ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 16: Nomeação (parte 1)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

06/05/2024

Nomeação

Essência

Nomes são usados para denotar/referenciar entidades em um sistema (distribuído). Para operar em uma entidade, precisamos acessá-la em um ponto de acesso. Pontos de acesso são entidades que são nomeadas por meio de um endereço.

Observação 1

Entidade: praticamente qualquer coisa, como computadores, dispositivos (e.g. impressora, disco), arquivos, processos, usuários, conexões, mensagens.

Observação 2

Um nome independente de localização para uma entidade E é independente dos endereços dos pontos de acesso oferecidos por E.

Identificadores

Nome puro

Um nome que não tem significado algum; é apenas uma string aleatória.
Nomes puros podem ser usados apenas para comparação.

Identificador: Um nome com algumas propriedades específicas

1. Um identificador se refere a, no máximo, uma entidade.
2. Cada entidade é referida por, no máximo, um identificador.
3. Um identificador sempre se refere à mesma entidade (ou seja, nunca é reutilizado).

Observação

Um identificador não necessariamente precisa ser um nome puro, ou seja, pode ter conteúdo.

Como resolver nomes/identificadores

Principal desafio

Como resolver nomes e identificadores para um endereço?

Estratégias básicas

- Tabela (distribuída) de pares (nome, endereço)
- Uso do próprio nome para roteamento (e.g. redes P2P estruturadas)

Observações

1. Endereçamento está intimamente ligado com roteamento
2. Um nome pode ser machine-friendly ou human-friendly

Nomeação plana

Características

- Uso de sequências aleatórias de bits como identificadores
- Consequentemente, nome não contém nenhuma informação de como localizar o ponto de acesso associado à entidade

Broadcast

Transmitir o ID em broadcast, solicitando que a entidade retorne seu endereço atual

- Não pode escalar além de redes locais
- Requer que todos os processos escutem as solicitações de localização recebidas

Protocolo de Resolução de Endereços (ARP)

Para descobrir qual endereço MAC está associado a um endereço IP, pergunta para todos no enlace ”quem possui este endereço IP”?

