Controlador Resiliente com Distribuição Eficiente para Redes Definidas por *Software*

Diogo M. F. Mattos, Martin Andreoni Lopez,
Lyno Henrique G. Ferraz e Otto Carlos M. B. Duarte

[1]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

Resumo. *A distribuição do controle em Redes Definidas por Software melhora a segurança, o desempenho e a escalabilidade da rede. Contudo, essa solução carece de consistência da base de dados e cria hierarquias de controle. Este artigo propõe uma arquitetura eficiente de distribuição de controladores. As contribuições principais da arquitetura proposta são: i) um controlador distribuído em zonas para garantir segurança, desempenho e escalabilidade da rede; ii) uma base de dados única mantida por um controlador designado para prover consistência ao plano de controle; iii) uma heurística de localização otimizada dos controladores para reduzir a latência no plano de controle; iv) um mecanismo resiliente de escolha de controlador designado para assegurar o funcionamento da rede quando há falhas. Um protótipo da proposta foi implementado e a heurística de localização foi analisada em diversas topologias reais. Os resultados mostram que a conectividade é mantida, mesmo em cenários em que há alta taxa de falhas em nós da rede. Por fim, é comprovado que a otimização da localização reduz a latência média dos controladores.*

Abstract. *Control plane distribution on Software Defined Networking enhances security, performance and scalability of the network. In this paper, we propose an efficient architecture for distribution of controllers. The main contributions of the proposed architecture are: i) A distributed controller in zones to ensure security, performance and scalability of the network; ii) A single database maintained by a designated controller to provide consistency to the control plane; iii) An optimized heuristic for locating controllers to reduce latency in the control plan; iv) A resilient mechanism of choosing the designated controller to ensure the proper functioning of the network, even when there are failures. A prototype of the proposal was implemented and the location heuristic was analyzed in real topologies. The results show that connectivity is maintained even in scenarios where there is a high failure rate in network nodes. Finally, it is shown that the optimization of location reduces the average latency of controllers.*

1. Introdução

Redes Definidas por *Software* (*Software Defined Networking* - SDN) advogam pela centralização lógica do plano de controle da rede [Kobayashi et al. 2014, Moraes et al. 2014]. A centralização do controle e a visão global consistente da rede

Este trabalho foi realizado com recursos da CNPq, CAPES, FINEP, FUNTTEL e FAPERJ.

permitem que operadores inovem e implementem novos serviços no núcleo da rede de forma ágil. No entanto, a centralização física do controle em Redes Definidas por *Software* implica desafios para a segurança, o desempenho e a escalabilidade da rede [Kobayashi et al. 2014]. A escalabilidade é prejudicada tanto no diâmetro da rede, comutadores distantes podem não estar próximos de um controlador, assim como no número de comutadores na rede, já que o número de controladores pode não ser suficiente para responder em tempo hábil às requisições de todos comutadores. A centralização física do controle gera um ponto único de falha. O controlador pode ser alvo de ataques de negação de serviço (DoS) [Lopez and Duarte 2015]. Os ataques DoS podem exaurir tanto recursos de processamento, exigindo o atendimento a uma quantidade anômala de requisições pelo controlador, como também pode ocorrer através da poluição (*jamming*) do canal entre controlador e comutadores [Mattos and Duarte 2014].

A distribuição do controle e a replicação de controladores são as principais técnicas para aumentar a resiliência das Redes Definidas por *Software* [Muller et al. 2014]. No entanto, a adoção de controladores que agem como um sistema distribuído [Berde et al. 2014, Koponen et al. 2010] ou a simples replicação de controladores [Kreutz et al. 2013] não é suficiente para garantir a segurança, o desempenho e a escalabilidade, pois dependem ainda do mapeamento otimizado entre comutadores e controladores, criando domínios de controle. Outro desafio que desponta da distribuição do controle é manutenção da consistência da visão global unificada [Levin et al. 2012, Schmid and Suomela 2013]. A visão global inconsistente acarreta em perda de desempenho e erros na execução de aplicações de controle da rede [Schmid and Suomela 2013]. Assim, a distribuição do controle em SDN pode ser entendida sob duas vertentes. A primeira é a provisão de um controlador distribuído com visão global da rede consistente [Koponen et al. 2010, Berde et al. 2014]. A segunda é a otimização da localização e da quantidade de controladores necessários para uma determinada topologia de rede [Heller et al. 2012, Bari et al. 2013].

Este artigo propõe uma arquitetura eficiente para a distribuição de controladores em uma Rede Definida por *Software*. Para tanto, o artigo aborda a distribuição do controle sob a ótica das duas vertentes. Primeiro, desenvolve-se um controlador distribuído para SDN. A proposta consiste em criar zonas de controle independentes com um controlador responsável. Os controladores das zonas se reportam a um controlador designado responsável por manter a consistência da visão global conhecida pelos controladores. Após, modela-se o problema da localização de controladores. O problema é modelado considerando a minimização da latência entre controlador e comutadores e, também, a maximização da conectividade entre controladores. A minimização da latência é um critério importante para diminuir o tempo de configuração de um novo fluxo. A maximização da conectividade entre controladores é importante para garantir-se a resiliência da rede. Um protótipo do controlador proposto foi implementado. A otimização da localização de controladores foi analisada em diferentes topologias. A proposta ainda cria um mecanismo de controladores de salvaguarda (*backup*). No caso de queda de um controlador designado, outro controlador assume. A lista de controladores de salvaguarda é ordenada pelas latências médias dos controladores de zona em relação aos demais.

