Redes Auto-Reconfiguráveis Randomizadas para Comunicação Distribuída em Datacenters

Otávio A. O. Souza[1], **Caio Caldeira**[1], **Olga Goussevskaia**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)
31270-901 - Belo Horizonte - MG - Brasil

{oaugusto,caio.caldeira,olga}@dcc.ufmg.br

Abstract. *In this paper we investigate the benefits of randomization in the design of self-adjusting network topologies: networks that dynamically adapt themselves toward the demand they currently serve, in an online manner. We present a randomized self-adjusting tree network, which leverages randomization to reduce the expected network reconfiguration cost by a constant factor, compared to existing deterministic solutions. The new solution is simple, easy-to-implement, fully distributed and concurrent. We prove algorithm correctness, provide worst-case amortized cost limits, and present simulation results on workloads with variable spatial and temporal complexity.*

Resumo. *Neste artigo nós estudamos os benefícios de randomização no projeto de topologias de redes auto-reconfiguráveis, ou seja, redes que se adaptam dinamicamente à demanda que atendem em tempo real, de forma online. Apresentamos uma rede randomizada auto-reconfigurável com topologia em árvore, que utiliza randomização para reduzir o custo esperado de reconfiguração por um fator constante, comparado a soluções determinísticas existentes. A nova solução é simples, de fácil implementação, totalmente distribuída e concorrente. Nós provamos que o algoritmo proposto é correto, provamos limites de custo amortizado no pior caso e apresentamos resultados de simulação em sequências de requisições com localidade de referência espacial e temporal variáveis.*

1. Introdução

Estudos empíricos mostram que os padrões de comunicação em datacenters apresentam estrutura espacial e temporal [Avin et al. 2020], ou seja, o tráfego é intermitente e as matrizes de tráfego desequilibradas. Esta estrutura representa um potencial inexplorado para construir redes de comunicação mais eficientes: as redes de datacenters atuais são projetadas em uma maneira *alheia* ao padrão de comunicação; em vez disso, essas redes visam fornecer um bom desempenho para padrões de tráfego arbitrários (ou seja, tráfego de todos para todos), por exemplo, uma alta largura de banda da bisseção. No entanto, tecnologias ópticas emergentes, como switches óticos 3D MEMS [Chen et al. 2017, Zerwas et al. 2021, Farrington et al. 2010, Chen et al. 2014], podem anunciar uma mudança de paradigma: essas tecnologias permitem ajustes topológicos rápidos, introduzindo uma visão de redes auto-reconfiguráveis[1]: topologias dinâmicas que se adaptam à demanda que atendem em tempo real.

[1]Utilizamos os termos "auto-reconfigurável", "auto-adaptativo"e "auto-ajustável"como sinônimos.

Um desafio fundamental no projeto de redes auto-reconfiguráveis é encontrar um equilíbrio entre os benefícios e os custos de reconfigurações. Enquanto as reconfigurações permitem reduzir os custos de comunicação, tornando nós (por exemplo, switches ou topos de rack (*Top-Of-Rack*, ToR) em um *datacenter*) que se comunicam com maior frequência topologicamente mais próximos, tais reconfigurações devem ser usadas com moderação. Reconfigurações demandam tempo e podem levar temporariamente à perda de pacotes. Em particular, os padrões de comunicação podem não ser perfeitamente previsíveis e, portanto, as decisões de reconfiguração precisam ser feitas de forma *online*.

Este trabalho investiga o problema algorítmico subjacente a tais redes auto-reconfiguráveis, abordados no estado da arte na literatura em [Avin and Schmid 2018], e, em particular, no *SplayNet*, [Schmid et al. 2016] , baseado em árvores binárias de busca auto-reconfiguráveis, (*splay trees*) [Sleator and Tarjan 1985]. A comunicação em redes, entretanto, se difere dessas principalmente no ponto de que qualquer nó da rede pode se comunicar com qualquer outro, ao contrário das requisições originarem-se da raiz, como é o caso nas árvores.

Neste trabalho, apresentamos uma versão randomizada de redes auto-reconfiguráveis, a qual nos referimos como *RanDisplayNet*. O novo algoritmo para ajustar a topologia da rede é simples e de fácil implementação. Demonstramos formalmente que esse esquema randomizado tem o mesmo desempenho assintótico que os esquemas determinísticos [Schmid et al. 2016, Peres et al. 2021], mas melhora as constantes no custo esperado de reconfiguração, além de não necessitar do custo adicional para manter consistência do estado global da rede.

Além da análise amortizada teórica, nós realizamos um detalhado estudo experimental, no qual mostramos como a localidade de referência temporal e espacial inerente à sequência de requisições de comunicação impacta os benefícios de redes auto-reconfiguráveis randomizadas. Em particular, em cenários com alta localidade temporal, apesar da abordagem randomizada reduzir o número de reconfigurações, o custo total de comunicação, que reflete a soma das distâncias percorridas pelas mensagens, é maior do que de abordagens determinísticas. Os maiores ganhos são obtidos em cenários com baixa localidade temporal e alta localidade espacial. Nesses cenários, o desempenho da abordagem randomizada, em termos de número total de reconfigurações, se aproxima do desempenho de CBNet, cuja implementação é mais complexa, e o seu custo total de comunicação não aumenta significativamente em comparado a abordagens determinísticas.

O restante deste artigo é organizado da seguinte forma. Na Seção 2, discutimos os trabalhos relacionados; na Seção 3, apresentamos o modelo de custos para redes auto-reconfiguráveis randomizadas; na Seção 4, o algoritmo proposto e sua implementação; na Seção 5, a análise amortizada para os cenários sequencial e concorrente; e nas Seções 6, 7 e 8, os resultados experimentais e as conclusões finais.

2. Trabalhos Relacionados

Tradicionalmente, os projetos de rede em *datacenters* dependem de topologias estáticas, como a topologia *Clos* [Al-Fares et al. 2008], topologias hipercúbicas como *BCube* [Guo et al. 2009], ou redes baseadas em expansores [Kassing et al. 2017]. Recentemente, topologias reconfiguráveis têm recebido bastante atenção, que vêm em

dois tipos, alheios à demanda, [Ballani et al. 2020] ou ciente da demanda, por exemplo, [Souza et al. 2021, Ghobadi et al. 2016, Chen et al. 2014, Schmid et al. 2016, Peres et al. 2021, Farrington et al. 2010].

