UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUCAS FERNANDO MÜLLER

Survivor: Estratégias de Posicionamento de Controladores Orientadas à Sobrevivência em Redes Definidas por Software

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Marinho Pilla Barcellos
Orientador

Porto Alegre, dezembro de 2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Müller, Lucas Fernando

Survivor: Estratégias de Posicionamento de Controladores Orientadas
à Sobrevivência em
Redes Definidas por Software / Lucas Fernando Müller. – Porto Alegre: PPGC da UFRGS, 2014.

86 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2014. Orientador: Marinho Pilla Barcellos.

1. Redes Definidas por Software. 2. Problema do Posicionamento de Controladores. 3. OpenFlow. 4. Otimização. 5. Algoritmos. 6. Resiliente. 7. Sobrevivência de Redes. 8. Segurança. I. Barcellos, Marinho Pilla. II. Title.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Prof. Luís da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“Have the courage to follow your heart and intuition. They somehow already know what you truly want to become. Everything else is secondary.”

— STEVE JOBS

AGRADECIMENTOS

Eu gostaria de agradecer, primeiramente, aos que me ensinaram a linguagem da vida e do amor: meus pais e meu irmão, Ângela, Rony e Tomás. Obrigado pelo apoio, incentivo, compreensão e amor. Agradeço tudo que fizeram por mim. Sou eternamente grato a vocês.

Agradeço ao meu orientador, Marinho Barcellos, pela sua dedicação, experiência e conhecimentos compartilhados comigo e acima de tudo pela grande amizade, confiança, paciência, apoio e força, que me ajudou muito a seguir em frente ao longo dessa caminhada e transformar este trabalho em uma realidade. É sobre condições difíceis e adversas que a vida nos impõe que descobrimos com quem podemos contar. Tens minha gratidão, admiração e respeito. Agradeço também ao professor Luciano Gaspary pela amizade, confiança, conselhos e experiências compartilhadas, além das contribuições realizadas durante a realização deste trabalho.

Agradeço também a dois amigos que são praticamente meus irmãos, Rodrigo Mansilha e Rodrigo Oliveira. Citados aqui na ordem em que os conheci e estabeleci esse vínculo forte. Muito obrigado por toda a ajuda, apoio, paciência e incentivo em todos os momentos, mostrando quão forte uma relação de amizade pode se tornar, obrigado por tudo mesmo. Rodrigo Oliveira o teu companheirismo foi muito importante para mim durante esta caminhada.

Agradeço aos meus tios, Liége e Melo com quem ao longo destes anos de muitas mudanças partilhei de dificuldades e sempre estiveram prontamente dispostos a ajudar. Além de todo apoio e força que concederam desde meu primeiro dia em Porto Alegre. Não posso deixar de agradecer nesta mesma linha meu tio Beto e tio Marcos, sempre prontos a ajudar. Meu muito obrigado.

Agradeço a todos os amigos, sem exceção. E isso posso dizer que o nosso grupo de redes proporcionou muitos, que me ensinaram cada qual a seu modo, o significado de parceria e amizade. Tanto aqueles que posso conviver diariamente, Marcelo, Rodolfo, Matheus L., Daniel, Miguel, Esteves, Ricardo, Raniery, Tobias, Rafael L., Gustavo(s), Jeff, Bays, Oscar, Faganello, Cadori, Cristiano, Juliano, Luine, Catarina, como aqueles que podem não estar mais presentes no dia a dia mas, são amizades importantes e preservadas, Bruna, Sergio, Weverton, Jedi, Flávio e Goiano. Obrigado à todos pelas conversas e momentos de descontração.

Ao longo dessa caminhada algumas relações mudaram e novas foram construídas. Pela importância que teve na minha história e por ter feito parte da base desta etapa na minha vida, agradeço Aroa e família Suleiman a amizade e o carinho. Por fim, agradeço a todos os meus familiares e amigos pelo incentivo ao longo deste trabalho e compreensão nos momentos de ausência. Contribuindo para o sucesso deste trabalho. MUITO OBRIGADO A TODOS.

Survivor: Enhanced Controller Placement Strategies for Improving SDN Survivability

ABSTRACT

The SDN paradigm simplifies network management by focusing all control tasks into a single entity, the controller. In this way, forwarding devices can only operate correctly while connected to a logically centralized controller. Within this context, recent literature identified fundamental issues, such as device isolation due to disruptions in the network and controller overload, and proposed controller placement strategies to tackle them. However, current proposals have crucial limitations: (i) device-controller connectivity is modeled using single paths, yet in practice multiple concurrent connections may occur; (ii) peaks in the arrival of new flows are only handled on-demand, assuming that the network itself can sustain high request rates; and (iii) failover mechanisms require predefined information which, in turn, has been overlooked.

This dissertation presents Survivor, a novel controller placement approach for WAN networks that addresses these challenges. The approach explicitly considers the following three aspects in the network design process: connectivity, capacity and recovery. Moreover, these aspects are planned for two distinct states of the network: pre and post-disruption. In other words, the network is configured optimally for both normal operation and for operation after disruption events. To this end, the approach is divided into two steps. The first defines the positioning of the controller instances, and the second specifies a list of backup controllers for each device on the network. Moreover, two strategies based on Survivor are developed. The first strategy, implemented with Integer Linear Programming, guarantees an optimal solution with a high computational cost. The second strategy, implemented using heuristics, provides sub-optimal solutions with a much lower computational cost. Comparisons to the state-of-the-art show that the Survivor approach provides significant increases in network survivability (identified with the lowest probability of connectivity loss) and converged network state through smarter recovery mechanisms.

Keywords: Software Defined Networking, Controller Placement Problem, OpenFlow, Optimization, Algorithms, Resilient, Network Survivability, Security.

RESUMO

O paradigma SDN simplifica o gerenciamento da rede ao concentrar todas as tarefas de controle em uma única entidade, o controlador. Nesse modo de operação, os dispositivos de encaminhamento só funcionam de forma completa enquanto conectados a um controlador. Neste contexto, a literatura recente identificou questões fundamentais, como o isolamento de dispositivos em função de disrupções na rede e a sobrecarga de um controlador, e propôs estratégias de posicionamento do controlador para enfrentá-las. Contudo, as propostas atuais têm limitações cruciais: (i) a conectividade dispositivo-controlador é modelada usando um único caminho, ainda que na prática possam ocorrer múltiplas conexões concorrentes; (ii) alterações no comportamento da chegada de novos fluxos são manipulados sob demanda, assumindo que a rede em si pode sustentar altas taxas de requisição; e (iii) mecanismos de recuperação de falhas requerem informações pré-definidas, que, por sua vez, não são otimizadas.

Esta dissertação apresenta Survivor, uma nova abordagem de posicionamento do controlador para redes WAN que visa enfrentar esses desafios. A abordagem trata três aspectos de forma explícita durante o projeto da rede: a conectividade, a capacidade e a recuperação. Além disso, tais aspectos são planejados para dois estados distintos da rede: pré e pós-disrupção. Em outras palavras, a rede é configurada da melhor forma tanto para operação normal, quanto para operação após eventos de disrupção. Para este fim, a abordagem é dividida em duas etapas. A primeira define o posicionamento de instâncias do controlador, enquanto a segunda especifica uma lista de controladores de backup para cada dispositivo na rede. Ademais, são desenvolvidas duas estratégias com base na abordagem Survivor. A primeira, implementada em Programação Linear Inteira, garante uma solução ótima a um custo computacional alto. A segunda, implementada através de heurísticas, fornece soluções sub-ótimas a um custo computacional muito mais baixo. Comparações com o estado-da-arte mostram que a abordagem Survivor provê ganhos significativos na sobrevivência (identificado na probabilidade mais baixa de perda de conectividade) e no estado convergente da rede através de mecanismos de recuperação mais inteligentes.

Palavras-chave: Redes Definidas por Software, Problema do Posicionamento de Controladores, OpenFlow, Otimização, Algoritmos, Resiliente, Sobrevivência de Redes, Segurança.