A distribuição do controle entre diferentes nós físicos, mantendo a visão global de rede centralizada, é um dos principais desafios em SDN [Koponen et al. 2010,

Berde et al. 2014]. Algumas propostas apontam os problemas das Redes Definidas por *Software* e definem soluções iniciais para a distribuição de controle e para a resiliência da rede [Heller et al. 2012, Kreutz et al. 2013]. Outras propostas buscam criar novos controladores distribuídos do ponto de vista de implementação de um sistema operacional distribuído para controlar uma SDN, mas não otimizam a localização dos controladores [Tootoonchian and Ganjali 2010, Koponen et al. 2010, Berde et al. 2014, Hassas Yeganeh and Ganjali 2012]. Há ainda estudos que focam na otimização da localização dos controladores seguindo diferentes heurísticas e objetivos [Muller et al. 2014, Bari et al. 2013]. Essas propostas não visam criar uma arquitetura única que considere desde a concepção do controlador até a localização otimizada dos controladores. Por sua vez, este artigo propõe a arquitetura de controle distribuído com a localização otimizada e mecanismos para garantir a resiliência da rede.

O restante do artigo está organizado da seguinte forma. Os trabalhos relacionados são discutidos na Seção 2. A Seção 3 discute o problema da distribuição e localização de controladores em Redes Definidas por *Software*. O controlador distribuído proposto e a heurísica de localização são apresentados na Seção 4. A Seção 5 apresenta a análise da avaliação do controlador proposto. A Seção 6 conclui o artigo.

2. Trabalhos Relacionados

Levin *et al.* argumentam que a distribuição do controle físico, enquanto o controle lógico mantém-se centralizado, prejudica o desempenho da rede, quando as aplicações de controle centralizado são agnósticas quanto à distribuição do estado [Levin et al. 2012]. Os autores identificam a existência de duas relações de compromisso. A primeira é entre o desempenho das aplicações de controle e a sobrecarga de distribuição de estados entre controladores. A segunda relação é entre a complexidade da lógica das aplicações de controle e a robustez contra inconsistência. Por sua vez, Schmid e Suomela defendem que há aplicações que podem ser executadas por controladores locais, sem que necessitem de uma visão global consolidada [Schmid and Suomela 2013]. Schmid e Suomela identificam dois tipos de planos de controle, o plano de controle horizontal e o plano de controle hierárquico. No plano de controle horizontal cada controlador é responsável por uma área disjunta dos demais e a estrutura se organiza a partir de restrições administrativas. Já no plano de controle hierárquico, os controladores se organizam verticalmente, convergindo para uma visão global da rede conforme se sobe na hierarquia.

A presença de um controlador fisicamente centralizado não está intrinsicamente ligada à arquitetura SDN. Portanto, há propostas de controladores fisicamente distribuídos, mas que mantêm a visão global da rede. A proposta ONIX age como um *middleware* para sistemas distribuídos, propagando as informações para os controladores físicos, e fornece uma API (*Application Programming Interface*) para as aplicações se comunicarem entre as diferentes réplicas e controlarem a rede [Koponen et al. 2010]. Baseado nos mesmos preceitos do controlador ONIX, há também a proposta ONOS [Berde et al. 2014]. O controlador ONOS baseia-se no controlador Floodlight e distribui o estado da rede com o registro distribuído Zookeeper. Outro controlador distribuído é o HyperFlow [Tootoonchian and Ganjali 2010]. Os eventos no HyperFlow são divulgados através do modelo Publicador/Assinante (*Publish/Subscriber*). Tais propostas se baseiam na distribuição do estado entre todos os controladores, mas não preveem uma política de localização e otimização do número de controladores na rede.

Yeganeh *et al.* argumentam que a escalabilidade do controle de Redes Definidas por *Software* pode ser alcançada através de controladores multiprocessados ou através de diversos controladores distribuídos [Yeganeh et al. 2013]. A proposta Kandoo é um controlador SDN hierárquico em que o escopo das aplicações é dividido em local ou global [Hassas Yeganeh and Ganjali 2012]. Aplicações locais e globais coexistem no controle da rede. As aplicações locais são as que podem operar somente com o estado local de cada comutador e, portanto, são implantadas nos controladores mais próximos do plano de dados. Mais próximas dos comutadores, as aplicações podem tratar mais rapidamente as requisições mais frequentes e evitam a sobrecarga no restante da rede de controle. Por sua vez, as aplicações globais são executadas por controladores no topo da hierarquia.

A otimização da localização e o número de controladores para atender uma Rede Definida por *Software* também é um desafio de pesquisa atual. Müller *et al.* propõem uma estratégia de localização de controladores, denominada *Survivor*, que considera a diversidade de caminhos até o controlador, a capacidade dos controladores e os mecanismos de recuperação de falhas [Muller et al. 2014]. A estratégia se baseia em um modelo de Programação Linear Inteira para encontrar a localização ótima dos controladores. O modelo considera que a localização ótima é a que maximiza a conectividade entre controladores e comutadores da rede. Paralelamente, Zhang *et al.* também propõem uma estratégia de localização de controladores para aumentar a resiliência da rede [Zhang et al. 2011]. A ideia central de Zhang *et al.* consiste em minimizar a probabilidade de falha em cada partição de controle da rede. Para tanto, os autores usam um algoritmo de *min-cut* para minimizar duas funções. A primeira minimiza a distância de todos os nós até seu controlador, resultando em uma minimização intra-clusters. A segunda minimiza a conectividade inter-cluster, reduzindo o número de enlaces passíveis de falha entre controlador e comutadores. Bari *et al.* também propõem um esquema de otimização da localização de controladores [Bari et al. 2013].

A proposta deste artigo é um controlador distribuído, com controle horizontal, cuja localização maximize o número de caminhos dos comutadores ao controlador, mas também minimize a latência entre controlador e comutadores de uma partição de controle. Tais funções objetivo são importantes, pois a maximização da quantidade de caminhos entre controlador e comutadores garante resiliência a falhas de enlaces, enquanto a minimização da latência influi diretamente no tempo de configuração de novos fluxos. Assim, a proposta deste artigo age tanto na resiliência quanto na eficiência da rede, enquanto outras propostas atacam apenas uma das vertentes.

