Árvores Geradoras Mínimas Distribuídas e Autonômicas

Luiz A. Rodrigues[1,2] Elias P. Duarte Jr.[2] Luciana Arantes[3]

[1]Universidade Estadual do Oeste do Paraná - UNIOESTE

[2]Universidade Federal do Paraná - UFPR

[3]Université Pierre et Marie Currie - CNRS/INRIA/REGAL

SBRC 2014 – Florianópolis – SC
5-9 de maio de 2014

Roteiro

1. Árvores Geradoras
2. Trabalhos Relacionados
3. O Algoritmo Autonômico para Árvores Geradoras
4. Broadcasts com VCube
5. Avaliação Experimental
6. Conclusão

Árvores Geradoras

- **Árvores geradoras** (*Spanning trees*) possuem diversas aplicações
  - Exclusão mútua, agrupamento, fluxo em redes, sincronização, *broadcast*, etc.
- Alternativa eficiente à difusão por *flooding*
- **Falhas** em sistemas distribuídos são inevitáveis
- **Sistemas autonômicos**: serviço se adapta de forma transparente

Contribuição

1. Algoritmo de Árvore geradora
   - Distribuído
   - Autonômico
   - Baseado em hipercubo virtual
2. Algoritmo de Broadcast de melhor-esforço
3. Algoritmo de Broadcast confiável

Trabalhos Relacionados

- Algoritmos clássicos
  - Kruskal e Joseph (1956): vértices de menor peso são conectados até formar uma componente conexa
  - Prim (1957): cortes mínimos para escolher arestas de menor peso
- Soluções derivadas: Gallager et al. (1983) (Kruskal e Joseph), Dalal (1987) (Prim)
- Avresky (1999)
  - Solução em hipercubos
  - Filhos são reconectados em caso de falhas
  - Pode bloquear de acordo com as combinações de falhas
- HyParView: híbrido árvore+*gossip*
- Probabilísticos: Eugster et al. (2003)
- Pereira et al. (2004): hierárquico, se adapta de acordo com a capacidade dos nodos
- Flocchini et al. (2012): em caso de falhas, árvores alternativas pré-computadas

Modelo do Sistema

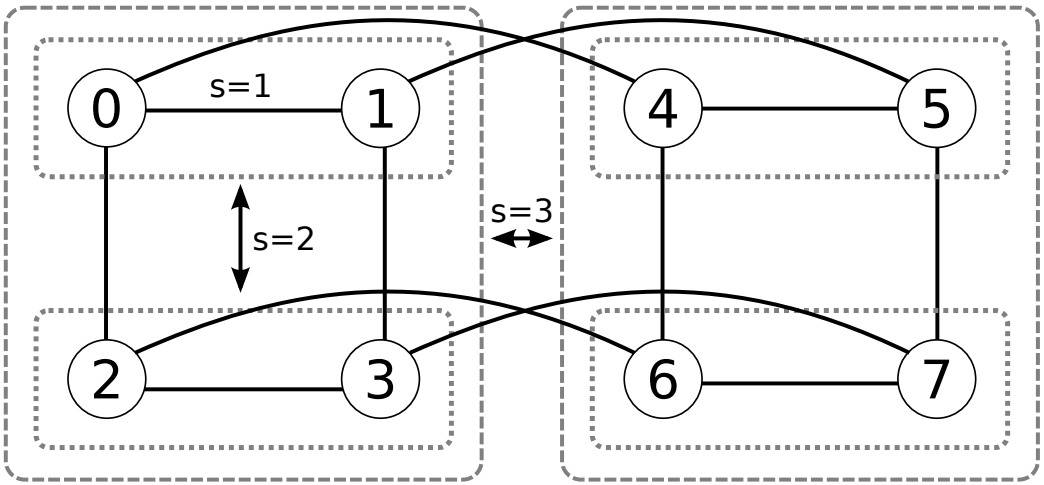
- **Sistema Distribuído**: conjunto finito Π de $n \geq 2$ processos independentes $\{p_0, ..., p_{n-1}\}$ que se comunicam por troca de mensagens
  - *nodo* = *processo*
- $G = (V, E)$: grafo conexo e não-direcionado que representa Π
- $T = (V, E')$: grafo conexo e acíclico, $E' \subseteq E$, $|E'| = |V| - 1$
- A rede é um grafo completo, mas processos são organizados baseando-se em um *hipercubo virtual* (VCube)
- Processos podem falhar por *crash* (permanente)
  - Falhas são detectadas por um serviço de detecção de falhas