LISTA DE ABREVIATURAS E SIGLAS

AMPL	A Mathematical Programming Language
API	Application Programming Interface
CAPEX	Capital Expenditure
CDF	Cumulative Distribution Function
GDA	Grafo Direcionado Acíclico
MCC	MinCut-Centroid
OPEX	Operational Expenditure
PLI	Programação Linear Inteira
SDN	Software Defined Networking
SVVR-H	Survivor Heurístico
SVVR-O	Survivor Ótimo
WAN	Wide Area Network

LISTA DE ALGORITMOS

4.1 Pseudocódigo da solução Heurística Survivor 42
4.2 Pseudocódigo do procedimento que define posicionamento dos controladores 43
4.3 Pseudocódigo do procedimento que define conexões dispositivos – controlador 44
4.4 Arcabouço Heurístico para composição de listas de controladores de backup 46

5.1 Pseudocódigo do simulador 52

LISTA DE FIGURAS

Figura 2.1: Modelo de referência SDN. 26
Figura 2.2: Desafios no projeto de uma rede SDN. 28
Figura 2.3: Exemplos de situações não tratadas apenas com a técnica de redundância física. 29

Figura 5.1: Topologias consideradas na experimentação. 50
Figura 5.2: Probabilidade da perda de conectividade controlador-dispositivo para diferentes probabilidades de falhas 56
Figura 5.3: Distribuição cumulativa de dispositivos desconectados para todos os possíveis casos de falhas 1, 3 e 6 enlaces 57
Figura 5.4: Distribuição cumulativa de dispositivos desconectados para todos os casos possíveis na disrupção de 1, 3 e 6 enlaces 59
Figura 5.5: Número de cenários com sobrecarga durante operação normal e na situação após a fase de recuperação 60
Figura 5.6: Mínimo, máximo, média, e moda para a carga em cada instância do controlador após recuperação – RNP 62
Figura 5.7: Mínimo, máximo, média, e moda para a carga em cada instância do controlador após recuperação – GÉANT 63

Figura C.1: Internet2 – 20% de reserva de backup. 81
Figura C.2: Internet2 – 30% de reserva de backup. 82
Figura C.3: RNP – 20% de reserva de backup. 83
Figura C.4: RNP – 30% de reserva de backup. 84
Figura C.5: GÉANT – 20% de reserva de backup. 85
Figura C.6: GÉANT – 30% de reserva de backup. 86

LISTA DE TABELAS

Tabela 4.1: Resumo dos símbolos do modelo PLI. 39

Tabela 5.1: Desempenho das estratégias implementadas com base na abordagem Survivor. 54

Tabela 5.2: Quantidade de instâncias de controle alocadas em cada estratégia. . . 54

SUMÁRIO

SUMÁRIO 19

1 INTRODUÇÃO 21
1.1 Contexto 21
1.2 Motivação 21
1.3 Contribuições 22
1.4 Organização 23

2 FUNDAMENTOS 25
2.1 Redes Definidas por Software 25
2.2 Cenários de Disrupção 27
2.3 Projeto do Plano de Controle 27

3 ESTADO DA ARTE EM SOBREVIVÊNCIA DO PLANO DE CONTROLE 31
3.1 Sistemas Operacionais de Rede 31
3.1.1 Centralizados 31
3.1.2 Distribuídos 32
3.2 Mecanismos de Recuperação 33
3.2.1 Mecanismos para Operação Limitada 33
3.2.2 Mecanismos Nativos de Recuperação 34
3.3 Posicionamento do Controlador 34
3.4 Limitações 35

4 SURVIVOR 37
4.1 Visão Geral 37
4.1.1 Posicionamento das instâncias de controle 37
4.1.2 Composição da lista de controladores de backup 38
4.2 Estratégias para Posicionamento Sobrevivente 38
4.2.1 Posicionamento Ótimo com Modelo PLI (*SVVR-O*) 38
4.2.2 Posicionamento Utilizando Estratégia Heurística (*SVVR-H*) 41
4.3 Heurísticas para Definição da Lista de Controladores de Backup 45
4.3.1 Definições 45
4.3.2 Arcabouço 46
4.3.3 Heurísticas Propostas 46

5 AVALIAÇÃO 49
5.1 Metodologia 49
5.1.1 Carga de Trabalho 49
5.1.2 Método de Comparação 50
5.1.3 Métricas 51
5.2 Simulador 51
5.3 Resultados da Avaliação 53
5.4 Observações Finais 63

6 CONCLUSÃO 65

REFERÊNCIAS 67

APPENDIX A - LISTA DE PUBLICAÇÕES 71

APPENDIX B - ARTIGO PUBLICADO IEEE GLOBECOM 73

APPENDIX C - POSICIONAMENTO RESULTANTE DAS ESTRATÉGIAS . 81

1 INTRODUÇÃO

1.1 Contexto

O gerenciamento das redes de computadores atuais tornou-se difícil e custoso devido à crescente diversidade de requisitos impostos pelas aplicações que a utilizam (FEAMSTER; REXFORD; ZEGURA, 2013). Tais adversidades são potencializadas pelo fato das configurações dessas redes dependerem de interfaces proprietárias incompatíveis e, por vezes, serem realizadas de maneira manual (BENSON; AKELLA; MALTZ, 2009). Essa situação restringe a eficiência no uso de recursos, dificulta a inovação e reduz a flexibilidade e a capacidade de reação perante situações inesperadas e/ou indesejadas (incluindo falhas, ataques e variações de carga).

Diante desse cenário, o paradigma de Redes Definidas por Software (SDN – *Software Defined Networking*) trouxe diversos benefícios à gerência e operação das redes, facilitando a inovação e simplificando as atividades de controle (JARRAYA; MADI; DEBBABI, 2014). Nesse paradigma, a lógica de controle da rede fica concentrada em um controlador remoto, enquanto o procedimento de encaminhamento de dados continua localizado em cada dispositivo de rede. O controlador (remoto) da rede, nesse caso, atua como uma entidade logicamente centralizada, que possui visão global da infraestrutura e gerencia uma coleção distribuída e programável de dispositivos de encaminhamento de pacotes (NUNES et al., 2014).

1.2 Motivação

Muito embora traga vantagens para o gerenciamento da rede, o paradigma SDN criou uma relação de dependência inerente entre os dispositivos de encaminhamento e o controlador. Mais precisamente, o correto funcionamento da rede depende da comunicação entre controlador(es) e os dispositivos de encaminhamento. Sem esta comunicação, os dispositivos operam com regras potencialmente ultrapassadas (VISSICCHIO; VANBEVER; BONAVENTURE, 2014). Portanto, em SDN a comunicação entre controlador e dispositivos é crítica.

As arquiteturas de SDN evitam o ponto único de falha pela implementação de controladores como se fossem sistemas distribuídos (KOPONEN et al., 2010; YEGANEH; GANJALI, 2012). Embora a replicação aumente a sobrevivência, aspectos fundamentais podem deixá-la ineficiente. Particularmente, duas questões importantes devem ser apropriadamente tratadas. Primeiro, as disrupções na rede podem isolar fisicamente os dispositivos de encaminhamento das instâncias de controlador. Segundo, a alta demanda

da rede pode sobrecarregar uma das réplicas do controlador, afetando negativamente a capacidade de resposta.

Para lidar com essas questões, a literatura propôs estratégias diferentes de posicionamento de controladores. Esta escolha exerce uma influência sobre cada aspecto de um plano de controle dissociado, desde opções do estado de distribuição, a tolerância a falhas, a métricas de desempenho. Em redes de grande escala (por exemplo, WANs) o posicionamento é responsável por determinar limites fundamentais sobre a disponibilidade, o tempo de convergência, além de implicações práticas no desenvolvimento dos serviços de controle de rede.

Nesse sentido, propostas recentes de posicionamento tratam do problema sob a ótica de duas métricas diferentes: (i) sobrecarga do controlador (BARI et al., 2013) e (ii) disrupções da rede (ZHANG; BEHESHTI; TATIPAMULA, 2011; HU et al., 2013). Porém, estas propostas apresentam limitações. Primeiro, ignoram aspectos essenciais para a sobrevivência, tais como diversidade de caminhos, reserva de recursos e o estado de convergência da rede. Segundo, consideram que o controlador possui capacidade ilimitada (exceto Bari et al.), desprezando a possibilidade de sobrecarga (Bari et al. considera uma instância de controlador para cada dispositivo de encaminhamento).

1.3 Contribuições

Nesta dissertação propõe-se *Survivor*, uma nova abordagem de posicionamento do controlador aplicada a redes WAN que supera as limitações acima referidas. A abordagem Survivor possui três benefícios principais. Primeiro, a conectividade é aumentada considerando a diversidade de caminhos. Segundo, a sobrecarga do controlador é proativamente evitada adicionando a consciência da capacidade no posicionamento do controlador. Terceiro, os mecanismos de recuperação são melhorados por meio de uma metodologia para construir a lista de backups nos dispositivos de encaminhamento. A fim de comprovar os benefícios, são realizadas comparações para avaliar o desempenho da solução proposta.

De forma resumida, destacam-se como principais contribuições desta dissertação:

- **Nova abordagem de posicionamento de controladores mais realista.** A abordagem *Survivor* é a primeira a considerar a capacidade dos controladores de forma pró-ativa, enquanto que trabalhos anteriores trataram apenas de forma reativa. O impacto de tal consideração é visualizada nos experimentos realizados, provando que o planejamento da capacidade é essencial para evitar a sobrecarga do controlador, especialmente durante a recuperação da rede.

- **Formalização de um modelo matemático.** A nova abordagem é apresentada como um modelo de otimização em Programação Linear Inteira (PLI) com a finalidade de gerar os *resultados ótimos*.

- **Projeto e desenvolvimento de uma heurística.** A abordagem Survivor requer a solução de um problema de otimização da classe de problemas de localização de instalações (*facility location problem*) (ARYA et al., 2001), de natureza combinatória, o qual é conhecidamente $\mathcal{NP}$-Difícil. Desse modo, uma estratégia heurística foi proposta visando a aplicação da abordagem em topologias maiores.