3. O Problema da Distribuição e Localização de Controladores

O problema da distribuição e da localização de controladores em SDN é definido por Heller *et al.* como duas características do controle da rede: i) o número de controladores necessários; e ii) o lugar de posicionamento desses controladores. No entanto, essas duas características não exaurem as possibilidades de otimização do controle da rede [Heller et al. 2012]. Elas apenas norteiam a busca por soluções de controle otimizadas. Em adição a esses dois pontos, deve-se considerar uma função objetivo. A função objetivo pode maximizar a distribuição de controladores para tolerar falhas, ao passo que pode também minimizar a latência para aumentar o desempenho e reduzir o tempo de configuração de fluxos.

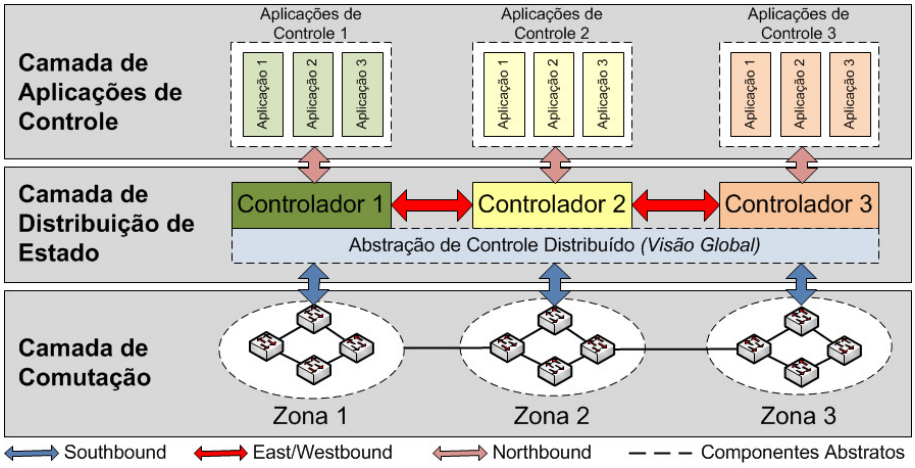


Figura 1. Rede Definida por *Software* com controle distribuído. A rede pode ser dividida em três camadas: comutação, distribuição do controle e aplicações. As setas indicam fluxos de dados de controle na rede. Cada ponto de troca de dados tem uma API (*Southbound, Northbound e East/Westbound*).

Paralelamente ao problema de otimização da localização e da quantidade de controladores, há também o desafio da distribuição do estado na rede de controle. Para tanto, considera-se a arquitetura de rede mostrada na Figura 1. A Figura 1 apresenta os pontos de troca de estado da rede em uma SDN. A ideia é que a rede é divida em três camada lógicas [Schmid and Suomela 2013]. A camada mais baixa, a camada de comutação, é representada pelos elementos de encaminhamento de pacote. Esses elementos armazenam as tabelas de encaminhamento e, no caso de redes OpenFlow [Kobayashi et al. 2014], armazenam também informações relativas aos fluxos e aos contadores de fluxo. A camada de comutação pode ser agrupada em domínios de controle distintos, seja por questões de escalabilidade, seja por questões administrativas.

A camada intermediária, a camada de distribuição de estado de rede, consiste na comunicação entre controladores, divulgando os eventos e o estado local de cada controlador. As propostas ONIX [Koponen et al. 2010] e ONOS [Berde et al. 2014] implementam tal camada de forma semelhante a um *middleware* de sistemas distribuídos. Já o HyperFlow a implementa como o canal de eventos sob o modelo de Publicador/Assinante. Já este artigo propõe a realização da camada de distribuição de estado através da ideia de controlador designado que é responsável por manter uma cópia atualizada da visão global da rede. A camada de distribuição de estado é normalmente referenciada como Sistema Operacional de Rede ("*Network Operating System*") [Levin et al. 2012], pois é essa camada que faz a mediação entre as aplicações de rede e a camada de comutação, permitindo a abstração da camada de comutação para as aplicações. Por fim, a terceira camada lógica é a camada das aplicações de controle. Cada aplicação se comunica com as demais através da camada de distribuição de estado. As aplicações podem executar em nós distintos ou mesmo em um único nó.

Na Figura 1 é possível ressaltar três pontos sensíveis de troca de informação e distribuição do estado. O primeiro é entre os comutadores e os controladores (sistemas operacionais de rede). A implementação mais comum do conceito de SDN, padroniza esse ponto de troca de informação através da API OpenFlow [Kobayashi et al. 2014].

Esse ponto de troca é comumente chamado de *southbound API*. O segundo ponto de troca de informação é entre controladores na camada de distribuição de estado. Nesse caso ocorre a comunicação horizontal entre controladores. A comunicação horizontal entre controladores é chamada de *East/Westbound API*. Por fim, há ainda o ponto de troca de informação entre o controlador, representado na camada de distribuição de estado, e as aplicações. Esse último ponto é chamado de *Northbound API* [Kreutz et al. 2015].

4. O Sistema Proposto

A ideia central do controlador distribuído proposto é dividir o controle da Rede Definida por *Software* em zonas de controle e gerar uma abstração da comunicação da visão global da rede entre todos os controladores de zona. Para tanto, a proposta considera a criação de uma camada de manutenção da consistência entre controladores. Essa camada distribui a visão global da rede entre todos controladores de zona.

4.1. O Controlador Distribuído

A Figura 2 mostra a arquitetura do controlador proposto. Um dos controladores de zona é escolhido como o Controlador Designado. O Controlador Designado é um controlador como os demais, mas que, a partir do momento da sua escolha, passa a ser o responsável por manter a visão global da rede e a distribuí-la para os demais. Nesse contexto, a visão global da rede é considerada como qualquer informação das aplicações de controle que sejam de interesse de mais de um controlador de zona ou que um controlador de zona deseja disponibilizar para os demais.

