ACH 2147 — Desenvolvimento de Sistemas de Informação Distribuídos

Aula 11: Comunicação (parte 3)

Prof. Renan Alves

Escola de Artes, Ciências e Humanidades — EACH — USP

08/04/2024

Multicasting

Essência

- De forma simplificada, comunicação multicast envolve enviar uma mensagem para vários destinos
- Tradicionalmente é um problema tratado nas camadas de rede ou transporte
  - Foco em descobrir e manter uma estrutura de encaminhamento

Multicasting

Essência

- De forma simplificada, comunicação multicast envolve enviar uma mensagem para vários destinos
- Tradicionalmente é um problema tratado nas camadas de rede ou transporte
  - Foco em descobrir e manter uma estrutura de encaminhamento
- Porém, em sistemas distribuídos (especialmente P2P), é comum haver uma rede overlay

Multicasting

Essência

- De forma simplificada, comunicação multicast envolve enviar uma mensagem para vários destinos
- Tradicionalmente é um problema tratado nas camadas de rede ou transporte
  - Foco em descobrir e manter uma estrutura de encaminhamento
- Porém, em sistemas distribuídos (especialmente P2P), é comum haver uma rede overlay
  - Neste caso, faz mas sentido aplicar técnicas de multicasting na camada de aplicação

Multicasting

Essência

- De forma simplificada, comunicação multicast envolve enviar uma mensagem para vários destinos
- Tradicionalmente é um problema tratado nas camadas de rede ou transporte
  - Foco em descobrir e manter uma estrutura de encaminhamento
- Porém, em sistemas distribuídos (especialmente P2P), é comum haver uma rede overlay
  - Neste caso, faz mas sentido aplicar técnicas de multicasting na camada de aplicação
  - Contudo, provavelmente não é a solução mais otimizada

Multicasting em nível de aplicação

Essência

Organizar nós de um sistema distribuído em uma rede overlay e usar essa rede para disseminar dados:

- Muitas vezes uma árvore, resultando em caminhos únicos
- Alternativamente, também redes em malha (mesh), exigindo alguma forma de roteamento

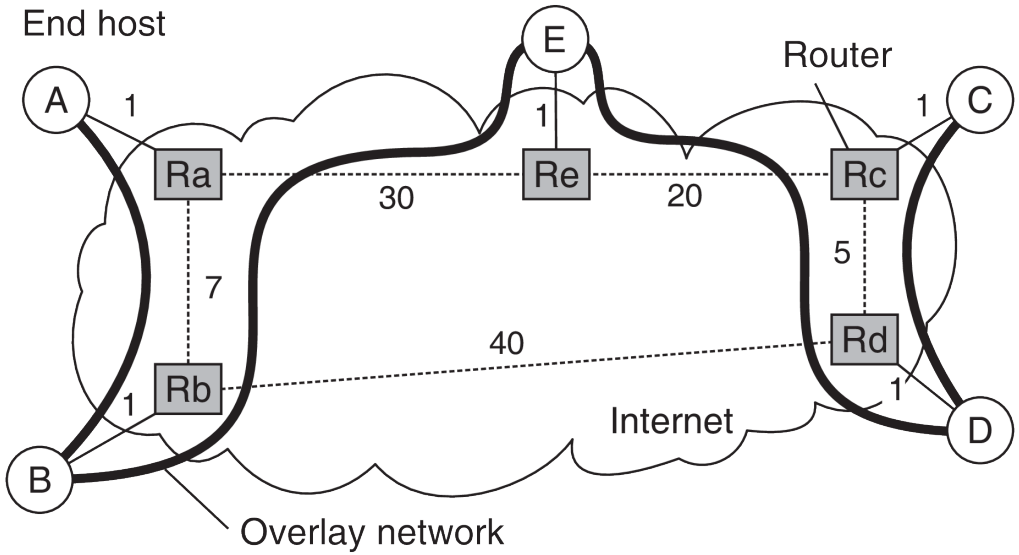
Multicasting em nível de aplicação em Chord