- **Mecanismos de recuperação mais inteligentes.** Survivor inclui heurísticas para definir uma lista de controladores de backup. Uma metodologia para compor tais

listas, modeladas como um arcabouço de heurísticas, é desenvolvida e três heurísticas são implementadas. Como resultado, o estado convergente da rede pode ser melhorado consideravelmente dependendo da heurística escolhida.

- **Redução significativa da perda de conectividade.** Os potenciais benefícios da abordagem proposta são medidos através de simulações. De acordo com os resultados, a exploração correta da diversidade de caminhos da rede (isto é, a consciência da conectividade) reduz a probabilidade de dispositivos desconectados em falhas de enlace único. Além disso, mesmo com diversidade de caminhos as soluções atuais são menos eficientes, pois as estratégias seguindo a abordagem proposta tem entre 2 a 3 vezes menos dispositivos desconectados nos piores casos de falhas.

1.4 Organização

O restante desta dissertação está organizado da seguinte forma. O Capítulo 2 apresenta os conceitos fundamentais utilizados ao longo dos capítulos desta dissertação. No Capítulo 3, apresenta-se, de maneira sistematizada, uma discussão dos principais trabalhos relacionados às arquiteturas de sistemas operacionais de rede, os mecanismos de recuperação nativos e o problema de posicionamento do controlador. No Capítulo 4 propõe-se a abordagem *Survivor* para o posicionamento de controladores, enquanto experimentos numéricos executados em um ambiente simulado são detalhados no Capítulo 5. Por fim, o Capítulo 6 conclui esta dissertação com considerações finais e indicações de possíveis trabalhos futuros.

2 FUNDAMENTOS

Neste capítulo são apresentados os conceitos chave, no escopo desta dissertação, relacionados ao paradigma de Redes Definidas por Software. O objetivo é esclarecer questões importantes relacionadas à sobrevivência neste cenário. Para isso, na Seção 2.1 apresenta-se uma descrição do paradigma SDN com enfoque na relação de dependência entre os dispositivos e o controlador. Segundo, na Seção 2.2 são definidos os cenários de falhas considerados no projeto de redes SDN. Por fim, na Seção 2.3 discute-se os aspectos envolvidos no projeto do plano de controle, com foco nos desafios do posicionamento.

2.1 Redes Definidas por Software

Esta seção apresenta a arquitetura conceitual de SDN dividida em três partes. Primeiro apresenta-se a relação entre os componentes da arquitetura. Após, descreve-se o papel do controlador na manutenção da consistência da rede, salientando-se a importância da conectividade deste com os dispositivos físicos. Por fim, ressalta-se a importância da distribuição física do controlador, bem como a relevância da definição de um posicionamento ótimo neste contexto.

Visão geral

A arquitetura considerada nesta dissertação segue o modelo de referência definido pelo grupo IETF-SDNRG (SHIN et al., 2013) e está alinhado também com *surveys* específicos da área (KREUTZ et al., 2014; JARRAYA; MADI; DEBBABI, 2014; NUNES et al., 2014). Tal arquitetura é composta por três componentes individuais conectados por camadas de abstração, conforme ilustrado na Figura 2.1. Na camada superior, cada *serviço de rede* (ou serviço de controle da rede) assume a responsabilidade de fornecer determinada tarefa de controle à rede, por exemplo roteamento, isolamento e engenharia de tráfego.

Na camada intermediária, o *controlador* concentra a comunicação com todos os elementos programáveis que compõem a rede e mantém uma visão unificada do estado da mesma (CASADO et al., 2010). Essa visão simplifica o gerenciamento e a manutenção da rede por meio da abstração de um sistema distribuído aos serviços de controle da rede.

Por fim, a camada inferior é composta por um conjunto de *dispositivos de rede* (comutadores e roteadores simplificados), que contam com a função de encaminhamento e suporte a um protocolo de controle remoto, por exemplo, OpenFlow (MCKEOWN et al., 2008). Tais dispositivos executam apenas funções de encaminhamento de pacotes, de acordo com uma ou mais entradas da tabela de fluxo.

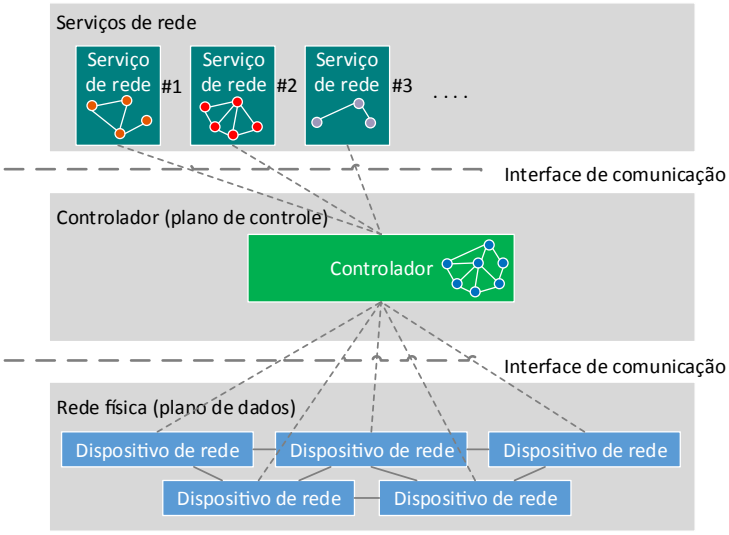


Figura 2.1: Modelo de referência SDN.

Consistência

Cabe ao controlador manter atualizados os outros dois componentes da rede. Essa tarefa pode ser traduzida em duas ações principais: (i) os serviços de rede devem ser atualizados com estado corrente da rede (visão global da topologia); e (ii) os dispositivos de rede devem estar atualizados com as regras de controle dos serviços. No primeiro caso, o controlador é responsável por notificar a camada de serviços sobre eventos que modifiquem o comportamento da infraestrutura da rede (p.ex., disrupções). No segundo caso, o controlador deve repassar novas políticas de controle aos dispositivos físicos (por meio da inserção e remoção de entradas na tabela de fluxo) mediante a atuação dos serviços de rede em operação.

Nesse cenário, é estabelecida uma relação de dependência entre o controlador e os dispositivos da rede. Mais especificamente, a não existência de conectividade entre ambos acarreta mal funcionamento da rede, uma vez que os dispositivos passam a ficar desatualizados e deixam de receber instruções de como encaminhar novos fluxos, tornando-se praticamente indisponíveis.

Controlador fisicamente distribuído

Ao fornecer uma visão logicamente centralizada do estado da rede, o controlador cria a ilusão de ser composto por uma única entidade centralizada. Apesar disso, a centralização é apenas lógica pois, em se tratando da disposição física do controlador, abordagens atuais argumentam em prol do uso de diversas instâncias fisicamente distribuídas para garantir sobrevivência e evitar sobrecargas (BERDE et al., 2014; YEGANEH; TOOTOONCHIAN; GANJALI, 2013; JAIN et al., 2013).

No modelo fisicamente distribuído, um subconjunto de instâncias de controlador podem coordenar-se de forma a criar uma sub-rede consistente. De forma geral, nas soluções atuais que implementam um controlador fisicamente distribuído, cada instância torna-se responsável por um subconjunto de dispositivos da rede física. Assim sendo, para garantir um estado consistente da rede, torna-se necessário manter todas essas instâncias coordenadas (BERDE et al., 2014; KOPONEN et al., 2010). Entretanto, dado um partici-

onamento físico, duas partições podem manter-se em funcionamento independente desde que exista, pelo menos, uma instância de controlador ativo em cada partição (TOOTOONCHIAN; GANJALI, 2010).

Nesse contexto, na ocorrência de falhas na infraestrutura, instâncias mal posicionadas podem acarretar formação de partições sem controle. Portanto, a definição do posicionamento de cada instância de controlador torna-se essencial para o funcionamento correto da rede (ZHANG; BEHESHTI; TATIPAMULA, 2011; HU et al., 2012, 2013).

2.2 Cenários de Disrupção

De acordo com a seção anterior (§2.1), SDN estabelece uma relação de dependência entre o controlador e os dispositivos de encaminhamento. Mais especificamente, a falta de conectividade entre ambos sistemas causa mau funcionamento da rede porque os dispositivos ficam desatualizados e param de receber instruções sobre como encaminhar novos fluxos. Dessa forma, a principal ameaça considerada nesta dissertação é a disrupção de comunicação entre os nós da rede e as instâncias do controlador.

A disrupção pode ocorrer por meio de falhas ou ataques na infraestrutura da rede física. Em linha com trabalhos anteriores (HU et al., 2013; ZHANG; BEHESHTI; TATIPAMULA, 2011), os cenários de falha considerados acontecem de maneira independente e em ambos planos de dados e de controle, conforme detalhado a seguir.

Plano de dados. São consideradas disrupções quaisquer eventos que acarretem a interrupção de um enlace ou dispositivo de encaminhamento. Estão inclusos desligamento manual dos dispositivos, falta de energia, falha de hardware, corte do cabo do enlace, superaquecimento e manutenção.