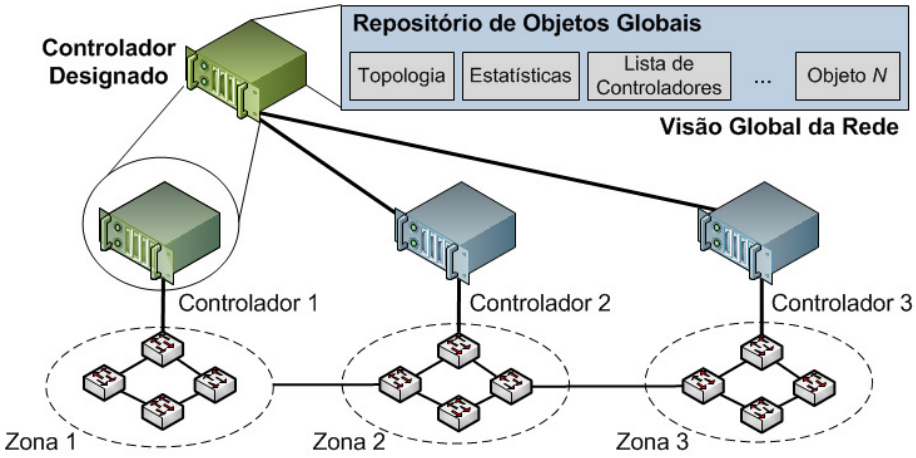


Figura 2. Proposta do controlador distribuído. O Controlador 1 é o Controlador Designado e, portanto, mantém o Repositório de Objetos Globais. O repositório é acessível a todos controladores e permite a distribuição da visão global.

O Controlador Designado armazena e disponibiliza um repositório de objetos de visão global da rede. Um objeto de visão global da rede é uma estrutura de dados que qualquer controlador registra na camada de abstração e distribuição do controle. Quando o objeto é registrado na camada de abstração e distribuição do controle, ele torna-se acessível aos demais controladores, tanto para leitura, quanto para a atualização. O repositório do controlador designado armazena todos objetos registrados.

O registro de objetos na camada de distribuição do controle é importante para abstrair o controle distribuído. Ao acessar um objeto que está registrado no controlador

designado, o controlador de zona pode operá-lo como se fosse local. O controle da consistência do objeto é realizado por parte do controlador designado. Para tanto, um objeto registrado só pode ser atualizado através de um único acesso por vez. As aplicações que executam nos controladores de zona podem realizar o controle local sem que haja a necessidade de se reportarem para o controlador designado, dado que para o controle local não há a necessidade de atualizar a visão global da rede.

No caso de aplicações que atualizam a visão global da rede, como a aplicação de descoberta de topologia, sempre que necessário, cada controlador de zona acessa o objeto referente àquela informação e o atualiza. No caso da descoberta de topologia, o objeto compartilhado é uma coleção de enlaces conhecidos entre os nós e o conjunto dos nós conhecidos da rede. Paralelamente à descoberta de topologia, também pode-se citar o cálculo da árvore de cobertura. Vale notar que, dada a visão global consistente e única da rede, o cálculo da árvore de cobertura pode ser realizado localmente. Cada controlador de zona só é responsável por ativar e desativar a inundação de portas de comutadores da sua zona, sem interferir nos demais [Schmid and Suomela 2013].

A resiliência é alcançada no cenário de controladores distribuídos proposto da seguinte forma. Cada controlador de zona é capaz de assumir o papel de Controlador Designado. Assim, cada controlador de zona possui uma lista ordenada de quais controladores devem assumir o papel de designado em caso de falha. Essa lista é distribuída e mantida ordenada através de um objeto registrado no repositório do controlador designado ativo. Os controladores de zona, de tempos em tempos, consultam a lista ordenada e mantém uma cópia local do estado global da rede, pois no caso de falha do Controlador Designado, o estado global da rede é mantido. A ordenação da lista se dá através da ordem crescente média $L(C_k)$ da latência de um controlador a todos os outros. Ou seja, a cada controlador C_k é associado um valor:

$$L(C_k) = \frac{\sum_{\forall i \in N} Latencia(C_k, C_i)}{N}, \tag{1}$$

onde i é cada nó da rede e N o conjunto de todos os nós da rede, e a lista é ordenada em função do valor $L(C_k)$ de cada controlador. A $Latencia(C_k, C_i)$ na Equação 1 é calculada *a priopri*, de acordo com a topologia da rede, no momento da localização dos controladores. O controlador designado apenas mantém esse valor atualizado de acordo com a monitoração da rede.

4.2. A Localização dos Controladores

A localização dos controladores é definida como um problema de otimização com múltiplos objetivos. Para atingir a máxima resiliência da rede é importante que a localização dos controladores considere que em um cenário de falha, a rede continue conexa e que, caso haja partição da rede, cada partição possua um controlador em seu interior. Paralelamente, é importante que para o funcionamento adequado da rede, o tempo de configuração de um novo fluxo seja o mínimo possível [Heller et al. 2012, Bari et al. 2013]. Assim, é importante que o tempo de comunicação entre o controlador e o conjunto de comutadores controlados seja o mínimo possível. Assim, os objetivos da otimização da localização de controladores são dados por:

$$min \prod_{\forall i \in N} (PercentualRede(C_i) * Prob(C_i))^{y_i}, \quad (2)$$

onde y_i identifica se o nó i é controlador ($y_i = 1$) ou não ($y_i = 0$), $PercentualRede(C_i)$ indica o percentual da rede que fica sem controlador caso haja uma falha em C_i e N representa o conjunto de todos os nós na rede. A equação

$$min \sum_{\forall i \in N} \frac{\sum_{\forall k \in N} y_i * y_{i,k} * latencia(C_k, C_i)}{N} \quad (3)$$

minimiza a latência, em que $y_{i,k}$ representa que o controlador i controla o nó k e y_i representa se o nó de índice i é um controlador ou não. As equações valem para $i, k \in N$, onde N é o conjunto de todos os nós da rede.