Encaminhamento de ponteiros

Quando uma entidade se move, deixa um ponteiro para sua próxima localização

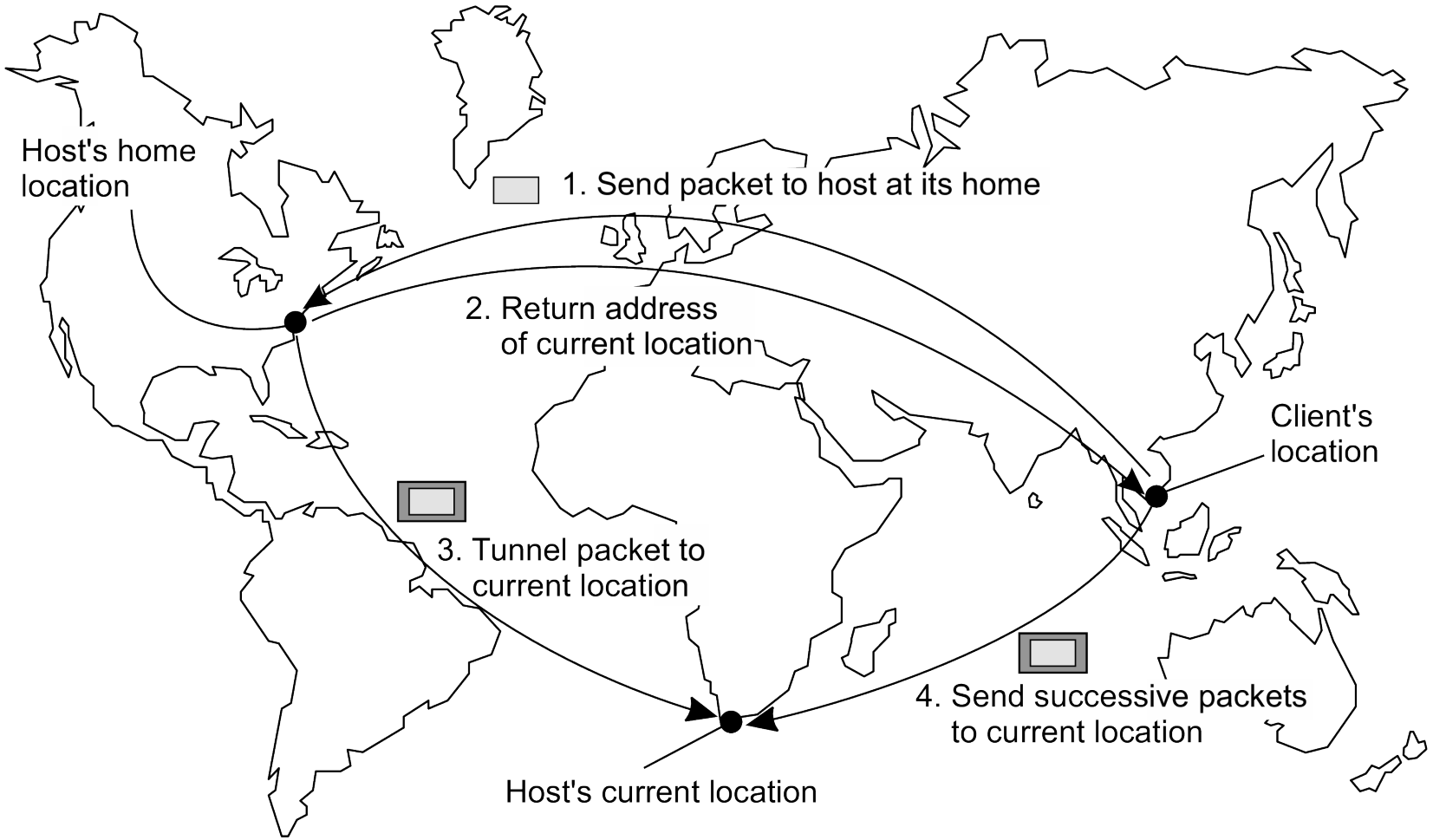
- O desreferenciamento pode ser feito de forma completamente transparente para os clientes simplesmente seguindo a cadeia de ponteiros
- A referência de um cliente é atualizada quando a localização atual for encontrada
- Problemas de escalabilidade geográfica (para os quais são necessários mecanismos de redução de cadeia):
  - Cadeias longas não são tolerantes a falhas
  - Aumento da latência ao desreferenciar

Abordagens baseadas em localização nativa (home-based)

Esquema de um único nível: um agente nativo acompanha onde a entidade está

- O endereço nativo da entidade é registrado em um serviço de nomeação
- O agente nativo registra o endereço remoto da entidade
- O cliente entra em contato com o agente nativo primeiro e depois continua com a localização remota

O princípio do IP móvel



Abordagens baseadas em localização nativa

Problemas com abordagens baseadas em localização nativa

- O endereço do agente nativo precisa ser suportado durante toda a vida útil da entidade
- O endereço do agente nativo é fixo $\Rightarrow$ ônus desnecessário quando a entidade se move permanentemente
- Escalabilidade geográfica ruim (a entidade pode estar próxima do cliente)

Observação

Movimentações permanentes podem ser tratadas com outro nível de nomeação (DNS)

Tabelas de hash distribuídas

Exemplo: **Chord**

Considere a organização de muitos nós em um anel lógico

- Cada nó é atribuído a um identificador aleatório de m bits.
- Cada entidade é atribuída a uma chave única de m bits.
- A entidade com a chave k está sob a jurisdição do nó com menor id tal que $id \geq k$ (chamado de seu sucessor $succ(k)$).

Uma **não** solução

Cada nó acompanha seu vizinho e faz uma pesquisa linear ao longo do anel.

Tabelas de derivação do Chord

Princípio

- Cada nó p mantém uma tabela de derivação (finger table) $FT_p[]$ com no máximo m entradas:

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

Nota: a i-ésima entrada aponta para o primeiro nó com id pelo menos 2^{i-1} maior que p.

Tabelas de derivação do Chord

Princípio

- Cada nó p mantém uma tabela de derivação (finger table) $FT_p[]$ com no máximo m entradas:

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

  Nota: a i-ésima entrada aponta para o primeiro nó com id pelo menos 2^{i-1} maior que p.