Abordagem básica

1. O iniciador gera um identificador de multicast *mid*.
2. Pesquisa *succ*(*mid*), o nó responsável por *mid*.
3. A solicitação é roteada para *succ*(*mid*), que se tornará a raiz.
4. Se *P* deseja ingressar no chord, ele envia uma solicitação de `entrar` para a raiz (basicamente vai buscar por *mid*).
5. Quando uma solicitação chega em *Q*:
   - *Q* não viu uma solicitação de `entrar` antes $\Rightarrow$ ele se torna um encaminhador; *P* se torna filho de *Q*. A solicitação de junção continua sendo encaminhada.
   - *Q* sabe sobre a árvore $\Rightarrow$ *P* se torna filho de *Q*. Não é mais necessário encaminhar a solicitação de junção.
6. Para enviar uma mensagem multicast: basta enviá-la para a raiz, que a dissemina através da árvore criada

Application Layer Multicasting (ALM): alguns custos

Diferentes métricas



- Estresse de enlace: Com que frequência uma mensagem ALM atravessa o mesmo link físico?
  Ex: mensagem de A para D precisa atravessar $\langle Ra, Rb \rangle$ duas vezes.
- Esticamento: Proporção no custo (tipicamente atraso) entre o caminho a nível ALM e o caminho a nível de rede.
  Exemplo: mensagens de B para C seguem um caminho de comprimento 73 em ALM, mas 47 em nível de rede $\Rightarrow$ esticamento = 73/47.

Application Layer Multicasting (ALM): custo da árvore

Custo da árvore

- Métrica global: custo total dos links da árvore

Se o custo for atraso

- No momento que um novo nó entrar, como escolher seu pai?
- Poderia colocar todos os novos nós como filhos da raiz...
- Na prática, limitar em k filhos
- Heurística: verificação periódica se há um pai melhor

Falhas

- Se um nó da árvore falhar...

Flooding

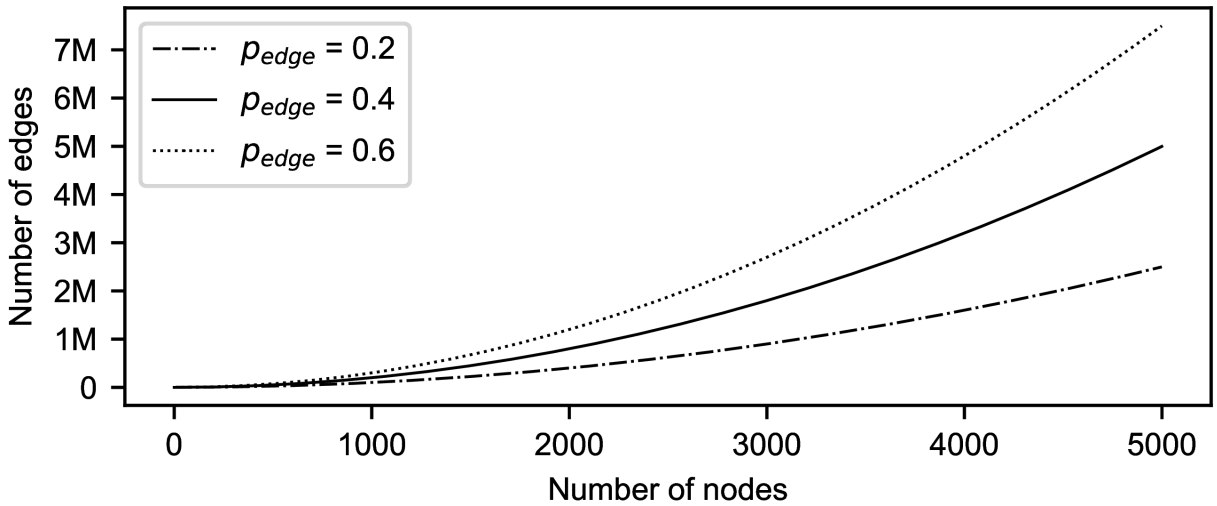
Enviando mensagens

- Uma vez que o grupo de multicast esteja construído, será desejado enviar mensagens.
- Se houver uma rede overlay para cada grupo de multicast diferente, podemos usar uma estratégia simples como o flooding.