A Equação 2 é confusa a primeira vista. Contudo, sua interpretação é fácil. A ideia da equação é minimizar o percentual da rede, representado pela função $0 < PercentualRede(C_i) \leq 1$, que fica sem controle caso o controlador C_i falhe. A função $Prob(C_i)$ modela a probabilidade de falha do nó C_i. No entanto, a falha do nó C_i só é prejudicial para o controle da rede caso C_i seja um controlador. Assim, cada termo do produtório é elevado a y_i que determina se o nó de índice i é controlador ou não. No caso de ser controlador, o termo é considerado. No caso contrário, o termo é elevado a 0, resultando no valor 1, que não interfere no produtório. Assim, ao minimizar a Equação 2, procura-se uma solução que mantenha a rede conectada e com controle, mesmo em cenários de falha. A Equação 3 é de interpretação imediata. Essa equação busca uma solução de localização que minimize a latência média entre todos os controladores e os comutadores que ele controla.

4.3. A Heurística de Localização

As Equações 2 e 3 definem a ideia central da estratégia de localização proposta. No entanto, para aplicar a estratégia proposta, desenvolve-se uma heurística de cálculo da localização de controladores otimizada. A heurística se baseia no método de otimização de Arrefecimento Simulado[1] [Cardoso et al. 2014].

O problema da localização foi modelado em Arrefecimento Simulado da seguinte forma. A solução do problema é um vetor com Y posições, onde Y é o número de controladores que se deseja alocar na rede. A cada iteração é gerada uma nova solução candidata que pode ser aceita caso tenha um custo menor do que a anterior. Se a solução não for melhor que a anterior, ela tem uma probabilidade associada a ela. Sorteia-se um número aleatório e verifica-se se o número é menor que a probabilidade de aceitação da solução. Em caso positivo, a solução é aceita mesmo tendo um custo maior do que a anterior. Esse comportamento é necessário para evitar que a otimização convirja para um mínimo local.

[1]O método de otimização de Arrefecimento Simulado foi escolhido como meta-heurística, pois esse método tem a convergência para o mínimo global comprovada, em tempo indeterminado, mesmo no cenário de decaimento linear de temperatura. Para tanto, a perturbação usada para a geração de novas soluções segue distribuição Cauchy.

Algoritmo 1: Mapeamento dos comutadores em controladores pertencentes ao vetor solução. A função $Distancia$ é definida como a latência entre os nós.

```
G = Grafo(topologia_da_rede)
controladoresArray = array de controladores calculados
mapArray = array [tamanho(G)] de inteiro
mapArray[k] = 0 ∀k ∈ G
for i ∈ tamanho(mapArray) do
    for j ∈ tamanho(controladoresArray) do
        if Distancia(controladoresArray[j], G[i]) <
        Distancia(mapArray[i], G[i]) then
            mapArray[i] = controladoresArray[j]
        end
    end
end
return mapArray
```

Dado o vetor solução retornado pela meta-heurística de Arrefecimento Simulado, dois algoritmos são aplicados sobre ele. O Algoritmo 1 mostra o mapeamento dos comutadores nos controladores do vetor solução. Esse algoritmo define as zonas de controle. O Algoritmo 2 mostra o cálculo da função objetivo que é o quanto da rede permanece conectado mesmo em um cenário com probabilidade de falha de nós.

O Algoritmo 2 recebe como entrada o grafo G, contendo toda a topologia da rede, um vetor de probabilidade $prob$, no qual cada posição i representa a probabilidade de falha do nó i da rede, e o vetor de mapeamento de comutadores em controladores. A saída do algoritmo é o inverso do percentual de nós da rede que permanecem conectados e com controle, mesmo no cenário de falhas representado pelo vetor $prob$. É importante ressaltar que o valor retornado é o inverso do percentual de nós que permanecem controlados, pois o problema de localização foi desenvolvido e modelado como um problema de minimização. Sendo assim, a minimização do inverso é a maximização dos nós que permanecem controlados mesmo no cenário de falhas.

5. A Avaliação do Sistema Proposto

A proposta foi avaliada em duas etapas. A primeira etapa foi a implementação de um protótipo do controlador distribuído. O protótipo foi submetido a experimentos para verificar a consistência da visão global da rede e o ganho de desempenho ao distribuir o controle e submeter o sistema a altas taxas de requisição de fluxos/segundo. A segunda etapa de avaliação consiste na otimização da localização de controladores em diferentes topologias reais e, posteriormente, a localização otimizada encontrada é avaliada quanto à latência média entre controladores e comutadores e quanto à resiliência provida à rede. A heurística de otimização proposta é comparada com heurísticas com diferentes funções objetivo: menor latência entre controlador e comutadores, centroides da rede, menor número de saltos entre controlador e comutadores.

O protótipo do controlador proposto, seguindo o modelo de Controlador Designado, considera uma rede com comutadores OpenFlow e implementa o controle dis-

Algoritmo 2: Função objetivo da otimização. Medida da partição da rede em cenário de falha.

