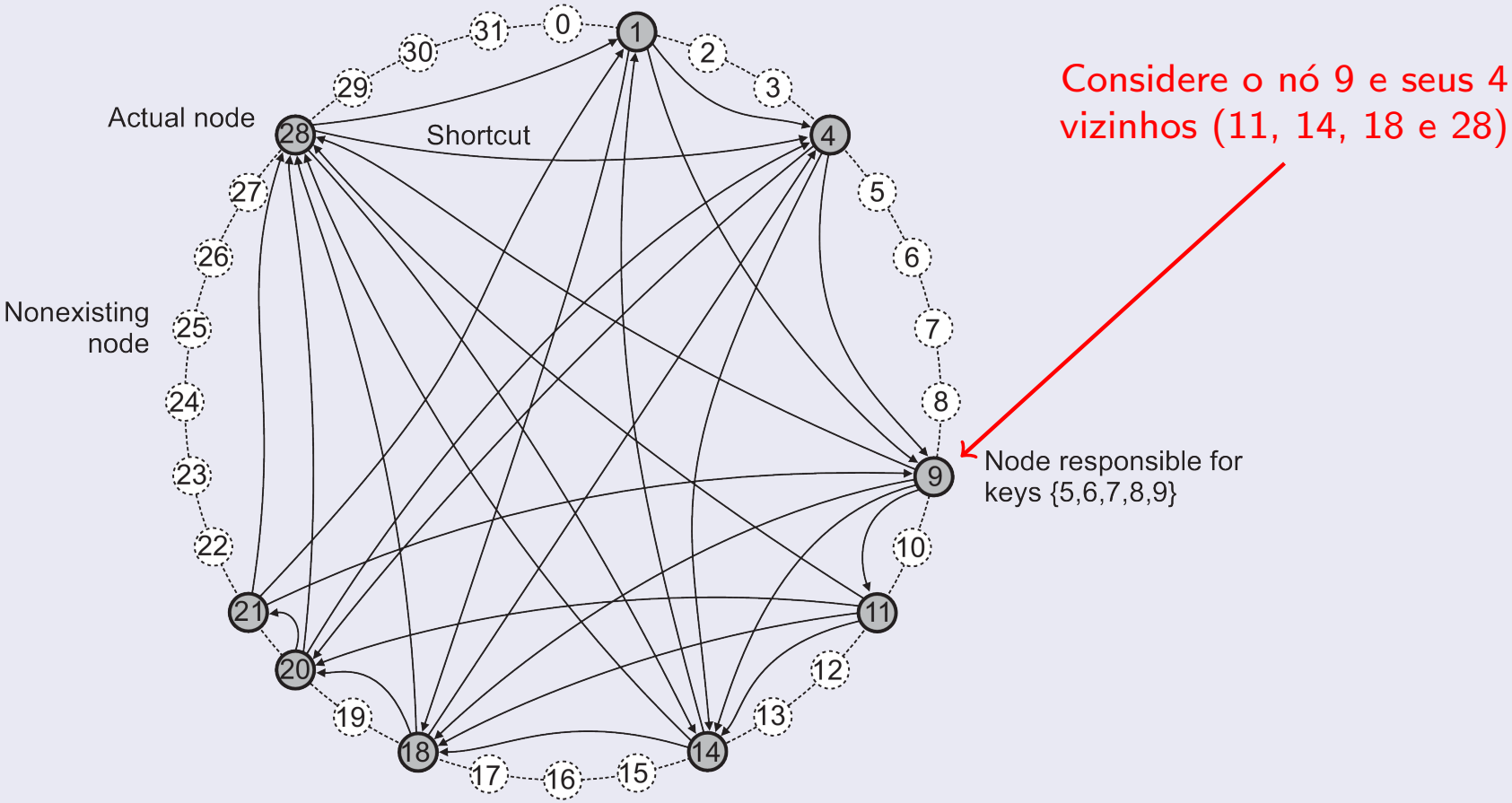
E como fazemos para buscar uma entidade (uma chave) nesse modelo?

Faça com que cada nó mantenha o registro de seus vizinhos e faça uma busca linear ao longo do anel (**solução ruim**)

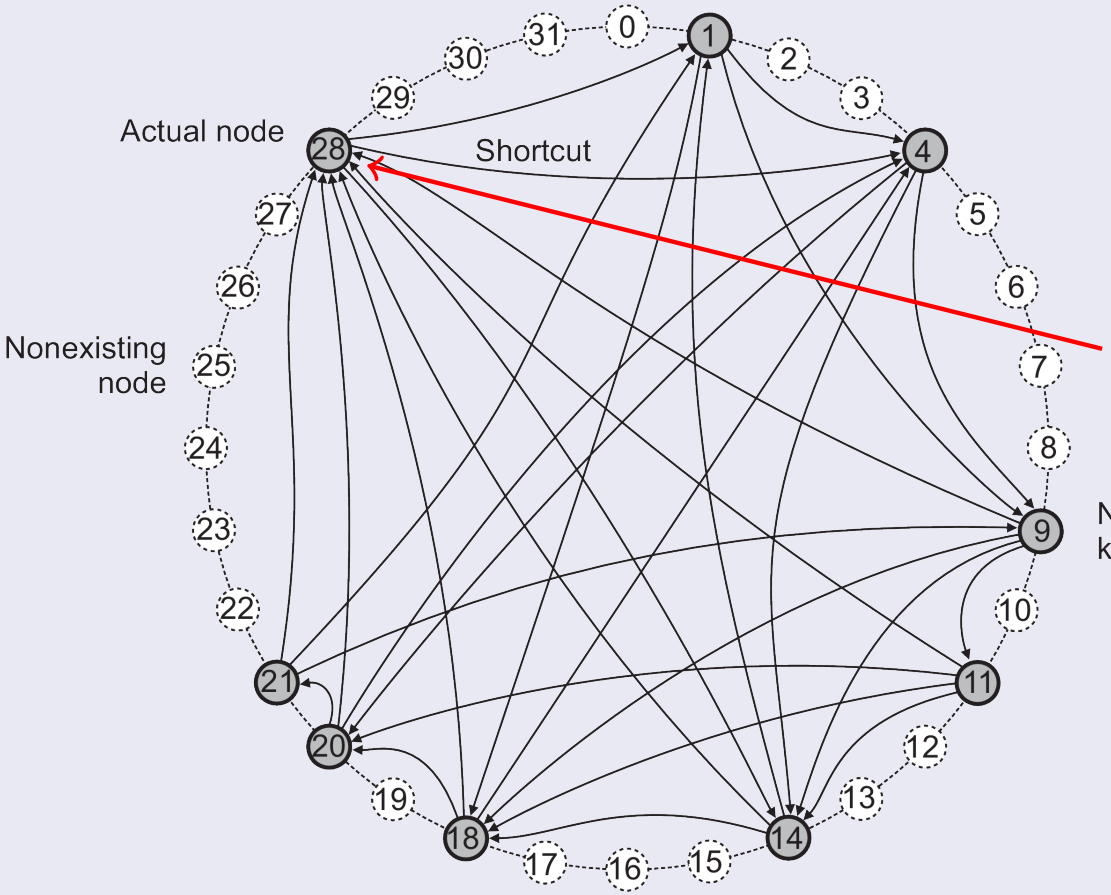
Ou tente passar a requisição o mais longe possível, mas sem ultrapassar o nó responsável por ela

ENP – Tabelas de *Hash* Distribuídas

Chord – Mecanismo Geral



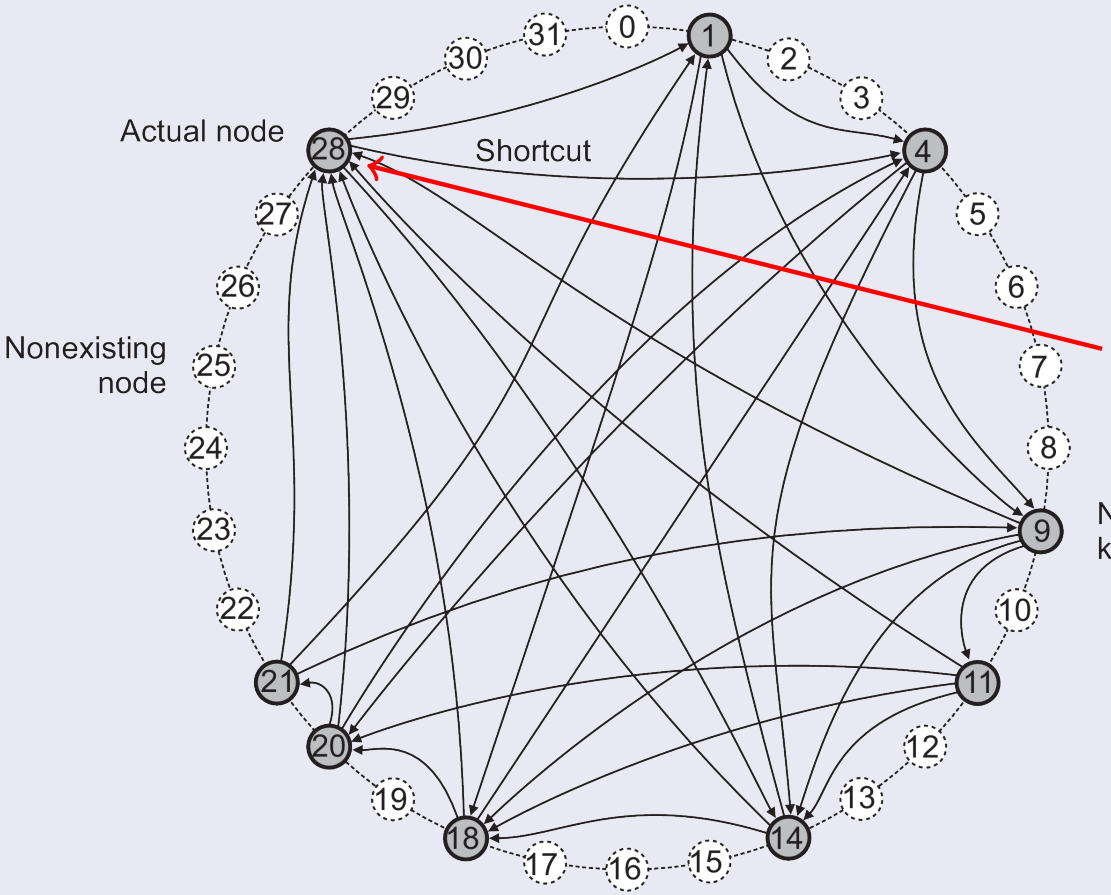
Chord – Mecanismo Geral



Considere o nó 9 e seus 4 vizinhos (11, 14, 18 e 28)

Se ele receber uma requisição para a chave 3 (nó 4), a repassará para o vizinho mais longe dele (28), mas que ainda preceda o nó responsável pela chave

Chord – Mecanismo Geral



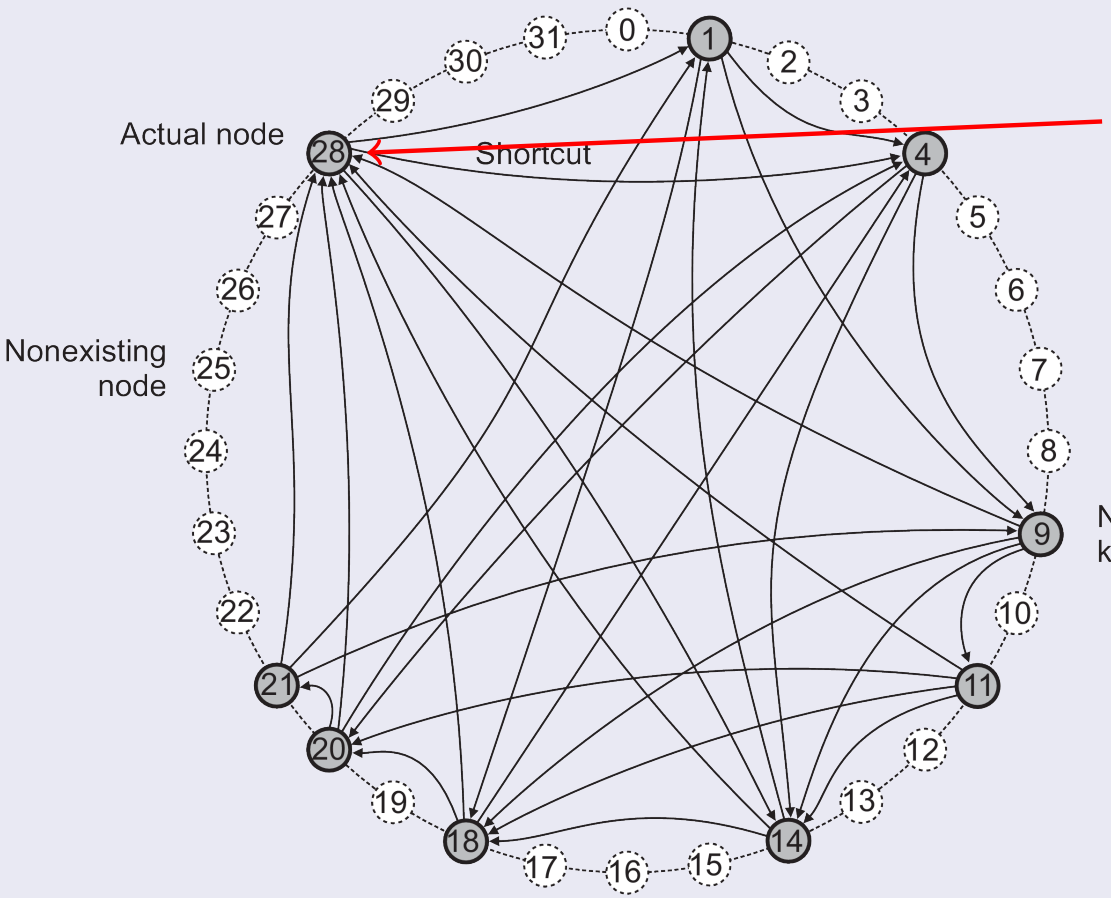
Considere o nó 9 e seus 4 vizinhos (11, 14, 18 e 28)

Se ele receber uma requisição para a chave 3 (nó 4), a repassará para o vizinho mais longe dele (28), mas que ainda preceda o nó responsável pela chave

Lembre que o sistema é organizado como um anel

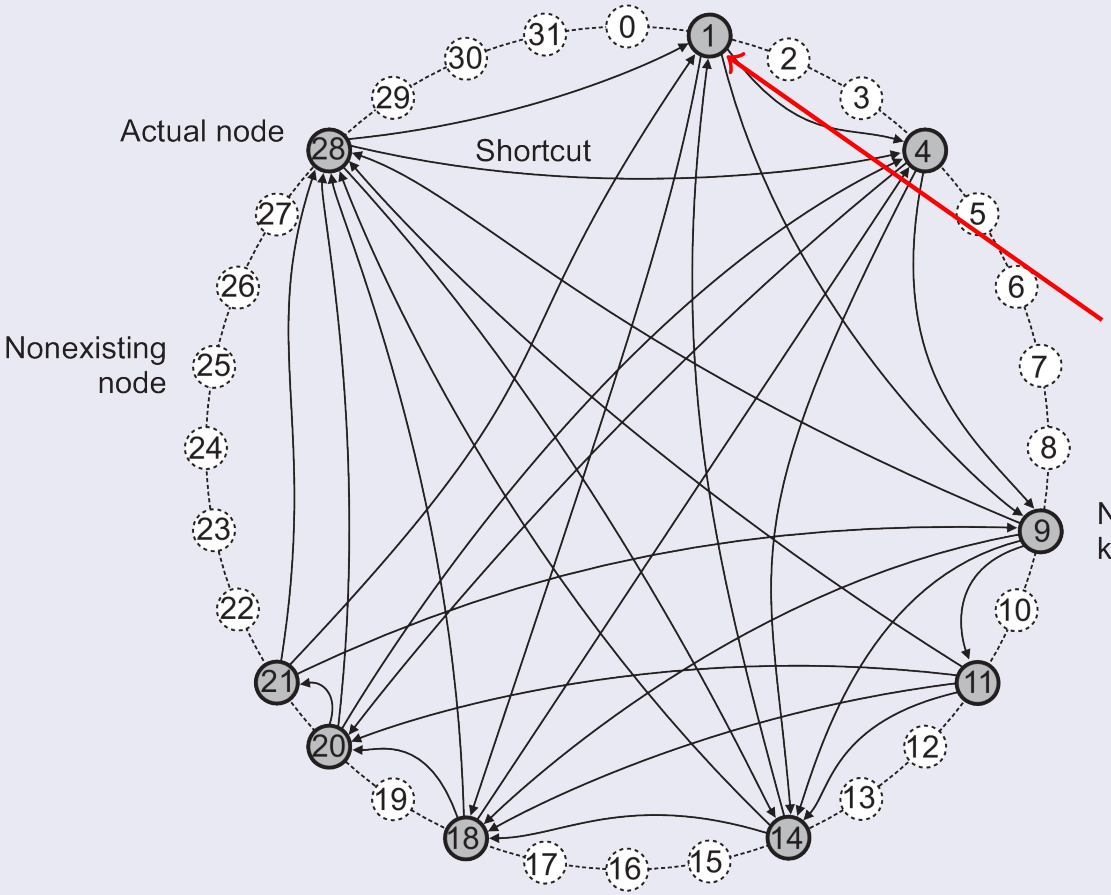
ENP – Tabelas de *Hash* Distribuídas

Chord – Mecanismo Geral



O nó 28, por sua vez, possui 3 vizinhos: 1, 4 e 14

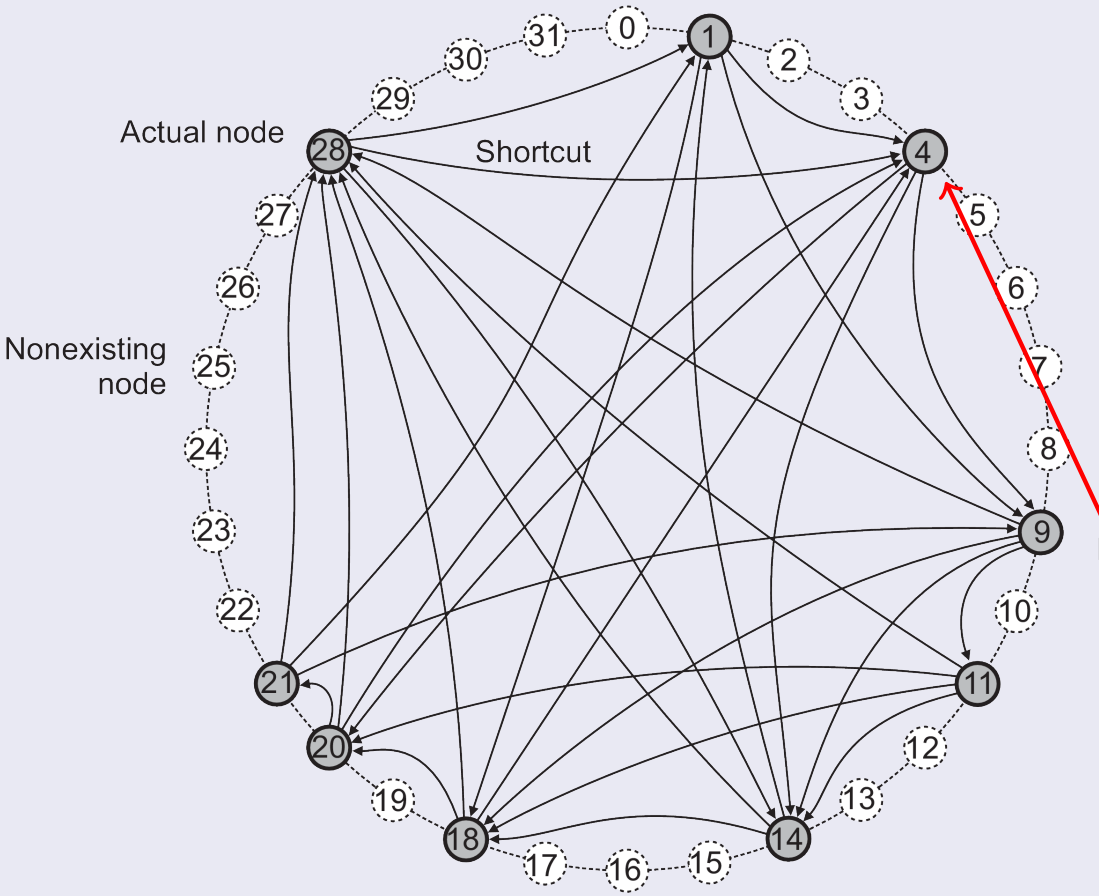
Chord – Mecanismo Geral



O nó 28, por sua vez, possui 3 vizinhos: 1, 4 e 14

Como ele não sabe se há um nó 3 (em que estaria a chave), repassa a requisição a 1, o nó mais longínquo que não ultrapassa a chave