Plano de controle. São consideradas quaisquer interrupções no funcionamento do controlador ou na conectividade com os dispositivos. Estão inclusos eventos similares aos anteriores, como desligamento do hardware ou encerramento do software de forma manual, por falta de energia, falhas, superaquecimento e manutenção. Também estão inclusos a falta de resposta do controlador por sobrecarga, latência muito grande, ou negação de serviço, desde que o dispositivo de encaminhamento perceba e remova a conexão com o controlador.

Fora do escopo. Não estão inclusos eventos que mantenham os dispositivos de encaminhamento ou instâncias do controlador em estado de funcionamento incorreto (por exemplo, inversão de bits e mensagens mal formatadas ou maliciosas), visto que os mesmos permanecem conectados a rede.

2.3 Projeto do Plano de Controle

Conforme discutido anteriormente, em razão da externalização das funções de controle, criou-se um vínculo entre os dispositivos de encaminhamento e as instâncias do controlador (§2.1). Tal vínculo tornou fundamental o desenvolvimento de um projeto adequado para o plano de controle de modo a lidar da melhor forma com os diferentes cenários de disrupção apresentados (§2.2). Esta seção argumenta que tal situação acarreta um grande número de possibilidades de configuração para uma rede SDN. Mais especificamente, são discutidos os impactos relacionados ao posicionamento físico de cada instância na rede, a quantidade de instâncias de controle e a atribuição das conexões dispositivo-controlador.

Importância do Posicionamento

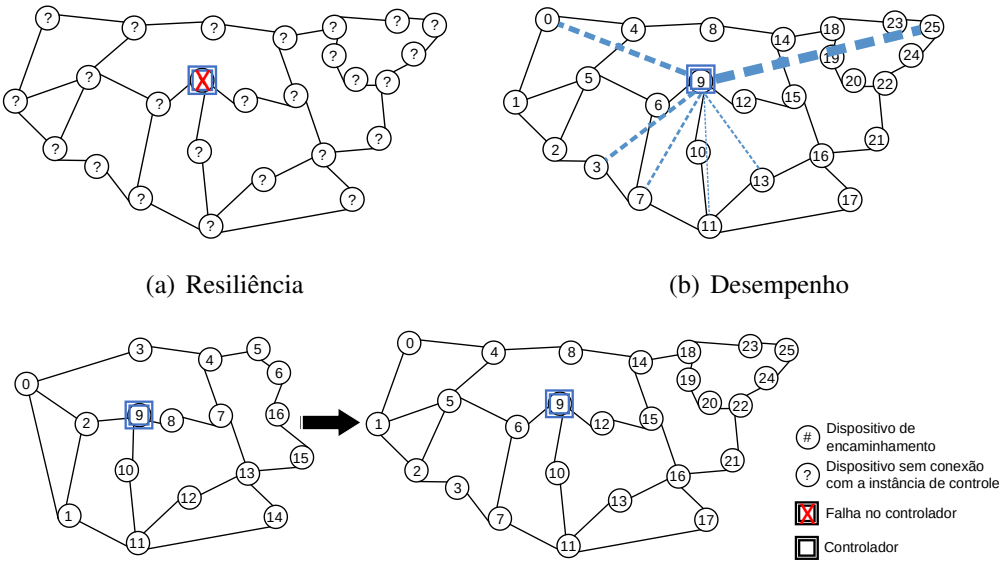
Para obter maior controle sobre os desafios e reduzir o conjunto de possíveis problemas na rede, é de fundamental importância definir estrategicamente o posicionamento do controlador. Com relação a isso, Heller et al. (HELLER; SHERWOOD; MCKEOWN, 2012) formalizaram o Problema do Posicionamento de Controladores. O desafio consiste em analisar a distribuição física de controladores na topologia da rede, além da eficiência no atendimento de propriedades tais como escalabilidade, desempenho e resiliência.

O posicionamento das instâncias do controlador irá determinar a habilidade da rede em responder aos eventos que nela ocorrem. Identificar *quantas* instâncias devem ser utilizadas, *onde* devem ser posicionadas e *quais* são os dispositivos de encaminhamento atendidos por cada controlador tratam-se de pré-requisitos para responder aos desafios impostos à redes SDN. No geral, busca-se otimizar o número de instâncias de controlador na rede, dado que cada instância tem custos inerentes em termos de CAPEX, OPEX e sincronização de estado com o restante das instâncias presentes na rede.

Desafios

O projeto do plano de controle apresenta desafios fundamentais no que se refere a resiliência, escalabilidade e desempenho. Tais desafios estão presentes tanto no caso de um único controlador quanto no caso de múltiplos controladores fisicamente distribuídos. Por uma questão de simplicidade, os desafios serão apresentados em dois passos. No primeiro são ilustradas as situações específicas para um único controlador fisicamente centralizado e no segundo os desafios são generalizados para o caso distribuído.

A definição original do paradigma SDN (MCKEOWN et al., 2008) assumiu um único controlador fisicamente centralizado. Essa centralização logo foi reconsiderada devido a diversas preocupações (LEVIN et al., 2012; SEZER et al., 2013), tais como:



(a) Resiliência (b) Desempenho

(c) Escalabilidade

Figura 2.2: Desafios no projeto de uma rede SDN.

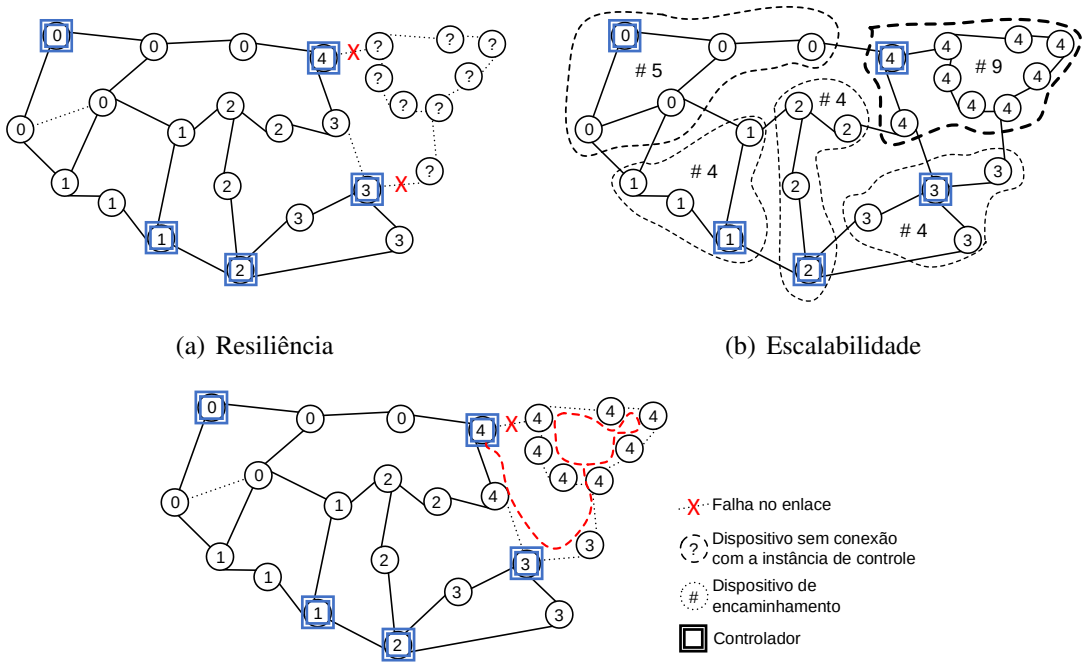
- Resiliência: dado que o controlador torna-se um ponto único de falha e possível alvo de ataques, a disponibilidade dos serviços prestados pela rede pode ser com-

prometida. A Figura 2.2(a) representa o pior caso, onde o nó ao qual o controlador está conectado falha deixando a rede sem comando. É importante lembrar que mesmo com o controlador operacional, parte dos dispositivos podem ser desconectados do controlador devido a falhas em enlaces ou outros nós que ficam no caminho entre os dois.

- Desempenho: em redes com grande diâmetro a localização dos dispositivos de encaminhamento pode ficar muito distante do controlador, de forma que atrasos podem ocorrer na comunicação entre dispositivos de encaminhamento e controlador. Isso pode levar ao aumento do tempo para a configuração dos fluxos de rede. A Figura 2.2(b) demonstra a situação dos atrasos de comunicação (provocados, por exemplo, por um número alto de *hops* ou enlaces de comprimento diferentes) ilustrados através da variação na espessura das linhas tracejadas, quanto mais espessa a linha, maior o atraso na comunicação;

- Escalabilidade: conforme ilustrado na Figura 2.2(c), quando o número de dispositivos de encaminhamento cresce, o tráfego em direção ao controlador é diretamente proporcional;

Para tentar resolver essas questões, arquiteturas SDN recorrem à implementação distribuída de controladores (KOPONEN et al., 2010; YEGANEH; GANJALI, 2012; BERDE et al., 2014) utilizando a estratégia de replicação de instâncias na topologia da rede (YEGANEH; TOOTOONCHIAN; GANJALI, 2013; SEZER et al., 2013).