A maioria das arquiteturas sensíveis à demanda existentes dependem de uma estimativa de matrizes de tráfego [Farrington et al. 2010], o que pode limitar a granularidade e a reatividade da rede. Neste artigo, consideramos abordagens mais distribuídas, como ProjectToR [Ghobadi et al. 2016] ou DiSplayNet [Peres et al. 2019], com suporte a reconfiguração por fluxo ou mesmo por pacote.

Os artigos mais intimamente relacionados ao nosso trabalho são [Schmid et al. 2016, Peres et al. 2019, Souza et al. 2021, Albers and Karpinski 2002], que também se baseiam em conceitos de estruturas de dados auto-reconfiguráveis, mais precisamente em árvores binárias de busca auto-reconfiguráveis, também chamadas de *splay trees* [Sleator and Tarjan 1985].

DiSplayNet [Peres et al. 2021, Peres et al. 2019] foi a primeira proposta de rede auto-reconfigurável totalmente distribuída com garantias formais de desempenho. DiSplayNet utiliza-se de diretivas de concorrência para permitir que requisições independentes possam ser atendidas simultaneamente. No entanto, esta e as soluções anteriores sofrem de uma sobrecarga significativa de custos de reconfiguração, devido à sua estratégia "agressiva" que, para cada requisição, dispara uma sequência de reconfigurações que reduzem a distância entre o nó de origem e o nó de destino a um só enlace.

Em [Souza et al. 2021], foi proposta a CBNet, uma rede auto-reconfigurável inspirada em uma estrutura de dados diferente e concorrente, a *CBTree* [Afek et al. 2012]. Diferentemente de soluções anteriores, nas quais os nós de origem e destino de cada solicitação são movidos para seu menor ancestral comum (*Least Common Ancestor*, LCA) por meio de uma sequência de rotações, CBNet realiza reconfigurações de topologia com frequência baixa (uma subconstante amortizada $O(1)$ por operação), trocando rotações por manutenção de um histórico global de comunicação. Cada nó mantém uma contagem do número de requisições a ele, e CBNet utiliza os contadores para decidir quando realizar uma rotação. CBNet não apenas evita o alto custo pago pela reconfiguração frequente da rede, mas também escala melhor com o nível de concorrência. A desvantagem dessa abordagem consiste na relativa complexidade e custo de comunicação adicional para se manter a consistência global entre os contadores, já que todo incremento de contador deve ser propagado até a raiz da árvore.

Nosso trabalho pode ser visto como uma generalização de *splay trees* randomizadas [Albers and Karpinski 2002] para o contexto de redes de comunicação e sistemas distribuídos. No entanto, enquanto SplayNet [Schmid et al. 2016] ainda é centralizado, DiSplayNet [Peres et al. 2019] vem com custos de ajuste significativos, e CBNet [Souza et al. 2021] vem com custo extra para atualizar os contadores, ou seja, manter a consistência de um estado global da rede, a abordagem que apresentamos neste trabalho é distribuída, de fácil implementação, e tem custos de comunicação e reconfiguração competitivos em relação a abordagens determinísticas.

3. Modelagem

Nossa rede é composta por um conjunto V de n nós de comunicação (por exemplo, *switches top-of-rack* ou *peers*). A entrada para o problema é dada por uma sequência σ de m

mensagens $\sigma_i(s,d) \in V \times V$ ocorrendo ao longo do tempo, com origem s e destino d; m pode ser infinito. Denotamos por b_i o instante de tempo em que uma mensagem σ_i é gerada, e por e_i o instante de tempo em que ela é entregue. O tempo entre as chegadas sucessivas de solicitações é assumido ser pelo menos um.

A sequência σ é revelada ao longo do tempo, de forma online: o algoritmo não possui nenhuma informação sobre as requisições futuras σ_j no tempo $t < b_j$. Além disso, a sequência σ pode ser arbitrária: em nossa análise formal consideramos um cenário de pior caso, onde σ é escolhido *adversarialmente*, para maximizar o custo de um dado algoritmo.

Para minimizar o custo de comunicação e ajustar a topologia suavemente ao longo do tempo, a árvore é reconfigurada localmente por meio de rotações que preservam as propriedades de uma BST, uma vez que são uma família básica de grafos e temos que as conexões auto-reconfiguráveis constituem apenas um subconjunto da topologia, uma suposição usual em tais redes [Ghobadi et al. 2016]. Uma rotação atualiza um número constante de enlaces a custo constante. Assim, denotaremos a árvore no tempo t calculada por um dado algoritmo distribuído (possivelmente contabilizando as solicitações de comunicação $\sigma_{t'}$ com $t' < t$) por $T_t \in \mathcal{T}$. A partir de agora, usamos os termos *rotação* e *reconfiguração* alternadamente para nos referirmos a atualizações topológicas locais na árvore.

Modelo de custo refinado. Em nossa rede auto-reconfigurável, para cada mensagem rotações ou roteamentos são realizados por meio de um fator probabilístico. Portanto, distinguimos entre o trabalho necessário para encaminhar uma mensagem até o destino por meio de reconfigurações e o trabalho para entregar a mensagem por meio de roteamento. Na prática, o custo de realizar uma reconfiguração de topologia de rede é tipicamente maior do que o de encaminhar uma mensagem por um conexão de comunicação. Trabalhos em estruturas de dados auto-reconfiguráveis já destacaram este problema [Sleator and Tarjan 1985, Afek et al. 2012]. Em particular, este problema também foi abordado em [Albers and Karpinski 2002], onde uma versão aleatória de *SplayTree* é apresentada para reduzir a sobrecarga de custos de comunicação causada pela reconfiguração da rede. Neste trabalho, pretendemos aproximar as redes auto-reconfiguráveis da prática, reduzindo bastante o custo total de reconfiguração da rede, preservando caminhos curtos de origem-destino na sequência de mensagens.

Assumimos que o roteamento de uma mensagem resulta em um custo de 1 unidade por salto e que uma rotação resulta em um custo de $R = O(1)$.