- Para procurar uma chave k, o nó p encaminha a solicitação para o nó com índice j que satisfaz

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

Tabelas de derivação do Chord

Princípio

- Cada nó p mantém uma tabela de derivação (finger table) $FT_p[]$ com no máximo m entradas:

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

  Nota: a i-ésima entrada aponta para o primeiro nó com id pelo menos 2^{i-1} maior que p.

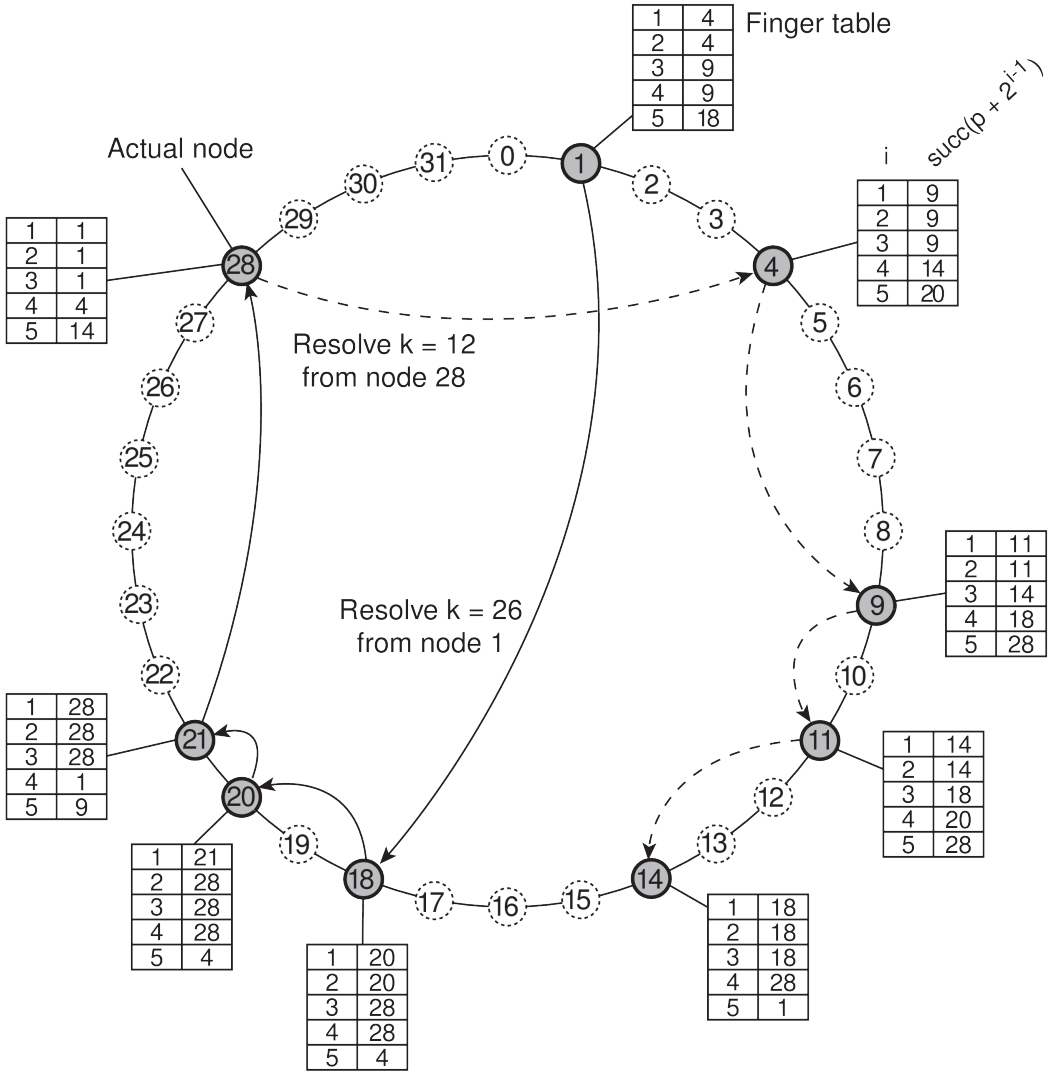
- Para procurar uma chave k, o nó p encaminha a solicitação para o nó com índice j que satisfaz