Chord – Mecanismo Geral

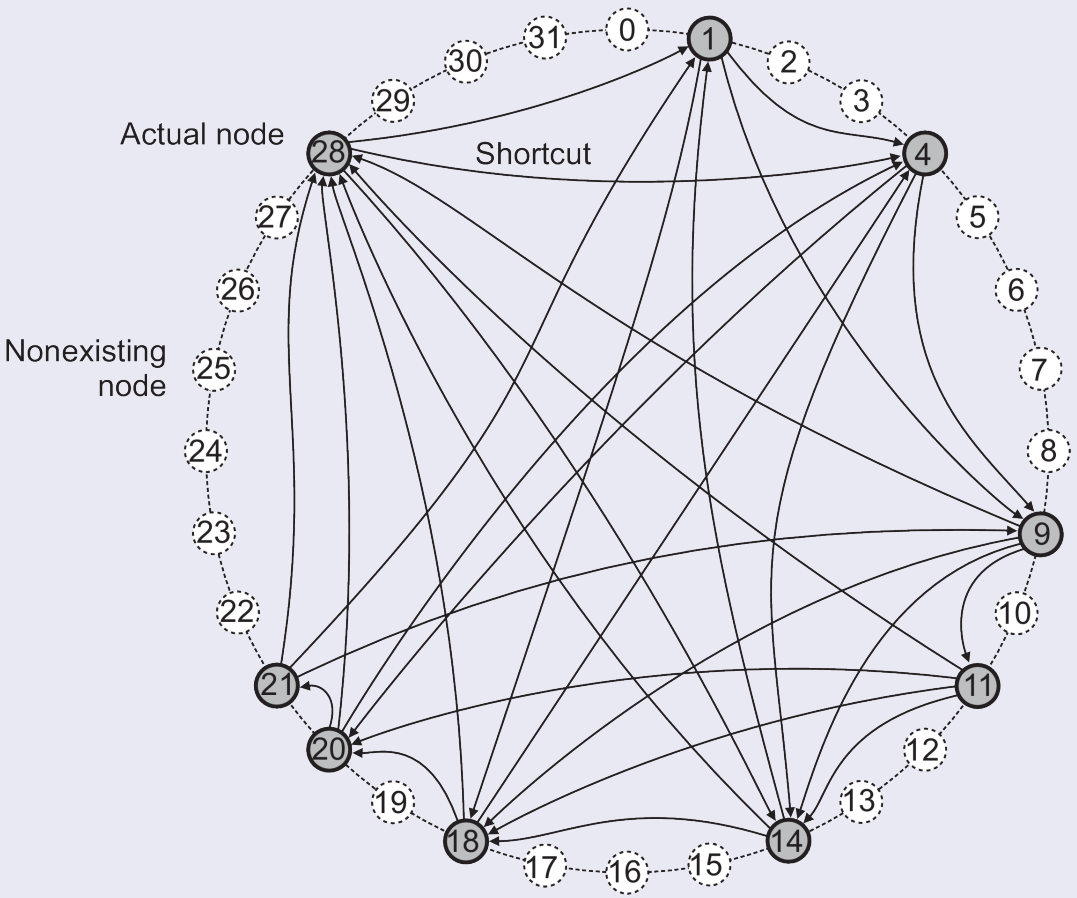


O nó 28, por sua vez, possui 3 vizinhos: 1, 4 e 14

Como ele não sabe se há um nó 3 (em que estaria a chave), repassa a requisição a 1, o nó mais longínquo que não ultrapassa a chave

Este sabe que seu sucessor é o 4 e que, portanto, não há como haver um nó 3, e envia a requisição a este, possuidor da entidade 3

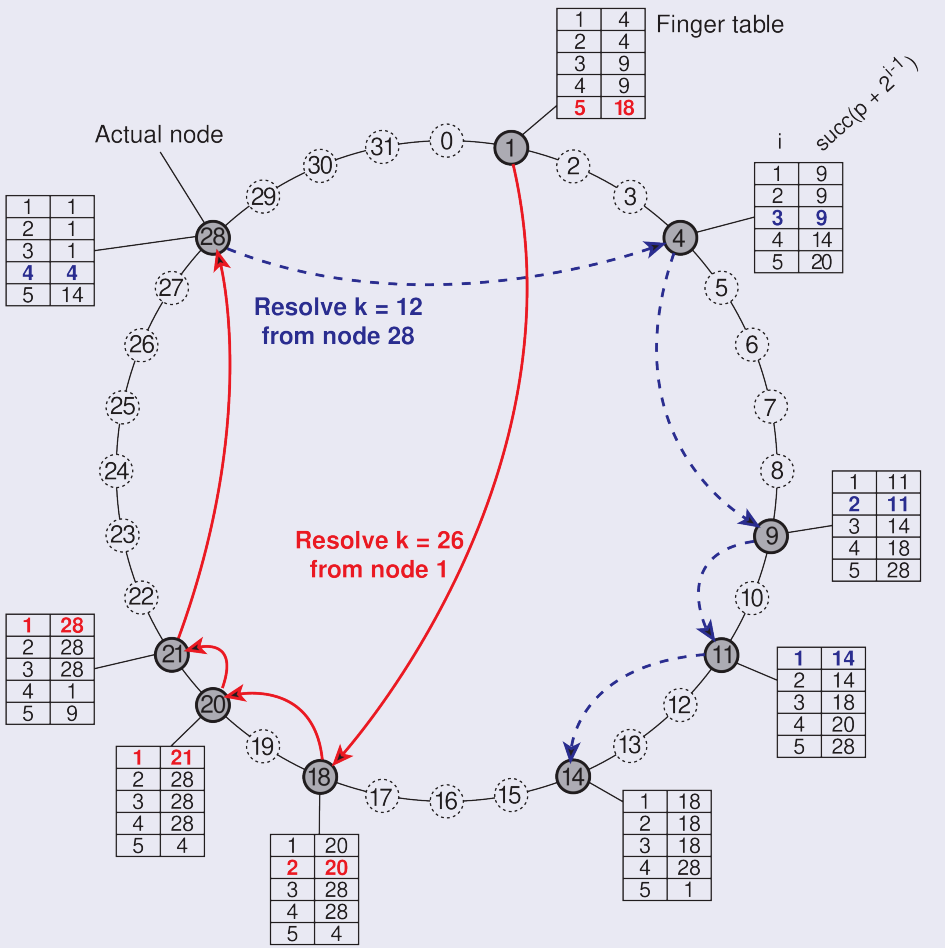
Chord – Mecanismo Geral



E como podemos implementar isso tudo?

ENP – Tabelas de *Hash* Distribuídas

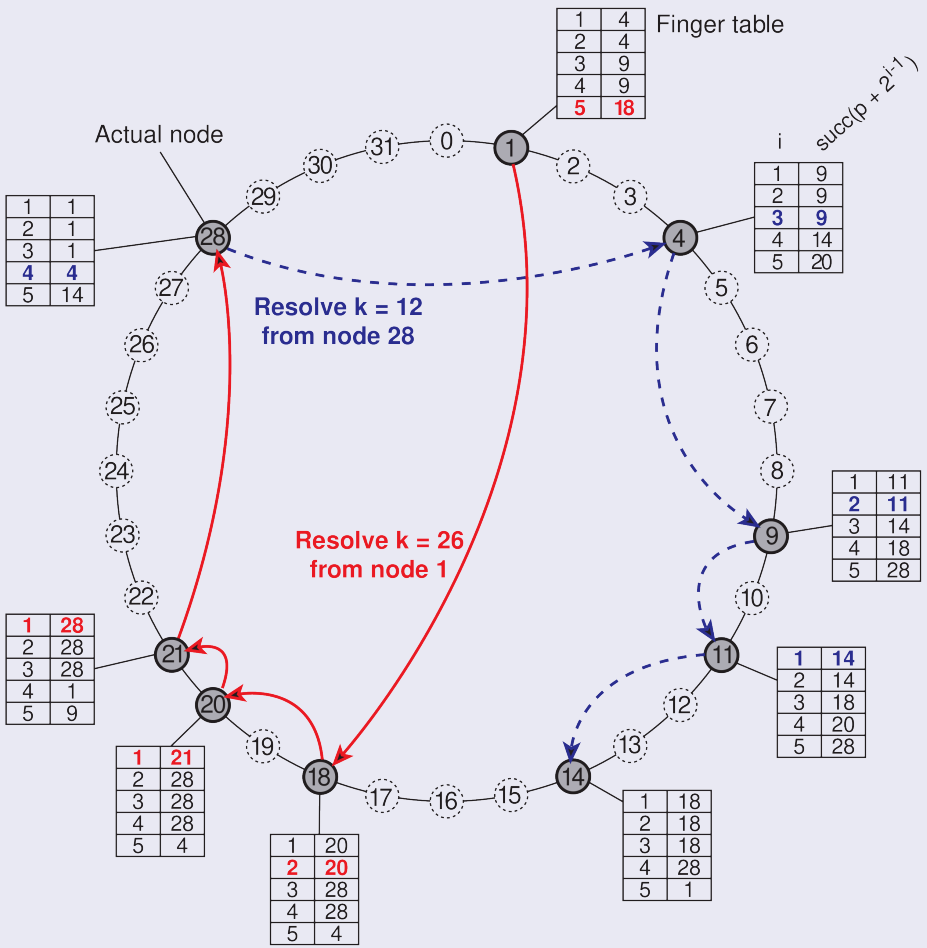
Chord – Mecanismo Geral



E como podemos implementar isso tudo?

Fazendo com que cada nó p mantenha uma *finger table* $FT_p[\,]$ com no máximo m entradas

Chord – Mecanismo Geral

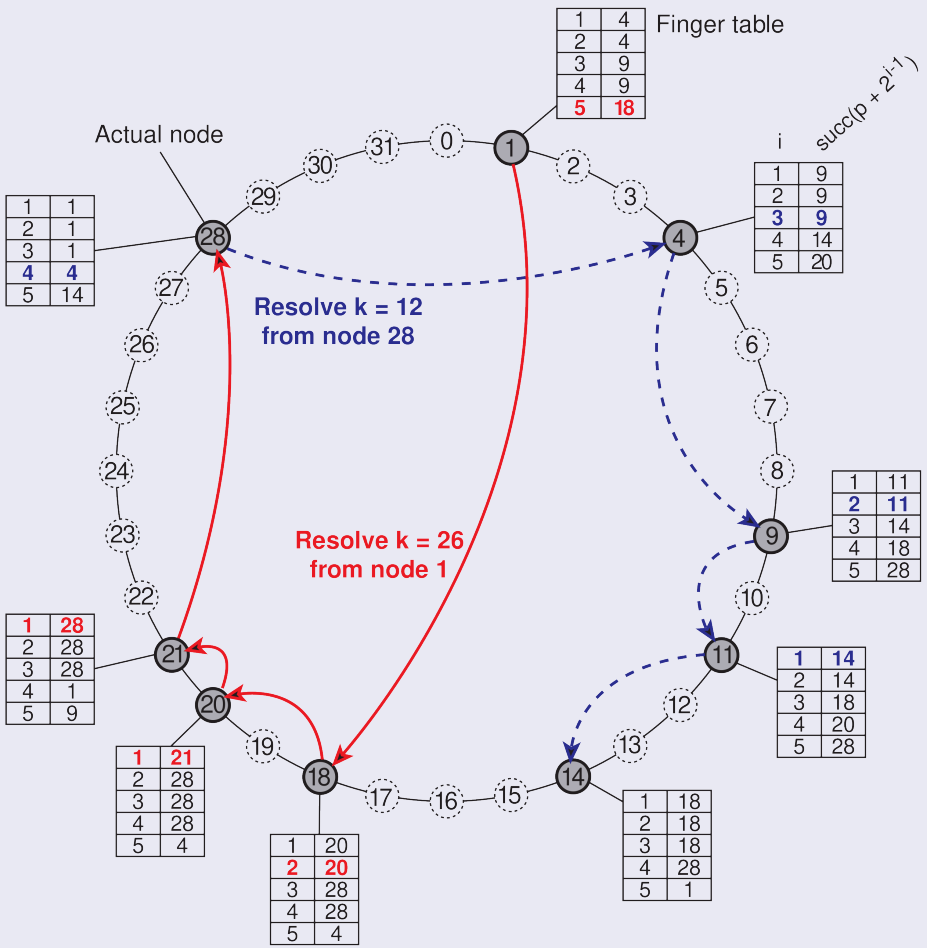


E como podemos implementar isso tudo?

Fazendo com que cada nó p mantenha uma *finger table* $FT_p[\,]$ com no máximo m entradas

$$FT_p[i] = succ(p + 2^{i-1})$$

Chord – Mecanismo Geral



E como podemos implementar isso tudo?

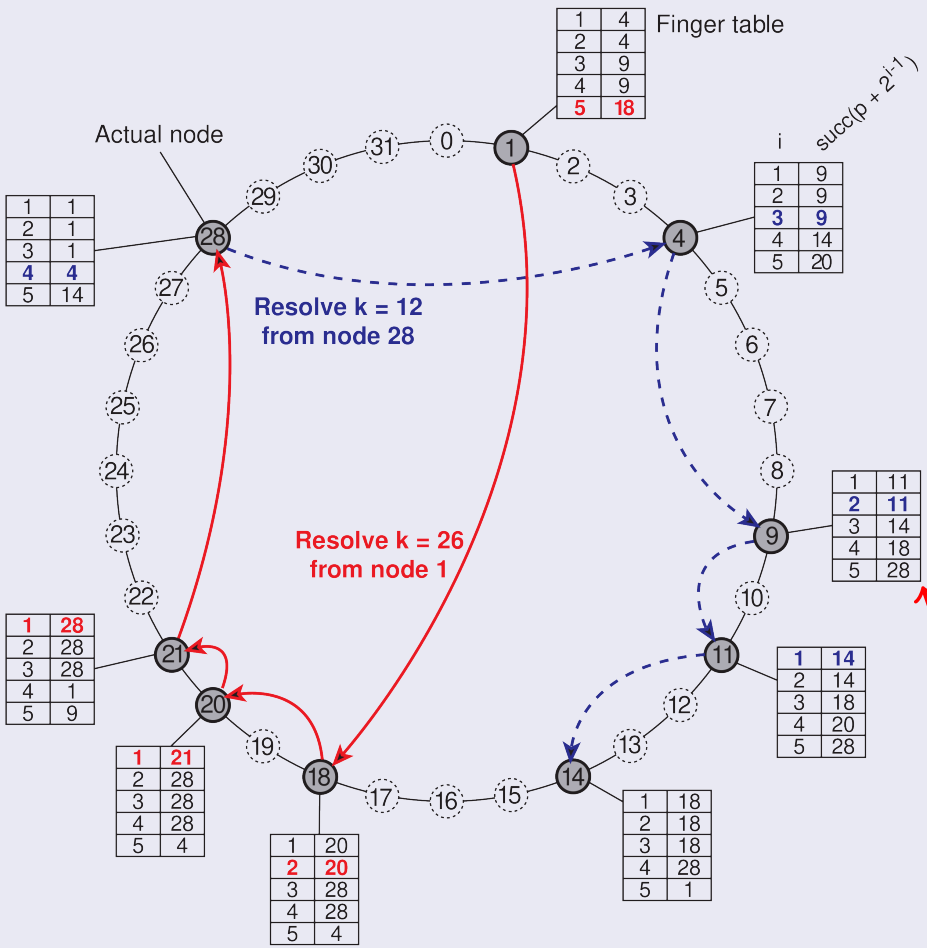
Fazendo com que cada nó p mantenha uma *finger table* $FT_p[\,]$ com no máximo m entradas

$$FT_p[i] = succ(p + 2^{i-1})$$

$FT_p[i]$ aponta para o primeiro nó que sucede p com distância de pelo menos 2^{i-1} posições

ENP – Tabelas de *Hash* Distribuídas

Chord – Mecanismo Geral



E como podemos implementar isso tudo?

Fazendo com que cada nó p mantenha uma *finger table* $FT_p[\,]$ com no máximo m entradas

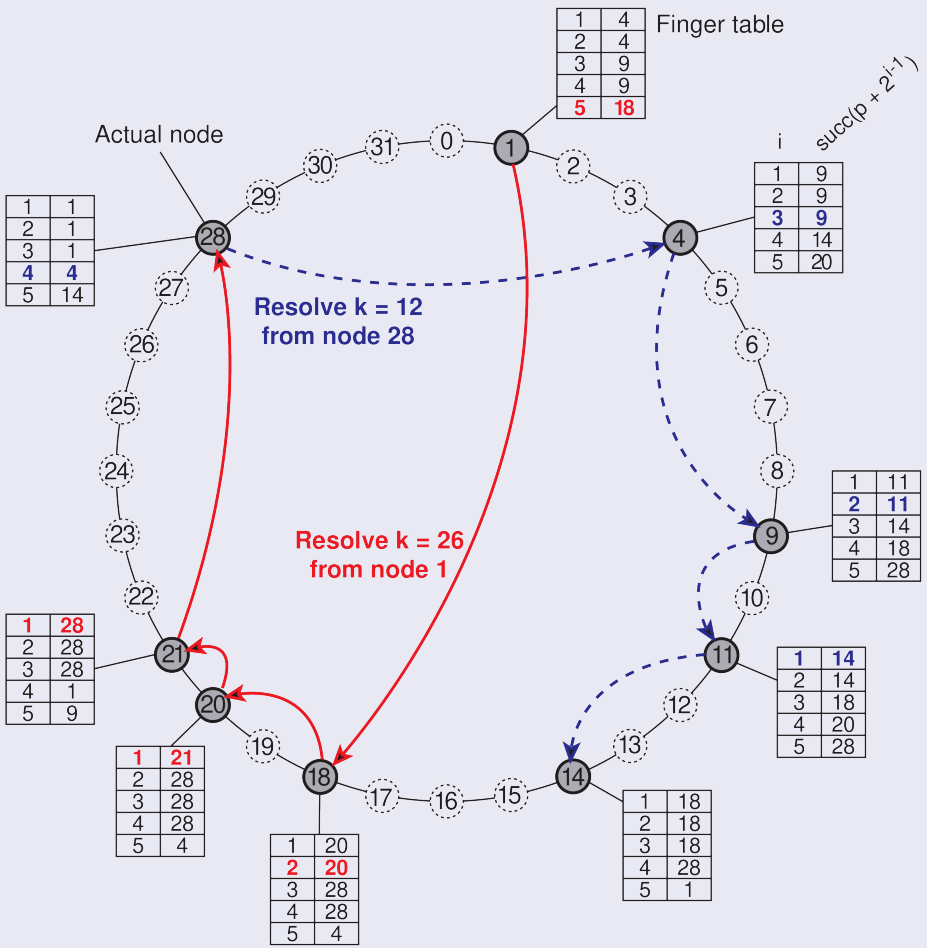
$$FT_p[i] = succ(p + 2^{i-1})$$

$FT_p[i]$ aponta para o primeiro nó que sucede p com distância de pelo menos 2^{i-1} posições

Então a chave 10, a 1 posição de 9, estaria na posição $2^{1-1} = 1$, correspondente ao nó $FT_p[1] = 11$

ENP – Tabelas de *Hash* Distribuídas

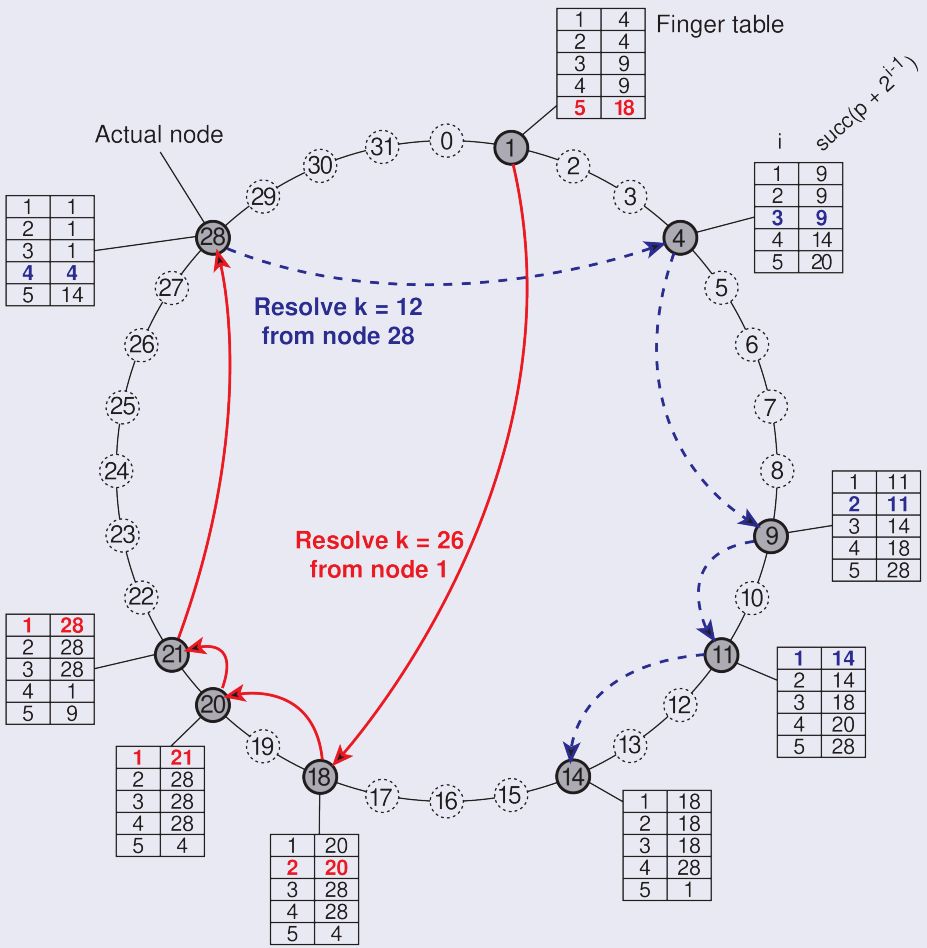
Chord – Mecanismo Geral



Para procurar por uma chave k, o nó p encaminha o pedido para o nó q com índice j tal que $q = FT_p[j] \leq k < FT_p[j+1]$ (posições consecutivas na tabela)