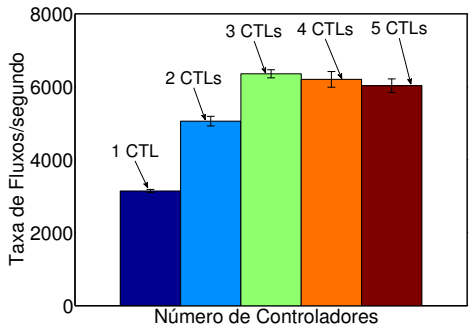
$G = Grafo(topologia\_da\_rede)$
$mapArray =$ mapeamento de cada comutador em um controlador
$prob =$ array com a probabilidade de falha de cada nó em G
for $k \in G$ **do**
  $aleatorio = random()$
  if $aleatorio \leq prob[k]$ **then**
    $G.remove(k)$
  end
end
$componentesConexas = conjunto\_componentes\_conexas(G)$
$soma = 0$ %soma de todos os nos controlados em componentes conexas
for $componente \in componentesConexas$ **do**
  for $no \in componente$ **do**
    if $mapArray[no] \in component$ **then**
      $soma+ = 1$
    end
  end
end
return $tamanho(G)/soma$ %inverso do percentual da rede com controle

tribuído sobre o controlador POX[2]. O repositório de objetos globais foi implementado usando-se uma versão adaptada do módulo para a programação distribuída em Python DOPY [3]. Assim, o Controlador Designado escolhido, inicia um servidor em que os demais controladores se conectam a ele para manter a consistência do repositório de objetos globais. Vale ressaltar que para realizar os experimentos com o protótipo, as aplicações `forwarding.l2_learning` e `openflow.spanning_tree` do controlador POX foram adaptadas para registrar e consultar o repositório de objetos globais ao invés de manterem a visão de cada controlador somente local.

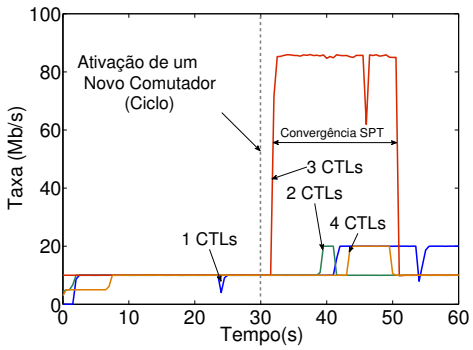
O protótipo do controlador foi executado como diversos processos em um computador pessoal Intel i7-2600 @ 3.40 GHz , com 16 GB de RAM, executando Debian Linux. Os comutadores OpenFlow são quatro computadores pessoais Core 2 Duo @ 2.40 GHz, com 3 GB de RAM, executando Debian Linux e o comutador por software com suporte a OpenFlow, Open vSwitch [4].

A Figura 3 mostra a avaliação do controlador distribuído proposto. O primeiro experimento visa avaliar o número de fluxos/segundo que o plano de controle consegue atender. Para avaliar o plano de controle, foi usada a ferramenta `cbench`[5]. `Cbench` emula diversos comutadores conectados a um controlador e gera eventos de `packet_in` para medir a capacidade do controlador a reagir a esses eventos. A Figura 3(a) evindencia que ao adicionar mais controladores ao plano de controle, a taxa de fluxos/segundo

[2]http://www.noxrepo.org/pox
[3]http://www.mindhog.net/ mmuller/projects/dopy/.
[4]http://openvswitch.org/.
[5]http://www.openflowhub.org/display/floodlightcontroller/Cbench.



(a) Execução do `cbench` para 1, 2, 3, 4 e 5 controladores (CTLs no gráfico). Para 4 e 5 controladores o número de fluxos/s atendidos é menor do que para 3 controladores devido a limitações de processamento da máquina de teste.



(b) Ativação de um novo comutador na rede, em 30 s, gerando um ciclo. Durante a convergência do algoritmo de árvore de cobertura (SPT) há duplicação de pacotes na rede.

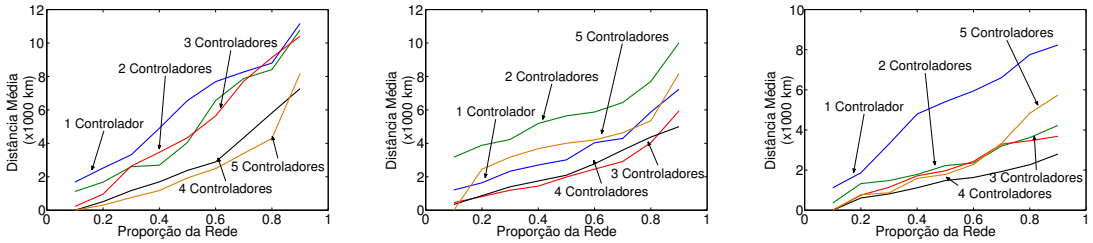
Figura 3. Avaliação do protótipo de controlador distribuído. a) Avaliação quanto ao número agregado de fluxos atendidos pelos controladores na rede. b) Reação do controle distribuído à entrada de um novo comutador na rede.

aumenta. Com três controladores, a taxa/fluxos por segundo atendida é quase o dobro de quando há somente um controlador na rede. Contudo, com 4 ou 5 controladores a taxa de fluxos atendidos é menor do que com três controladores, pois há concorrência entre os processos de controle e geração de fluxo e, portanto, há um gargalo de processamento, já que com 4 controladores os 8 núcleos (`cores`) do computador que hospeda o teste são usados para gerar fluxos e reagir aos eventos de `packet_in`. O segundo experimento verifica a reação do plano de controle à ativação de um novo comutador, fechando uma topologia em anel com os 4 comutadores. O comutador é ativado aos 30 s. Percebe-se que após a ativação há a duplicação de pacotes devido à inconsistência na árvore de cobertura. No pior caso, com três controladores, a convergência da árvore de cobertura chega a demorar 20 s, e converge para que todos os comutadores tenham acesso à visão global.

A avaliação da heurística de localização proposta considera três topologias: topologia da rede da RNP no Brasil, da rede GEANT na Europa e da rede MPLS da AT&T nos Estados Unidos. Os grafos das topologias usadas foram obtidos no *The Internet Topology Zoo*[6]. A rede da RNP conta com 31 nós, a GEANT, com 40 nós e a AT&T conta com 25 nós. A otimização considera que cada nó na rede é capaz de suportar um controlador.