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

- Se $p < k < FT_p[1]$, a solicitação também é encaminhada para $FT_p[1]$

Exemplo de busca do Chord

Resolvendo a chave 26 a partir do nó *1* e chave *12* a partir do nó *28*



Alterando topologia do Chord

Adicionando nó

- Novo nó que procura nó $s = succ(q+1)$
- Nó s faz $pred(s) = q$

Manutenção

- Periodicamente recalcular FT
- Periodicamente verificar se $pred(FT[1])$ é o próprio nó

Explorando proximidade

Problema

A organização lógica de nós na rede overlay pode levar a transferências de mensagens erráticas na rede subjacente: o nó p e o nó $succ(p+1)$ podem estar muito distantes.

Soluções

Explorando proximidade

Problema

A organização lógica de nós na rede overlay pode levar a transferências de mensagens erráticas na rede subjacente: o nó p e o nó $succ(p+1)$ podem estar muito distantes.

Soluções

- Atribuição de nós ciente de topologia: Ao atribuir um ID a um nó, certificar-se de que os nós próximos no espaço de IDs também estejam próximos na rede. Pode ser muito difícil.

Explorando proximidade

Problema

A organização lógica de nós na rede overlay pode levar a transferências de mensagens erráticas na rede subjacente: o nó p e o nó $succ(p+1)$ podem estar muito distantes.

Soluções

- Atribuição de nós ciente de topologia: Ao atribuir um ID a um nó, certificar-se de que os nós próximos no espaço de IDs também estejam próximos na rede. Pode ser muito difícil.
- Encaminhamento por proximidade: Manter mais de um sucessor possível e encaminhar para o mais próximo.
  Exemplo: no Chord, $FT_p[i]$ aponta para o primeiro nó em $INT = [p+2^{i-1}, p+2^i-1]$. O nó p também pode armazenar ponteiros para outros nós no INT.

Explorando proximidade

Problema

A organização lógica de nós na rede overlay pode levar a transferências de mensagens erráticas na rede subjacente: o nó p e o nó $succ(p+1)$ podem estar muito distantes.

Soluções

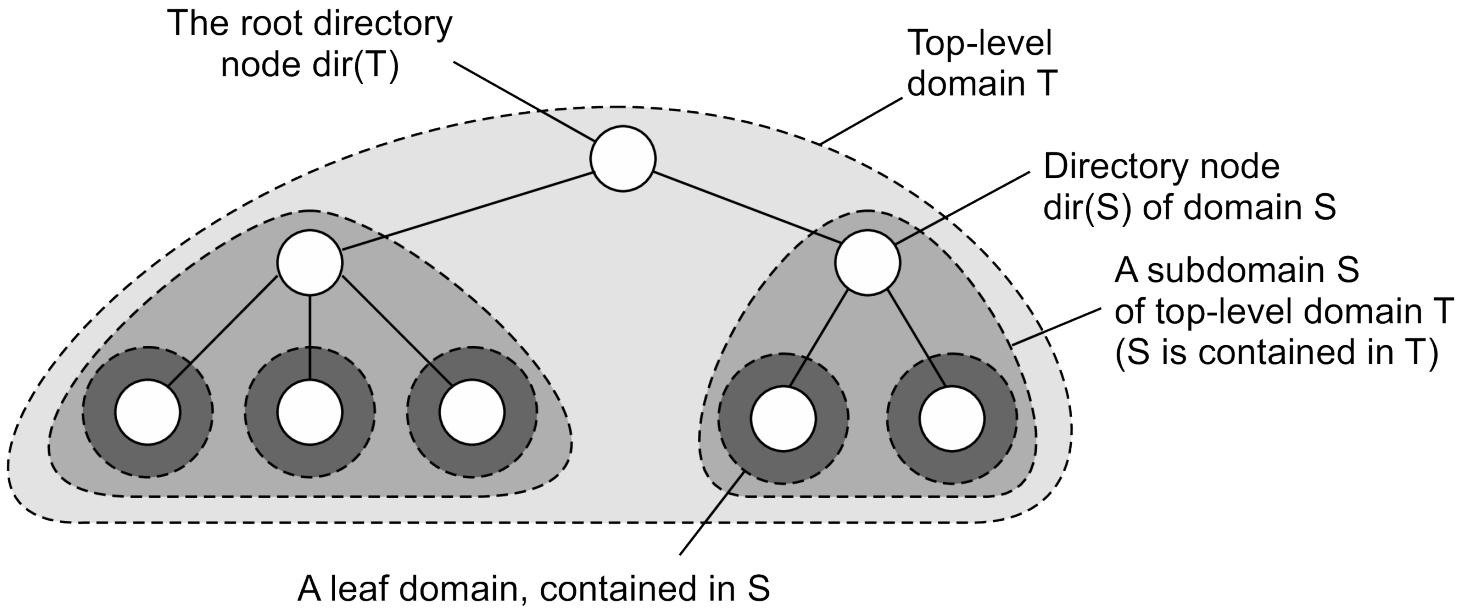
- Atribuição de nós ciente de topologia: Ao atribuir um ID a um nó, certificar-se de que os nós próximos no espaço de IDs também estejam próximos na rede. Pode ser muito difícil.
- Encaminhamento por proximidade: Manter mais de um sucessor possível e encaminhar para o mais próximo.
  Exemplo: no Chord, $FT_p[i]$ aponta para o primeiro nó em $INT = [p+2^{i-1}, p+2^i-1]$. O nó p também pode armazenar ponteiros para outros nós no INT.
- Seleção de vizinho por proximidade: Quando houver a possibilidade de escolher quem será seu vizinho (não é o caso do Chord), escolher o mais próximo.