ENP – Tabelas de *Hash* Distribuídas

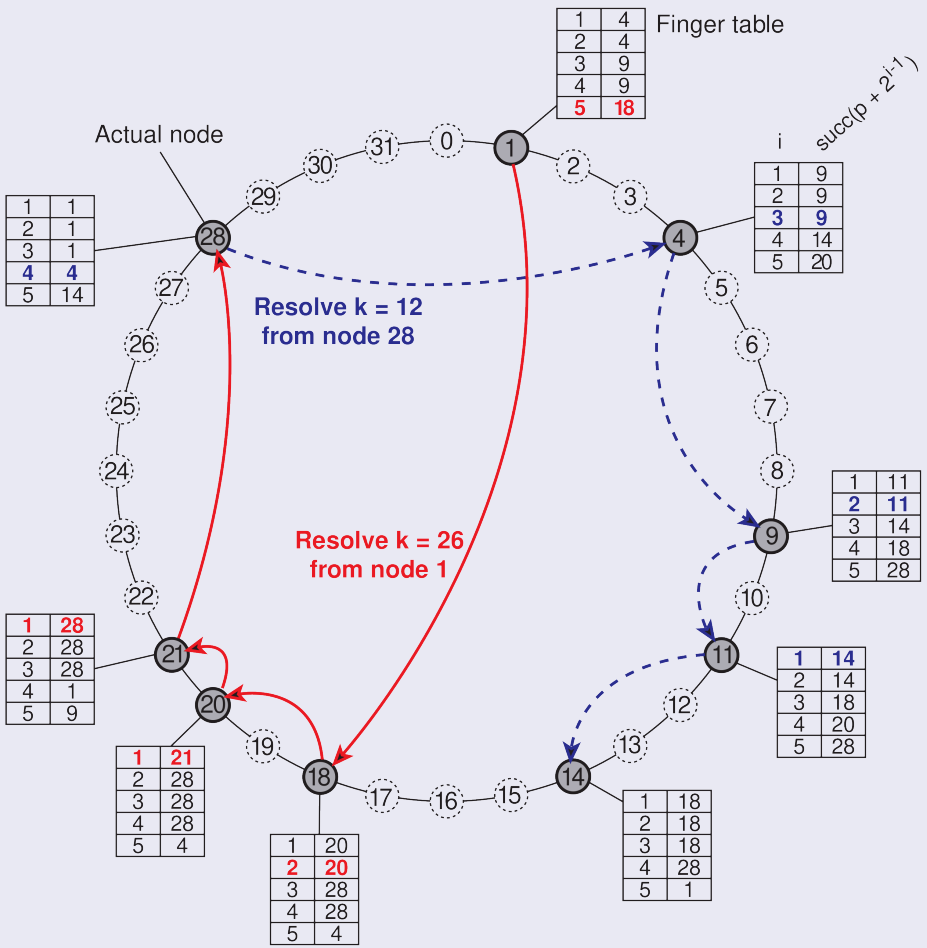
Chord – Mecanismo Geral



Para procurar por uma chave k, o nó p encaminha o pedido para o nó q com índice j tal que $q = FT_p[j] \leq k < FT_p[j+1]$ (posições consecutivas na tabela)

Se $p < k < FT_p[1]$, a requisição é encaminhada para $q = FT_p[1]$

Chord – Mecanismo Geral

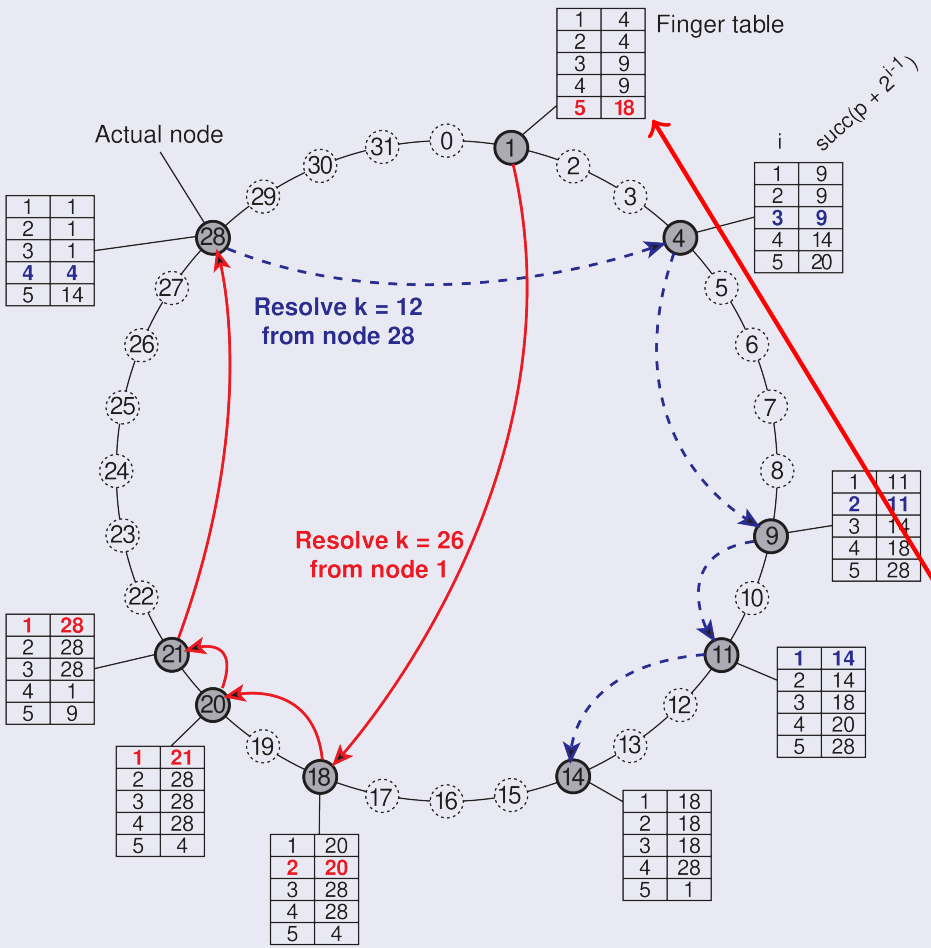


Para procurar por uma chave k, o nó p encaminha o pedido para o nó q com índice j tal que $q = FT_p[j] \leq k < FT_p[j+1]$ (posições consecutivas na tabela)

Se $p < k < FT_p[1]$, a requisição é encaminhada para $q = FT_p[1]$

Ex: Resolver $k = 26$ a partir do nó 1

Chord – Mecanismo Geral



Para procurar por uma chave k, o nó p encaminha o pedido para o nó q com índice j tal que $q = FT_p[j] \leq k < FT_p[j+1]$ (posições consecutivas na tabela)

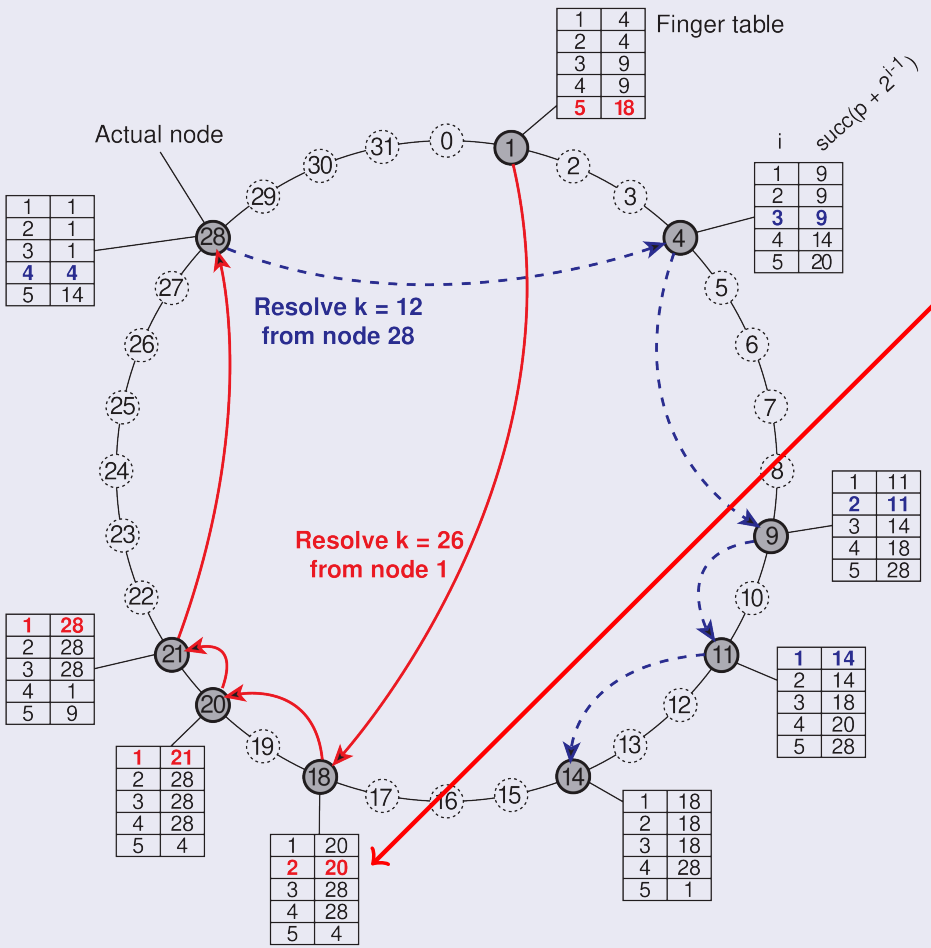
Se $p < k < FT_p[1]$, a requisição é encaminhada para $q = FT_p[1]$

Ex: Resolver $k = 26$ a partir do nó 1

O nó 1 busca $k = 26$ em sua tabela. Como $26 > FT_1[5]$, a requisição será transmitida ao nó 18

ENP – Tabelas de *Hash* Distribuídas

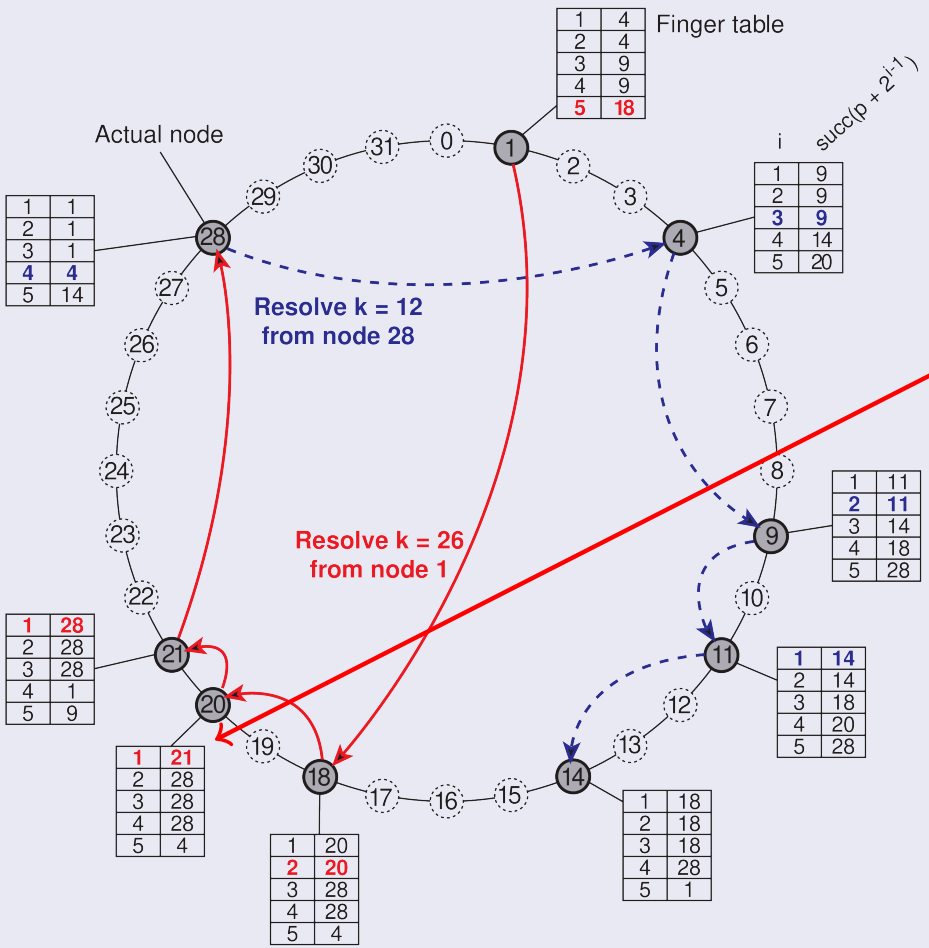
Chord – Mecanismo Geral



O nó 18, por sua vez, seleciona o nó 20, pois $FT_{18}[2] \leq 26 < FT_{18}[3]$

ENP – Tabelas de *Hash* Distribuídas

Chord – Mecanismo Geral

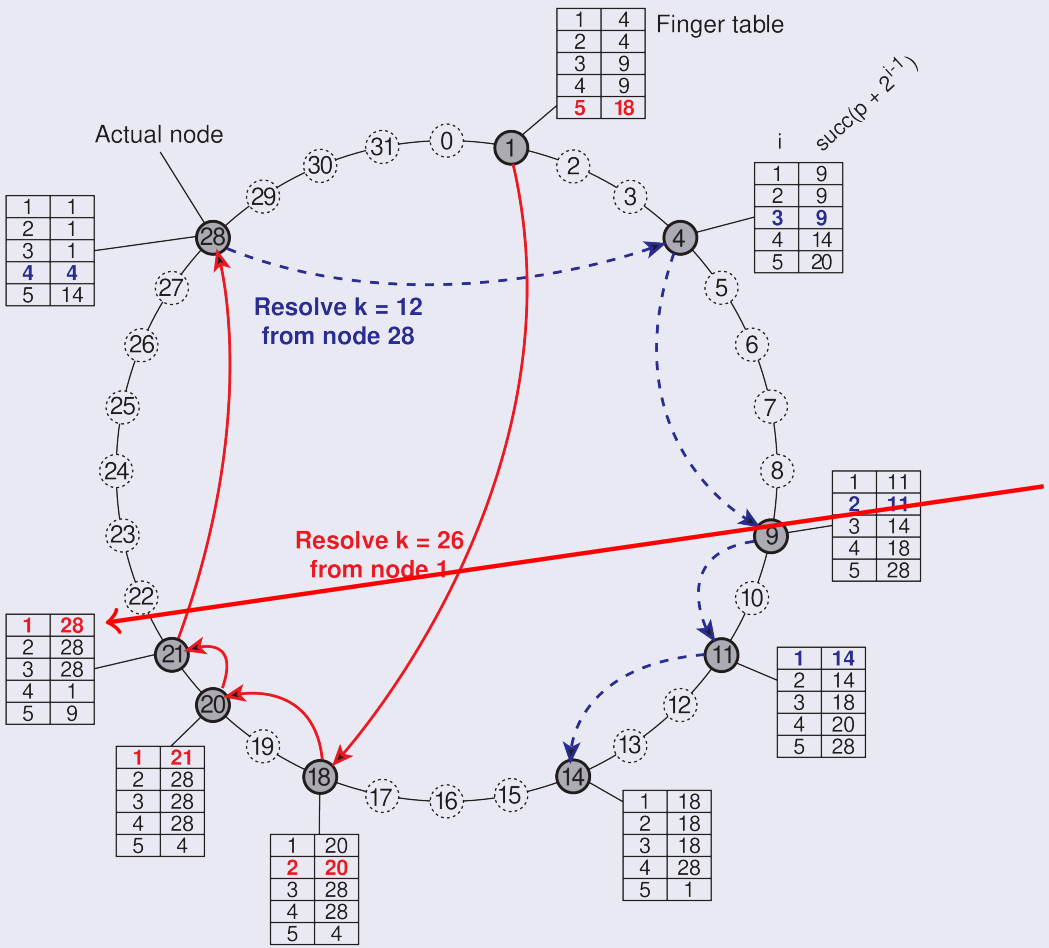


O nó 18, por sua vez, seleciona o nó 20, pois $FT_{18}[2] \leq 26 < FT_{18}[3]$

Como $FT_{20}[1] \leq 26 < FT_{20}[2]$, o nó 20 repassa ao nó 21

ENP – Tabelas de *Hash* Distribuídas

Chord – Mecanismo Geral

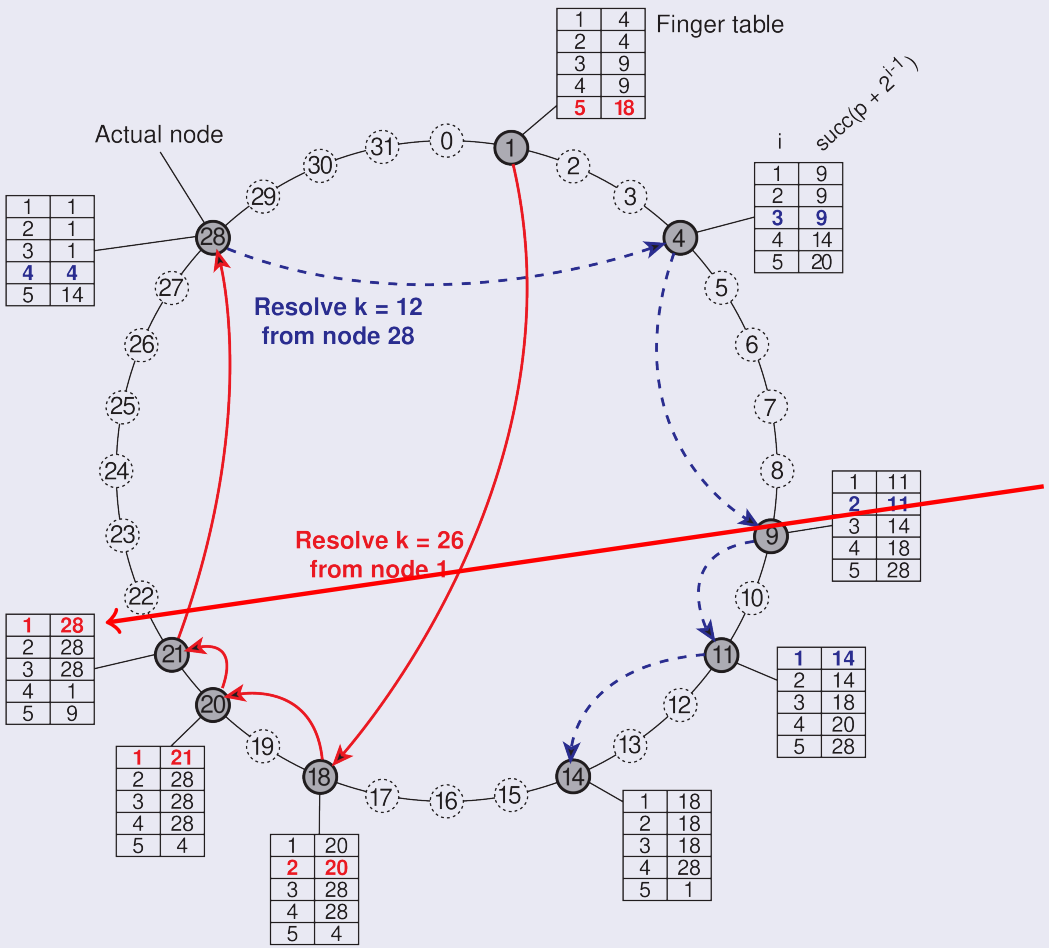


O nó 18, por sua vez, seleciona o nó 20, pois $FT_{18}[2] \leq 26 < FT_{18}[3]$

Como $FT_{20}[1] \leq 26 < FT_{20}[2]$, o nó 20 repassa ao nó 21

Com $21 < 26 < FT_{21}[1]$, o nó 21 repassa ao nó 28, responsável por $k = 26$

Chord – Mecanismo Geral



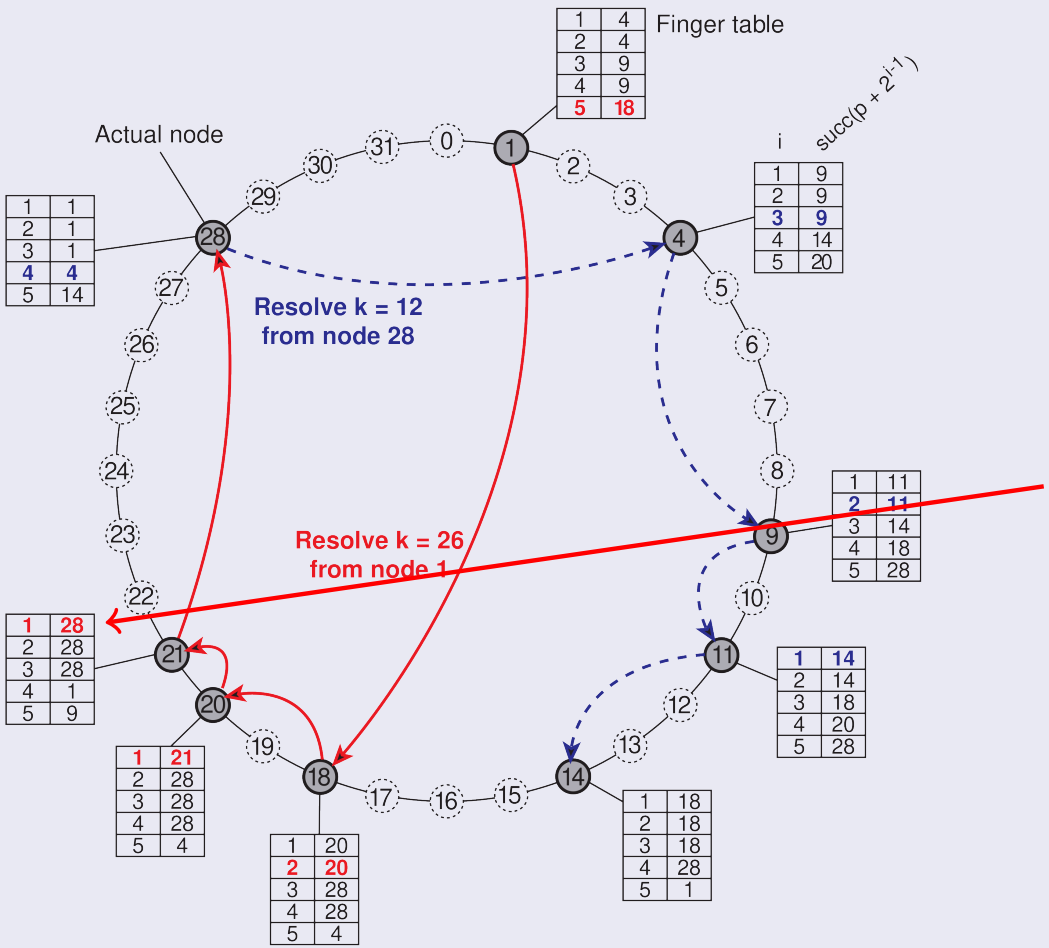
O nó 18, por sua vez, seleciona o nó 20, pois $FT_{18}[2] \leq 26 < FT_{18}[3]$