Flooding

Essência

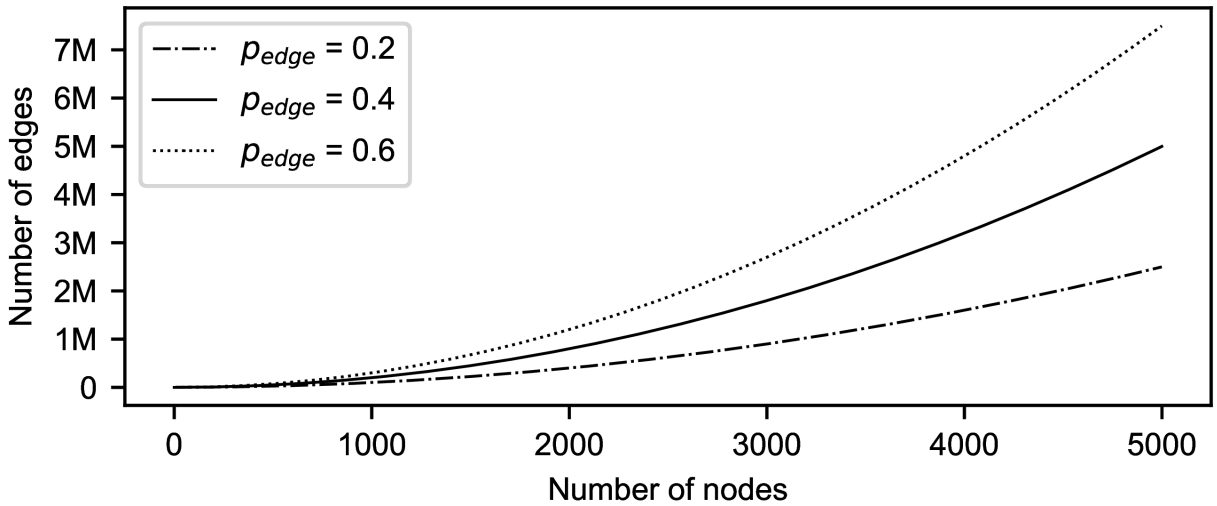
P simplesmente envia uma mensagem *m* para cada um de seus vizinhos.
Cada vizinho encaminhará essa mensagem, exceto para *P*, e somente se não tiver visto *m* antes.
Quantidade de mensagens proporcional ao número de arestas



Flooding

Essência

P simplesmente envia uma mensagem *m* para cada um de seus vizinhos.
Cada vizinho encaminhará essa mensagem, exceto para *P*, e somente se não tiver visto *m* antes.
Quantidade de mensagens proporcional ao número de arestas



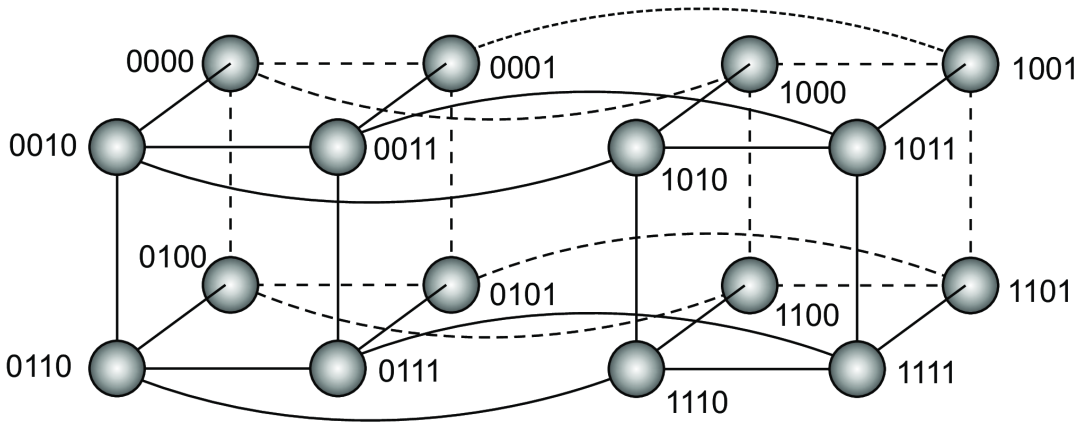
Variação

Fazer com que *Q* encaminhe uma mensagem com uma certa probabilidade p_{flood}, possivelmente dependente do seu número de vizinhos (ou seja, grau do nó) ou do grau de seus vizinhos.