VCube

- Processos são agrupados em clusters progressivamente maiores
  - Quando não há falhas, um **hipercubo** é formado
- Processos executam **testes** para realizar o diagnóstico do sistema
- Os elementos de cada cluster s do processo i são dados por:

$$c_{i,s} = \{i \oplus 2^{s-1}, c_{i \oplus 2^{s-1},1}, \ldots, c_{i \oplus 2^{s-1},s-1}\}$$

Figura: 3-VCube



O Algoritmo Autonômico para Árvores Geradoras (Algoritmo 1)

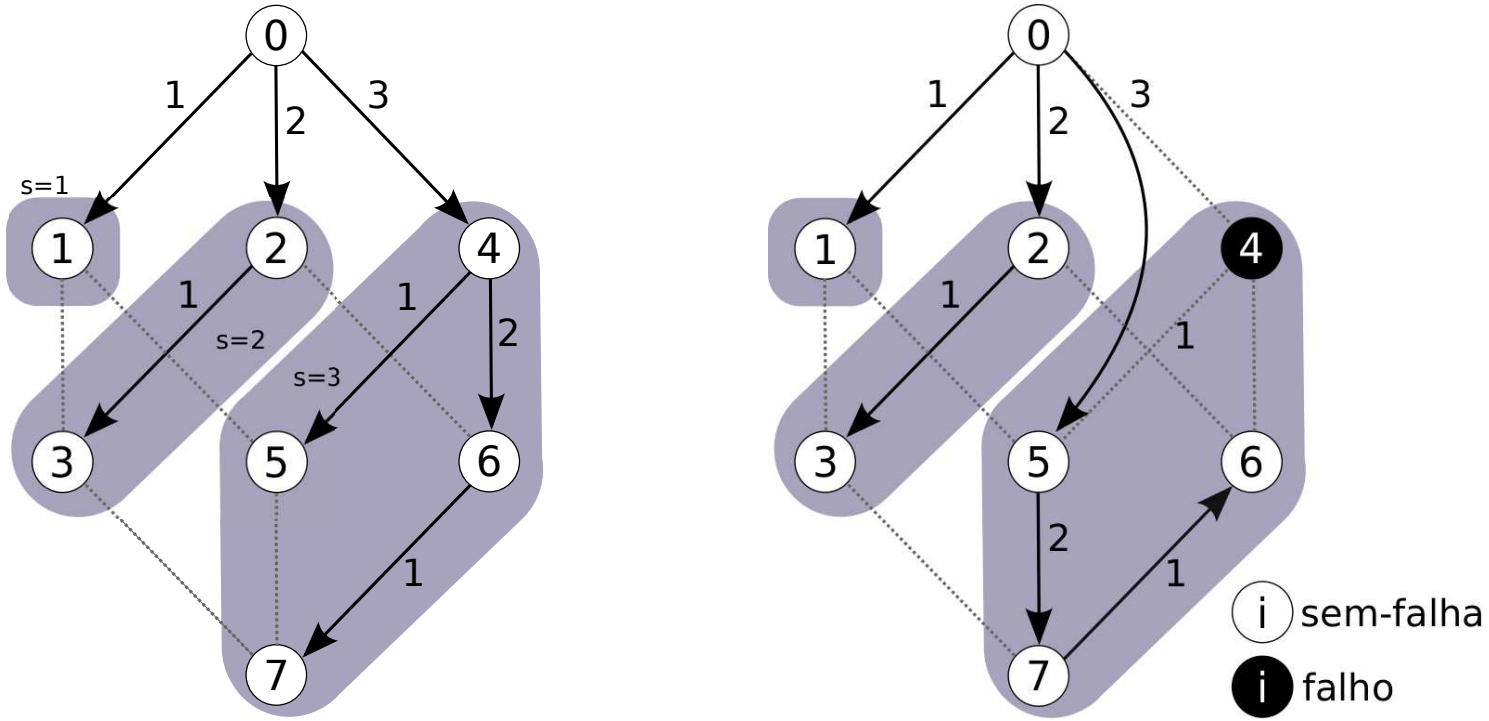
- $correct_i$
  - processos considerados corretos por i
- $cluster_i(j) = s$
  - processo j pertence ao cluster s do processo i
- $FF\_neighbor_i(s) = j$
  - o primeiro processo sem-falha do cluster s ou $\perp$
- $neighborhood_i(h) = \{j \mid j = FF\_neighbor_i(s), j \neq \perp, 1 \leq s \leq h\}$
  - vizinhos de i nos clusters de $1..h$

```
1:  procedure StartTree( )
2:                                   ▷ envia a todos os vizinhos
3:      for all k ∈ neighborhood_i(log_2 n) do
4:          Send(⟨TREE⟩) para p_k

5:  procedure Receive(⟨TREE⟩) de p_j
6:      if j ∈ correct_i then
7:                          ▷ retransmite aos vizinhos internos
8:          for k ∈ neighborhood_i(cluster_i(j) − 1) do
9:              send(⟨TREE⟩) para p_k

10: procedure Crash(processo j)          ▷ j está falho
11:     correct_i ← correct_i ∖ {j}
12:     if k = FF_neighbor_i(cluster_i(j)), k ≠ ⊥ then
13:         send(⟨TREE⟩) para p_k
```