Como $FT_{20}[1] \leq 26 < FT_{20}[2]$, o nó 20 repassa ao nó 21

Com $21 < 26 < FT_{21}[1]$, o nó 21 repassa ao nó 28, responsável por $k = 26$

Nesse ponto, o endereço do nó 28 é retornado ao nó 1, e a chave foi resolvida

Chord – Mecanismo Geral



O nó 18, por sua vez, seleciona o nó 20, pois $FT_{18}[2] \leq 26 < FT_{18}[3]$

Como $FT_{20}[1] \leq 26 < FT_{20}[2]$, o nó 20 repassa ao nó 21

Com $21 < 26 < FT_{21}[1]$, o nó 21 repassa ao nó 28, responsável por $k = 26$

Nesse ponto, o endereço do nó 28 é retornado ao nó 1, e a chave foi resolvida

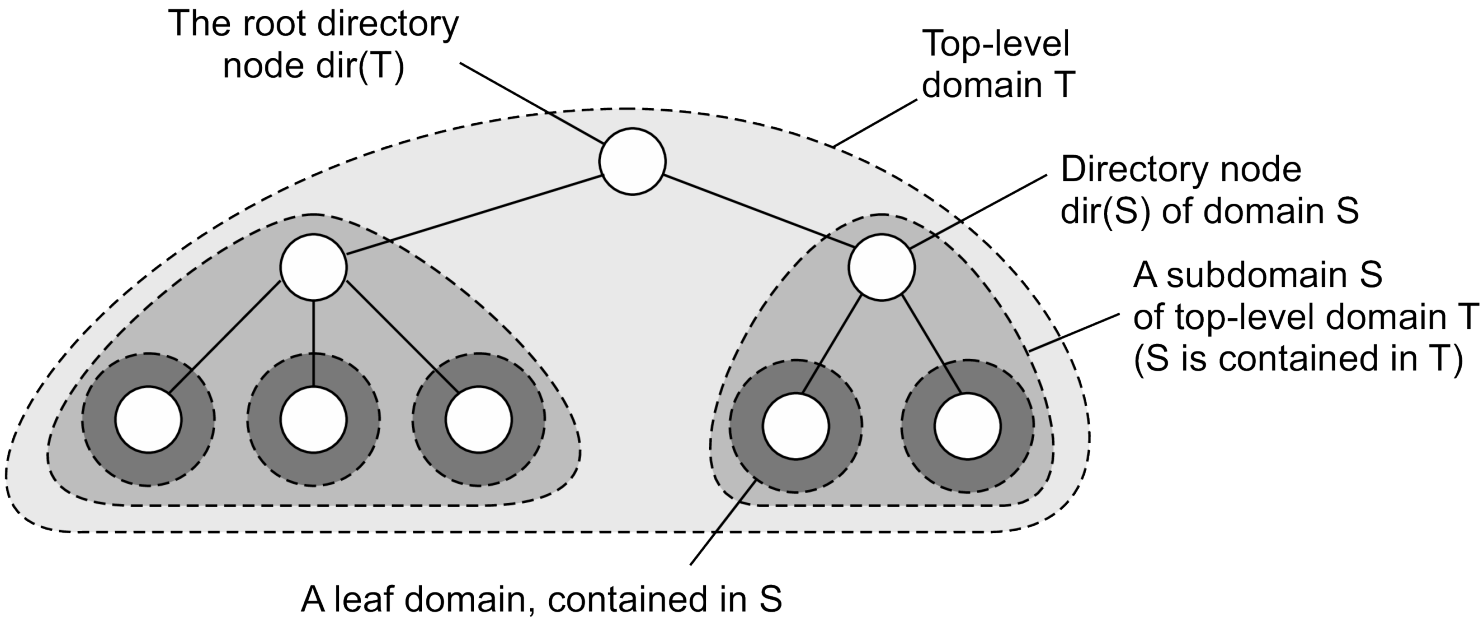
Note que $FT_q[1]$ se refere ao próximo nó no anel, sucessor de q

Sistemas de Nomeação

- Nomes, Identificadores e Endereços
- **Espaço de Nomes Plano**
  - Soluções Simples
  - Abordagens *Home-based*
  - Tabelas de *Hash* Distribuídas
  - **Abordagens Hierárquicas**

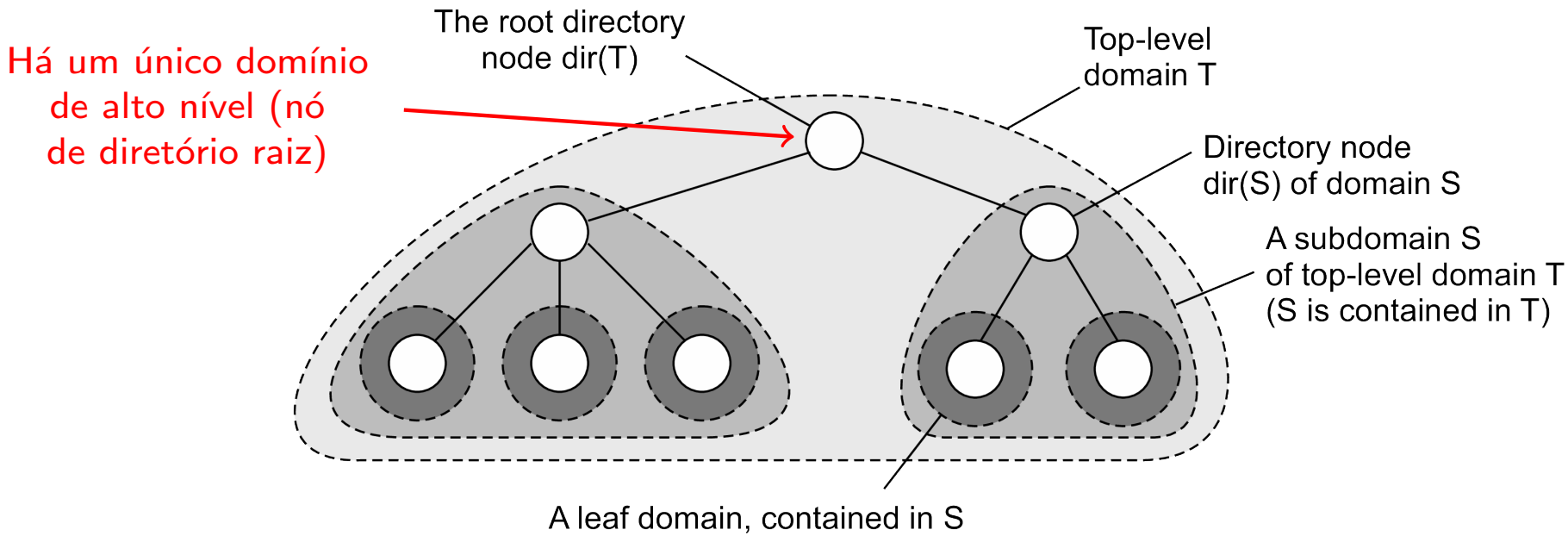
ENP – Abordagens Hierárquicas

Construa uma árvore de busca em que a rede é dividida em **domínios** hierárquicos. Cada domínio é representado por um nó de diretório separado



ENP – Abordagens Hierárquicas

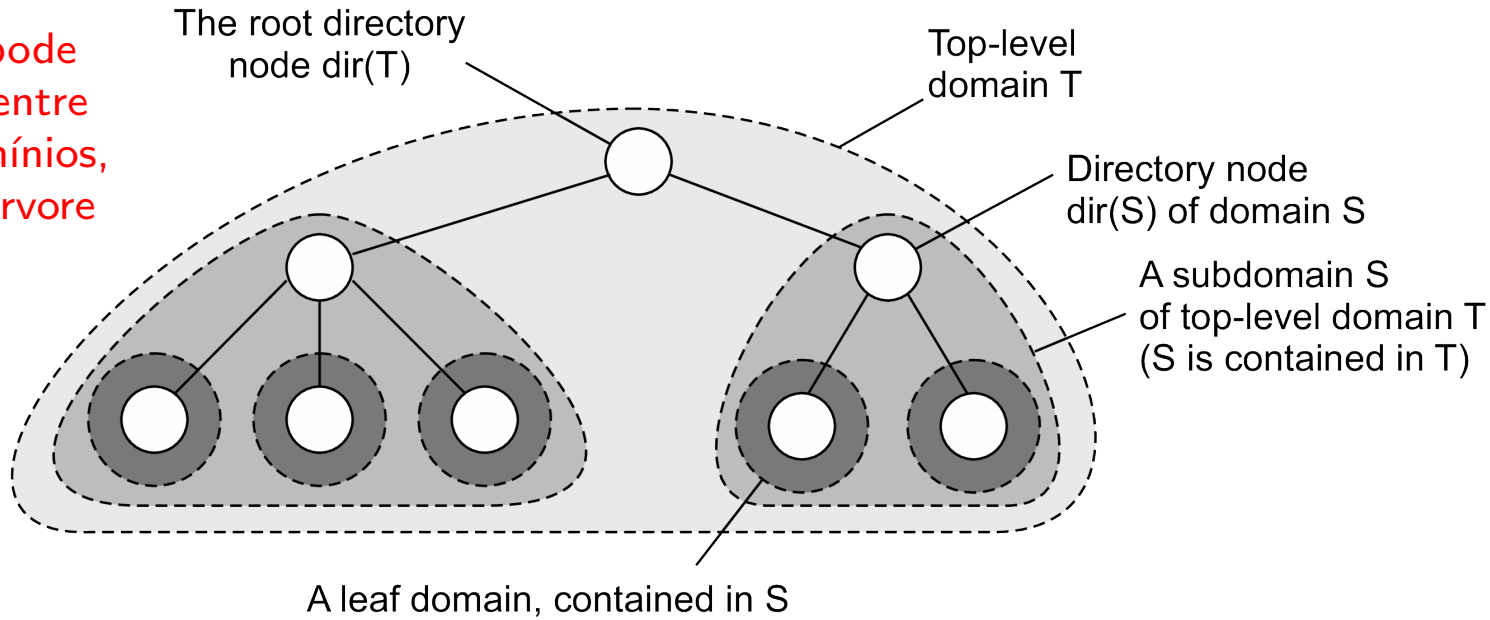
Construa uma árvore de busca em que a rede é dividida em **domínios** hierárquicos. Cada domínio é representado por um nó de diretório separado



ENP – Abordagens Hierárquicas

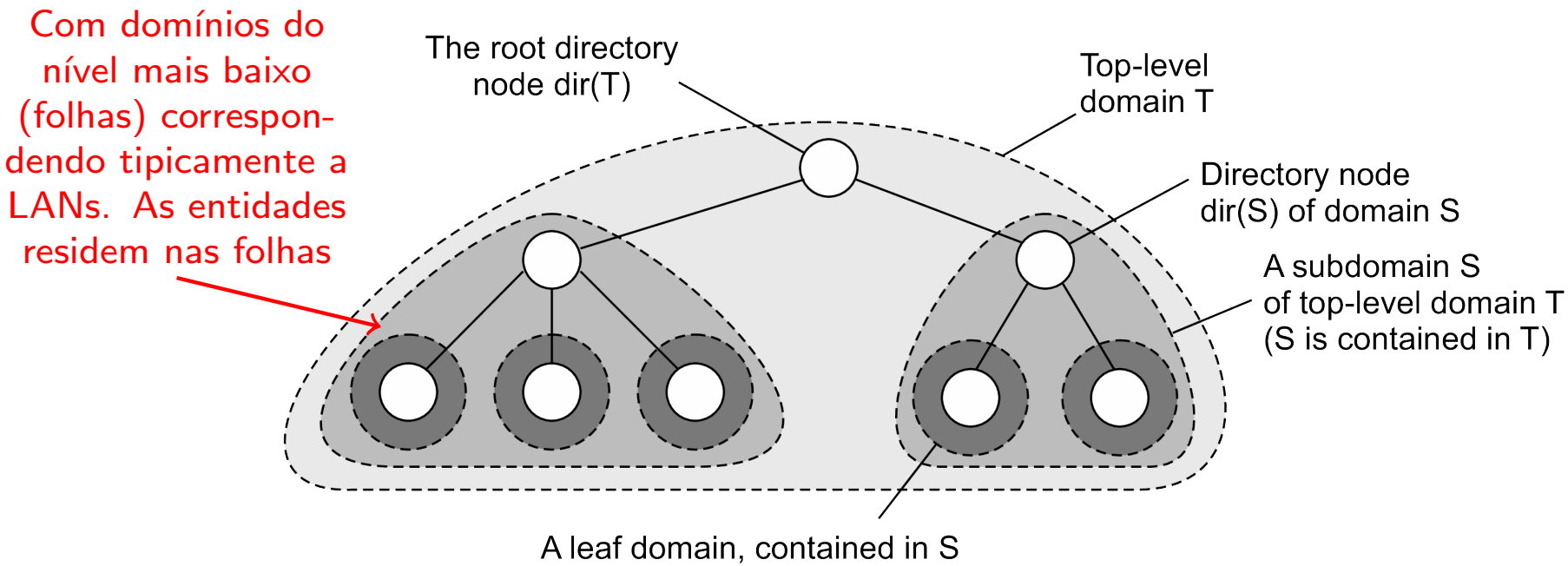
Construa uma árvore de busca em que a rede é dividida em **domínios** hierárquicos. Cada domínio é representado por um nó de diretório separado

Cada domínio pode ser subdividido entre múltiplos subdomínios, formando uma árvore



ENP – Abordagens Hierárquicas

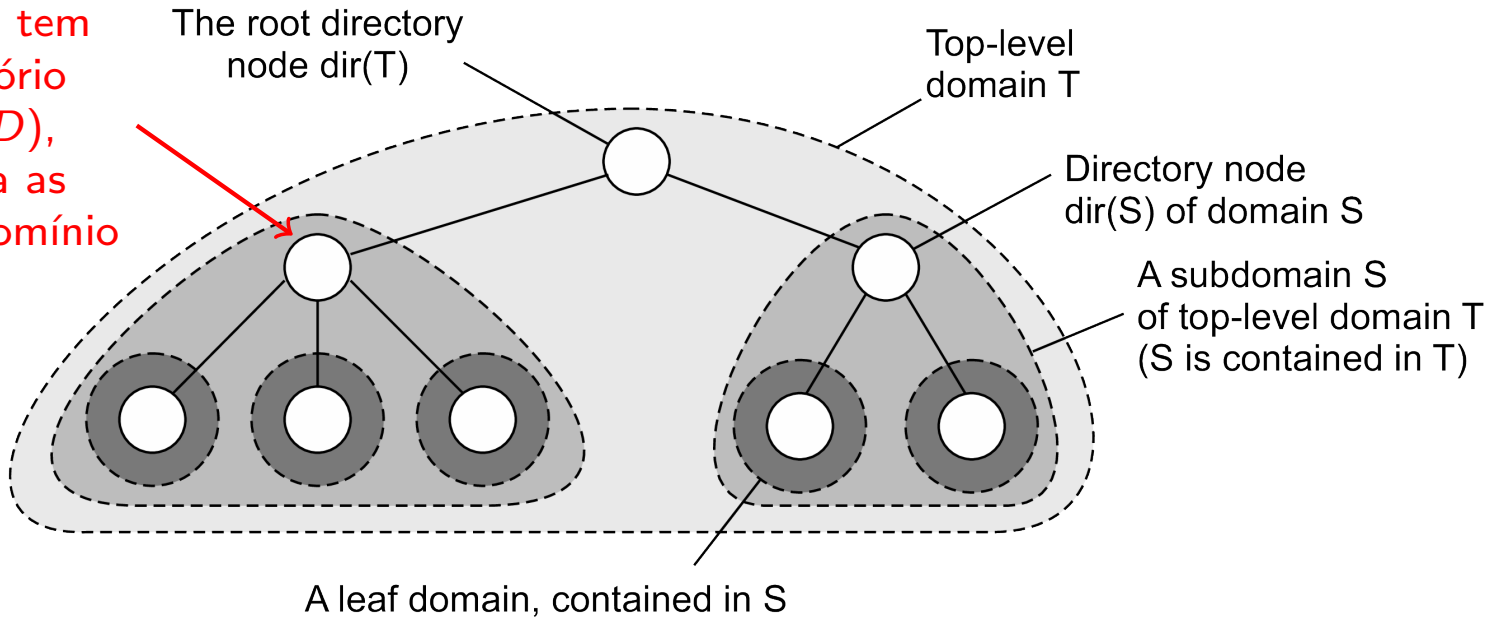
Construa uma árvore de busca em que a rede é dividida em **domínios** hierárquicos. Cada domínio é representado por um nó de diretório separado



ENP – Abordagens Hierárquicas

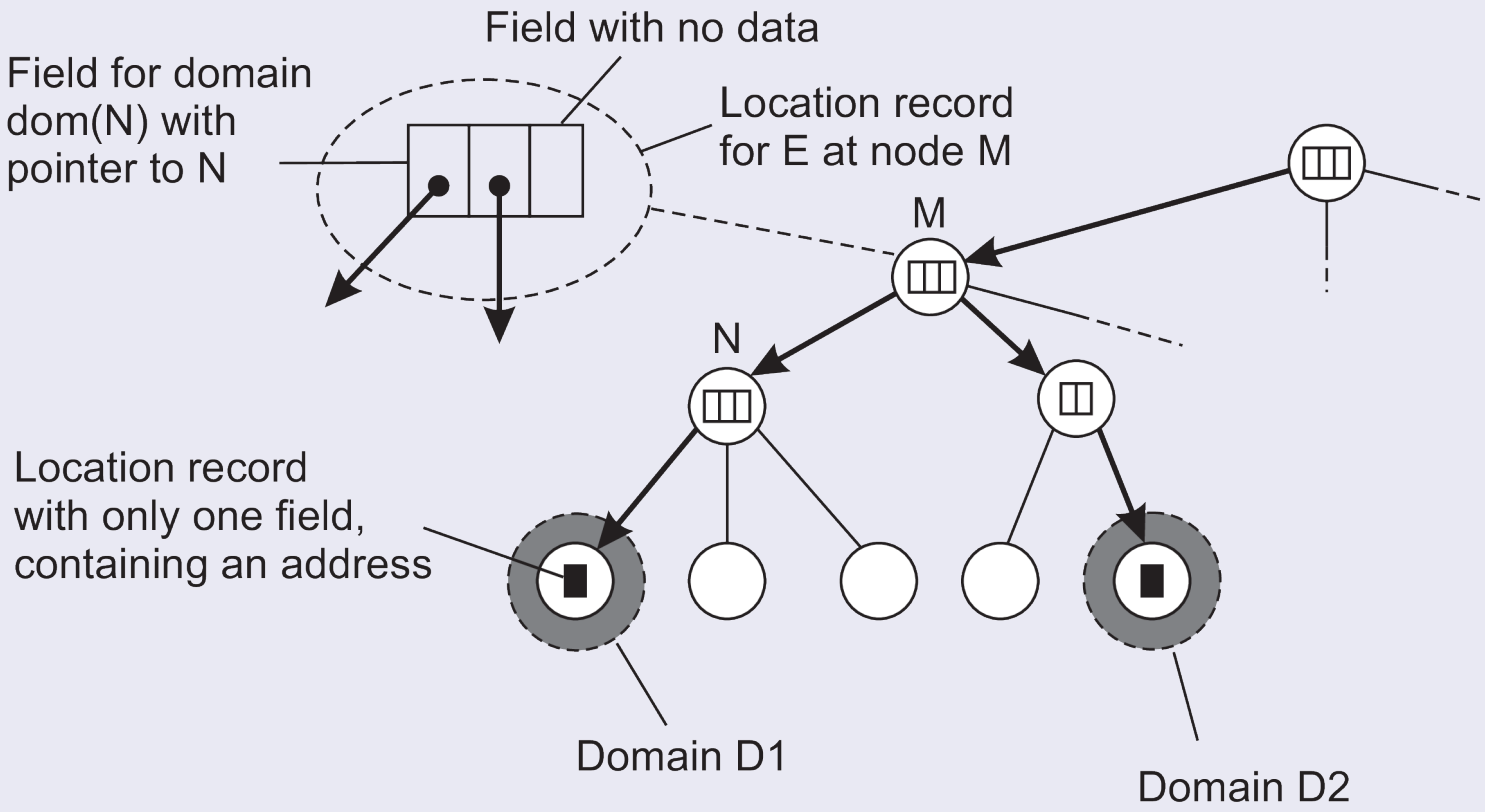
Construa uma árvore de busca em que a rede é dividida em **domínios** hierárquicos. Cada domínio é representado por um nó de diretório separado