Utilizando estrutura da rede

Evitando flooding

- Se a rede for estruturada, é possível utilizar a estrutura para reduzir a quantidade total de mensagens utilizada para fazer a transmissão multicast
- Exemplo: hipercubo
  - Limitar arestas que podem ser usadas: apenas alguns bits
  - 1001 envia mensagem m: (m,1) para 0001; (m,2) para 1101; (m,3) to 1011; (m,4) to 1000
  - 1101 envia: (m,3) para 1111; (m,4) para 1100



Protocolos epidêmicos (gossip)

Há nós infectados e nós suscetíveis

- Objetivo é infectar todos
- Qualquer nó deve conseguir alcançar qualuer outro (difícil na prática)

Hipótese de que não há conflitos de escrita–escrita

- Atualização de um item de dados são inicializadas em um único nó
- Uma réplica passa o estado atualizado para apenas alguns vizinhos
- A propagação de atualização é lenta, ou seja, não imediata
- Eventualmente, valores atualizados devem alcançar todas as réplicas

Protocolos epidêmicos (gossip)

Duas formas de epidemia

- Anti-entropia: Cada réplica escolhe periodicamente outra réplica aleatória e verifica as diferenças de estado, levando a estados idênticos em ambos após a troca de informações
- Propagação de boatos: Uma réplica que acabou de ser atualizada (ou seja, foi contaminada), conta a várias outras réplicas sobre sua atualização (contaminando-as também).

Anti-entropia

Operações principais

- Um nó P seleciona outro nó Q do sistema aleatoriamente.
- Pull: P apenas pega novas atualizações de Q
- Push: P apenas envia suas próprias atualizações para Q
- Push-pull: P e Q enviam atualizações um ao outro

Observação 1

Assume-se que há alguma forma de discernir dados novos dos antigos (e.g. timestamps)

Observação 2

A operação push-pull leva $\mathcal{O}(log(N))$ rodadas para disseminar atualizações para todos os N nós (rodada = quando cada nó tomou a iniciativa de iniciar uma troca).