Exemplos



(a) sem-falhas

(b) processo p_4 falho

Figura: Árvore geradora no VCube de 3 dimensões.

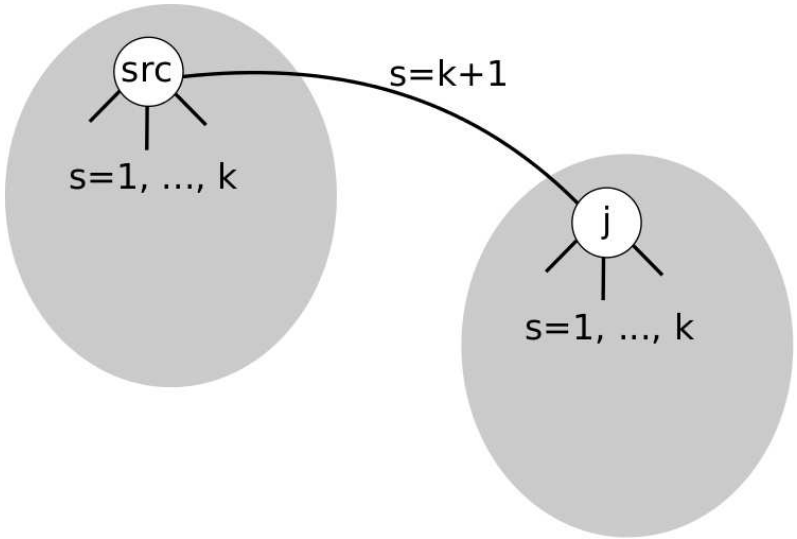
Especificação Formal

Teorema

Seja m uma mensagem propagada por um processo fonte src correto. Todo processo correto no sistema Π *recebe m.*

Prova por indução:

- Base: $n = 2$
- Hipótese: suponha que é valido para $n = 2^k$
- Passo: $n = 2^{k+1}$
  - O sistema consiste de dois subsistemas com $n = 2^k$



Broadcasts com VCube

- Duas soluções tolerantes a falhas:
  1. Broadcast de melhor-esforço: se o emissor é correto, todos os processos corretos recebem a mensagem
  2. Broadcast confiável: o mesmo conjunto de mensagens é entregue a todos os processos corretos, mesmo se o emissor falhar

Broadcast de Melhor-esforço

- Dois tipos de mensagem:
  1. $\langle TREE, m \rangle$
     - Contém o identificador de origem ($source(m) = source$) e o *timestamp* $ts(m) = ts$
  2. $\langle ACK, m \rangle$ usado para confirmar o recebimento de uma mensagem TREE
- Variáveis locais:
  - $last_i[n]$: última mensagem recebida de cada processo
  - ack_set_i: conjunto de ACKs pendentes. Ex.: $\langle j, k, m \rangle$
  - $correct_i$: conjunto dos processos considerados corretos por i

Algoritmo de Melhor-esforço (Algoritmo 2)

Emissor

1. Executa Broadcast de TREE para todos os vizinhos sem-falha
2. Espera pelos ACKs

Receptores

1. Recebe uma mensagem TREE do processo j
2. Entrega a mensagem à aplicação, se ela é nova (*timestamp*)
3. Propaga a mensagem na sub-ávore
4. Espera pelos ACKs
5. Envia o ACK para j

Após Crash(j)

1. Para cada ack pendente de j, envia a mensagem para o próximo processo sem-falha do mesmo cluster de j, se há algum

Expecificação Formal

Teorema

O Algoritmo 2 é uma solução para broadcast de melhor-esforço: se o emissor é correto, todo processo correto receberá o mesmo conjunto de mensagens.