Serviços de Localização Hierárquica (Hierarchical Location Services – HLS)

Ideia básica

Construir uma árvore de busca em grande escala na qual a rede subjacente é dividida em domínios hierárquicos. Cada domínio é representado por um nó de diretório separado.

Princípio



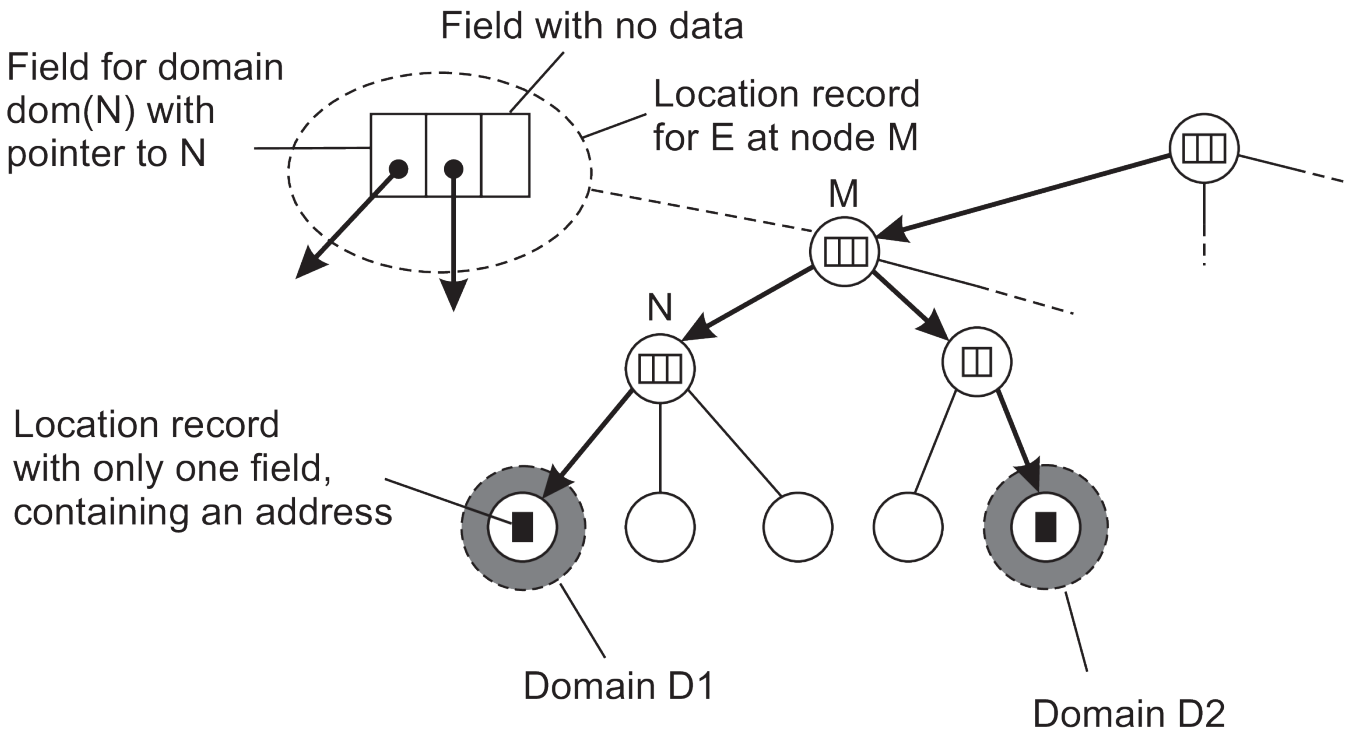
HLS: Organização em árvore

Invariantes

- O endereço da entidade *E* é armazenado em um nó folha ou intermediário
- Nós intermediários contêm um ponteiro para um filho se e somente se a subárvore com raiz no filho armazenar um endereço da entidade
- A raiz sabe sobre todas as entidades

HLS: Organização em árvore

Armazenando informações de uma entidade com dois endereços em domínios folha diferentes