Anti-entropia: análise

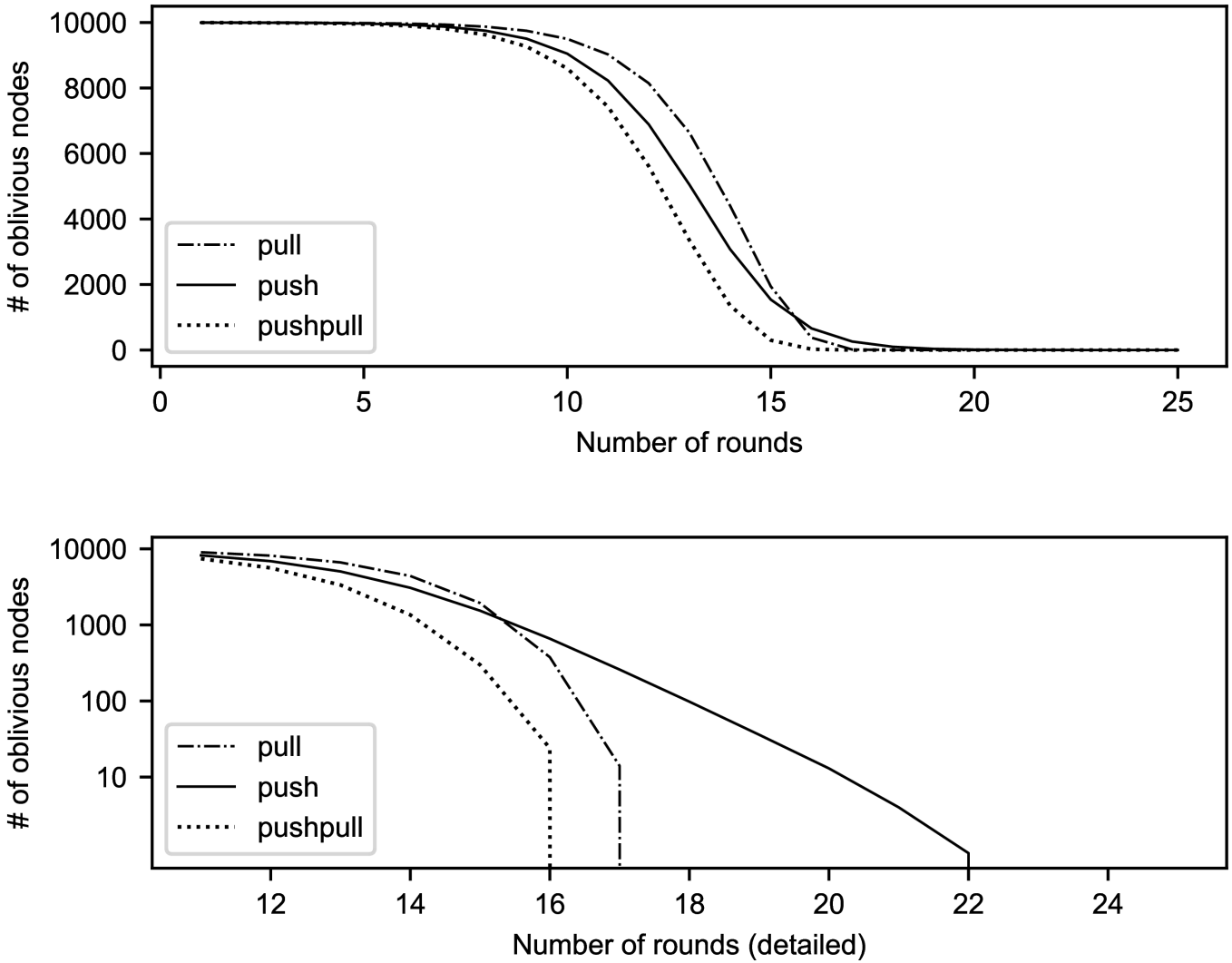
Básico

Considere uma única fonte, propagando sua atualização. Considere que p_i seja a probabilidade de um nó não ter recebido a atualização após a i-ésima rodada.

Análise: permanecendo ignorante

- Com pull, $p_{i+1} = (p_i)^2$: o nó não foi atualizado durante a i-ésima rodada e deve entrar em contato com outro nó ignorante durante a próxima rodada.
- Com push, $p_{i+1} = p_i(1 - \frac{1}{N-1})^{(N-1)(1-p_i)} \approx p_i e^{-1}$ (para p_i pequeno e N grande): o nó não foi atualziado durante a i-ésima rodada e nenhum nó atualizado escolhe contatá-lo durante a próxima rodada.
- Com push-pull: $(p_i)^2 \cdot (p_i e^{-1})$

Anti-entropia: desempenho



Propagação de boatos (rumor spreading)

Modelo básico

Um servidor S que tem uma atualização para relatar, entra em contato com outros servidores. Se um servidor para o qual a atualização já se propagou for contatado, S para de contatar outros servidores com probabilidade p_{stop}.

Observação

Se s é a fração de servidores ignorantes (ou seja, que não estão cientes da atualização), pode-se mostrar que com muitos servidores

$$s = e^{-(1/p_{stop}+1)(1-s)}$$

Análise formal

Notações

Defina s como a fração de nós que ainda não foram atualizados (ou seja, suscetíveis); i a fração atualizada (infectada) e ativa de nós; e r a fração de nós atualizados que desistiram (removidos).