Como o conjunto de dados disponibilizados não apresenta informação sobre latência entre os nós, a avaliação usa a distância física entre os nós como medida indireta da latência. Tal procedimento é aplicado de maneira semelhante por Heller *et al.* [Heller et al. 2012]. A Figura 4 mostra o comportamento da latência em função do percentual de nós na rede. A partir dessa figura, pode-se observar que ao adicionar o quinto controlador, a redução da latência não é tão significativa. No caso da topologia da AT&T, mostrada na Figura 4(c), com 5 controladores ocorre o processo inverso e há um aumento da latência. Portanto, os próximos resultados, por mérito de clareza, consideram somente os cenários com 4 controladores.

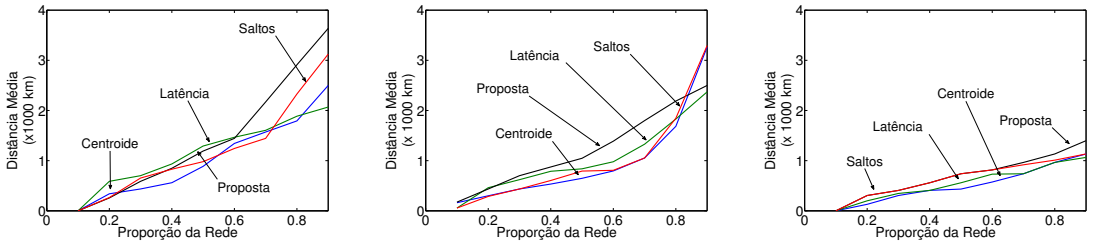
[6]http://www.topology-zoo.org/.



(a) Topologia da RNP no Brasil. (b) Topologia da GEANT na UE. (c) Topologia da AT&T nos EUA.

Figura 4. Latência média entre comutadores e o comutador a que foi mapeado em função do percentual da rede. Heurística proposta de localização considera maior resiliência e menor latência média. Os resultados mostram para cada topologia o uso de até cinco controladores.

Os experimentos seguintes comparam a heurística proposta com três outras: menor latência entre controlador e comutadores (`Latência`), centroides da rede (`Centroide`), menor número de saltos entre controlador e comutadores (`Saltos`). A heurística de menor latência é inspirada na proposta de [Heller et al. 2012]. A heurística da escolha dos centroides da rede é baseada na estratégia de [Zhang et al. 2011]. Por fim, a heurística de escolha dos controladores com o menor número de saltos até os comutadores é uma proposta simplista de comparação com as demais. As heurísticas foram usadas como função objetivo para a otimização do Arrefecimento Simulado.

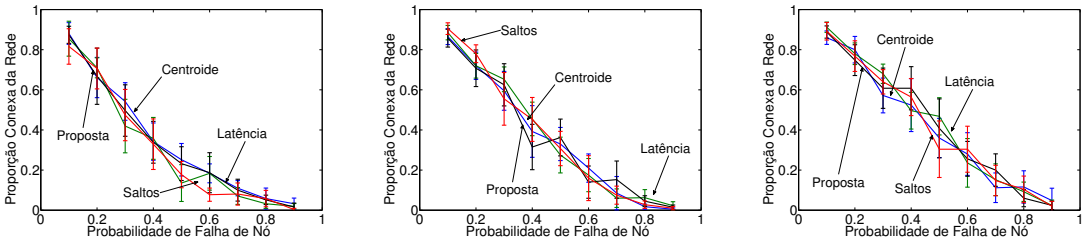


(a) Topologia da RNP no Brasil. (b) Topologia da GEANT na UE. (c) Topologia da AT&T nos EUA.

Figura 5. Latência média entre comutadores e o comutador a que foi mapeado em função do percentual da rede. Comparação entre a `Proposta`, a menor latência (`Latência`), os centroides da rede (`Centroide`) e o menor número de saltos (`Saltos`). Todas as heurísticas consideram a localização de 4 controladores.

A Figura 5 mostra as quatro heurísticas comparadas quanto a latência média entre os controladores e os comutadores. Na três topologias, evidencia-se que a heurística proposta apresenta um desempenho similar às demais. Vale observar, que a heurística do menor número de saltos entre comutador e controlador (`Saltos`), embora seja simplista, apresenta baixa latência média. Esse resultado é compatível com os cenários, pois os nós das topologias consideradas tendem conectarem diretamente com nós próximos.

O último experimento avalia a resiliência da rede, quando a localização dos controladores é realizada através das quatro heurísticas avaliadas. Para tanto, foi desenvolvido um simulador de falhas na topologia. Os resultados da Figura 6 mostram a proporção da rede que continua conexa e controlada em função da probabilidade de falha dos nós



(a) Topologia da RNP no Brasil. (b) Topologia da GEANT na UE. (c) Topologia da AT&T nos EUA.

Figura 6. Proporção da rede que continua conexa e controlada em função da probabilidade de falha dos nós. Comparação entre a `Proposta`, a menor latência (`Latência`), os centroides da rede (`Centroide`) e o menor número de saltos (`Saltos`). Todas as heurísticas consideram a localização de 4 controladores.

da rede. Uma partição da rede é considerada funcional, caso a partição contenha algum nó em que foi implantado um controlador. Essa suposição é feita, pois os comutadores dessa partição podem ter uma lista de controladores de salva-guarda que contenha todos os controladores da rede, ordenados[7].