Prova:

- não-duplicação e não-criação: propriedades dos enlaces confiáveis e uso de timestamp
- Entrega confiável: provado pelo teorema da árvore geradora

Broadcast Confiável (Algoritmo 3)

- Herda das propriedades do Melhor-esforço
  - Entrega confiável, não-duplicação e não-criação
- Inclui **acordo** (*agreement*): todo processo correto entrega o mesmo conjunto de mensagens

Algoritmo 3

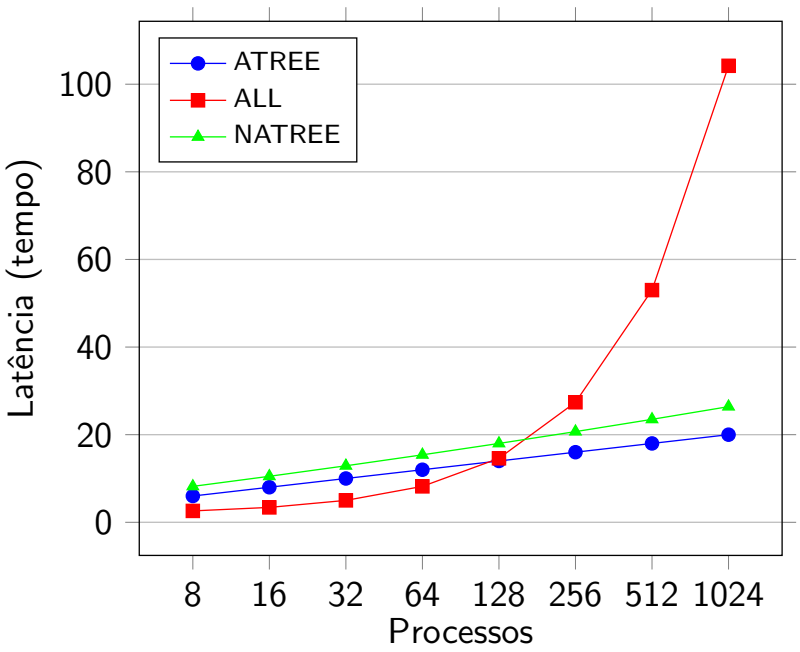
- Variação do algoritmo de melhor-esforço com tratamento da falha do emissor
  - Após detecção da falha de p, efetua broadcast da última mensagem recebida de p ($last_i[p]$)

Avaliação Experimental

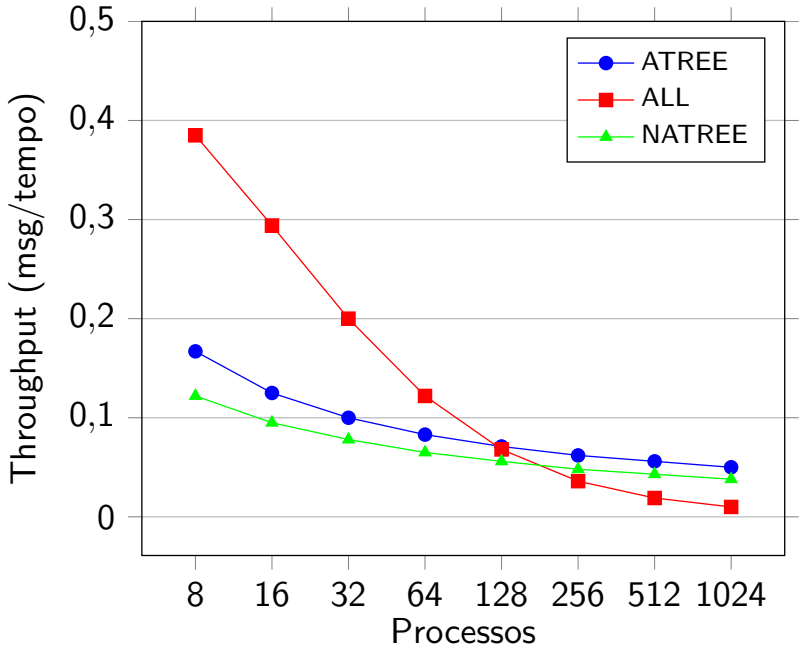
- Simulação com Neko
- Comparação com duas outras abordagens:
  1. ALL: todos-para-todos
  2. NATREE: árvore não-autonômica
- Parâmetros de simulação:
  - t_s: tempo de envio de uma mensagem
  - t_r: tempo de recebimento de uma mensagem
  - t_t: tempo de propagação de uma mensagem na rede
  - $t_s = t_r = 0,1$ e $t_t = 0,8$
  - *timeout*: $4 * (t_s + t_r + t_t)$

