ACH 2147 — DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO DISTRIBUÍDOS

COMUNICAÇÃO MULTICAST

Daniel Cordeiro

2 de maio e
“May the 4th be with you  ” de 2018

Escola de Artes, Ciências e Humanidades | EACH | USP

AVISO: 1ª PROVA

Data 16 (quarta) e 18 (sexta) de maio

Horário das 20h às 22h

Observações

- a prova começa e termina pontualmente nesses horários;
- não é permitido fazer a prova com a outra turma;
- o conteúdo da prova (capítulos 1 ao 4) não se restringe aos slides!

Multicast

COMUNICAÇÃO MULTICAST

- Transmissão de mensagens multicast no nível da aplicação
- Disseminação de dados com métodos de *gossiping*

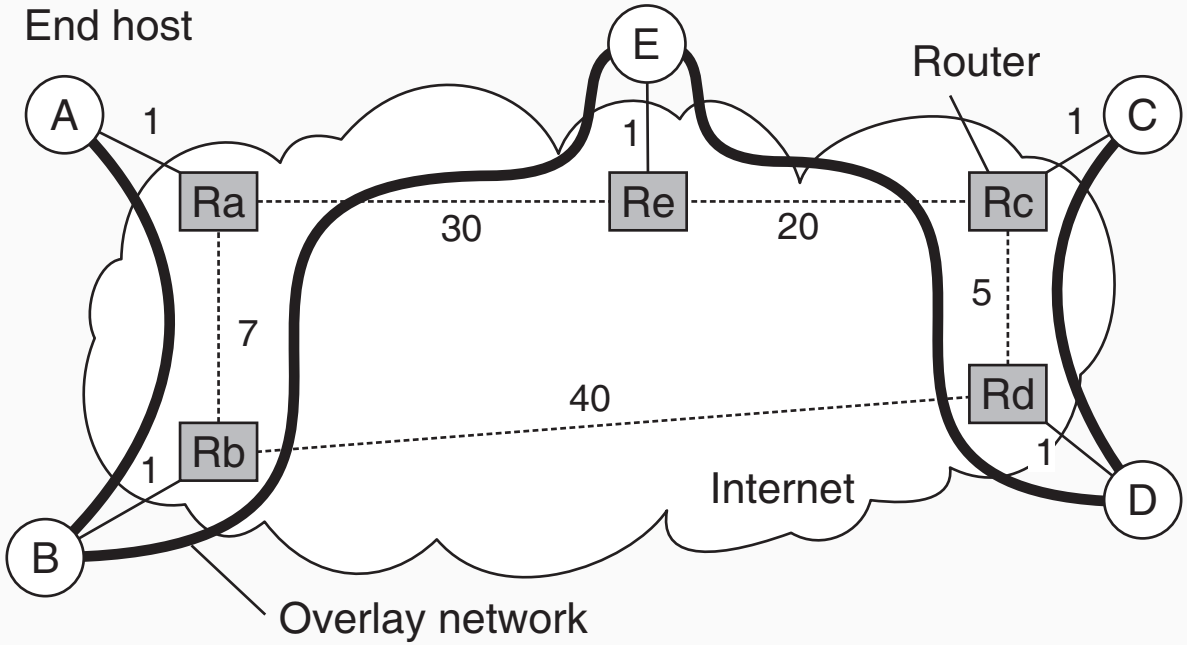
Ideia geral

Organizar os nós de um sistema distribuído em uma rede overlay e usar a rede para disseminar os dados.

Construção de árvores pelo protocolo Chord

1. O iniciador gera um identificador multicast `mid`
2. `Lookup` por `succ(mid)`, o nó responsável por `mid`
3. Requisição é roteada para `succ(mid)`, que será designado root
4. Se *P* quiser se juntar, ele envia uma requisição do tipo `join` ao root.
5. Quando uma requisição chegar em *Q*:
   - se *Q* não viu nenhuma requisição `join` antes, ele se torna um forwarder; *P* se torna filho de *Q* e a requisição de `join` continua a ser repassada
   - se *Q* sabe sobre a existência da árvore, *P* se torna filho de *Q* (como antes), mas não é mais necessário repassar a requisição de `join`

MULTICAST NO NÍVEL DA APLICAÇÃO: ALGUNS CUSTOS



- Stress nos links: com que frequência uma mensagem de multicast será enviada pelo mesmo enlace físico? Exemplo: uma mensagem de A para D precisa atravessar $\langle Ra, Rb \rangle$ duas vezes
- Stretch: razão entre o atraso da comunicação usando o caminho multicast e usando a rede. Exemplo: mensagens de B para C seguem o caminho de tamanho 73 no multicast, mas um de 47 existe no nível da rede. stretch = 73/47