(a) Resiliência

(b) Escalabilidade

(c) Desempenho

Figura 2.3: Exemplos de situações não tratadas apenas com a técnica de redundância física.

Embora a replicação possa minimizar o impacto gerado nos casos apresentados, existem diferentes situações em que a eficácia desta abordagem será afetada. Para ilustrar isso, a Figura 2.3 utiliza a mesma topologia exibida anteriormente, porém agora com cinco

instâncias do controlador distribuídas. A primeira situação representada na Figura 2.3(a) mostra que a ocorrência de partições de rede pode provocar o isolamento físico entre os dispositivos de encaminhamento e suas instâncias do controlador. Enquanto na segunda Figura 2.3(b), é visualizado o excesso de conexões de dispositivos de encaminhamento em determinadas réplicas do controlador, que acaba por gerar as situações de sobrecarga. E na terceira situação, demonstrada na Figura 2.3(c), percebe-se que a má distribuição das réplicas ocasiona atrasos na comunicação entre as instâncias de controlador e os dispositivos de encaminhamento. Além disso, mudanças no tráfego da rede podem sobrecarregar uma das réplicas do controlador, prejudicando a capacidade de resposta global da rede.

3 ESTADO DA ARTE EM SOBREVIVÊNCIA DO PLANO DE CONTROLE

Neste capítulo é apresentada uma visão geral dos principais trabalhos que compõem o estado da arte relacionados ao problema de sobrevivência do plano de controle. Mais especificamente, a Seção 3.1 apresenta aspectos de sobrevivência presentes nos principais sistemas operacionais de rede. A seguir, a Seção 3.2 identifica os mecanismos existentes para a recuperação da rede na situação de perda de conectividade entre controlador e dispositivos. A Seção 3.3 descreve os trabalhos relacionados ao problema de posicionamento de controladores, com enfoque em estratégias que visam aprimorar a sobrevivência do plano de controle. Por fim, a Seção 3.4 discute as limitações das propostas atuais, as quais serão tratadas pela estratégia proposta nesta dissertação.

3.1 Sistemas Operacionais de Rede

Existe uma grande diversidade de sistemas disponíveis que, por sua vez, apresentam diferentes decisões de projeto e arquitetura. Diante disso, as implementações existentes podem ser categorizadas com base em diferentes aspectos. Sob a perspectiva arquitetural, o aspecto de maior relevância para a sobrevivência é a capacidade de coordenar múltiplas instâncias de controle. Este é um dos principais eixos do projeto de plataformas de controle SDN e, portanto, gera grande impacto no planejamento de uma rede neste paradigma. No que segue, apresenta-se uma discussão de diferentes plataformas em torno deste aspecto.

3.1.1 Centralizados

Um controlador centralizado é composto por uma única entidade, a qual gerencia todos os dispositivos de encaminhamento da rede (vide Figura 2.2, Seção 2.3). Tal arquitetura facilita a manutenção da consistência do estado da topologia, principalmente em redes altamente dinâmicas que necessitam de garantias fortes de consistência. Essa categoria é composta por diversas propostas, tais como Maestro (CAI; COX; NG, 2010), NOX-MT (TOOTOONCHIAN et al., 2012), Floodlight (BIG SWITCH NETWORKS, 2013), e Beacon (ERICKSON, 2013), que têm em comum em seus projetos explorar meios de alcançar a vazão requisitada por redes empresariais e redes de *datacenters*. Estas plataformas de controle tem como base de seus projetos a exploração do paralelismo através de múltiplas *threads*, tirando proveito das arquiteturas de computadores *multi-core*. Nesta lista ainda estão presentes o controlador Ryu (TELEGRAPH; CORPORATION, 2013), Trema (TREMA, 2013), e Meridian (BANIKAZEMI et al., 2013), objetivando fornecer APIs (*Application Programming Interface*) bem definidas para o desenvolvimento

de soluções na implantação em ambientes como redes de *datacenters* e infraestruturas de nuvem.

Em suma, este conjunto de plataformas de controle busca maximizar o desempenho do controlador na rede, porém deixa a cargo do administrador de rede tratar de outras questões fundamentais, como escalabilidade e resiliência (§2.3). Uma entidade fisicamente centralizada representa um ponto único de falha, o que inviabiliza a sua utilização em redes que apresentem, por exemplo, requisitos de alta disponibilidade e tempo de resposta controlado.

3.1.2 Distribuídos

Em contraste ao modelo centralizado, controladores distribuídos são compostos por diversas entidades físicas, as quais podem estar distribuídas em pontos da topologia física ou agrupadas em *clusters* de controladores. Mais especificamente, o controlador fornece uma visão logicamente centralizada do estado da rede (criando a ilusão de ser composto por uma única entidade centralizada), muito embora existam diversas instâncias.

As arquiteturas definidas neste grupo incluem: Onix (KOPONEN et al., 2010), Hyperflow (TOOTOONCHIAN; GANJALI, 2010), Kandoo (YEGANEH; GANJALI, 2012), OpenDaylight (OPENDAYLIGHT, 2014) e ONOS (BERDE et al., 2014). Essas arquiteturas empregam diferentes técnicas/mecanismos para oferecer sobrevivência ao plano de controle e lidar com a sobrecarga.

Clusters

As plataformas Onix (KOPONEN et al., 2010) e OpenDaylight (OPENDAYLIGHT, 2014) adotam a prática de agrupamento (*clustering*) de servidores fisicamente distribuídos, que atuam como uma instância de controlador na rede. No entanto, essas plataformas não apresentam mecanismos para lidar com disrupções e particionamentos, devido a isso elas consideram que as aplicações de controle são responsáveis por tratar as ocorrências de eventos que levem a falhas de conectividade na infraestrutura.

Tolerância a Particionamentos

O Hyperflow (TOOTOONCHIAN; GANJALI, 2010), por outro lado, fornece um mecanismo explícito para lidar com situações de particionamento na rede, ao permitir que instâncias do controlador operem de maneira independente. A plataforma Kandoo (YEGANEH; GANJALI, 2012), que emprega uma abordagem similar ao Hyperflow, propõe a organização dos controladores em uma hierarquia de dois níveis, formada por um controlador raiz e múltiplos controladores locais. Por fim, o sistema operacional ONOS (BERDE et al., 2014) apresenta mecanismos que permitem a operação do sistema no caso de ocorrência de falha de um componente ou uma instância do controlador e isso acontece através da redistribuição das responsabilidades para as instâncias remanescentes.

Controle de Carga

Além desses trabalhos, duas propostas recentes apresentam abordagens distintas para efetuar o controle da carga a ser distribuída para as instâncias do controlador: a ElastiCon (DIXIT et al., 2013) e Pratyaastha (KRISHNAMURTHY; CHANDRABOSE; GEMBER J., 2014). Na primeira, é proposto um protocolo que habilita a migração das conexões de dispositivos de encaminhamento para diferentes instâncias de controlador presentes na rede. De acordo com os autores, essa funcionalidade permite alterações na

configuração da rede de acordo com a dinâmica do comportamento do tráfego vigente. Na segunda abordagem, os autores exploram uma técnica para o uso eficiente dos recursos de memória das instâncias de controle. Para isso propõem o particionamento do estado dos serviços de rede junto da atribuição dos dispositivos de encaminhamento, através de processos de tomada de decisão dinâmicos em relação ao conjunto de instâncias do controlador presentes na rede.

Conforme evidenciado, de fato, algumas plataformas permitem lidar de maneira mais robusta com eventos de falhas (tanto em dispositivos de rede quanto em enlaces) e de ataques à rede e aos próprios controladores. No entanto, nenhuma considera o posicionamento das réplicas, questão fundamental discutida na Seção 2.3 e, que traz consigo outros desafios. A distribuição acarreta em maior custo e dificuldade para manter a consistência do estado da rede entre os diferentes controladores, além de introduzir maior quantidade de tráfego de gerência referente à comunicação entre os nodos de controle. Desse modo, essas arquiteturas oferecem soluções complementares à proposta nesta dissertação.

3.2 Mecanismos de Recuperação

Os trabalhos mencionados até aqui se concentram em manter a conectividade entre os planos de dados e controle. Porém, há situações em que nenhuma das propostas anteriores consegue evitar a perda da conectividade. Em tais situações, alguns dispositivos de encaminhamento podem deixar de ser atendidos por uma instância de controlador e, da maneira como são concebidos no paradigma SDN (isto é, simples, responsáveis apenas pelo encaminhamento), não poderão operar de modo apropriado. Para mitigar o impacto desses casos, há trabalhos que se dedicam a garantir a recuperação do funcionamento do dispositivo afetado. A literatura se concentra em duas abordagens: (i) garantir um conjunto de funcionalidades básicas na ausência de uma conexão de controle funcional e (ii) explorar mecanismos nativos de recuperação.

3.2.1 Mecanismos para Operação Limitada

Esta abordagem se divide em três possibilidades principais: (i) regras de *backup* pré-calculadas (REITBLATT et al., 2013; STEPHENS; COX; RIXNER, 2013); (ii) modo *fail standalone* (isso é, operar como um comutador Ethernet padrão) (ONF, 2014); e (iii) mecanismos de recuperação no plano de dados (LIU et al., 2013).