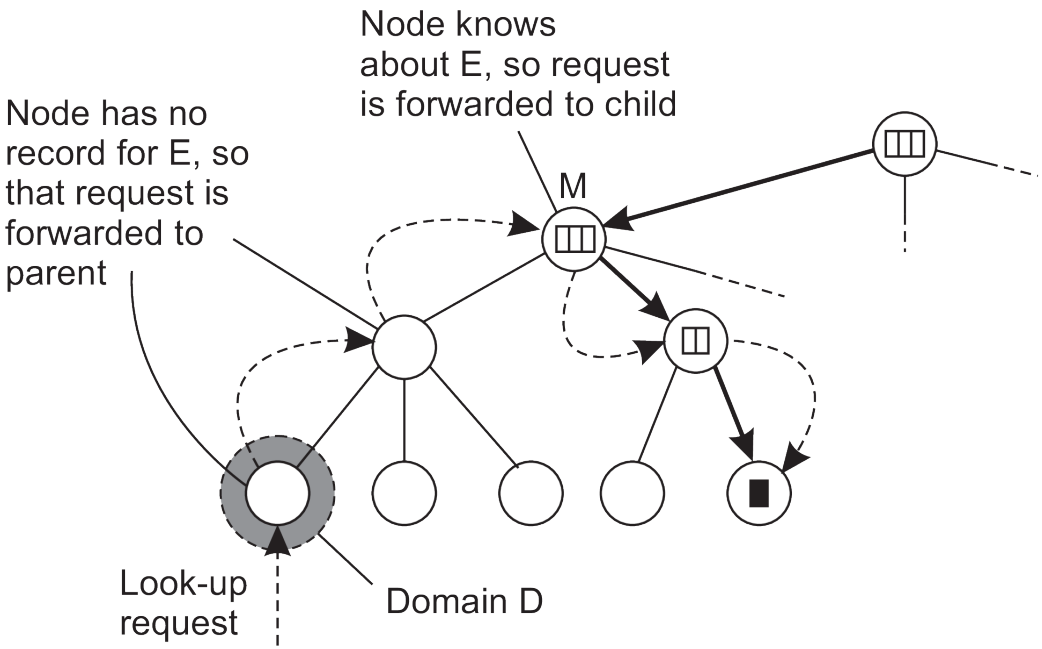


HLS: Operação de busca

Princípios básicos

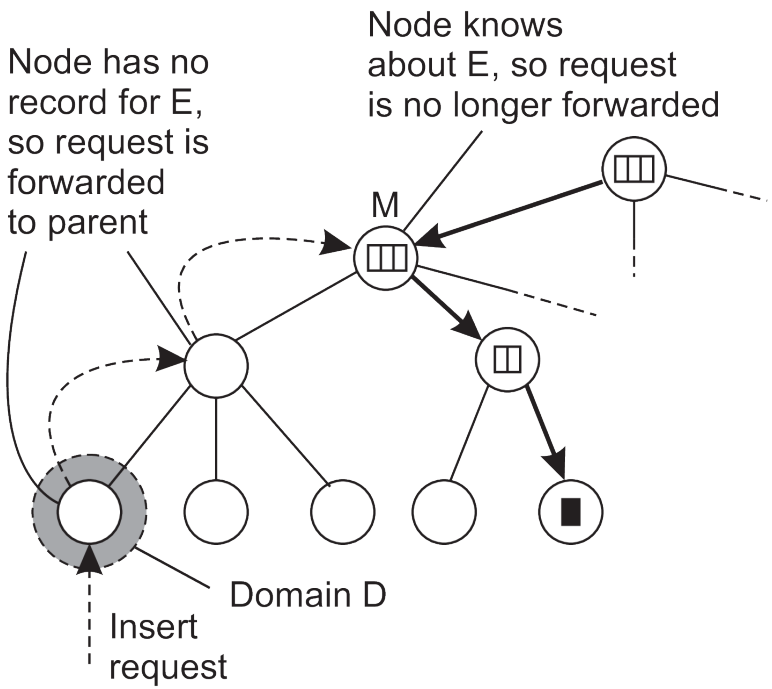
- Iniciar a busca no nó folha local
- Se o nó souber sobre $E \Rightarrow$ seguir o ponteiro descendente, caso contrário, suba
- A busca para cima sempre para na raiz

Procurando uma localização

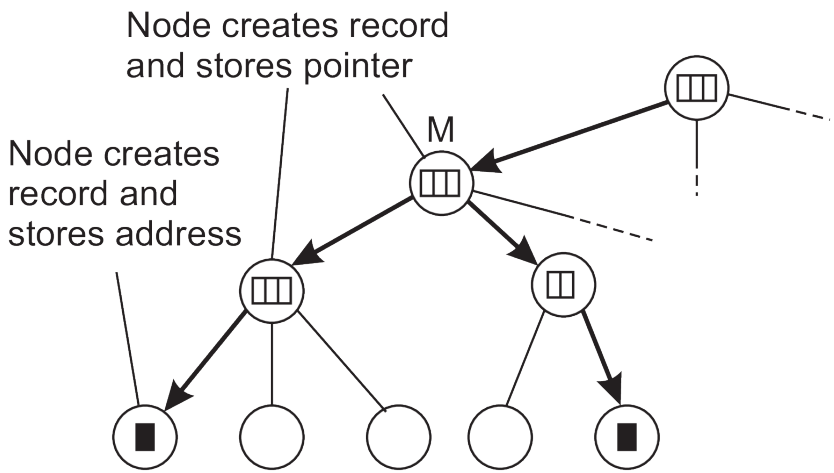


HLS: Operação de inserção

(a) Uma solicitação de inserção é encaminhada para o primeiro nó que conhece a entidade *E*.
(b) Uma cadeia de ponteiros de encaminhamento para o nó folha é criada



(a)



(b)

Segurança na nomeação plana

Conceitos básicos

Sem medidas especiais, precisamos confiar que o processo de resolução de nome retornará o que realmente está associado a um nome plano. Dois enfoques a seguir:

- Tornar segura a associação identificador-entidade
- Tornar seguro o processo de resolução de nomes

Nomes auto-certificados (self-certifying)

Usar um valor derivado da entidade e fazer dele (parte do) nome plano:

- $id(entidade) = hash(dados\ associados\ à\ entidade)$

ao lidar com entidades somente leitura. Caso contrário:

- $id(entidade) = chave\ pública(entidade)$

nesse caso, dados adicionais são retornados, como uma assinatura digital verificável.

Tornar seguro o processo de resolução de nomes

Mais complicada: discutiremos quando estudar DNS seguro.