PROTOCOLOS EPIDÊMICOS

- Contexto
- Modelos de atualização
- Remoção de objetos

PRINCÍPIOS

Ideia básica: assuma que não existem conflitos de write–write (má ideia)

- atualizações são realizadas em um único servidor
- uma réplica passa o estado atualizado para alguns vizinhos
- propagação da atualização é *lazy*, i.e., não é imediata
- eventualmente, todo update deveria alcançar todas as réplicas

Duas formas de epidemias:

Anti-entropy: cada réplica regularmente escolhe outra réplica ao acaso e uniformiza seus estados

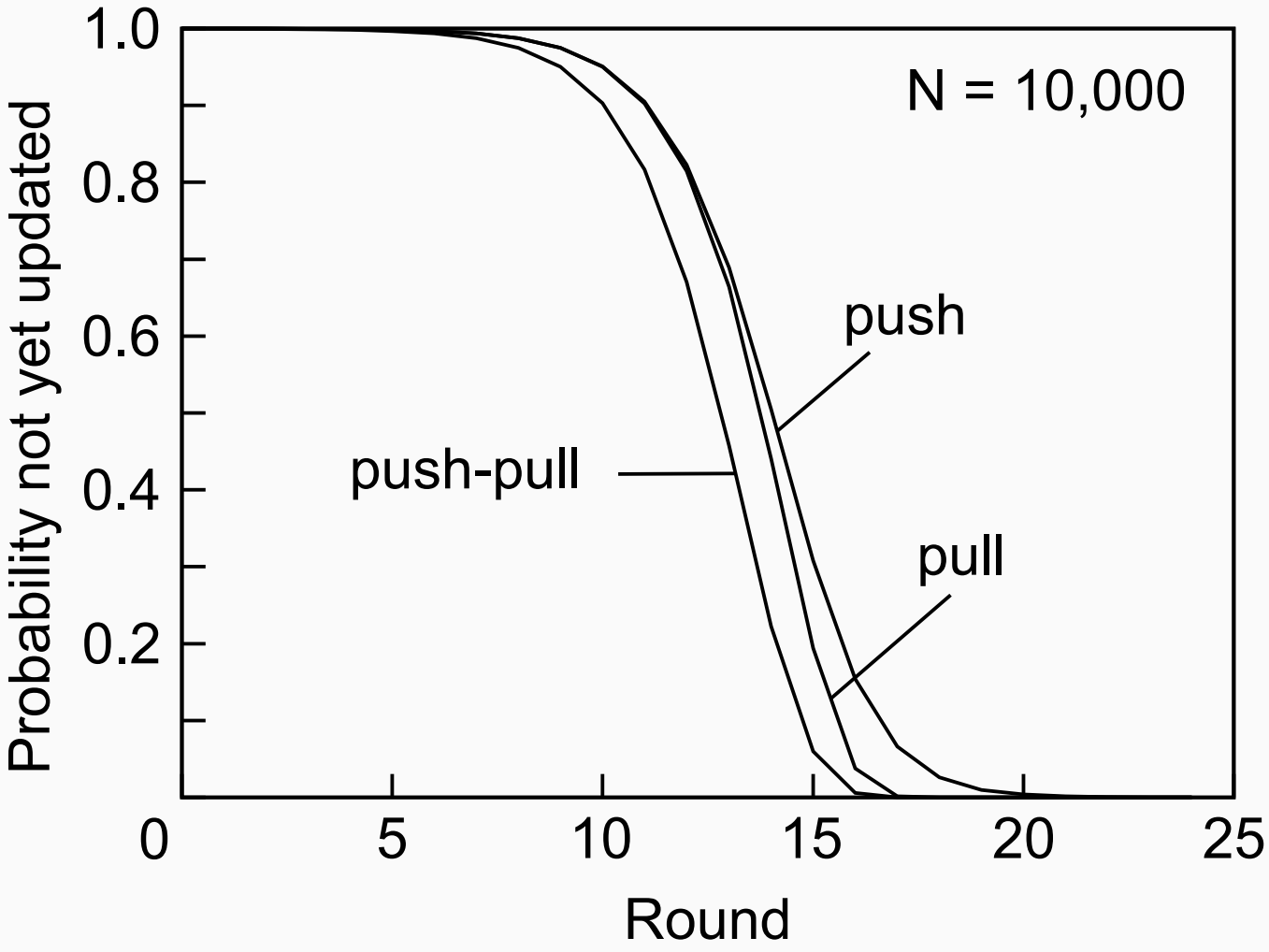
Gossiping: uma réplica que acaba de ser atualizada (*contaminada*) repassa a atualização a alguns vizinhos (contaminando elas)

ANTI-ENTROPY

- um nó P seleciona aleatoriamente outro nó Q do sistema
- Push: P só envia suas atualizações para Q
- Pull: P só recebe informações de Q
- Push–Pull: P e Q trocam atualizações entre si (e terminam com as mesmas informações)