Resultado dos Experimentos (I)

- Cenário sem-falhas
  - Broadcast de melhor-esforço = broadcast confiável



(a) Latência

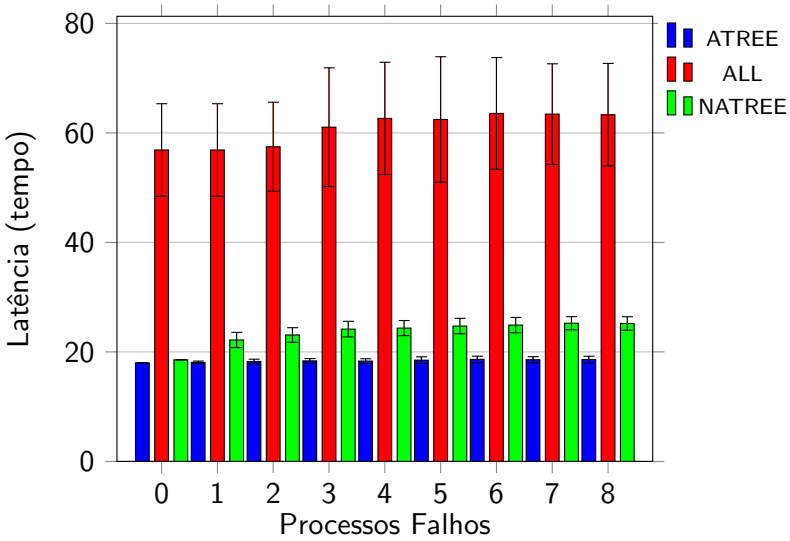


(b) Throughput

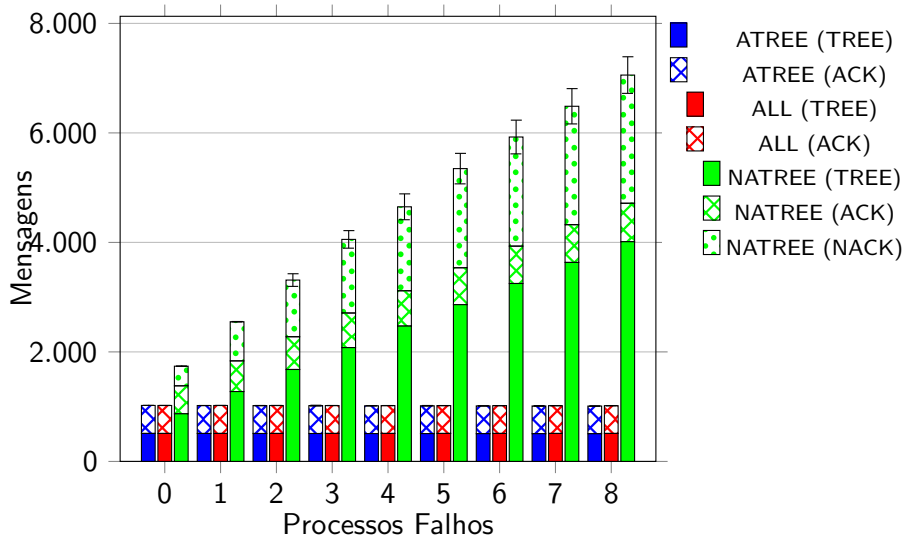
Figura: Broadcast de melhor-esforço em uma execução sem-falhas

Resultados dos Experimentos (II)

- Cenário com falhas



(a) Latência



(b) Total de mensagens

Figura: Broadcast confiável com $n = 512$ e diferente número de falhas

Conclusão

- Solução para criação **autonômica** de árvores geradoras mínimas em sistemas distribuídos sujeitos a falhas *crash*
- Emprego do VCube
- Duas aplicações propostas:
  1. Broadcast de melhor-esforço
  2. Broadcast confiável
- Resultados da simulação:
  - Eficiência
  - Escalabilidade
- Trabalhos Futuros:
  1. Tratar falhas *crash* com recuperação
  2. Estudar modelo com particionamento
  3. Implementar o algoritmo de árvore como um serviço

Perguntas?

Árvores Geradoras Mínimas Distribuídas e Autonômicas

Luiz A. Rodrigues
luiz.rodrigues@unioeste.br

Obrigado!