As regras de *backup* pré-calculadas são entradas adicionais que são inseridas de maneira preventiva nas tabelas de fluxos dos dispositivos. Tais regras são criadas pelo plano de controle, utilizando caminhos alternativos pré-computados, e entram em ação quando o dispositivo não consegue se conectar a qualquer instância de controlador. Essa medida pode tirar vantagem do conhecimento da topologia que o controlador possui para prever os pontos da rede cuja falha tenha maior impacto, permitindo uma recuperação rápida do dispositivo. Entretanto, os caminhos pré-computados oferecem garantias para um conjunto limitado de cenários e usualmente ignoram aqueles nos quais há múltiplas falhas associadas (LIU et al., 2013). Além disso, as regras adicionais ocupam memória nas tabelas dos dispositivos. As memórias utilizadas nessas tabelas (TCAM) constituem um recurso escasso nos atuais dispositivos de encaminhamento (MOSHREF et al., 2013; TRESTIAN; MUNTEAN; KATRINIS, 2013).

O modo *fail standalone* e os mecanismos de recuperação no plano de dados produzem resultados semelhantes de maneiras distintas. No primeiro caso, os dispositivos passam

a operar como comutadores Ethernet tradicionais (*standalone*), isto é, ignoram quaisquer regras que estejam em suas tabelas e recalculam rotas com base em endereços MAC alcançáveis. No segundo caso, os mecanismos de recuperação no plano de dados recalculam rotas com base em um Grafo Direcionado Acíclico (*GDA*), que representa a topologia da rede. Em suma, as regras originais são expressas como um GDA, que é recalculado quando algum evento causa remoção de um nó (LIU et al., 2013). A principal limitação com estas abordagens é que elas ignoram as garantias requisitadas por políticas de rede (incluindo balanceamento de carga, controle de acesso e latência).

3.2.2 Mecanismos Nativos de Recuperação

O protocolo OpenFlow (ONF, 2014) inclui, no lado do dispositivo, dois mecanismos que permitem reagir rapidamente sobre a perda de conectividade com uma instância de controlador (i) as conexões auxiliares e (ii) a lista de controladores de *backup*.

As conexões auxiliares com o controlador podem ser definidas, estabelecidas e mantidas de modo que permitam uma recuperação rápida pelo próprio dispositivo de encaminhamento nas situações de disrupção de enlaces. Tais conexões são realizadas pelo sistema operacional de rede (através do uso de *MultiPath TCP*) e evitam atrasos desnecessários na recuperação da conectividade. Além disto, elas podem ser estabelecidas sobre qualquer caminho na rede que alcance a instância de controlador (por exemplo, caminhos por enlace/vértice-disjuntos, caminhos mais curtos, caminhos mais longos). Não sendo possível o reestabelecimento, o dispositivo pode tentar iniciar a conexão com outra instância de controlador presente na rede. Para isso o mesmo recorre a uma lista de controladores.

A lista de controladores de *backup* define a ordem na qual o dispositivo tentará conectar a diferentes instâncias do controlador. Tal lista é configurada individualmente em cada dispositivo, possibilitando a existência de diferentes combinações tanto de ordem quanto do conjunto de instâncias de controlador presentes. Este mecanismo é utilizado quando todas as conexões à instância primária do controlador são perdidas (seja devido a uma grande falha de enlaces ou por falha na instância do controlador).

3.3 Posicionamento do Controlador

A organização física do plano de controle em redes SDN tem sido discutida em diversos trabalhos (HELLER; SHERWOOD; MCKEOWN, 2012; LEVIN et al., 2012; YEGANEH; TOOTOONCHIAN; GANJALI, 2013; PANDA et al., 2013). Essa decisão de projeto exerce influência sobre cada aspecto de um plano de controle dissociado, desde tolerância a falhas, a métricas de desempenho. Em redes de grande escala (por exemplo, WANs) o posicionamento é responsável por determinar limites fundamentais sobre a disponibilidade e o tempo de convergência. Além disso, têm implicações práticas no desenvolvimento dos serviços de rede (Pág. 25), determinando as situações onde as instâncias do controlador podem responder a eventos em tempo real, ou quando estas instâncias devem enviar instruções de encaminhamento antecipadamente para os dispositivos. Nesse sentido, a literatura tem proposto diferentes estratégias de posicionamento de controlador. Propostas recentes de posicionamento tratam do problema sob a ótica de duas métricas diferentes: (i) sobrecarga do controlador e (ii) disrupções da rede.

Sobrecarga do controlador

O trabalho de Bari et al. (BARI et al., 2013) estuda o aprovisionamento dinâmico de instâncias do controlador sobre a topologia de rede. Os autores assumem que a posição inicial é de algum modo pré-definida e os controladores são limitados a serem ligados ou desligados de acordo com a carga da rede particular, obtendo como único benefício o consumo energético.

Disrupções da rede

Em se tratando de disrupções, as propostas de posicionamento de controlador atuais buscam minimizar a probabilidade de desconexão. Zhang et al. (ZHANG; BEHESHTI; TATIPAMULA, 2011) definem um algoritmo de corte mínimo (Min-Cut) que especifica regiões de agrupamento dos dispositivos de encaminhamento e então posiciona uma instância de controlador em cada centroide das regiões. Por sua vez, Hu et al. (HU et al., 2013, 2012) escolhem a posição para as instâncias de controladores de tal forma que a chance de perda de conectividade é minimizada. Em ambas as estratégias, as conexões são definidas de acordo com o caminho mais curto entre as instâncias do controlador e os dispositivos de encaminhamento.

Recentemente, outras metodologias foram empregadas para analisar o problema do posicionamento de controlador. A abordagem formulada por Ros et al. (ROS; RUIZ, 2014) determina que cada nó de encaminhamento deve atender uma restrição de confiança. Tal restrição visa garantir a existência de caminhos operacionais que permitam a conexão com as instâncias do controlador com no mínimo uma probabilidade definida. Contudo, essa abordagem garante apenas o mínimo (ou seja, não faz o melhor possível com os recursos disponíveis).

Por fim, Hock et al. (HOCK et al., 2013) elabora um arcabouço multi-objetivos que oferece um conjunto de métricas (por exemplo, tolerância a falhas e latência máxima) para que seja computado o posicionamento mais adequado de acordo com as escolhas de um administrador de rede. No entanto, a otimização multi-objetivos apresenta uma diferença fundamental da otimização de um único objetivo. Ela encontra um conjunto de soluções onde o ganho em um objetivo significa o sacrifício do outro. Assim, necessita-se equacionar uma forma de comparar as aproximações obtidas. O ótimo, naquele trabalho, é definido através da técnica de Pareto, que retorna um conjunto de soluções para as quais não existem maneiras de melhorar algum critério sem piorar outro (OSYCZKA, 1981).

3.4 Limitações

No contexto de sobrevivência de redes SDN, podem ser identificadas duas importantes limitações: (i) a conectividade entre o controlador e dispositivos de encaminhamento; e (ii) a conectividade entre elementos de controle. No primeiro caso, o dispositivo de encaminhamento perde a capacidade de receber atualizações de regras de encaminhamento (VISSICCHIO; VANBEVER; BONAVENTURE, 2014). No segundo, o controle da rede pode ser prejudicado pela perda da consistência entre elementos particionados e da visão global da rede (PANDA et al., 2013).

Para fornecer garantias para tais limitações, é importante realizar o planejamento da infraestrutura do plano de controle. Mais especificamente, duas ações podem ser definidas, a pró-ativa e a reativa. Com base no estado-da-arte, foram apresentadas duas medidas, uma para cada ação, isto é, a estratégia de posicionamento do controlador e os

mecanismos de recuperação, cada qual com suas limitações, conforme a discussão que segue.

Em relação às estratégias de posicionamento do controlador (§3.3), a literatura propõe enfrentar dois desafios: as desconexões dispositivo-controlador e a sobrecarga das instâncias do controlador. Enquanto que para o primeiro desafio as estratégias propostas estão limitadas a usar apenas os caminhos mais curtos para estabelecer as conexões, no segundo, para lidar com a sobrecarga assumem que os controladores já foram colocados de forma otimizada e simplesmente determinam a atribuição de dispositivos para controladores. Uma solução adequada, no entanto deve fornecer um modo para recuperar de forma eficiente desconexões de dispositivos-controladores e otimizar o posicionamento das instâncias do controlador com objetivo de reduzir a chance de sobrecarga antes e depois de falhas.

Por fim, os mecanismos de recuperação atuais (§3.2) permitem que os dispositivos se recuperem de algumas falhas de enlace, dispositivo ou instância do controlador. Quando uma falha interrompe a conexão primária com um controlador, o dispositivo de encaminhamento pode usar conexões auxiliares pré-calculadas, isto é, caso o posicionamento tenha realizado a exploração correta da diversidade de caminhos na topologia da rede. E em caso da instância do controlador tornar-se indisponível, o dispositivo de encaminhamento pode seguir uma lista pré-definida, a fim de tentar se conectar a outras instâncias do controlador presentes na rede. Entretanto, a lógica de como definir conexões auxiliares e a lista de controladores de *backup* não estão atualmente especificadas.