Cada domínio D tem um nó de diretório associado $dir(D)$, que acompanha as entidades neste domínio



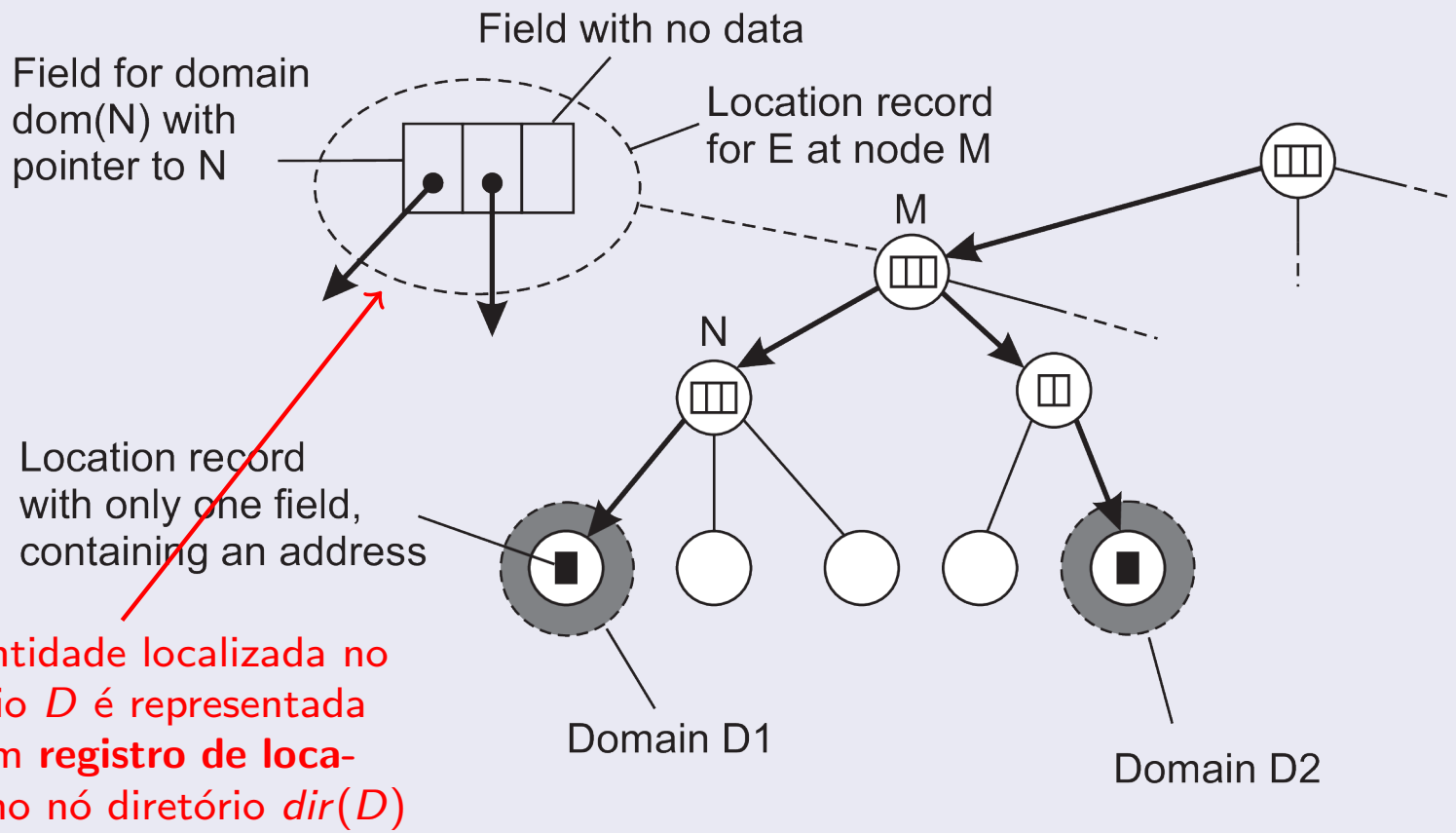
ENP – Abordagens Hierárquicas

Registro de Localização



ENP – Abordagens Hierárquicas

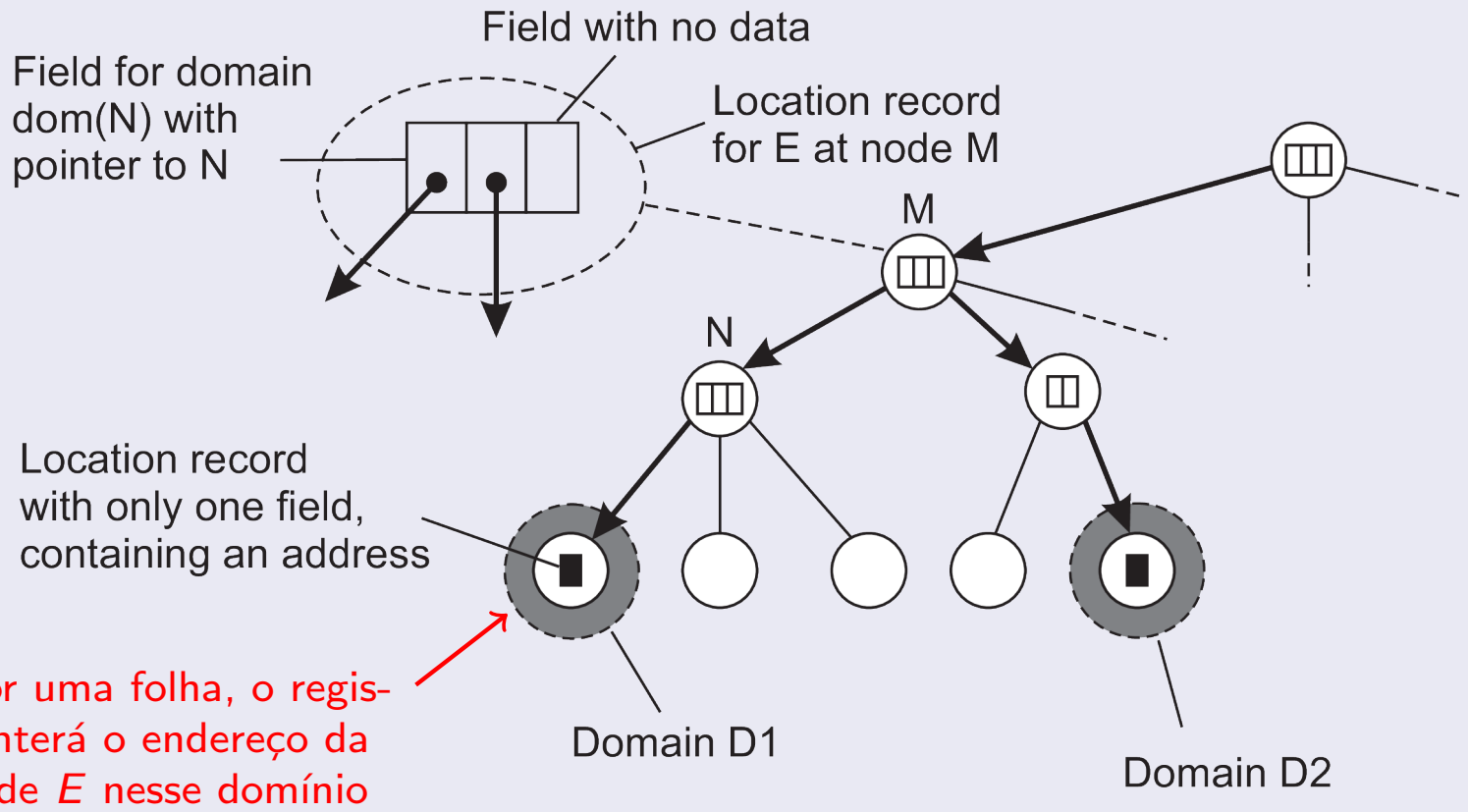
Registro de Localização



Cada entidade localizada no domínio D é representada por um **registro de localização** no nó diretório $dir(D)$

ENP – Abordagens Hierárquicas

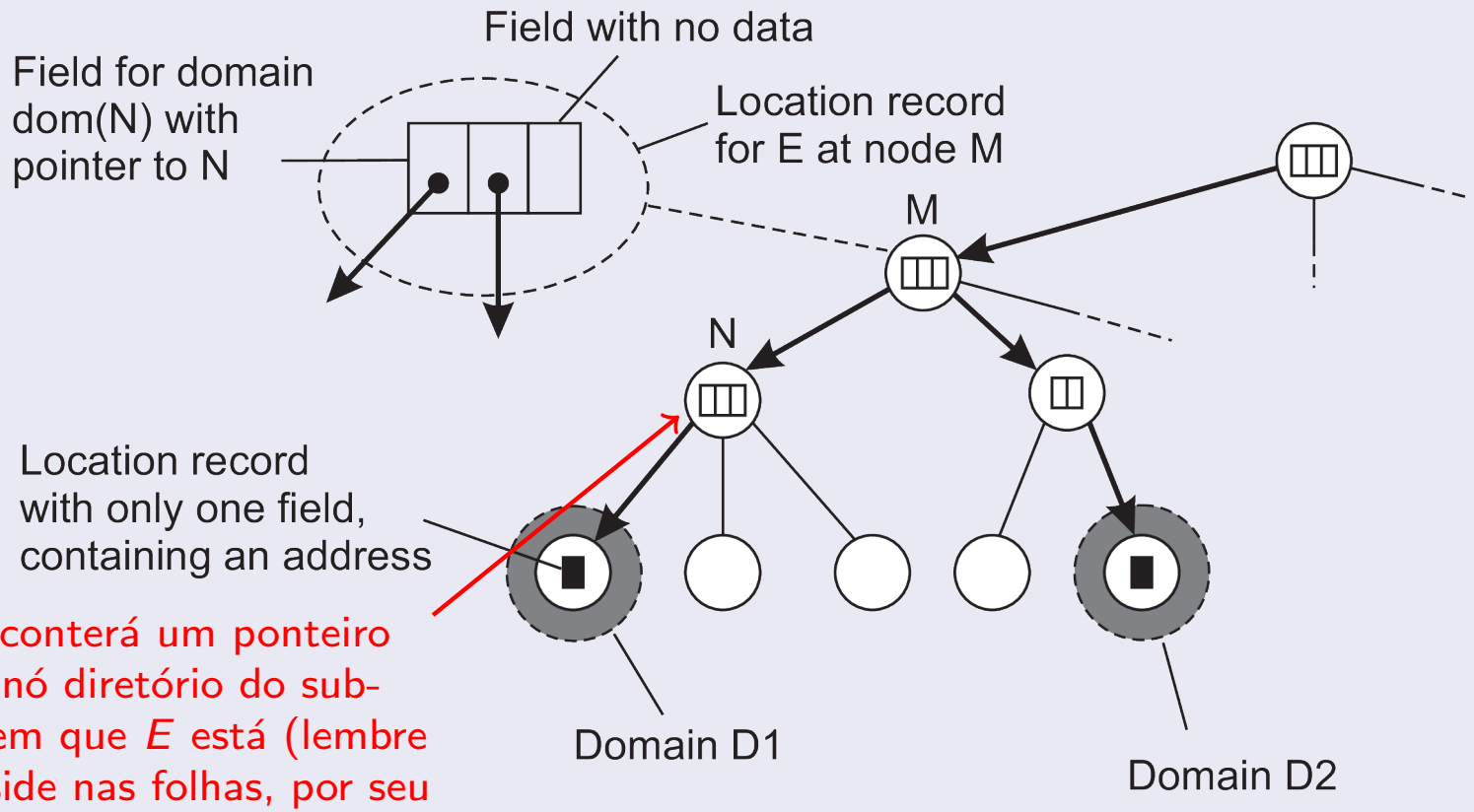
Registro de Localização



Se *D* for uma folha, o registro conterá o endereço da entidade *E* nesse domínio

ENP – Abordagens Hierárquicas

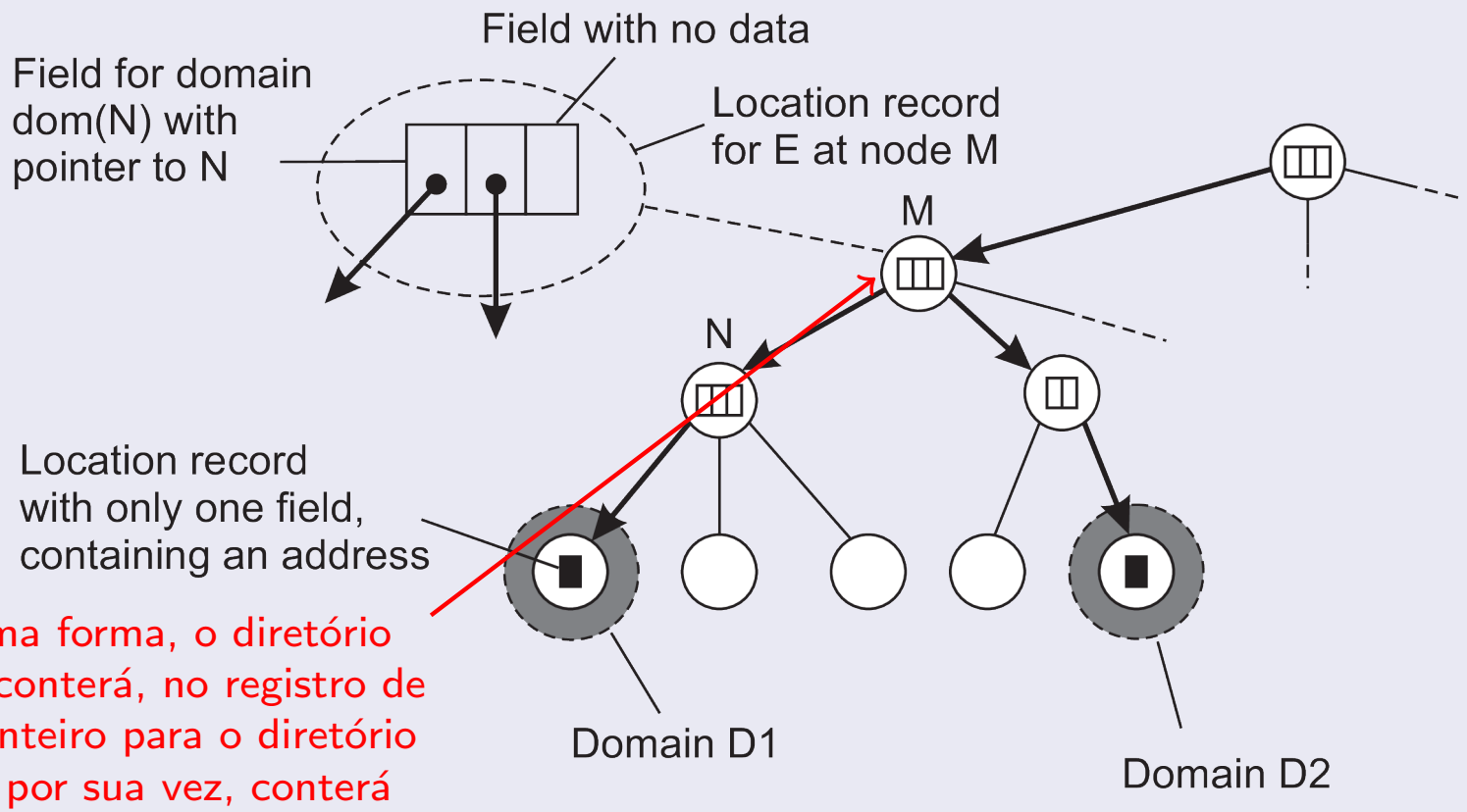
Registro de Localização



Senão, conterá um ponteiro para o nó diretório do sub-domínio em que *E* está (lembre que *E* reside nas folhas, por seu nome não possuir estrutura)

ENP – Abordagens Hierárquicas

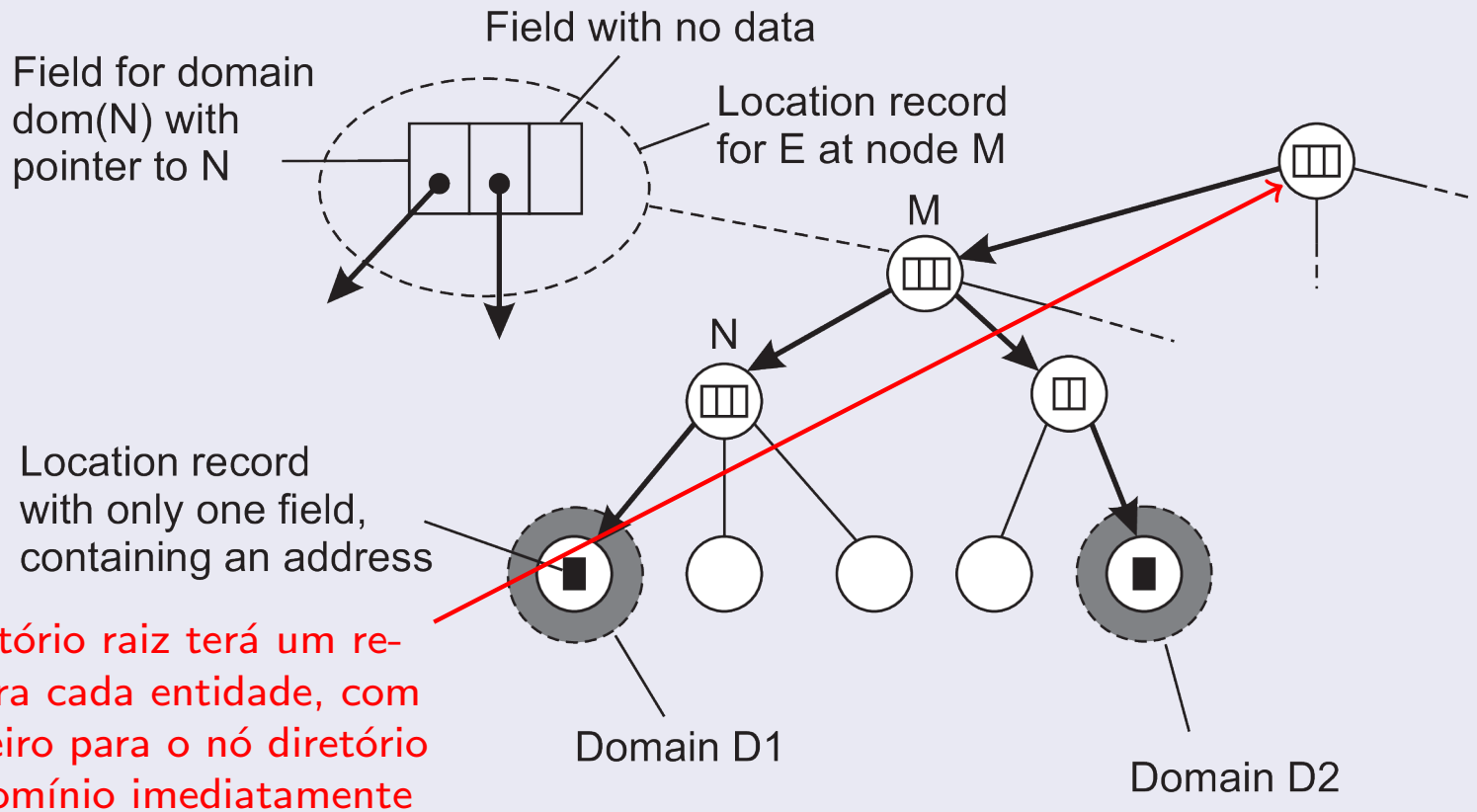
Registro de Localização



Da mesma forma, o diretório avô de E conterá, no registro de E, um ponteiro para o diretório pai que, por sua vez, conterá um para o diretório de E