Da teoria das epidemias

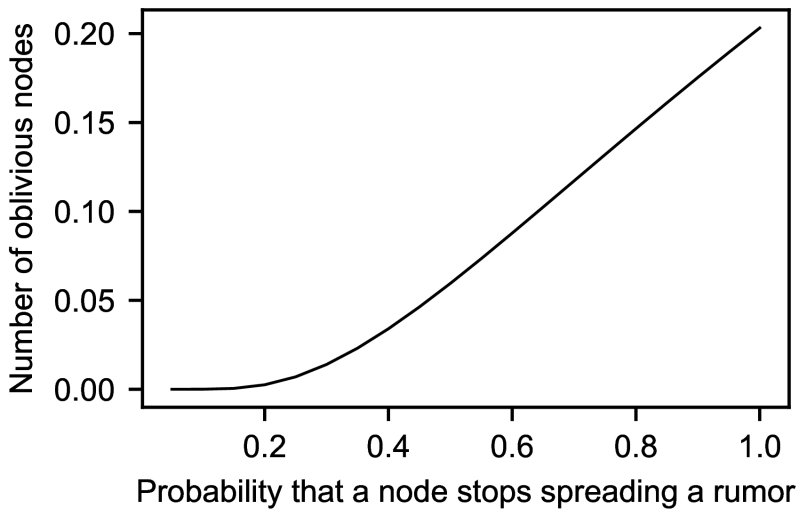
$$
\begin{array}{lrcl}
(1) & ds/dt & = & -s \cdot i \\
(2) & di/dt & = & s \cdot i - p_{stop} \cdot (1-s) \cdot i \\
\Rightarrow & di/ds & = & -(1+p_{stop}) + \frac{p_{stop}}{s} \\
\Rightarrow & i(s) & = & -(1+p_{stop}) \cdot s + p_{stop} \cdot \ln(s) + C
\end{array}
$$

Conclusão

$i(1) = 0 \Rightarrow C = 1 + p_{stop} \Rightarrow i(s) = (1+p_{stop}) \cdot (1-s) + p_{stop} \cdot \ln(s)$. Estamos procurando o caso $i(s) = 0$, o que leva a $s = e^{-(1/p_{stop}+1)(1-s)}$

Propagação de boatos

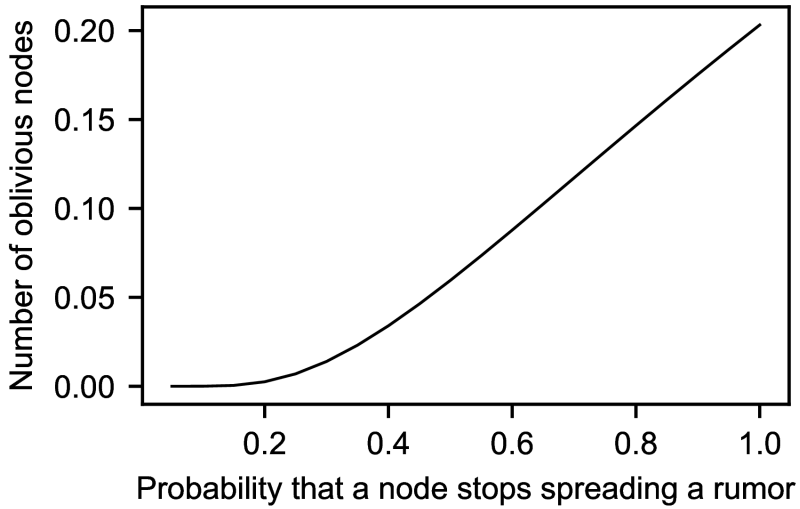
O efeito de parar



Considere 10.000 nós		
$1/p_{stop}$	s	N_s
1	0,203188	2032
2	0,059520	595
3	0,019827	198
4	0,006977	70
5	0,002516	25
6	0,000918	9
7	0,000336	3

Propagação de boatos

O efeito de parar



Considere 10.000 nós		
$1/p_{stop}$	s	N_s
1	0,203188	2032
2	0,059520	595
3	0,019827	198
4	0,006977	70
5	0,002516	25
6	0,000918	9
7	0,000336	3

Observação

Se realmente precisamos garantir que todos os servidores sejam eventualmente atualizados, a propagação de boatos por si só não é suficiente

Excluindo valores

Problema fundamental

Não podemos remover um valor antigo de um servidor e esperar que a remoção se propague: a remoção simples seria desfeita pelo algoritmo epidêmico ao receber uma cópia antiga

Solução

A remoção deve ser registrada como uma atualização especial inserindo um atestado de óbito

Excluindo valores

Quando remover um atestado de óbito (já que não deve ser mantido para sempre)

- Executar um algoritmo global para detectar se a remoção é conhecida em todos os lugares e, em seguida, coletar os atestados de óbito (semelhante a *garbage collection*)
- Supor que os atestados de óbito se propaguem em um tempo finito e associar um tempo de vida máximo para um certificado (pode ser feito com o risco de não alcançar todos os servidores, alguns nós podem manter o certificado como controle)

Nota

É necessário que a remoção alcance todos os servidores de fato.