Os resultados das Figura 6(a) e 6(b) demonstram que a heurística proposta tende a manter uma maior proporção da rede conexa quando a probilidade de falhas dos nós é até 0.5. Contudo, o ganho é limitado quando comparado com outras heurísticas, pois as topologias consideradas não apresentam redundância capaz de suportar falhas de múltiplos nós. Assim, mesmo com uma probabilidade pequena de falha de nós, as partições geradas na rede ficam sem controlador e, portanto, são consideradas como falhas também. Evidencia-se também que a heurística simples de alocação de controladores em nós com menor número médio de saltos até os demais (`Saltos`) apresenta bons resultados de resiliência. A heurística simples `Saltos`, com menor tempo de convergência, é então uma boa opção em topologias com baixo grau de redundância.

6. Conclusão

As Redes Definidas por *Software* (SDN) utilizam a centralização lógica do plano de controle para criar uma abstração da visão global da rede que facilita o desenvolvimento e a implantação de novos serviços no núcleo da rede. Esse artigo propôs uma arquitetura de controladores para SDN que mantém a visão global da rede, ao passo que cria zonas de controle com controle distribuído. A visão global da rede é alcançada através da ideia de Controlador Designado, que é um controlador de zona que assume o papel de manter a consistência da visão global da rede. O controlador designado mantém um repositório de objetos globais que é acessível a todos controladores. Paralelamente ao controlador distribuído, esse artigo também propôs duas heurísticas de otimização da localização dos controladores: uma baseada na maximização da resiliência da rede e, outra, mais simples, baseada na minimização do número de saltos entre controlador e comutadores. Um protótipo do controlador proposto foi implementado e avaliado. Os resultados mostraram que a visão global é mantida no cenário de controle fisicamente distribuído e todos os controladores a acessam e convergem no cálculo de uma árvore de

[7]Considera-se o uso do OpenFlow 1.2, ou superior, para a definição de papéis entre controladores, permitindo a configuração de controladores de salvaguarda.

cobertura comum. As heurísticas de otimização foram avaliadas e os resultados mostraram que a heurística proposta aumenta a resiliência da rede, em especial quando a taxa de falhas é de até 0.5, mas para topologia com menor redundância, a heurística mais simples, o menor número médio de saltos, apresenta desempenho comparável às demais.

7. Referências

[Bari et al. 2013] Bari, M., Roy, A., Chowdhury, S., Zhang, Q., Zhani, M., Ahmed, R., and Boutaba, R. (2013). Dynamic controller provisioning in software defined networks. In *Network and Service Management (CNSM), 2013 9th International Conference on*, pages 18–25.

[Berde et al. 2014] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W., and Parulkar, G. (2014). Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 1–6, New York, NY, USA. ACM.

[Cardoso et al. 2014] Cardoso, L. P., Ferraz, L. H. G., and Duarte, O. C. M. B. (2014). Migração de máquinas virtuais para economia de energia. In *Workshop de Gerência e Operação de Redes e Serviços (WGRS 2014) do SBRC'2014*.

[Hassas Yeganeh and Ganjali 2012] Hassas Yeganeh, S. and Ganjali, Y. (2012). Kandoo: A framework for efficient and scalable offloading of control applications. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 19–24, New York, NY, USA. ACM.

[Heller et al. 2012] Heller, B., Sherwood, R., and McKeown, N. (2012). The controller placement problem. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 7–12, New York, NY, USA. ACM.

[Kobayashi et al. 2014] Kobayashi, M., Seetharaman, S., Parulkar, G., Appenzeller, G., Little, J., Van Reijendam, J., Weissmann, P., and Mckeown, N. (2014). Maturing of openflow and software-defined networking through deployments. *Comput. Netw.*, 61:151–175.

[Koponen et al. 2010] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA. USENIX Association.

[Kreutz et al. 2013] Kreutz, D., Ramos, F. M., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 55–60, New York, NY, USA. ACM.

[Kreutz et al. 2015] Kreutz, D., Ramos, F. M. V., Veríssimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):63.

[Levin et al. 2012] Levin, D., Wundsam, A., Heller, B., Handigol, N., and Feldmann, A. (2012). Logically centralized?: state distribution trade-offs in software defined networks. In *Proceedings of the First workshop on Hot topics in software defined networks*, HotSDN '12, Helsinki, Finland. ACM.

[Lopez and Duarte 2015] Lopez, M. E. A. and Duarte, O. C. M. B. (2015). Providing elasticity to intrusion detection systems in virtualized software defined networks. In *IEEE ICC 2015 - Communication and Information Systems Security Symposium (ICC'15 (11) CISS)*, London, United Kingdom.

[Mattos and Duarte 2014] Mattos, D. M. F. and Duarte, O. C. M. B. (2014). AuthFlow: Um mecanismo de autenticação e controle de acesso para redes definidas por software. In *XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC'2014*.

[Moraes et al. 2014] Moraes, I. M., Mattos, D. M., Ferraz, L. H. G., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M., de Amorim, M. D., Velloso, P. B., Duarte, O. C. M., and Pujolle, G. (2014). Fits: A flexible virtual network testbed architecture. *Computer Networks*, (0):–.

[Muller et al. 2014] Muller, L. F., Oliveira, R. R., Luizelli, M. C., Gaspary, L. P., and Barcellos, M. P. (2014). Survivor: an enhanced controller placement strategy for improving sdn survivability. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, Austin, Texas, USA.

[Schmid and Suomela 2013] Schmid, S. and Suomela, J. (2013). Exploiting locality in distributed sdn control. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 121–126, New York, NY, USA. ACM.

[Tootoonchian and Ganjali 2010] Tootoonchian, A. and Ganjali, Y. (2010). Hyperflow: A distributed control plane for openflow. In *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, pages 3–3, Berkeley, CA, USA. USENIX Association.

[Yeganeh et al. 2013] Yeganeh, S., Tootoonchian, A., and Ganjali, Y. (2013). On scalability of software-defined networking. *Communications Magazine, IEEE*, 51(2):136–141.

[Zhang et al. 2011] Zhang, Y., Beheshti, N., and Tatipamula, M. (2011). On resilience of split-architecture networks. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6.