ENP – Abordagens Hierárquicas

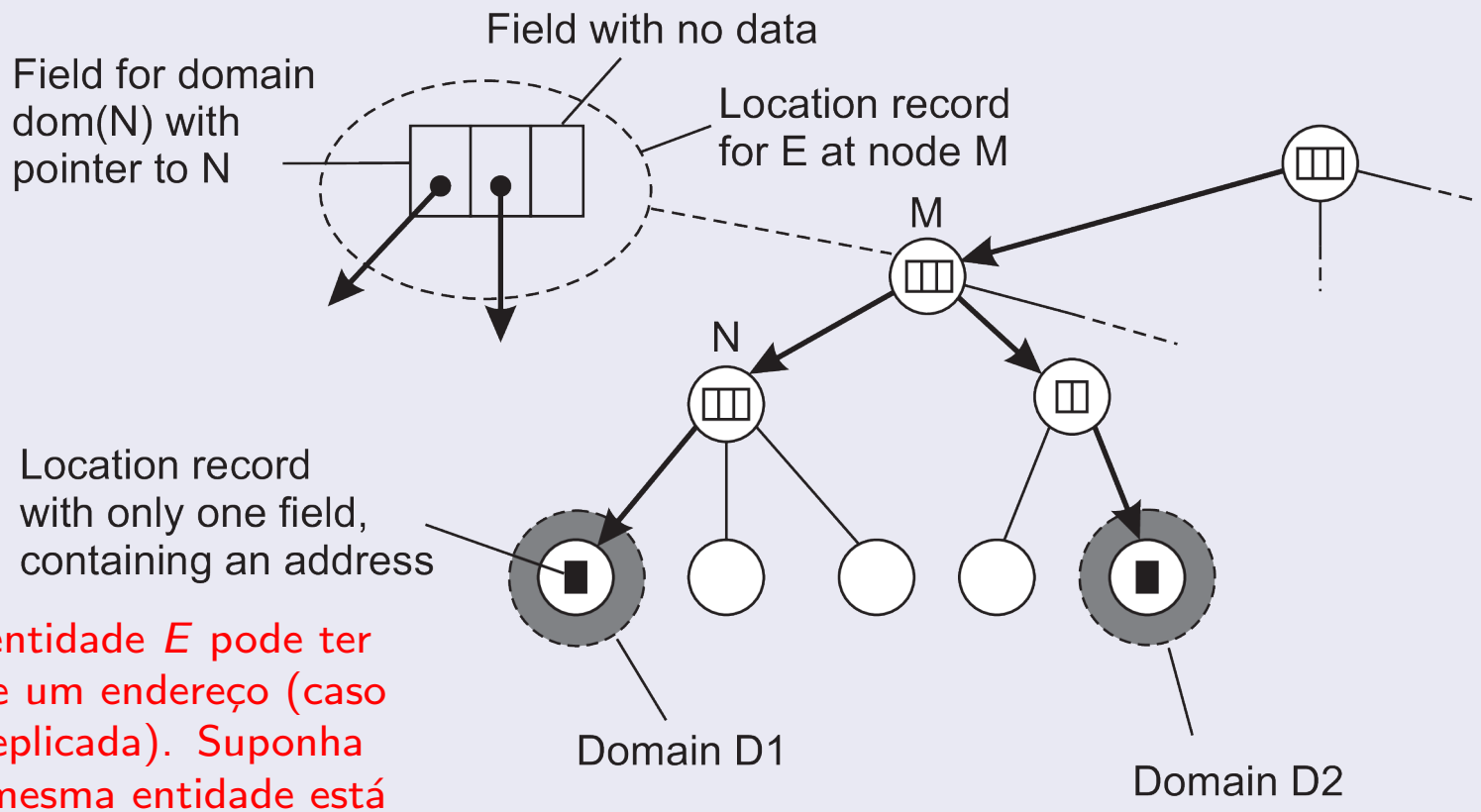
Registro de Localização



E o diretório raiz terá um registro para cada entidade, com um ponteiro para o nó diretório do subdomínio imediatamente abaixo contendo a entidade

ENP – Abordagens Hierárquicas

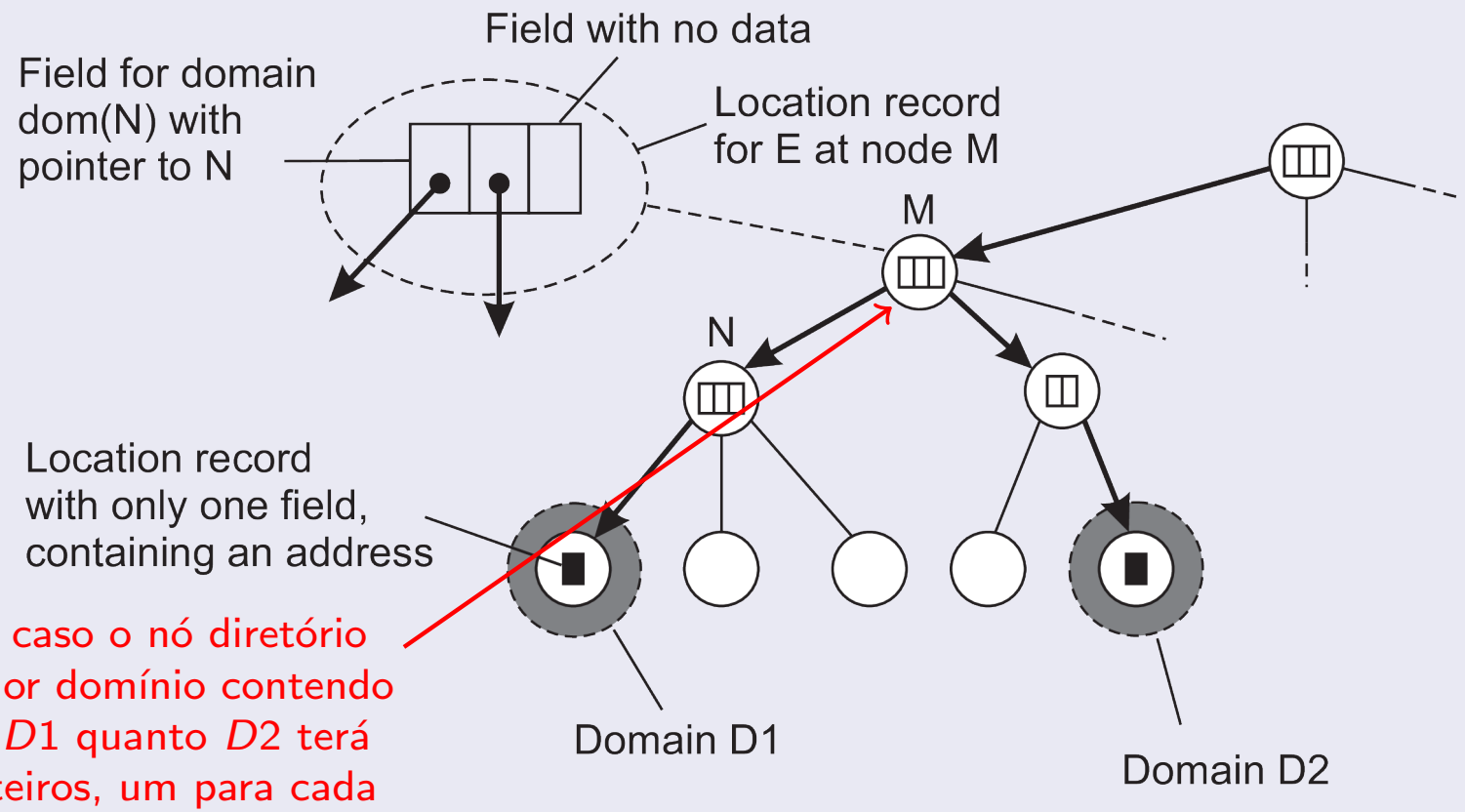
Registro de Localização



Uma entidade E pode ter mais de um endereço (caso seja replicada). Suponha que a mesma entidade está tanto em $D1$ quanto em $D2$

ENP – Abordagens Hierárquicas

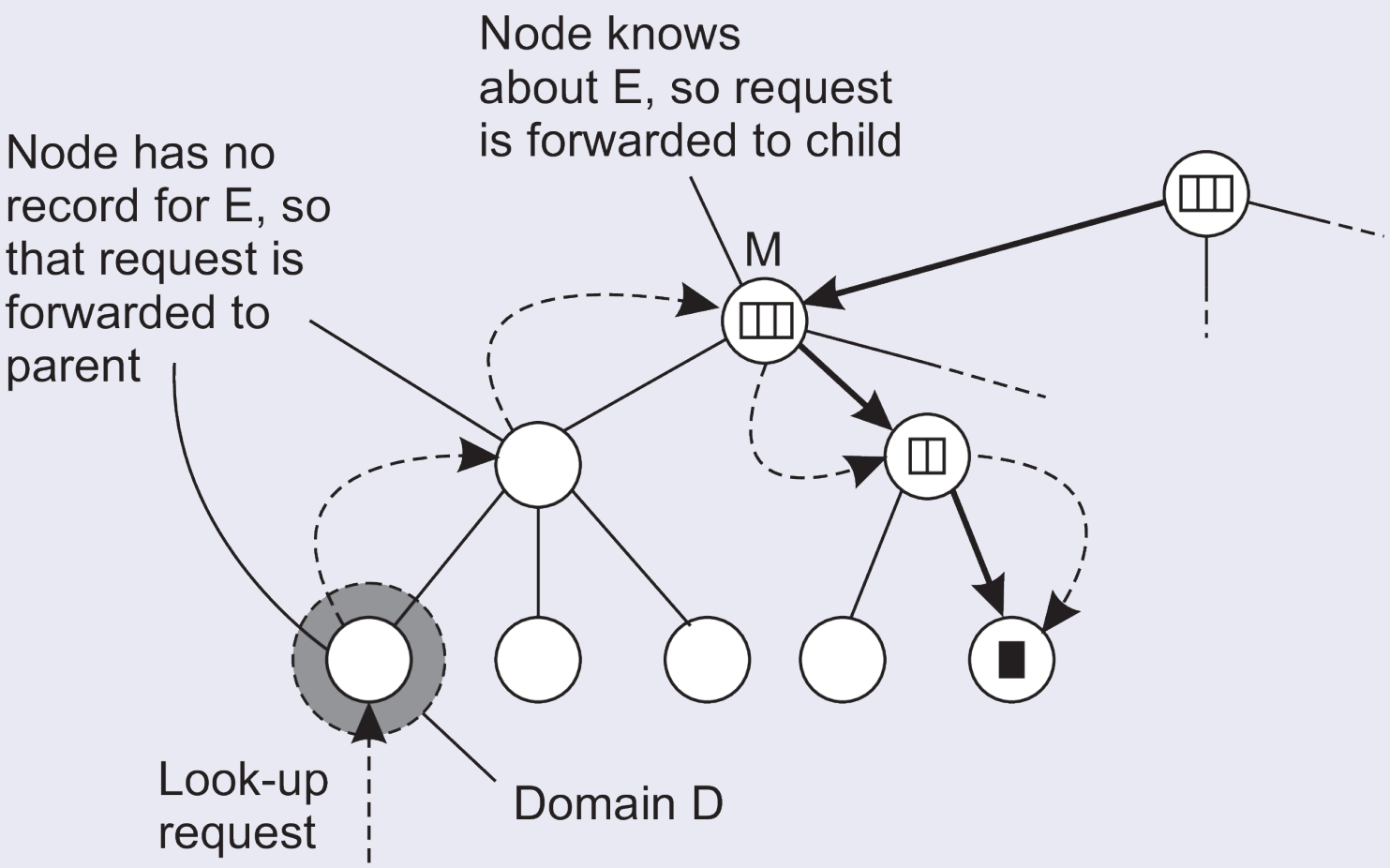
Registro de Localização



Nesse caso o nó diretório do menor domínio contendo tanto $D1$ quanto $D2$ terá 2 ponteiros, um para cada subdomínio contendo E

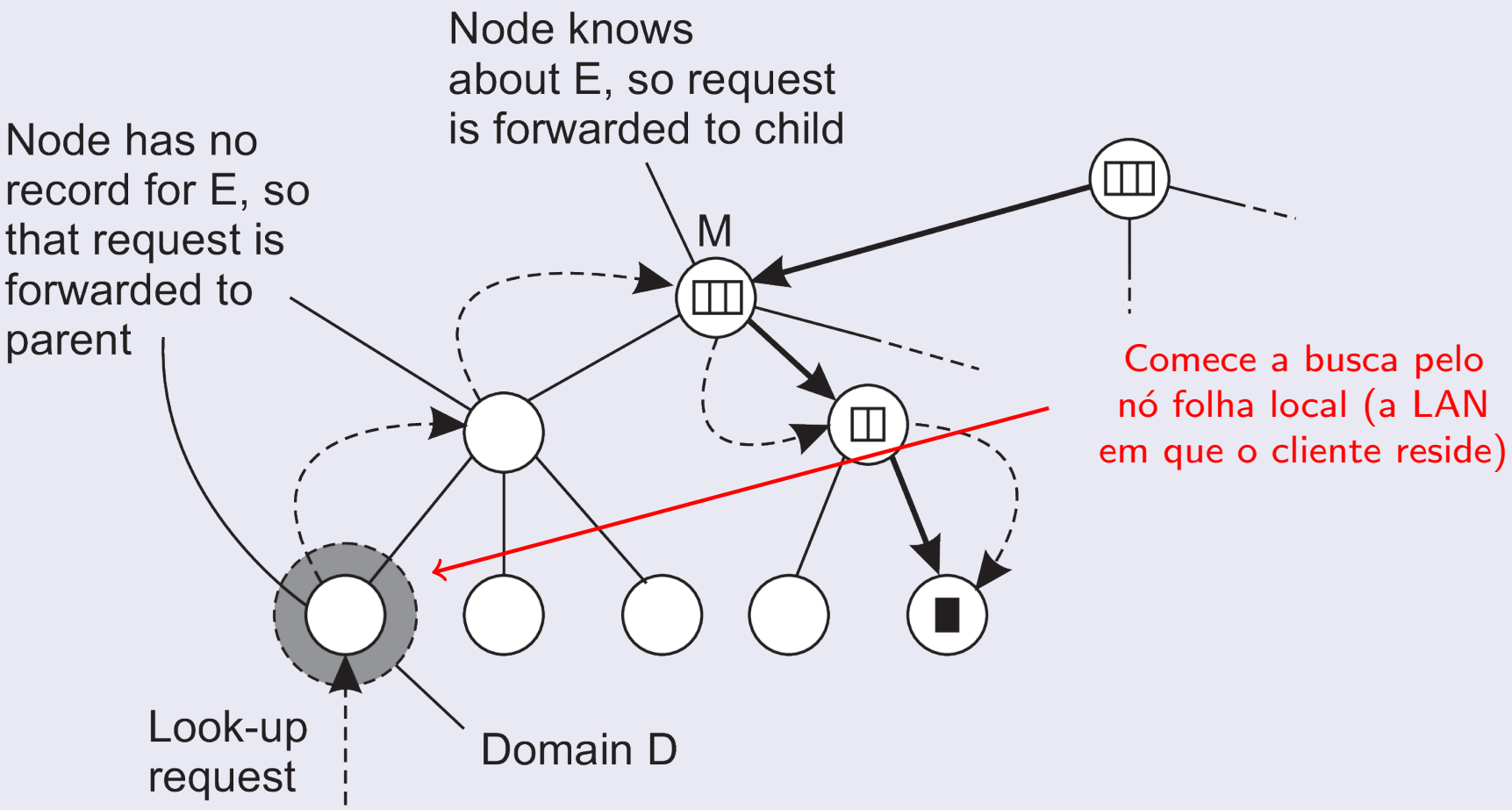
ENP – Abordagens Hierárquicas

Busca



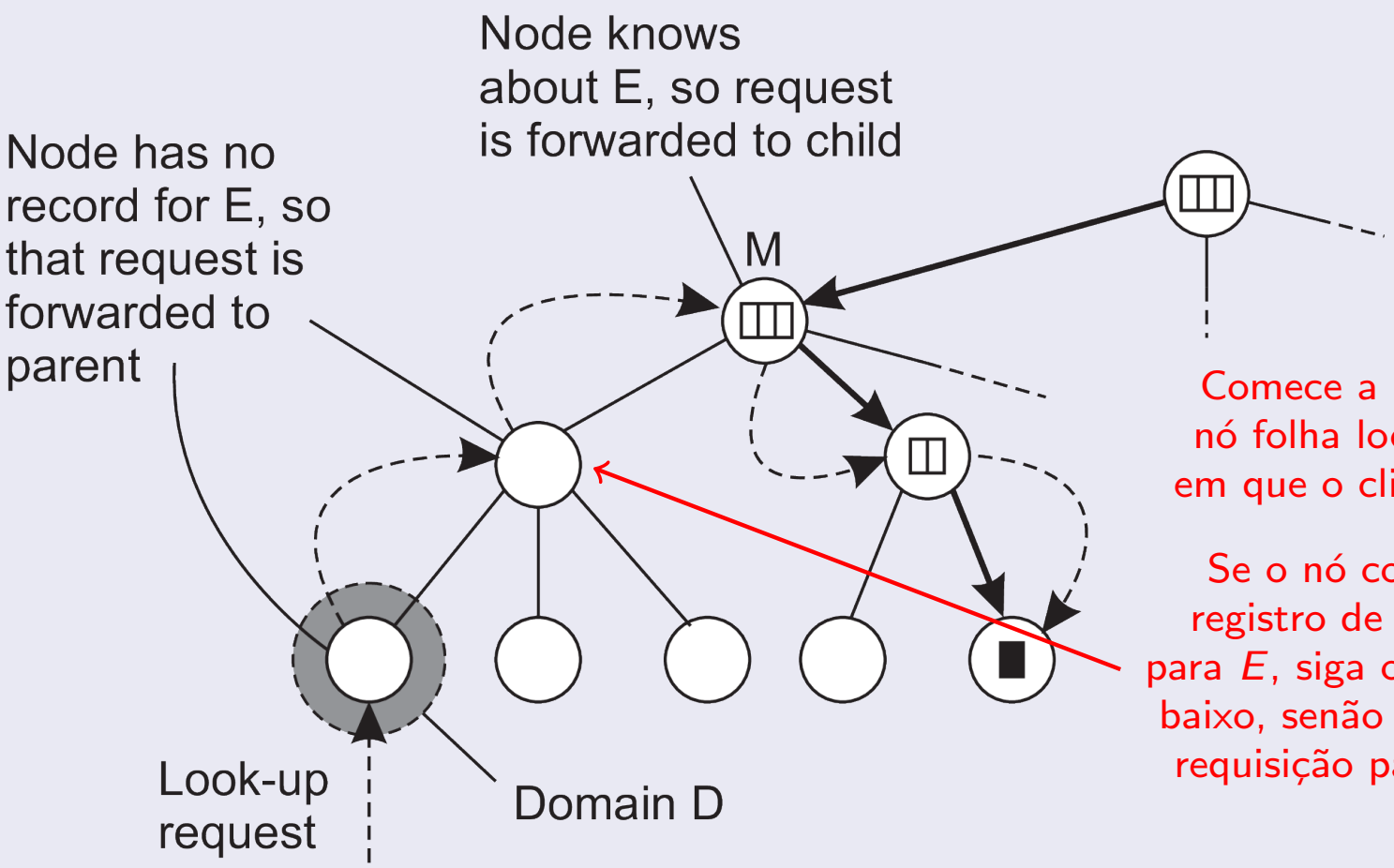
ENP – Abordagens Hierárquicas

Busca



ENP – Abordagens Hierárquicas

Busca

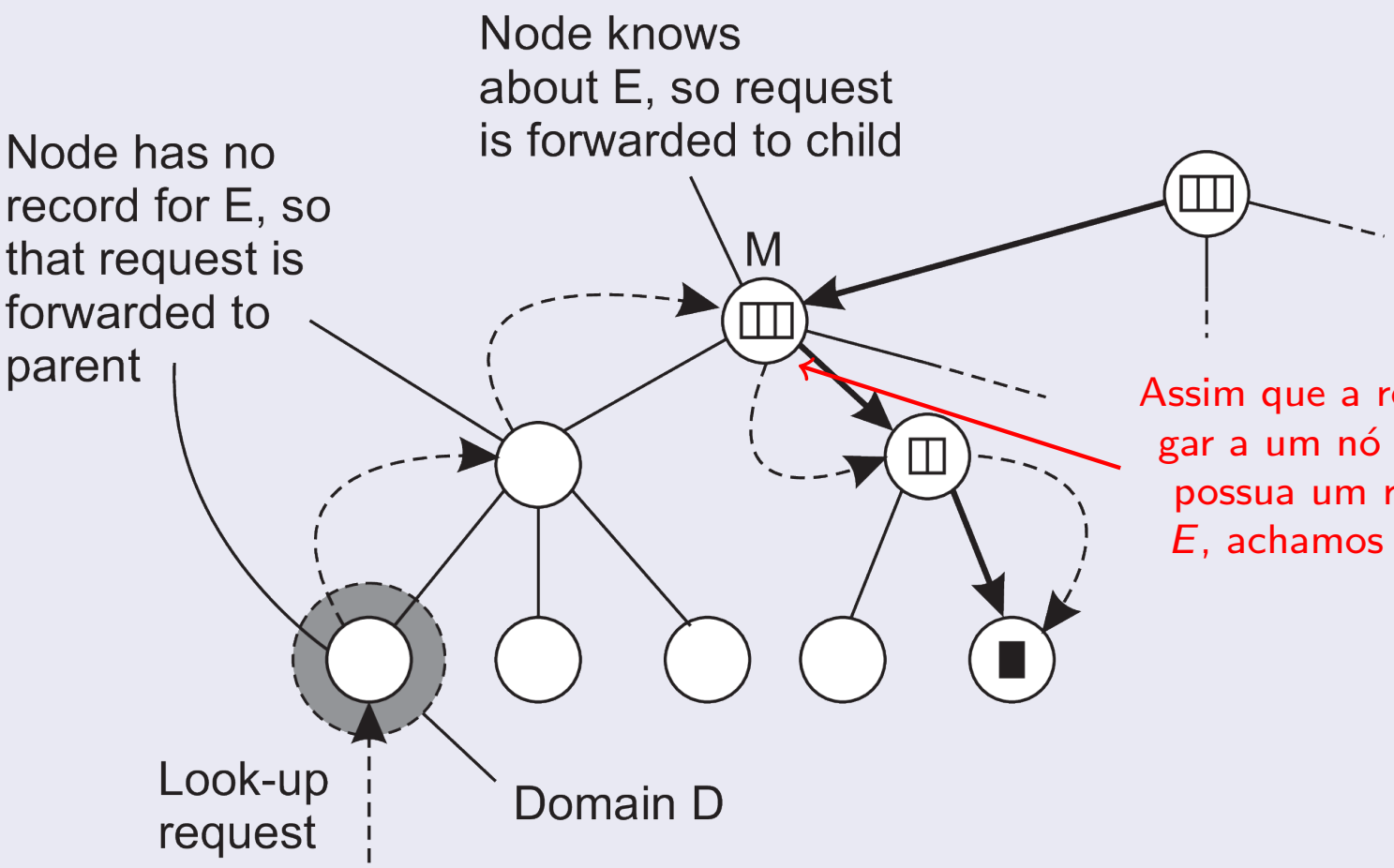


Comece a busca pelo nó folha local (a LAN em que o cliente reside)