4 SURVIVOR

Neste capítulo, é apresentada uma abordagem de posicionamento de controladores aplicada a redes WAN que visa garantir a sobrevivência do plano de controle da rede. Denominada *Survivor*, a abordagem busca aumentar a diversidade de caminhos entre dispositivo-controlador, evitar a sobrecarga do controlador e definir de forma inteligente os mecanismos de recuperação. Mais especificamente, na Seção 4.1 identifica-se os requisitos e propriedades da abordagem de posicionamento e apresenta-se uma visão geral da sua organização. A seguir, na Seção 4.2 formaliza-se o problema de posicionamento de controladores, através de um modelo de Programação Linear Inteira (PLI) que garante a solução ótima, e propõe-se uma estratégia de resolução heurística. Por fim, na Seção 4.3, um arcabouço é apresentado a fim de habilitar a criação de heurísticas que compõem as listas dos controladores de backup nos dispositivos de encaminhamento da rede.

4.1 Visão Geral

Como discutido anteriormente, a abordagem *Survivor* trata de três aspectos principais: a conectividade, a capacidade e a recuperação. Além disso, tais aspectos devem ser planejados para dois estados distintos da rede: pré e pós-disrupção. Em outras palavras, por um lado a rede deve ser configurada de forma ótima para operação normal, por outro também deve prever a reação, da melhor maneira possível, a eventos de disrupção. Para este fim, a abordagem é dividida em duas etapas. A primeira define o posicionamento de instâncias do controlador, enquanto a segunda especifica uma lista de controladores de backup para cada dispositivo na rede.

4.1.1 Posicionamento das instâncias de controle

A primeira parte requer encontrar posições ótimas para instâncias do controlador de tal forma que a *conectividade seja maximizada* e as *restrições de capacidade sejam satisfeitas*. A fim de maximizar a conectividade, o algoritmo escolhe posições que geram o maior número de caminhos de nó disjuntos entre os dispositivos de encaminhamento e a instância do controlador que eles se conectam.

Por sua vez, lidar com restrições de recursos requer alguma intuição sobre as demandas do dispositivo e capacidade do controlador. As estimativas para as demandas de dispositivos podem ser obtidas com as medições da rede (CUNHA et al., 2009), enquanto a capacidade de controladores pode ser inferida através da experimentação (TOOTOONCHIAN et al., 2012).

Na ocorrência de interrupções na rede, alguns dispositivos de encaminhamento podem ser transferidos de uma instância do controlador para outra. Esta situação pode fazer com

que um controlador fique sobrecarregado, gerando atrasos ou perdas nas solicitações. Para sustentar demandas excedentes, cada controlador tem uma percentagem de capacidade reservada como backup.

Este algoritmo é implementado de duas formas, primeiro utilizando Programação Linear Inteira (PLI), que garante a otimização e segundo, propõe-se uma abordagem de resolução heurística. O modelo ótimo é apresentado na Subseção 4.2.1 enquanto a estratégia heurística é detalhada na Subseção 4.2.2.

4.1.2 Composição da lista de controladores de backup

A segunda parte da abordagem *Survivor* consiste na definição de uma lista ordenada para cada dispositivo. Apesar de não haver nenhuma indicação no OpenFlow (ONF, 2014) de como se construir cada lista, o impacto da ordem escolhida é significativo. Conforme será demonstrado na Seção 5.3, a estabilidade da rede em um estado pós-disrupção é extremamente vulnerável à heurística empregada na definição da lista de backup.

Nesta dissertação são definidas três heurísticas para a composição da lista de backup: (i) heurística baseada em proximidade; (ii) heurística baseada em capacidade residual; e (iii) heurística baseada em confiança. Tais heurísticas visam representar as principais características consideradas durante o posicionamento. Apesar de não ser uma relação exaustiva, serve de base para o estudo mais aprofundado do procedimento de composição de listas.

Em linha com tal objetivo, será definido um arcabouço para a criação de heurísticas desse tipo. Isso é possível pois, conforme será observado, diferentes heurísticas para a composição de listas de backup seguem o mesmo principio: otimizar uma métrica local e gerar uma lista ordenada sem repetições. Assim, as heurísticas utilizadas neste trabalho foram implementadas utilizando um único arcabouço. Este arcabouço, as premissas relacionadas, e as heurísticas implementadas são descritas na Seção 4.3.

4.2 Estratégias para Posicionamento Sobrevivente

Esta seção apresenta duas estratégias de posicionamento de controladores. A primeira consiste em um modelo formalizado em Programação Linear Inteira. A segunda estratégia é uma heurística implementada por meio de um conjunto de algoritmos.

4.2.1 Posicionamento Ótimo com Modelo PLI (*SVVR-O*)

A seguir, são detalhadas as entradas, as variáveis, a saída, o objetivo e as restrições desse modelo.

Entrada. Tupla $I = \{G(N, L);\ C;\ U_i;\ R_i;\ \mathcal{DP}_{i,j};\ \mathcal{MC}_{n,m};\ maxHops;\ \alpha_i\}$ representa a entrada do PLI. A topologia física é representada por um grafo não-direcionado $G = (N, L)$, onde cada vértice $n \in N$ representa dispositivos de encaminhamento, enquanto arestas $(n, m) \in L$ representam enlaces bidirecionais. O conjunto de possíveis instâncias de controlador é definido por C. Seja $c \in C$ um controlador, sua capacidade é denotada por U_c, enquanto que a demanda requisitada por cada dispositivo n é representada por R_n. Além disso, $\mathcal{DP}_{n,m}$ fornece o número de caminhos totalmente disjuntos pré-calculados entre os nós n e m. O tamanho do menor caminho entre pares de nós n e m é definido por $\mathcal{MC}_{n,m}$ e $maxHops$ representa a distância máxima entre dispositivo-controlador. Finalmente, $\alpha_c : c \in C$ indica o percentual de capacidade de backup reservado para cada controlador.

Saída. A tupla $V = \{x_{i,j};\ y_{i,j};\ w_{i,j};\ aux_{c,n,m}\}$ representa as variáveis do PLI. Os

mapeamentos de dispositivos são definidos por $x_{n,c} \in \{0,1\}$; eles indicam se o dispositivo n é mapeado para o controlador c. O posicionamento dos controladores são denotados por $y_{c,n} \in \{0,1\}$; eles indicam se o controlador c está posicionado sobre o (ou seja, é conectado fisicamente ao) nó n. A variável $w_{c,n} \in \mathbb{N}$ conta (com o auxílio da entrada $\mathcal{DP}_{n,m}$) o número de caminhos disjuntos entre controlador c e dispositivo n; esta última contém 0 quando n não é mapeada para c. Por último, $aux_{c,n,m} \in \{0,1\}$ contém a multiplicação das variáveis x e y, que indica a existência de um mapeamento dispositivo-controlador.

Tabela 4.1: Resumo dos símbolos do modelo PLI.

Símbolo	Definição
N	Conjunto de nós da rede (ou seja, dispositivos de encaminhamento).
L	Conjunto de enlaces.
C	Conjunto de controladores.
U_c	Número máximo absoluto de solicitações que o controlador c pode manipular.
R_n	Número absoluto de requisições de cada dispositivo n.
$\mathcal{DP}_{n,m}$	Número de caminhos disjuntos entre pares de nós n e m.
$\mathcal{MC}_{n,m}$	Tamanho do menor caminho entre pares de nós n e m.
$maxHops$	Distância máxima entre dispositivo-controlador.
α_c	Percentual da capacidade reservada para backup no controlador c.
$x_{n,c} \in \{0,1\}$	Indica se o dispositivo n está mapeado para o controlador c.
$y_{c,n} \in \{0,1\}$	Indica se o controlador c está posicionado no nó n.
$w_{c,n} \in \mathbb{N}$	Indica o número de caminhos disjuntos entre dispositivo n e controlador c (0 se o dispositivo n não está mapeado para o controlador c).
$aux_{c,n,m} \in \{0,1\}$	Indica a multiplicação das variáveis x e y.

Objetivo. A estratégia proposta maximiza a conectividade entre dispositivos de encaminhamento e instâncias de controlador. Este objetivo é modelado na equação

$$\max \quad \frac{\sum_{c \in C} \sum_{n \in N} w_{c,n}}{|N|}, \tag{4.1}$$

que maximiza a média de caminhos disjuntos entre dispositivos e seu controlador.

Restrições. As restrições deste modelo de PLI podem ser divididas em três categorias: *relacionadas com o posicionamento*, *relacionadas com a capacidade*, e *relacionadas com a conectividade*.