Considere uma sequência σ de m mensagens, um algoritmo $\mathcal{A}$, um *BST* T_0 e uma mensagem $\sigma_i(s,d) \in \sigma$. Vamos definir o custo de reconfiguração ρ_i como o número de rotações realizadas por $\mathcal{A}$ para entregar σ_i. O custo de roteamento será igual a $d_{e_i}(s,d)$, o comprimento do caminho $\mathcal{P}_{e_i}(s,d)$ na árvore resultante T_{e_i} , ou seja, após a entrega de σ_i. Observe que $d_{e_i}(s,d)$ não é necessariamente um, como em [Schmid et al. 2016, Peres et al. 2019], mas é igual ao número de vezes que a mensagem foi encaminhada ao longo de $\mathcal{P}_{b_i}(s,d)$, em vez de acionar uma rotação. Assumimos que o valor do custo de roteamento para entregar uma mensagem, mesmo quando endereçada a si mesma ($\sigma_i(v,v)$), é pelo menos um.

Definição 1. ***Custo do trabalho:*** *Considere qualquer árvore binária inicial T_0, uma sequência de m mensagens $\sigma_i(s,d) \in \sigma$ e algoritmo $\mathcal{A}$. Definimos o* ***custo total de rote-***

***amento, custo total de reconfiguração e custo total de trabalho**, respectivamente, como segue:* $\mathcal{D}(\mathcal{A}, T_0, \sigma) = \sum_{i=1}^{m} (d_{e_i}(s,d)(\sigma_i) + 1)$, $\mathcal{R}(\mathcal{A}, T_0, \sigma) = R \times \sum_{i=1}^{m} \rho_i$, $\mathcal{C}(\mathcal{A}, T_0, \sigma) = \mathcal{D}(\mathcal{A}, T_0, \sigma) + \mathcal{R}(\mathcal{A}, T_0, \sigma)$.

Assumimos que o tempo é dividido em intervalos síncronos, nos quais uma mensagem pode viajar um número constante de saltos na rede ou uma reconfiguração local pode ser executada. Para estudar a simultaneidade, dividimos o tempo de execução em *rounds*: em uma rodada, vários nós (independentes) podem fazer reconfigurações locais simultaneamente. Consideramos que os nós e a comunicação entre eles são confiáveis e síncronos. Em termos de tempo, pretendemos minimizar o *makespan*:

Definição 2. ***Custo de tempo:*** *Considere uma árvore inicial T_0, uma sequência de m mensagens σ e um algoritmo $\mathcal{A}$.* $Makespan(\mathcal{A}, T_0, \sigma) = \max\limits_{1 \le i \le m} e_i - \min\limits_{1 \le i \le m} b_i$.

Nosso objetivo é *minimizar* o custo de comunicação tanto em termos de trabalho quanto de tempo. Estamos interessados no desempenho do pior caso em sequências arbitrárias de operações (em vez de operações individuais) e, portanto, conduzimos uma *análise amortizada*. Em nosso modelo, o custo amortizado pode ser descrito como o custo médio por mensagem para uma dada sequência σ.

Definição 3. ***Custo amortizado:*** *Dada uma sequência de m mensagens σ, se $\mathcal{C}(\sigma_i)$ é o custo (tempo ou trabalho) de $\sigma_i \in \sigma$, o custo amortizado é definido em relação à pior sequência σ e árvore inicial T_0 como:* $\mathcal{C}_A = \max\limits_{\sigma, T_0} \frac{1}{m} \sum\limits_{\sigma_i \in \sigma} \mathcal{C}(\sigma_i)$.

Outro conceito útil para a análise é a entropia empírica.

Definição 4. ***Entropia Empírica:*** *Dada uma sequência de m mensagens σ e árvore inicial T_0 em n nós, a entropia empírica é definida em relação a $\hat{S}$ como:* $H(\hat{S}) = \sum\limits_{i=1}^{n} f_s(v_i) \log \frac{1}{f_s(v_i)}$, *onde* $\hat{S} = \{f_s(v_1), \ldots, f_s(v_n)\}$ *são as frequências que um nó $v_i \in V$ é uma fonte em σ. Da mesma forma, $H(\hat{D})$ é definido para o conjunto de frequências de destino $\hat{D}$.*

4. Algoritmo

Neste trabalho visamos aproximar as redes auto-reconfiguráveis das aplicações reais visto que o alto número de reconfigurações têm se mostrado ser um problema na prática, dado que o custo das reconfigurações é mais elevado que o de roteamento. Com isso, nos inspiramos nos trabalhos de [Albers and Karpinski 2002] que utiliza de randomização para alcançar tais resultados. Para reduzir ainda mais os custos de reconfigurações e facilitar na implementação, nós utilizamos uma variação do algoritmo descrito em [Peres et al. 2019], adaptando suas reconfigurações com base em *semi-splaying*. A seguir, fornecemos uma breve descrição do algoritmo determinístico, que denominamos Semi-DiSplayNet Determinístico, e em seguida apresentamos sua versão randomizada.

4.1. Semi-DiSplayNet Determinístico

Em alto nível, Semi-DiSplayNet funciona da seguinte forma: quando uma mensagem é enviada de um nó s para um dado nó d, o algoritmo executa uma série de reconfigurações locais da topologia da rede, enquanto a mensagem vai sendo transmitida por nós intermediários. Semi-DiSplayNet gradativamente modifica a topologia de forma a tornar próximos nós que se comunicam com maior frequência.

Diferentemente de DiSplayNet, em que os nós carregam mensagens para transmitir somente quando estão a um enlace (*hop*) de distância, em Semi-DiSplayNet, as mensagens podem passar por nós intermediários no caminho através de reconfigurações. Quando um nó possui uma mensagem a ser transmitida, um tipo de operação é escolhido baseado nas posições relativas dos nós vizinhos na direção de transmissão (direção do destino). Após realizar uma operação, a mensagem pode continuar no mesmo ou pertencer a outro nó. Contudo, a cada operação uma mensagem sempre realiza progresso(reduz a distância) em direção ao nó destino.