Se o nó contiver um registro de localização para E, siga o ponteiro pra baixo, senão encaminhe a requisição para o nó pai

ENP – Abordagens Hierárquicas

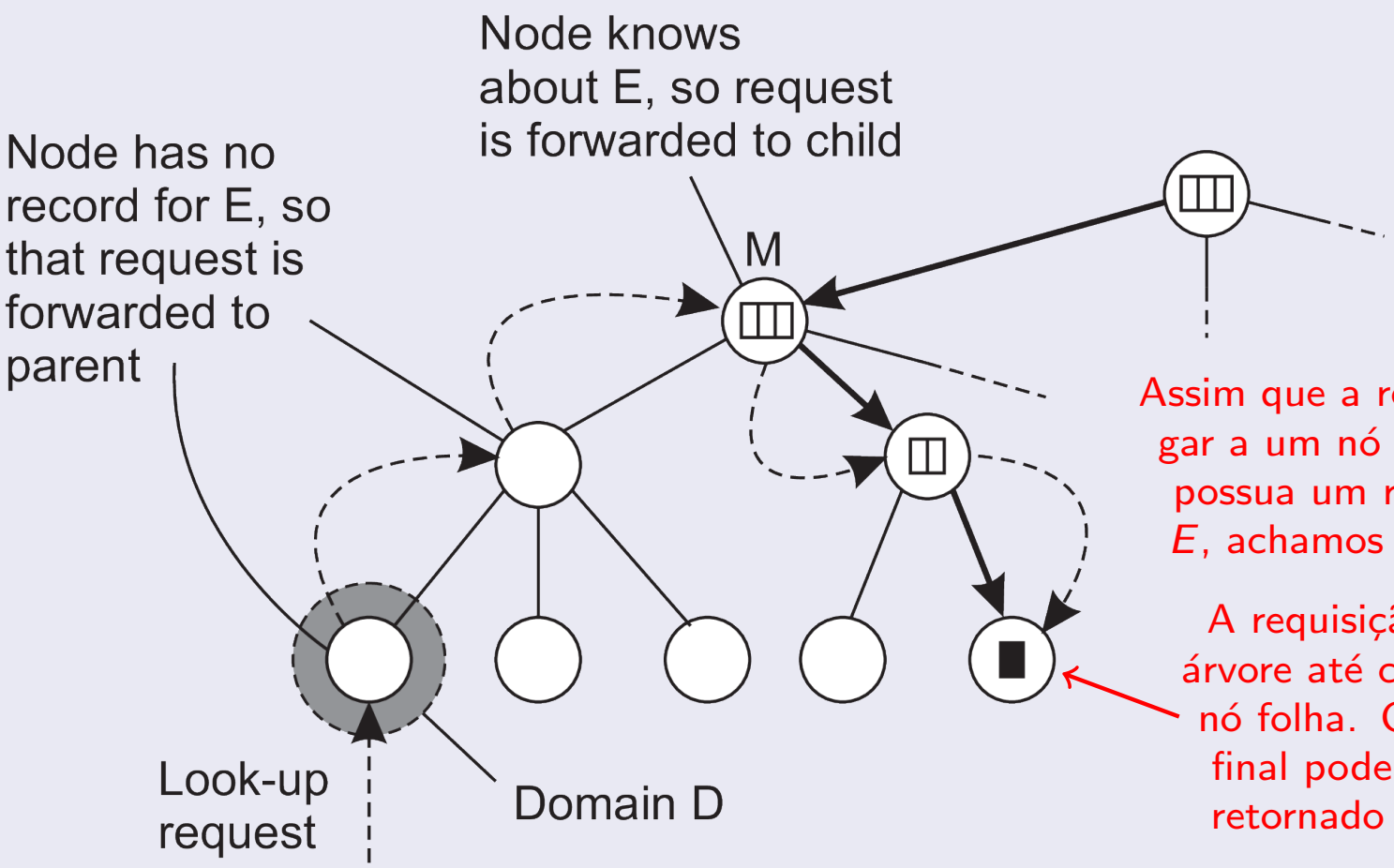
Busca



Assim que a requisição chegar a um nó diretório que possua um registro para E, achamos seu domínio

ENP – Abordagens Hierárquicas

Busca

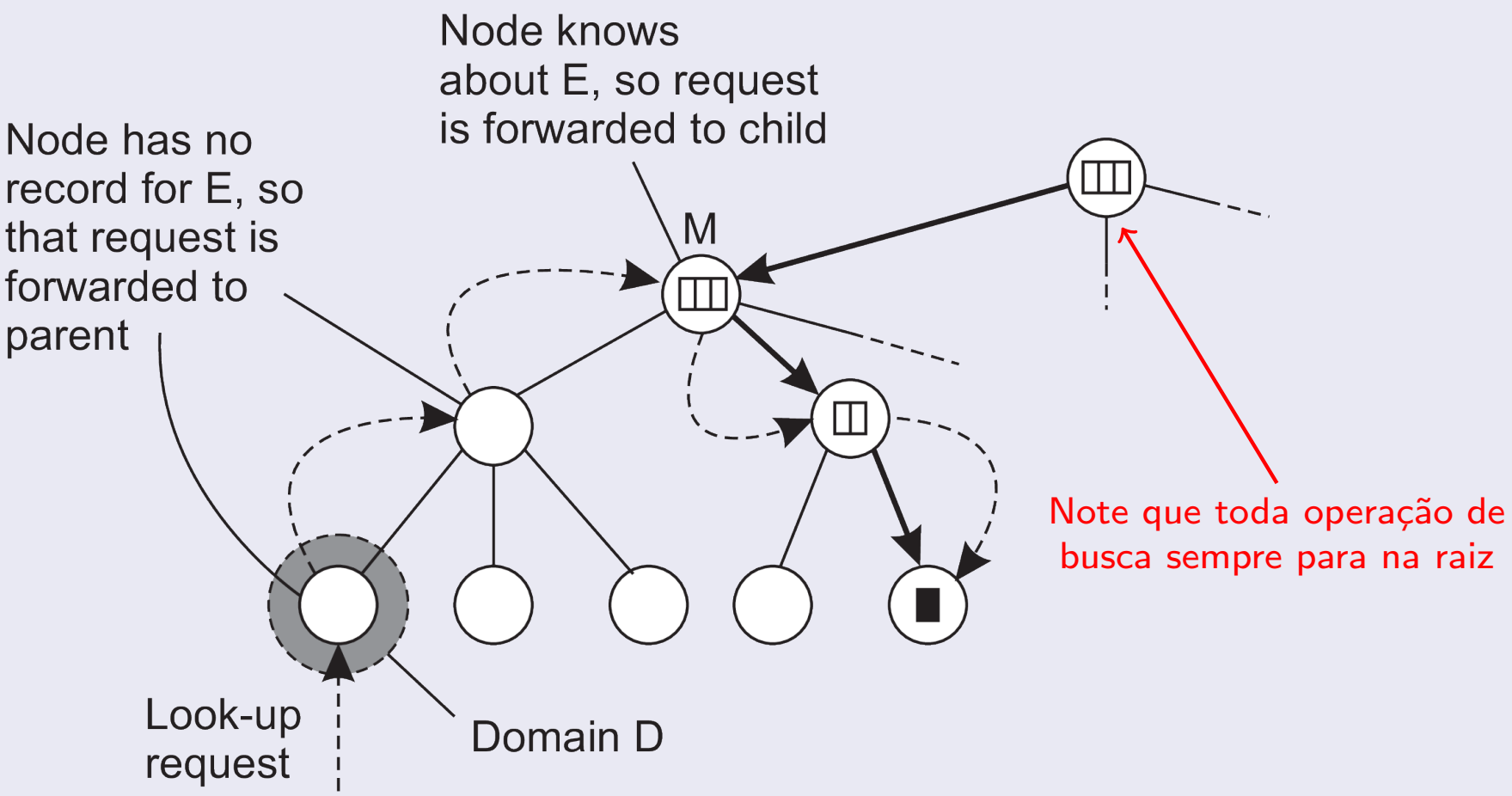


Assim que a requisição chegar a um nó diretório que possua um registro para E, achamos seu domínio

A requisição desce a árvore até chegar a um nó folha. O endereço final pode então ser retornado ao cliente

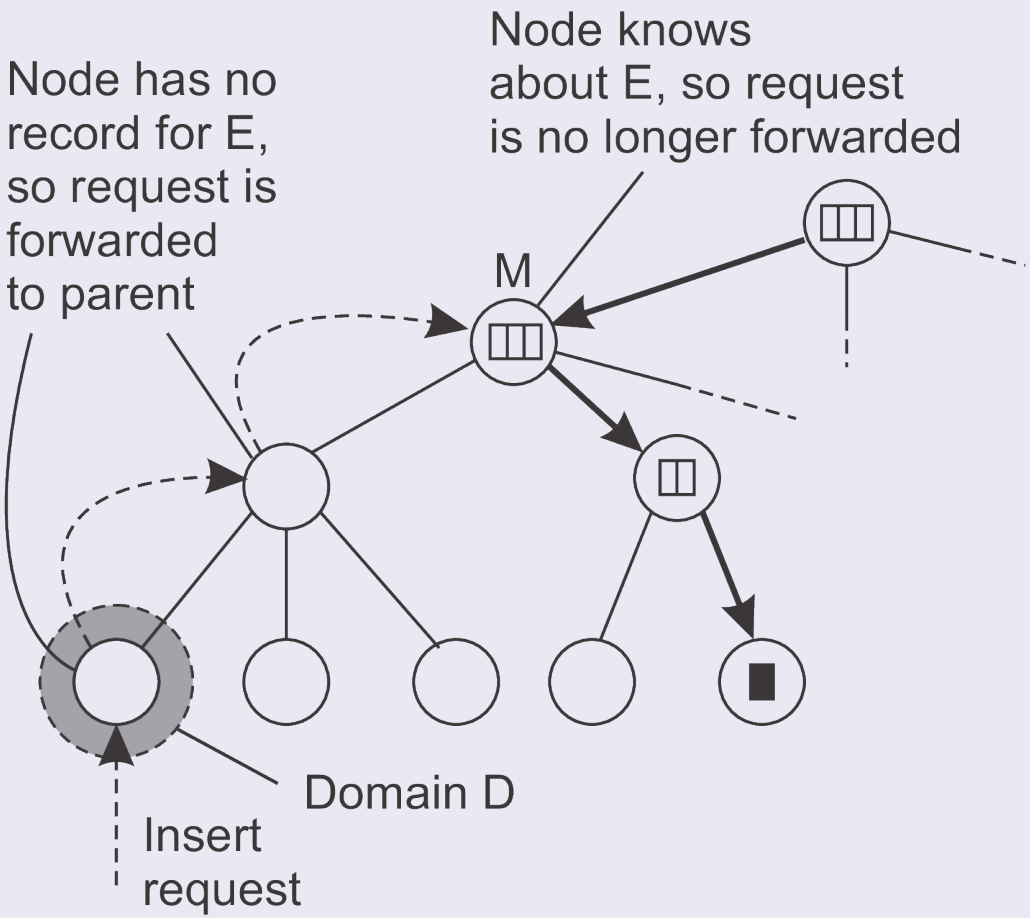
ENP – Abordagens Hierárquicas

Busca



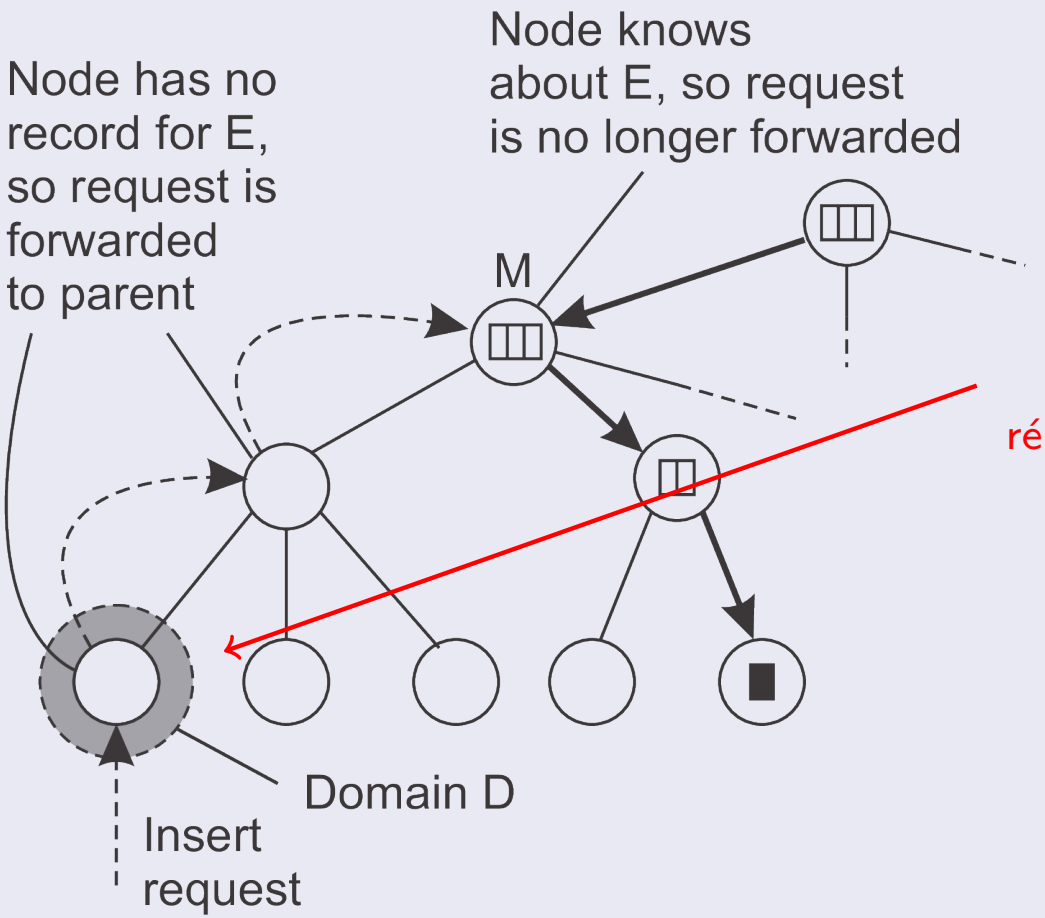
ENP – Abordagens Hierárquicas

Inserção



ENP – Abordagens Hierárquicas

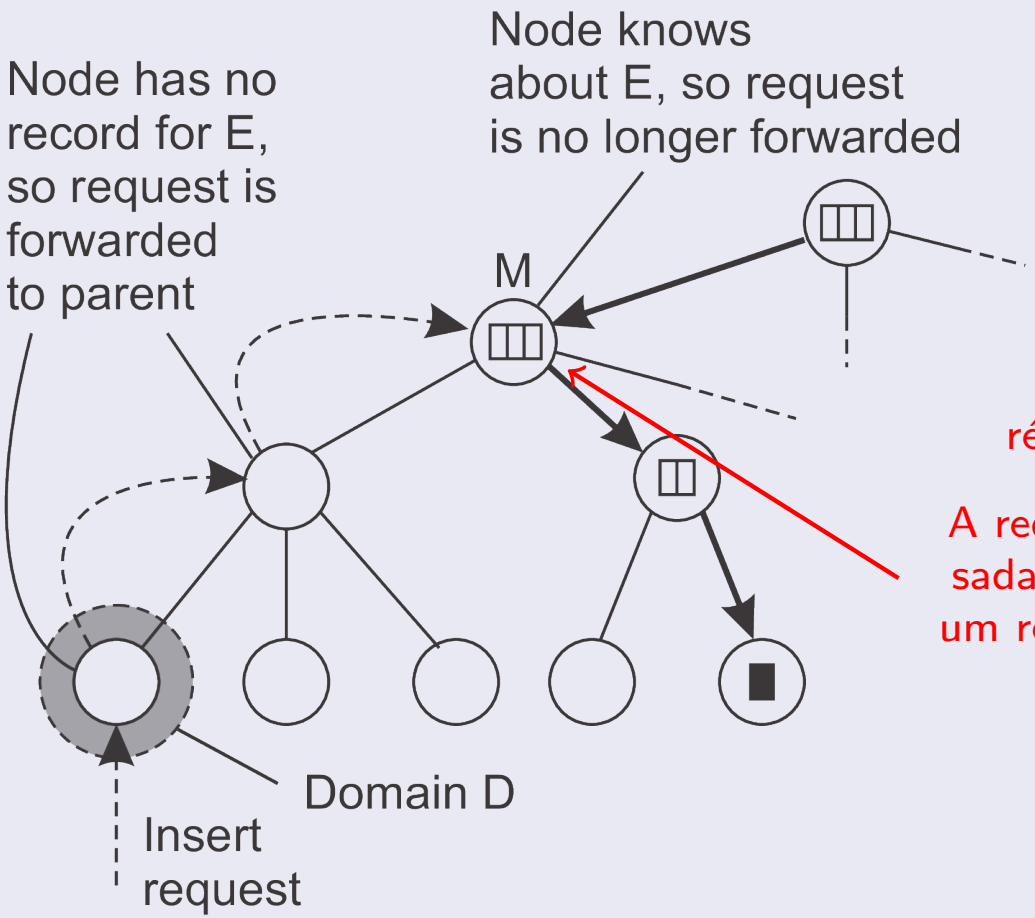
Inserção



Considere uma entidade E que criou uma réplica no domínio folha D

ENP – Abordagens Hierárquicas

Inserção

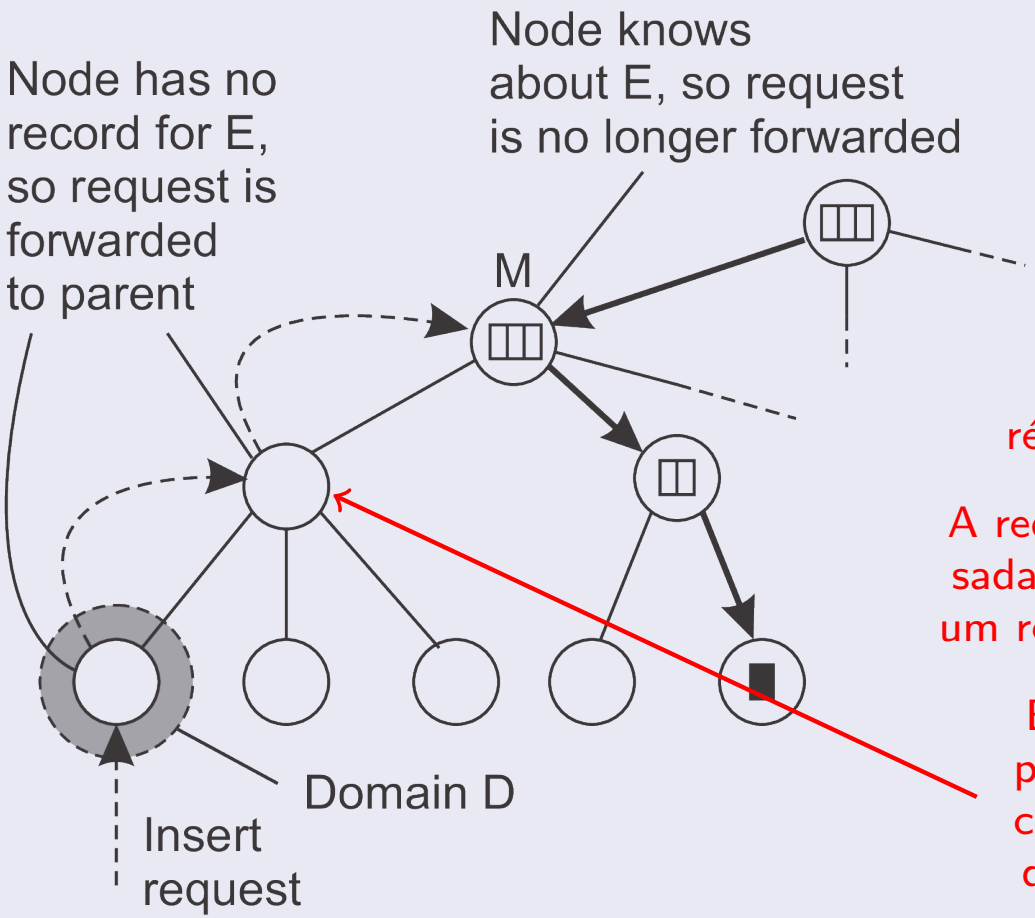


Considere uma entidade E que criou uma réplica no domínio folha D

A requisição de inserção é repassada até primeiro nó que possua um registro de localização para E