Observação:
Cada push–pull leva $\mathcal{O}(log(N))$ rodadas de comunicação para disseminar as atualizações para todos os N nós

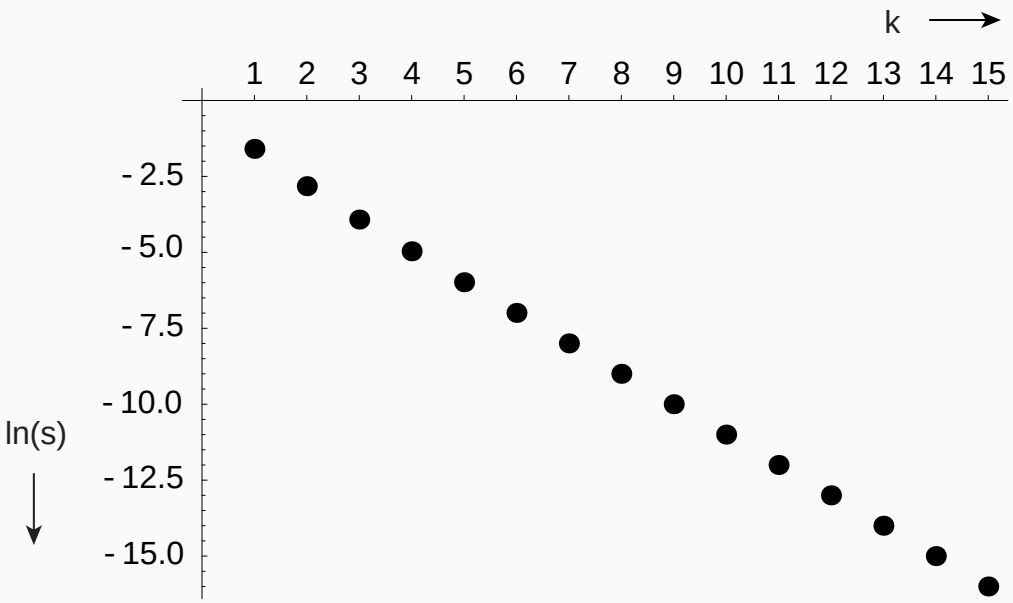
DESEMPENHO DE ANTI-ENTROPY



Um servidor S com uma atualização a reportar contacta outros servidores. Se o servidor contactado já compartilhou essa atualização, S para de conectar outros servidores com probabilidade $1/k$

Observação:
Se s for a fração de servidores que desconhecem a atualização, pode-se mostrar que, com muitos servidores,

$$s = e^{-(k+1)(1-s)}$$



Considere 10.000 nós		
k	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

Note que:
se você realmente quiser se assegurar de que todos os servidores serão atualizados eventualmente, gossiping sozinho não é suficiente

REMOÇÃO DE VALORES

Problema intrínseco:
Nós não podemos remover um valor antigo do servidor e esperar que a remoção se propague. Pior, uma remoção simples pode ser desfeita rapidamente se um protocolo epidêmico estiver sendo utilizado.

Solução
A remoção precisa ser registrada com um tipo especial de atualização: um atestado de óbito.

REMOÇÃO DE VALORES

Problema seguinte: como remover um atestado de óbito? (ele não pode ficar lá pra sempre)

- execute um algoritmo global para detectar se a remoção foi percebida por todos os nós e então remova os atestados
- assuma que os atestados não serão propagados para sempre e associe um tempo de vida máximo para o atestado

Observação:
É preciso que a remoção realmente alcance todos os servidores.

Problema de escalabilidade:
Quanto mais servidores, maior o tempo de propagação.

APLICAÇÕES DE EXEMPLO

- Disseminação de dados: (talvez a aplicação mais importante)
- Agregação: faça cada nó i manter uma variável x_i. Quando dois nós fizerem gossip, eles irão resetar suas variáveis para

$$x_i, x_j \leftarrow (x_i + x_j)/2$$

Resultado, no final cada nó terá calculado a média $\bar{x} = \sum_i x_i/N$.

O que acontece se inicialmente $x_i = 1$ e $x_j = 0, \forall j \neq i$?

APLICAÇÕES DE EXEMPLO

- Disseminação de dados: (talvez a aplicação mais importante)
- Agregação: faça cada nó i manter uma variável x_i. Quando dois nós fizerem gossip, eles irão resetar suas variáveis para

$$x_i, x_j \leftarrow (x_i + x_j)/2$$

  Resultado, no final cada nó terá calculado a média $\bar{x} = \sum_i x_i/N$.

O que acontece se inicialmente $x_i = 1$ e $x_j = 0, \forall j \neq i$?

No final todo nó terá uma estimativa do número de nós no sistema: $1/x_i$