Para realizar as reconfiguração de forma distribuída, Semi-DiSplayNet depende dos seguintes conceitos:

1. *Reconfigurações locais:* Para modificar a topologia da rede, Semi-DiSplayNet utiliza-se de reconfigurações locais de **semi-*splaying***, essas operações alteram de conexões em quantidade constante nós.
2. *Formação de clusters:* Para certificar a corretude das transformações, bem como para evitar *deadlocks*, cada operação de semi-*splaying* requer a formação de um *cluster*(conjunto de nós que terão conexões modificadas). Em cada *cluster* somente uma operação de semi-*splaying* pode ocorrer por *round*.
3. *Prioridade de execução:* Para evitar inanição, são utilizadas propriedades definidas pelo tempo de envio das mensagens na rede. Em caso de empates, o identificador do nó é utilizado como forma de desempate.

4.2. Reconfigurações Locais

Em particular, Semi-DiSplayNet emprega dois tipos de operações de reconfiguração enquanto a mensagem percorre o caminho entre o nó fonte e o destino. Uma operação de *bottom-up*, é utilizada quando a mensagem se move para cima na árvore essa operação termina quando o destino ou o menor ancestral comum (LCA) for alcançado. Caso o nó fonte já corresponde ao nó mais alto, não é realizada a operação de *bottom-up*. Caso a mensagem não tenha encontrado o destino, operações de *top-down*, são empregadas para mover a mensagem para baixo na árvore até que atinja o destinatário.

4.3. *Clusters*

Como as reconfigurações realizadas por um nó portando uma mensagem precisa de acesso exclusivo aos nós vizinhos para alterar as conexões, é utilizado o conceito de *clusters* definido em [Peres et al. 2019] para garantir a consistência da rede. Um *cluster* é formado por todos os nós que terão conexões alteradas durante uma operação e que concordam sobre qual nó tem preferência de operação. Somente um dos nós pertencente ao *cluster* pode realizar operação por *round*. Quando um nó está portando mensagem, uma operação de formação de *cluster* é inicializada antes de executar a operação em questão. Após a formação do *cluster* as conexões podem ser alteradas simultaneamente.

4.4. Randomização

Com esses conceitos em mente podemos agora apresentar o algoritmo não-determinístico. As mensagens realizam progresso em direção ao destino através de operações de reconfigurações locais na rede ou por meio de roteamentos.

A cada mensagem é atribuído um valor aleatório $\hat{p}$, entre 0 e 1, ao ser inicializada e, durante o processo de decisão do tratamento da mensagem, esse valor $\hat{p}$ é comparado

com um valor p pré-determinado pela rede, também entre 0 e 1, se $\hat{p}$ for maior que p, o *splay* entre s e d ocorre normalmente, caso contrário, a mensagem é roteada pelo do caminho da mensagem.

Uma vez decidida qual operação, roteamento ou rotação, o nó realiza a formação de *cluster*, como desenvolvido em [Peres et al. 2019]. Caso ocorra conflito na formação do *cluster*, o tempo de entrada na rede da mensagem é utilizado como prioridade. Com o *cluster* formado, a operação pode ser executada na rede. Após a operação ser aplicada, a mensagem reduz a distância ao nó destino, que dependendo do tipo de operação pode permanecer ou migrar para outro nó intermediário.

A invariante de *Semi-DiSplayNet* descrito pelo algoritmo 1 reside nas mensagens: enquanto uma mensagem não alcança o nó destino, operações de reestruturação ou roteamento são empregadas pelo nós que transmitem a mensagem pelo caminho.

Algorithm 1 RanDiSplayNet(p, σ_t, x_t)

```
proximoNo = próximo nó no caminho entre x_t e d
if p̂_t ≤ p then
    if proximoNo == x_t.pai then
        reconfiguração bottom-up
    else
        reconfiguração top-down
    end if
else
    roteamento de σ_t
end if
```

5. Análise Amortizada

Para calcular o custo do pior caso em sequências arbitrárias, realizamos uma análise amortizada do desempenho do algoritmo. Introduzimos uma função potencial para amortizar os custos reais. Considere uma instancia da árvore T. Seja $w(u), u \in T$ como o peso de um nó pertencente à árvore, denominamos tamanho de um nó u como $W(u) = \sum_{v \in T_u} w(v), u \in T$, sendo T_u a subárvore enraizada em u, incluindo u. Definimos o *rank* do nó u como $r(u) = \log W(u))$. Observe que o tamanho e o *rank* máximos de um nó são n e $\log n$, respectivamente.

Considerando a função potencial definida como $\Phi(T) = (1/p + \mathcal{R}) \times \sum_{u \in T} r(u)$. No método potencial, o custo amortizado $\hat{c}$ de uma operação é o custo real c, mais o aumento do potencial $\Delta\Phi$ devido à operação, onde $\Delta\Phi = \Phi(T') - \Phi(T)$. Isso nos dá: $\hat{c} = c + \Phi(T') - \Phi(T)$.

Para completar as definições, ao analisarmos o custo esperado amortizado, consideramos p como a probabilidade de realizar um *step* e nosso custo esperado amortizado, dado pela multiplicação do custo amortizado por p: $E[\hat{c}] = p \times \hat{c} = E[c] + E[\Delta\Phi]$.

5.1. Trabalho e tempo em cenário sequencial

A seguir, vamos provar o equivalente ao *Access Lemma* do artigo [Sleator and Tarjan 1985], com o objetivo de demonstrar que o custo amortizado esperado de acesso de um nó na árvore é logarítmico com as rotações.

Lema 1. *Suponha que existe uma mensagem entre* s *e* d $s, d \in V$. *Então o tempo amortizado esperado pelo Rand-SplayNet(p) para que a mensagem chegue a seu destino é no máximo* $2(1 + p\mathcal{R})(r(r) - r(s)) + 2(1 + p\mathcal{R})(r(r) - r(d)) = O\left((1 + p\mathcal{R})(\log \frac{W(r)}{W(s)} + \log \frac{W(r)}{W(d)})\right)$.