ENP – Abordagens Hierárquicas

Inserção



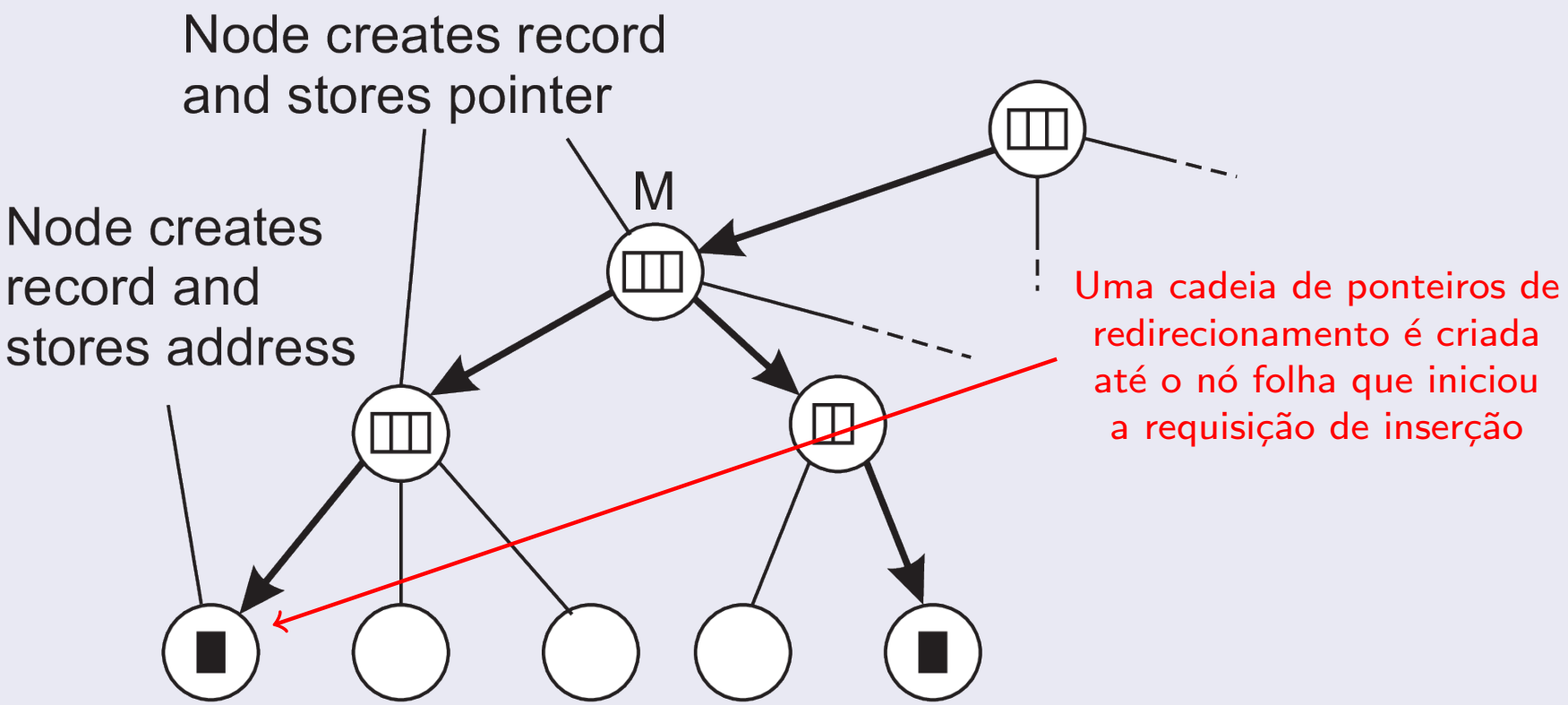
Considere uma entidade E que criou uma réplica no domínio folha D

A requisição de inserção é repassada até primeiro nó que possua um registro de localização para E

Este então armazena um ponteiro no registro de E, com uma referência ao nó de onde a requisição veio

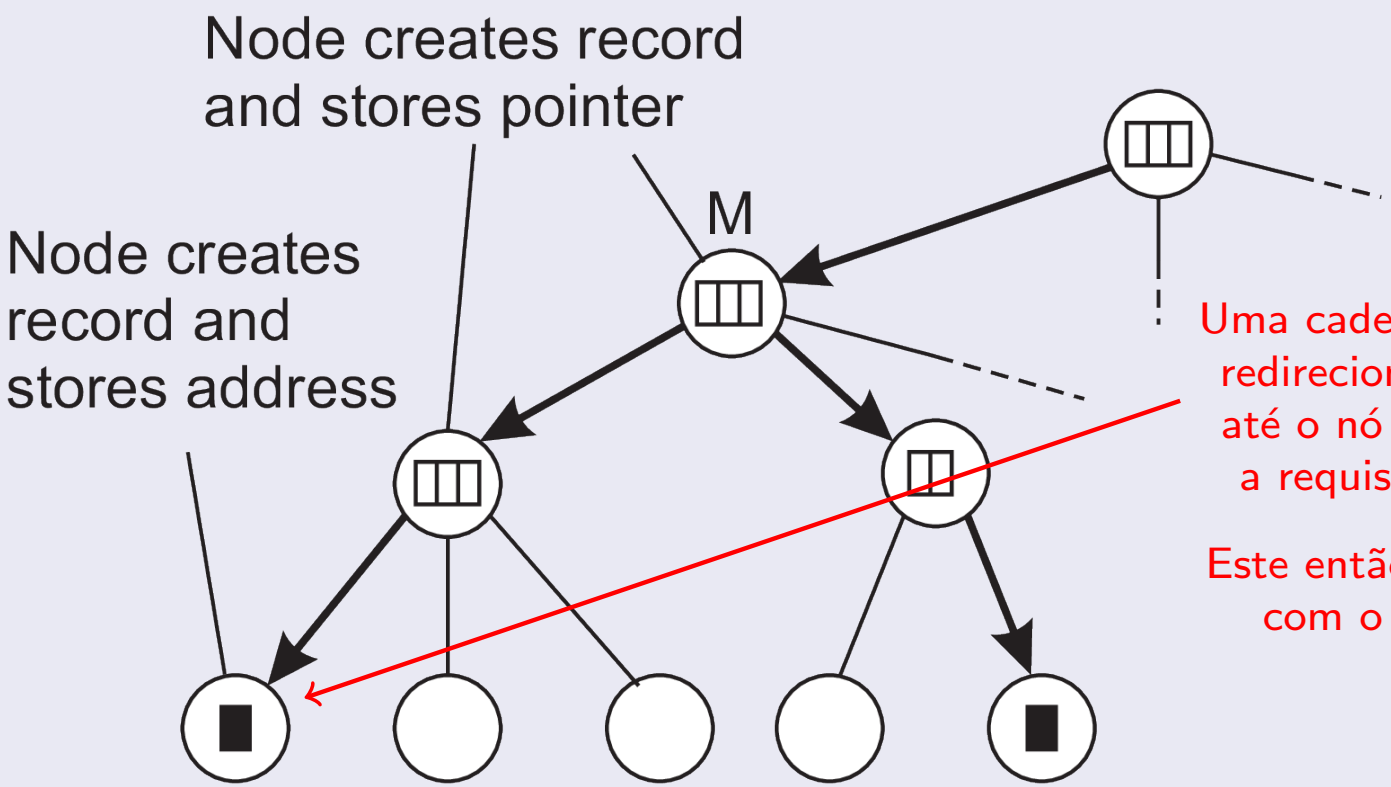
ENP – Abordagens Hierárquicas

Inserção



ENP – Abordagens Hierárquicas

Inserção

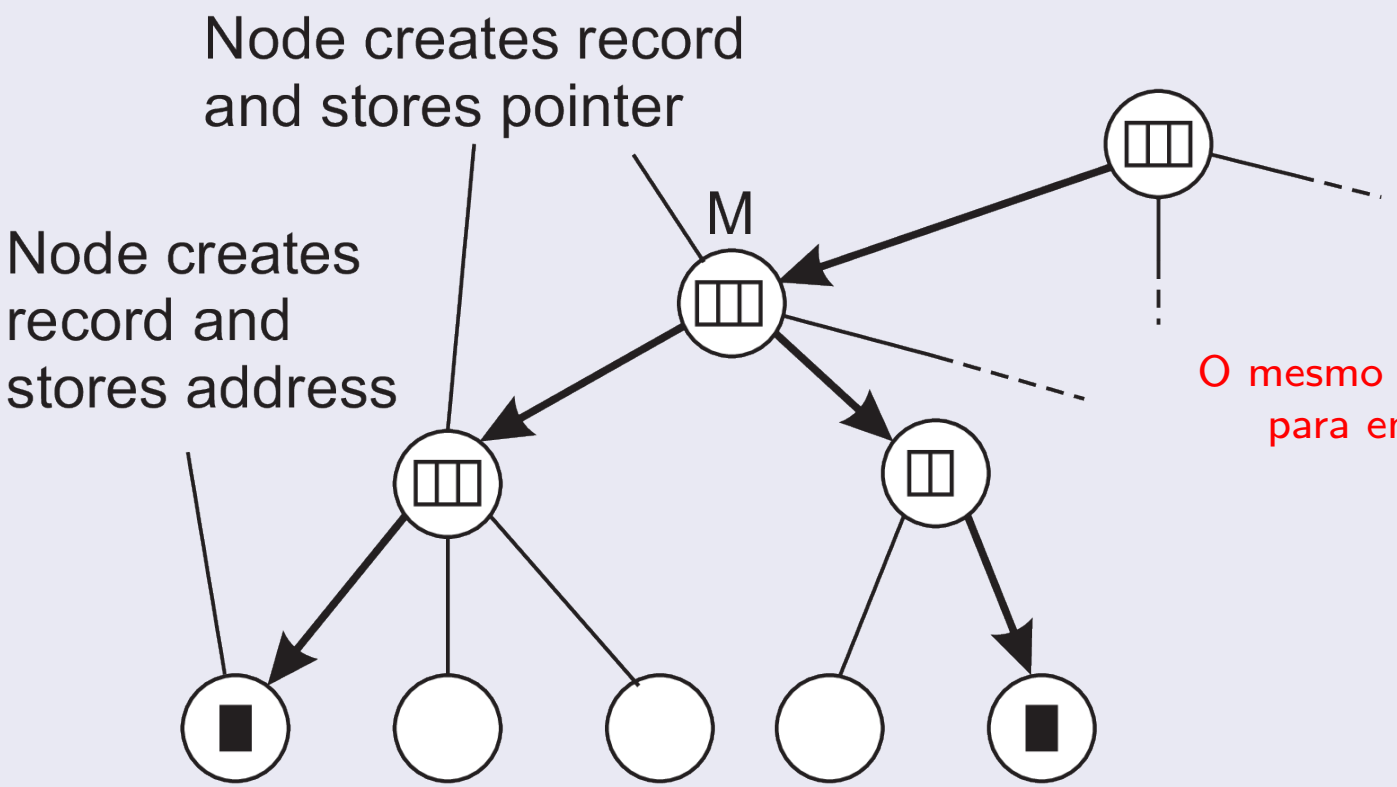


Uma cadeia de ponteiros de redirecionamento é criada até o nó folha que iniciou a requisição de inserção

Este então cria um registro com o endereço de *E*

ENP – Abordagens Hierárquicas

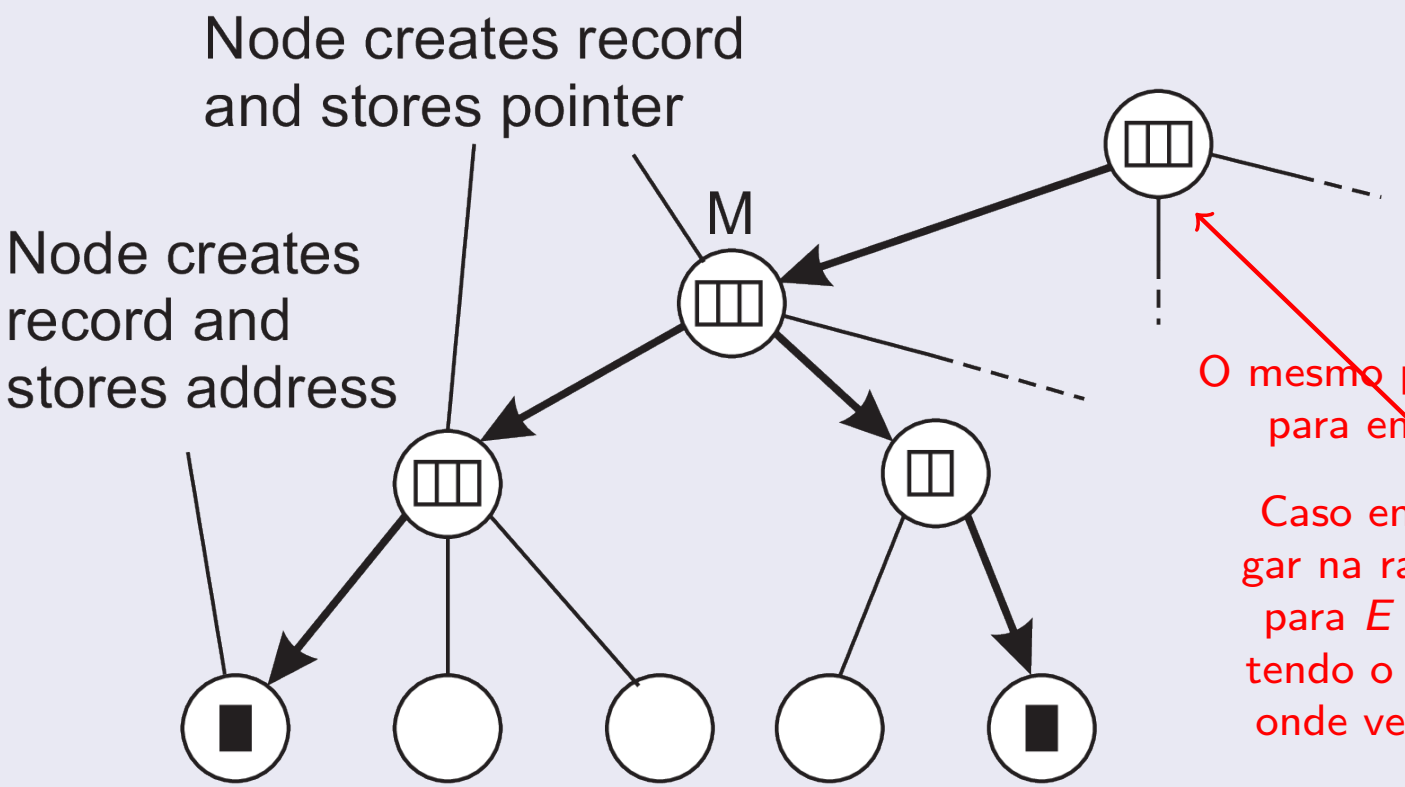
Inserção



O mesmo processo funciona para entidades novas

ENP – Abordagens Hierárquicas

Inserção

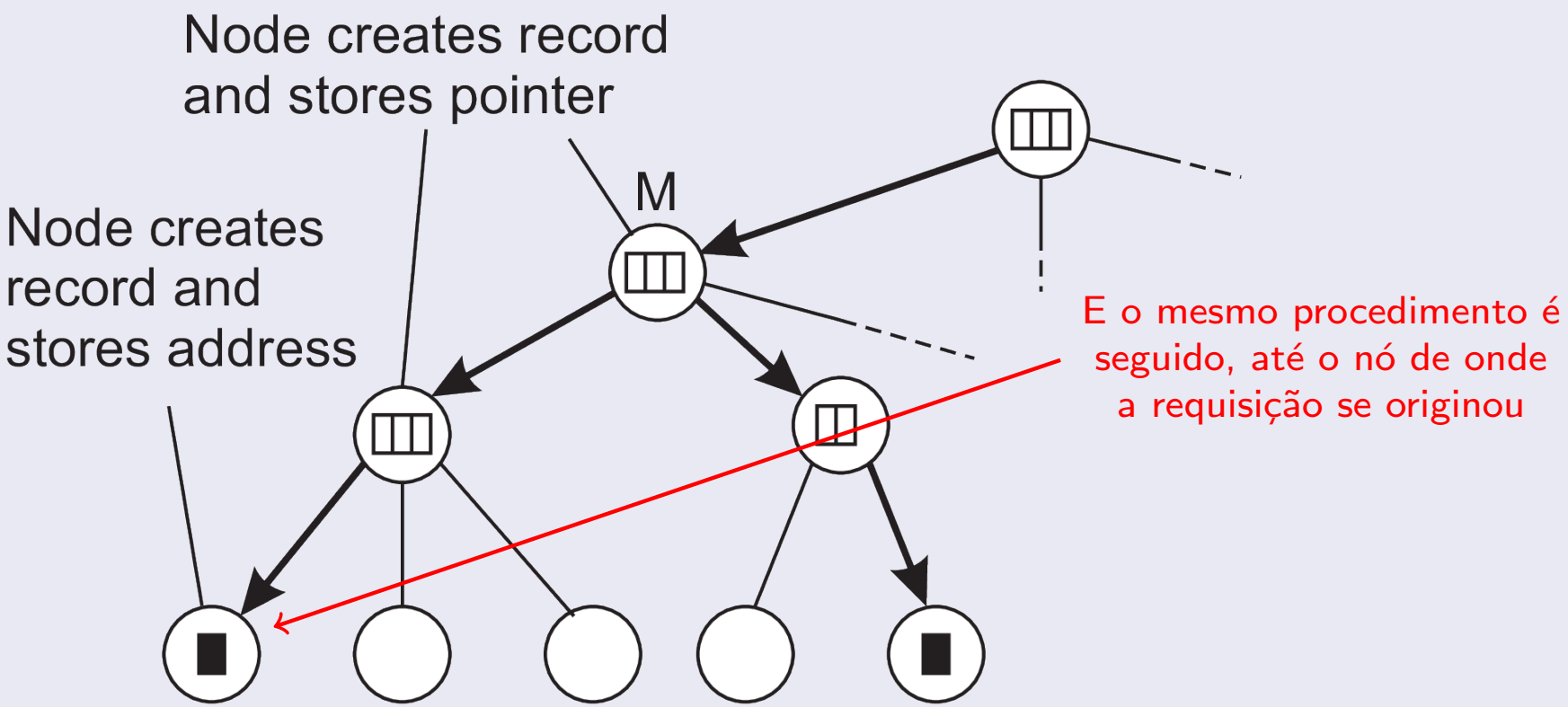


O mesmo processo funciona para entidades novas

Caso em que, ao chegar na raiz, um registro para E é criado, contendo o nó diretório de onde veio a requisição

ENP – Abordagens Hierárquicas

Inserção



ENP – Abordagens Hierárquicas

Remoção

- Suponha que queremos remover o endereço da entidade E do domínio folha D

ENP – Abordagens Hierárquicas

Remoção

- Suponha que queremos remover o endereço da entidade E do domínio folha D
- Pedimos que $dir(D)$ remova esse endereço do registro de E
  - Se o registro ficar vazio (sem outro endereço para E), remova-o e passe a requisição para o nó pai

ENP – Abordagens Hierárquicas

Remoção

- Suponha que queremos remover o endereço da entidade E do domínio folha D
- Pedimos que $dir(D)$ remova esse endereço do registro de E
  - Se o registro ficar vazio (sem outro endereço para E), remova-o e passe a requisição para o nó pai
- O mesmo procedimento é repetido no nó pai
  - O processo continua até encontrar um registro que não fique vazio, ou atingirmos a raiz