Demonstração. O RanDisplayNet utiliza de 5 tipos diferentes de rotação para realizar o *splay* dos nós acessados até a raiz, o *Zig Bottom-Up*, o semi Zig-Zig *Bottom-Up* e *Top-Down* e o semi Zig-Zag *Bottom-Up* e *Top-Down*. Suponha uma rotação Zig *Bottom-Up*, através do lema 2 em [Albers and Karpinski 2002], sabemos que, sendo x o nó com a mensagem, o tempo esperado de acesso é:

$$E[\hat{c}] = (1 + p\mathcal{R}) + (1 + p\mathcal{R}) \times (r'(x) - r(x)) \leq (1 + p\mathcal{R}) \times 2(r'(x') - r(x))$$

Pelo Lema 4.1 em [Afek et al. 2012], após um *semi-splay bottom-up step*, sendo x, r, x', r' o nó com a mensagem e a função de *rank* antes e depois do *step* respectivamente, e z o avô de de x antes do *step*, dado que $r'(x') = r(z)$, segue que a mudança de potencial causada é:

$$\Delta\Phi \leq (1/p + \mathcal{R})(2(r(z) - r(x)) - 2) \leq (1/p + \mathcal{R})2(r'(x') - r(x)) - 2(1/p + \mathcal{R})$$

Temos então que, para *bottom-up steps*, o valor esperado, $E[\hat{c}]$, para que o algoritmo realize o *step* é:

$$E[\hat{c}] \leq 2(1 + p\mathcal{R}) + (1 + p\mathcal{R}) \times 2(r'(x') - r(x)) - 2(1 + p\mathcal{R}) \leq 2(r'(x') - r(x))$$

Pelo Lema 4.5 em [Afek et al. 2012], após um *semi-splay top-down step*, sendo x, r, x', r' o nó atual e a função de *rank* antes e depois do *step*, respectivamente, segue que a mudança de potencial causada é:

$$\Delta\Phi \leq (1/p + \mathcal{R})(2(r(x) - r'(x')) - 2) \leq (1/p + \mathcal{R})2(r(x) - r'(x')) - 2(1/p + \mathcal{R})$$

Temos então que, para *top-down steps*, o valor esperado, $E[\hat{c}]$, para que o algoritmo realize o *step* é:

$$E[\hat{c}] \leq 2(1 + p\mathcal{R}) + (1 + p\mathcal{R})2(r(x) - r'(x')) - 2(1 + p\mathcal{R}) \leq 2(r(x) - r'(x'))$$

Identificamos que durante o *splay* entre s e d, o custo amortizado dos *steps bottom-up* é limitado pela diferença entre o *rank* do nó x' com a mensagem após o *step* e o *rank* do nó x com a mensagem antes do step w converge para uma série telescópica dada pela soma, seja $LCA(s, d)$ o menor ancestral comum entre os vértices s e d, $2(r(LCA(s, d)) - r(s))$. Já o custo amortizado dos *steps top-down*, é limitado pela diferença entre o *rank* do nó x com a mensagem antes do step e o *rank* do nó x' com a mensagem após o *step* e converge para uma série telescópica dada pela soma, $2(r(LCA(s, d)) - r(d))$. Como o *rank* da raiz da árvore é maior ou igual ao *rank* de qualquer nó da árvore temos que:

$$E[\hat{c}] \leq 2(r(LCA(s, d)) - r(s)) + 2(r(LCA(s, d)) - r(d)) \leq 2(r(r) - r(s)) + 2(r(r) - r(d))$$

□

O teorema a seguir estabelece uma relação entre a entropia empírica do padrão de requisições σ e a topologia da rede. Designando o peso de cada nó como sua frequência de comunicação, o peso total da nossa árvore é igual a 1. Com isso, podemos realizar uma análise que alcance um limite superior simples para o custo esperado amortizado de todas as mensagens em σ.

Teorema 1. *Sejam $H(\hat{S})$ e $H(\hat{D})$ as entropias empíricas de origem e destino de σ, conforme definido em (4). O* ***custo amortizado esperado total de comunicação*** *incorrido pelo Algoritmo 1 para entregar todas as mensagens em σ é $O\left((1+p\mathcal{R})(H(\hat{S})+H(\hat{D}))\right)$.*

Demonstração. Seja $s(\cdot)$ o número total de vezes i aparece como uma fonte em sigma, e seja $d(\cdot)$ o número total de vezes i aparece como destino. Considerando o lema 1 temos que o custo de RanDiSplayNet (RSN) é dado por:

$$\begin{aligned}
\mathcal{C}(RSN, T_0, \sigma) &\leq \frac{1}{m}\sum_{t=1}^{m}(2(1+p\mathcal{R}) \times (r(LCA(\sigma_t)) - r(src(\sigma_t))) \\
&\quad +2(1+p\mathcal{R}) \times (r(LCA(\sigma_t)) - r(dst(\sigma_t))) \\
&= O\left((1+p\mathcal{R}) \times \frac{1}{m}\left(\sum_{i=1}^{n} s(i)\log\frac{m}{s(i)} + \sum_{j=1}^{n} d(i)\log\frac{m}{d(i)}\right)\right) \\
&= O\left((1+p\mathcal{R})(H(\hat{S})+H(\hat{D}))\right) \qquad (1)
\end{aligned}$$

□

5.2. Análise no cenário concorrente

A análise do RanDiSplayNet no cenário concorrente é equivalente à análise do DiSplayNet, apresentada em [Peres et al. 2019]. Podemos mostrar que todas as atualizações de enlaces são consistentes e o algoritmo é livre de *deadlocks* e inanição. Além disso, podemos mostrar que o valor esperado do custo adicional (*overhead*) de trabalho gerado pela concorrência entre reconfigurações é $O((1+p\mathcal{R})(m(m+n)\log n)$.

6. Experimentos

Neste trabalho, antes de executar cada experimento, medimos e classificamos a localidade da entrada em termos de seus componentes temporais e não-temporais. A localidade temporal é a tendência de comunicação contínua entre nós por algum intervalo de tempo (Comunicação em rajada). A localidade não-temporal é a tendência de alguns pares de nós se comunicarem com mais frequência do que os outros pares (Comunicação assimétrica). Essa classificação é útil para avaliar redes auto-reconfiguráveis porque alguns algoritmos podem operar melhor com um componente do que com outro. As cargas de trabalho do mundo real podem apresentar uma mistura dessas localidades em vários níveis, e identificá-las pode facilitar a interpretação dos resultados.

6.1. Complexidade dos dados

Como mostrado em [Avin et al. 2020], a quantidade de localidade presente em uma carga de trabalho pode ser medida baseando se na entropia presente da sequência de comunicação. Aqui, o conceito de entropia está ligado a quantidade de informação ou a capacidade de compressão dos dados. Intuitivamente, cargas de trabalho com baixa estrutura de localidade tende a apresentar sequências aleatórias de pares de comunicação e por consequência, baixa taxa de compressão do dados. Já sequências com alta localidade tendem a apresentar certa estrutura entre requisições que permite comprimir melhor. Utilizando se dessa propriedade é possível identificar e medir o tipo de localidade presentes em cada dataset valendo se apenas de operações de randomização e compressão de dados.

6.1.1. Medindo a localidade

Para medir a localidade de referência presente em uma carga de trabalho, usamos a definição de *trace complexity*, introduzida em [Avin et al. 2020], que utiliza apenas operações de randomização e compactação de dados.

Definição 5. ***Complexidade de uma carga de trabalho [Avin et al. 2020]:*** *A complexidade de uma carga de trabalho σ é dada pelo produto da complexidade temporal e não-temporal:* $\Psi(\sigma) = T(\sigma) \times NT(\sigma)$.

Na Tabela 1 listamos os atributos de todas as cargas de trabalho utilizadas em nossos experimentos.

Cenários	Parâmetros	Valores
ProjecToR[Ghobadi et al. 2016] *i.i.d.*	(n, m) $\Psi(\sigma) = (T(\sigma), NT(\sigma))$	(128, 10,000) (0.997, 0.467)
Skewed	(n,m) $(T(\sigma), NT(\sigma))$	(1024, 10,000) (1, 0.4)
Bursty	$(T(\sigma), NT(\sigma))$	(0.4, 1)

Tabela 1. Parâmetros dos datasets

6.2. Datasets

Nesta seção, apresentamos os datasets utilizados nos experimentos e sua classificação quanto ao tipo de localidade. Os datasets estão listados na tabela 1 com os parâmetros e suas complexidades de trace. Para termos pontos de referência quanto a complexidade dos traces reais, nós reproduzimos dois traces artificiais com complexidades opostas de localidade temporal e não-temporal, denominados de *Bursty* e *Skewed*.

A partir da análise da Tabela 1, agrupamos as cargas de trabalho de acordo com sua complexidade e discutimos suas principais características. A carga de trabalho do ProjectToR [pro 2016] descreve a distribuição de probabilidade de comunicação entre 8.367 pares de nós em uma rede de $n = 128$ nós (topo dos racks), selecionados aleatoriamente de 2 grupos de produção, executados entre operações Map-Reduce, construtores de índice, banco de dados e armazenamento de sistemas. Amostramos uma sequência de $m = 10.000$ pedidos independentes e distribuídos identicamente ($i.i.d.$) no tempo pela matriz de comunicação fornecida e repetimos cada experimento 30 vezes.

O *Skewed*, [Avin et al. 2020], possue cargas de trabalho com alta localidade não-temporal, NT, e baixa temporal, T, cujo componente de localidade temporal foi produzido usando a distribuição *Zipf*. Esta distribuição de entropia torna possível calcular seus parâmetros analiticamente dado o valor de entropia que queremos reproduzir e, finalmente, a carga de trabalho do Bursty, com alta localidade temporal e baixa não-temporal, foi gerada artificialmente com o intuito de ter uma localidade temporal extremamente alta e quase nenhuma localidade não-temporal.

7. Simulações

Todos os nossos experimentos foram realizados trinta vezes, usando a plataforma de simulação Sinalgo [Group 2007], que fornece uma abstração de rede por passagem de

mensagens em um modelo de comunicação síncrona. Esses experimentos correspondem a redes de 128, 256, 512 e 1024 nós, identificável no topo dos gráficos de resultados. O projeto foi implementado no repositório `https://github.com/oaugusto/CBNetProject.git`.

Para espaçar as requisições no tempo e tornar as sequencias de *timestamps* mais realistas, utilizamos uma distribuição Poisson com $\lambda = 0,05$, para determinar o tempo de entrada de cada mensagem na rede. Fizemos medições empíricas do trabalho total de roteamento e reconfiguração e do *makespan* do experimento, para isso assumimos que ambos o custo de roteamento e reconfiguração, $R = 1$, apesar de que em instalações reais a reconfiguração costume ser uma operação mais cara.

Rodamos os experimentos sobre o RanDisplayNet definindo valores p diferentes sendo RSN_i o experimento relacionado ao RanDisplayNet com probabilidade $1/i$ de não realizar uma rotação durante determinado *splay*. Para analisar o desempenho do nosso trabalho, comparamos nossos resultados com os *baselines* de duas árvores estáticas, BT, uma BST balanceada e OPT, a BST ótima para a sequência σ de mensagens, tal como é calculada em [Schmid et al. 2016], o que só é possível com conhecimento prévio de σ, e CBN, uma implementação concorrente de CBNet, como apresentado em [Souza et al. 2021].

7.1. Resultados

Dividimos nossos experimentos em grupos, de acordo com o tipo de localidade das respectivas sequências de entrada, e discutimos os resultados obtidos. Estruturamos a discussão os resultados para cada grupo.

Alta localidade não temporal (ProjectToR e Skewed): Nas Figuras 1 e 2 analisamos o custo do trabalho, que é composto pelos componentes de reconfiguração (rotações) e encaminhamento de mensagens (roteamento). Nas cargas de trabalho do ProjectToR e Skewed, a semi-DiSplayNet randomizado realizou menos rotações a medida que a probabilidade diminui em comparação com rede determinística (RSN1) e mantendo a mesma quantidade de trabalho total, exceto para OPT, devido à baixa localidade temporal dessas cargas de trabalho. A diferença de trabalho entre BT e OPT é coerente com a presença de localidade não-temporal nas sequências, como mostrado em ambos os gráficos.

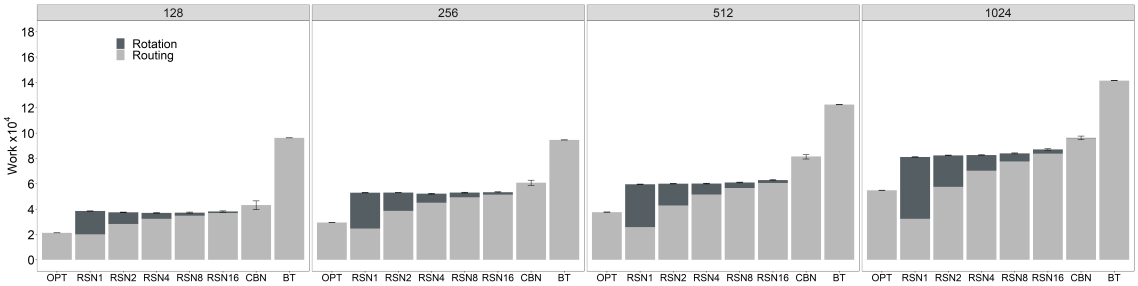

Figura 1. O trabalho total sobre a carga de trabalho do Projector se manteve constante a medida que a probabilidade de rotações diminuiu

Alta localidade temporal (Bursty): Na carga de trabalho Bursty, a matriz de comunicação é quase uniforme e, consequentemente, o trabalho total de BT e OPT é quase o mesmo, como pode ser visto nas Figuras 5. Entre todos os algoritmos, a rede determinística (RSN1) teve o melhor desempenho em termos de trabalho total, mostrando

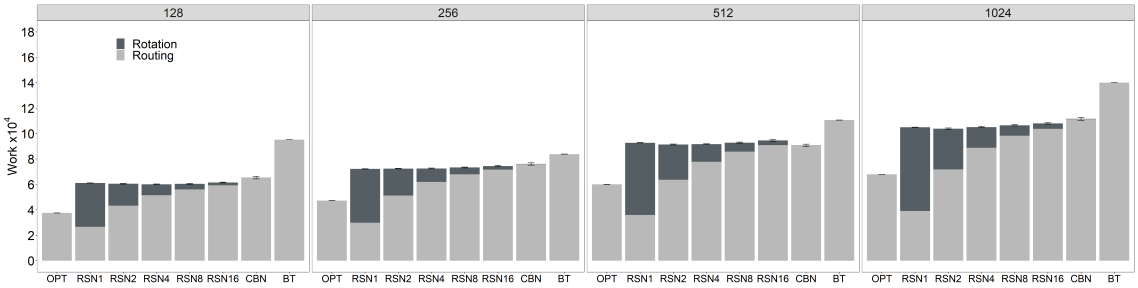

Figura 2. O trabalho total sobre a carga assimétrica do *Skewed* também se manteve constante com a diminuição da probabilidade

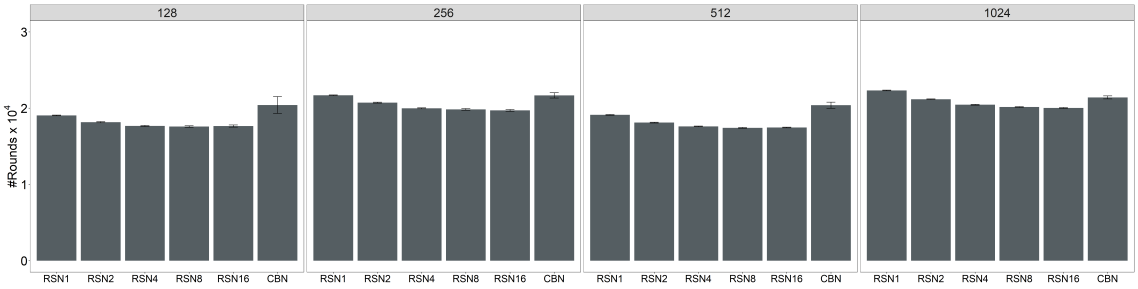

Figura 3. ProjecToR: Makespan

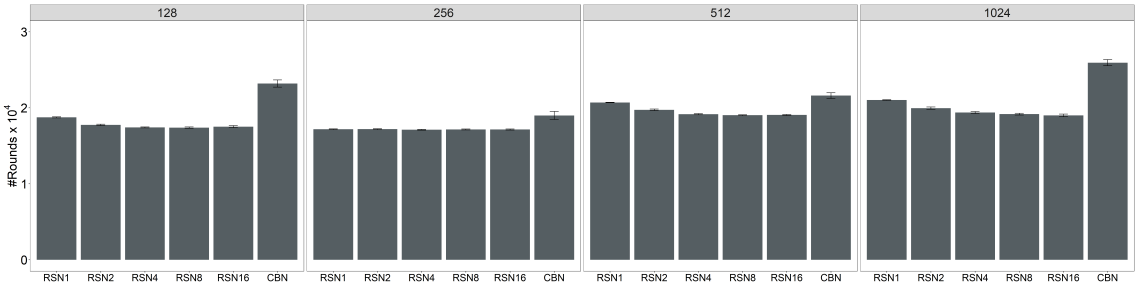

Figura 4. Skewed: Makespan

quão bem este algoritmo explora a localidade temporal (especialmente requisições consecutivas repetidas) realizando rotações de forma agressiva.

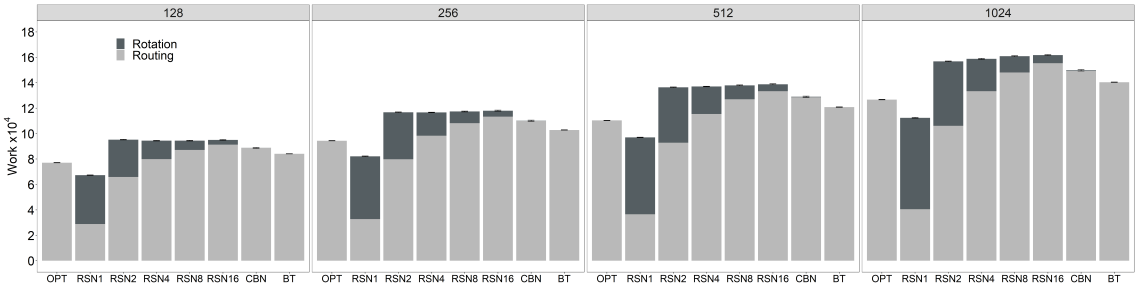

Figura 5. A carga de trabalho do *Bursty*, com alta localidade temporal, é a mais favorecida pelo alto número de reconfigurações

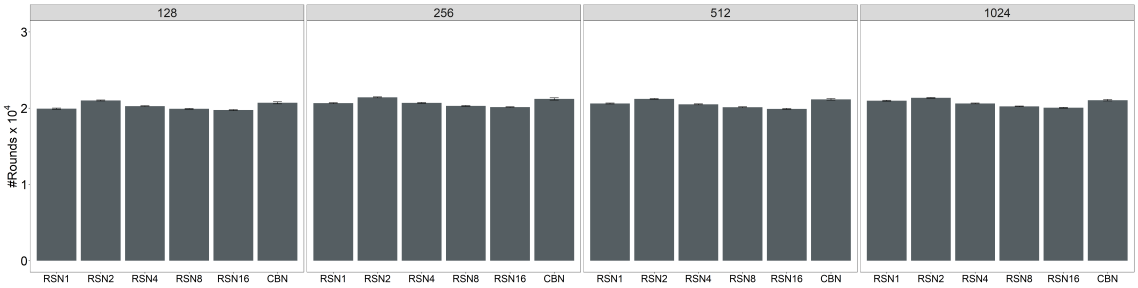

Figura 6. Bursty: Makespan

8. Conclusão

A abordagem randomizada ao estudo de algoritmos auto-reconfiguráveis e distribuídos apresenta-se como uma alternativa válida para as abordagens determinísticas apresentada em [Peres et al. 2019] e em [Souza et al. 2021]. Com resultados semelhantes aos do CBNet, a facilidade na implementação da decisão de rotação e o abandono da necessidade de manter contadores sincronizados ao longo da rede se mostram vantagens significantes em relação ao CBNet. O trabalho total sobre as diferentes sequências de requisições de comunicação, com foco naquelas com maior localidade não-temporal, permaneceu constante com a redução da probabilidade de rotação sobre o *DiSplayNet* mostrando como um ganho considerável de performance, considerando que o custo de reconfiguração é tipicamente mais caro que o de roteamento, com um aumento de complexidade de implementação relativamente baixo.[2]

Referências

(2016). Projector dataset. `www.microsoft.com/en-us/research/project/projector-agile-reconfigurable-data-center-interconnect`.

Afek, Y., Kaplan, H., Korenfeld, B., Morrison, A., and Tarjan, R. E. (2012). Cbtree: A practical concurrent self-adjusting search tree. In *Proceedings of the 26th International Conference on Distributed Computing*, DISC'12, pages 1–15, Berlin, Heidelberg. Springer-Verlag.

Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM.

Albers, S. and Karpinski, M. (2002). Randomized splay trees: Theoretical and experimental results. *Inf. Process. Lett.*, 81(4):213–221.

Avin, C., Ghobadi, M., Griner, C., and Schmid, S. (2020). On the complexity of traffic traces and implications. In *Proc. ACM SIGMETRICS*.

Avin, C. and Schmid, S. (2018). Toward demand-aware networking: A theory for self-adjusting networks. In *ACM SIGCOMM Computer Communication Review (CCR)*.

Ballani, H., Costa, P., Behrendt, R., Cletheroe, D., Haller, I., Jozwik, K., Karinou, F., Lange, S., Shi, K., Thomsen, B., et al. (2020). Sirius: A flat datacenter network with nanosecond optical switching. In *Proc. ACM SIGCOMM*, pages 782–797.

[2]Este projeto recebeu apoio de CAPES, Fapemig e CNPq.

Chen, K., Singla, A., Singh, A., Ramachandran, K., Xu, L., Zhang, Y., Wen, X., and Chen, Y. (2014). Osa: An optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Transactions on Networking (TON)*, 22(2):498–511.

Chen, L., Chen, K., Zhu, Z., Yu, M., Porter, G., Qiao, C., and Zhong, S. (2017). Enabling wide-spread communications on optical fabric with megaswitch. NSDI'17, page 577–593.

Farrington, N., Porter, G., Radhakrishnan, S., Bazzaz, H. H., Subramanya, V., Fainman, Y., Papen, G., and Vahdat, A. (2010). Helios: a hybrid electrical/optical switch architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*, 40(4):339–350.

Ghobadi, M., Mahajan, R., Phanishayee, A., Devanur, N., Kulkarni, J., Ranade, G., Blanche, P.-A., Rastegarfar, H., Glick, M., and Kilper, D. (2016). Projector: Agile reconfigurable data center interconnect. In *Proceedings of the 2016 ACM SIGCOMM*, SIGCOMM '16, pages 216–229, New York, NY, USA. ACM.

Group, D. C. (2007). Sinalgo - simulator for network algorithms. `https://sinalgo.github.io`. Accessed 22-April-2022.

Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., and Lu, S. (2009). Bcube: a high performance, server-centric network architecture for modular data centers. *Proc. ACM SIGCOMM Computer Communication Review (CCR)*, 39(4):63–74.

Kassing, S., Valadarsky, A., Shahaf, G., Schapira, M., and Singla, A. (2017). Beyond fat-trees without antennae, mirrors, and disco-balls. In *Proc. ACM SIGCOMM*, pages 281–294.

Peres, B., Souza, O., Goussevskaia, O., Schmid, S., and Avin, C. (2019). Distributed self-adjusting tree networks. In *Proc. IEEE INFOCOM*.

Peres, B., Souza, O. A. d. O., Goussevskaya, O., Avin, C., and Schmid, S. (2021). Distributed self-adjusting tree networks. *IEEE Transactions on Cloud Computing*.

Schmid, S., Avin, C., Scheideler, C., Borokhovich, M., Haeupler, B., and Lotker, Z. (2016). Splaynet: Towards locally self-adjusting networks. *IEEE/ACM Trans. Netw.*, 24(3):1421–1433.

Sleator, D. and Tarjan, R. (1985). Self-adjusting binary search trees. *Journal of the ACM (JACM)*, 32(3):652–686.

Souza, O. A. d. O., Goussevskaia, O., and Schmid, S. (2021). Cbnet: Minimizing adjustments in concurrent demand-aware tree networks. In *IEEE Int. Parallel and Distributed Processing Sym. (IPDPS)*, pages 382–391.

Zerwas, J., Kellerer, W., and Blenk, A. (2021). What you need to know about optical circuit reconfigurations in datacenter networks. In *The 33nd International Teletraffic Congress (ITC 33)*.