As cinco primeiras restrições [Restrições (R1)–(R5)] são relacionadas com o posicionamento. Elas garantem a corretude para o posicionamento de instâncias do controlador na topologia e o mapeamento de dispositivos para instâncias do controlador. Restrição (R1) garante que, para todos os dispositivos (isto é, $\forall n \in N$), cada dispositivo n será controlado exatamente por um controlador c (isto é, $\sum_{c \in C} x_{n,c} = 1$).

$$\sum_{c \in C} x_{n,c} = 1 \qquad \forall n \in N. \text{ (R1)}$$

Restrição (R2) é formada por duas partes: (i) controladores atribuídos devem estar ativos (ou seja, $x_{n,c} \leq \sum_{m \in N} y_{c,m}$); e (ii) um controlador não pode ser colocado em mais de um local (isto é, $\sum_{m \in N} y_{c,m} \leq 1$).

$$x_{n,c} \leq \sum_{m \in N} y_{c,m} \leq 1 \qquad \forall c \in C, \forall n \in N. \text{ (R2)}$$

Restrição (R3) assegura que dois controladores não serão colocados no mesmo local, enquanto a Restrição (R4) garante que, se o controlador c é colocado no nó n (isto é, $y_{c,n} = 1$), então, o dispositivo n deve ser mapeado para o controlador c (ou seja, $x_{n,c} = 1$).

$$\sum_{c \in C} y_{c,n} \leq 1 \qquad \forall n \in N; \text{ (R3)}$$

$$y_{c,n} \leq x_{n,c} \qquad \forall n \in N, \forall c \in C. \text{ (R4)}$$

A última restrição relacionada ao posicionamento limita a distância entre dispositivo-controlador a um máximo definido por $maxHops$. Essa restrição é primeiro apresentada de forma simplificada, em Restrição (R5), e após definida de forma mais estrita para a implementação do modelo.

A Restrição (R5) compara o valor do menor caminho ($MC_{n,m}$) com o valor do número máximo de *hops* ($maxHops$). Essa restrição é executada para as possibilidades de mapeamento dispositivo-controlador ($x_{n,c} = 1, y_{c,m} = 1$), e ignorada para o resto (ou seja, $x_{n,c} = 0, y_{c,m} = 0$).

$$MC_{n,m} \leq maxHops \qquad \forall n, m \in N, \forall c \in C | x_{n,c} = 1, y_{c,m} = 1. \text{ (R5)}$$

A definição de restrições com base no valor de uma variável não é possível em um modelo linear, porque elas devem ser definidas de forma estática. Portanto, no modelo considerado a restrição foi implementada com equação quadrática $x_{n,c} \cdot y_{c,m} \cdot MC_{n,m} \leq maxHops$, linearizada a seguir.

$$aux_{c,n,m} \in \{0, 1\}; \text{ (R5.1)}$$

$$aux_{c,n,m} \leq x_{n,c}; \text{ (R5.2)}$$

$$aux_{c,n,m} \leq y_{c,m}; \text{ (R5.3)}$$

$$aux_{c,n,m} \geq x_{n,c} + y_{c,m} - 1; \text{ (R5.4)}$$

$$aux_{c,n,m} \cdot MC_{n,m} \leq maxHops; \text{ (R5.5)}$$

$$\forall n, m \in N, \forall c \in C.$$

A Equação (R5.1) define a variável $aux_{c,n,m}$ que representa a multiplicação entre as variáveis x e y. As Equações (R5.2)–(R5.4) definem a linearização da multiplicação. Por fim, a Equação (R5.5) re-define a restrição de forma linearizada.

A restrição *relacionada com a capacidade* [Restrição (R6)] garante que a capacidade do controlador não seja ultrapassada. Ela considera capacidades normais e de backup na mesma formulação. Especificamente, para todas as instâncias do controlador (isto é, $\forall c \in C$), a capacidade normal do controlador [isto é, $(1 - \alpha_c) \cdot U_c$] não pode ser ultrapassada pelo somatório de toda a demanda que lhe é atribuída (ou seja, $\sum_{n \in N} x_{n,c} R_n$).

$$\sum_{n \in N} x_{n,c} R_n \leq (1 - \alpha_c) \cdot U_c \qquad \forall c \in C. \text{ (R6)}$$

As duas restrições finais [Restrições (R7.1) e (R7.2)] são *relacionadas a conectividade*. Elas são utilizadas para contar o número de caminhos disjuntos em cada mapeamento controlador-dispositivo. Essas restrições são mais complicadas de interpretar, uma vez que elas trabalham em conjunto e dependem da função objetivo. As restrições representam dois casos. Quando o dispositivo n é mapeado para o controlador c ($x_{n,c} = 1$), $w_{c,n}$ é definido o limite superior pelos caminhos disjuntos entre m e o posicionamento de c (n), que é dado por $\mathcal{DP}_{n,m}$ [Restrição (R7.1)]. Em contraste, quando o dispositivo n não está mapeado para o controlador c ($x_{n,c} = 0$), $w_{c,n}$ tem o limite superior delimitado por 0 [Restrição (R7.2)]. Em suma, a Restrição (R7.1) é sempre definida por $\mathcal{DP}_{n,m}$, enquanto a Restrição (R7.2) é 0 ou infinito, dependendo de $x_{n,c}$.

$$w_{c,n} \leq \sum_{m \in N} y_{c,m} \cdot \mathcal{DP}_{n,m} \qquad \forall c \in C, \forall n \in N; \text{ (R7.1)}$$

$$w_{c,n} \leq x_{n,c} \cdot \infty \qquad \forall c \in C, \forall n \in N. \text{ (R7.2)}$$

Restrições (R7.1) e (R7.2) definem limites superiores para $w_{c,n}$, que são os valores exatos que a variável deve conter. Em outras palavras, $w_{c,n}$ deve ter sempre atribuído o seu limite superior. Isto é assegurado pela função objetivo [Equação (4.1)], que sempre tenta maximizar o valor para o $w_{c,n}$.

4.2.2 Posicionamento Utilizando Estratégia Heurística (*SVVR-H*)

Nesta subseção é apresentada a estratégia heurística proposta para o problema de posicionamento de controladores. A estratégia heurística foi implementada devido à característica combinatória do problema em questão, o que dificulta a realização de uma análise de um conjunto extenso de topologias e também impossibilita a avaliação de topologias maiores que aquelas utilizadas na avaliação realizada (Seção 5.1.1). O problema do posicionamento de controladores se enquadra na classe de problemas de localização de instalações (*facility location problem*) (ARYA et al., 2001), o qual é conhecidamente $\mathcal{NP}$-Difícil. Portanto, para solucionar este problema de modo eficiente, a estratégia heurística Survivor propõe utilizar técnicas de busca com base analítica (busca local). As soluções obtidas através desta heurística são sub-ótimas, mas são computadas dentro de um tempo factível para entradas de diferentes tamanhos. A seguir, é apresentada uma visão global do procedimento algorítmico e detalhados procedimentos específicos utilizados para construir uma solução. A Tabela 4.1 serve como referência para os símbolos utilizados nesta subseção.

Heurística Survivor

O Algoritmo 4.1 apresenta uma visão geral da estratégia proposta para o posicionamento de controladores em pseudocódigo; seus detalhes são explicados a seguir. Em essência, ao invés de explorar o espaço global de soluções em busca do posicionamento ótimo, a estratégia executa uma busca local de forma a encontrar uma boa solução de posicionamento, de acordo com um conjunto de heurísticas definidas. No início (Linha 1) a estratégia calcula a quantidade mínima necessária de instâncias de controle, dada em função da capacidade e da demanda dos dispositivos de encaminhamento. Seguindo, o algoritmo é composto por *três etapas principais*.

A primeira etapa (Linha 2) consiste em identificar as melhores posições para a instalação das instâncias de controle de acordo com as características topológicas. Esse procedi-

mento será descrito em mais detalhes no Algoritmo 4.2. A segunda etapa (Linha 6) define uma estratégia para estabelecer as conexões dos dispositivos de encaminhamento com as instâncias do controlador dispostas sobre a topologia. Similarmente à etapa anterior, esse procedimento será descrito em mais detalhes no Algoritmo 4.3. Por fim, é possível que dispositivos de encaminhamento permaneçam sem conectividade com as instâncias de controle previstas inicialmente (Linha 1). Isso ocorre em função de restrições topológicas (G) combinadas com o resultado do posicionamento efetuado na primeira etapa. A terceira etapa (Linhas 7–9) resolve esse impasse adicionando uma nova instância do controlador à rede e estabelecendo as conexões que faltam.

Diferente da solução ótima, a heurística pode eventualmente retornar uma solução que implica custo maior para a infraestrutura da rede. Isso acontece porque as etapas de posicionamento e de definição de conexões não garantem resultados ótimos. Tal simplificação foi necessária para permitir a execução do algoritmo em tempo factível. A seguir, os principais procedimentos da heurística são apresentados, isto é: (i) o posicionamento das instâncias do controlador sobre a topologia e (ii) a definição das conexões entre dispositivos e controladores.

Algoritmo 4.1: Pseudocódigo da solução Heurística Survivor

Entrada: G, C, U_c, R_n, $\mathcal{DP}_{n,m}$, $\mathcal{MC}_{n,m}$, α_c, $maxHops$
Saída: $x_{n,c}$, $y_{c,n}$

```
1  qtdeCtrls ← ⌈(|N| · R_n)/(U_c · (1 − α_c))⌉ ;   //quantidade mínima de controladores
   /* Decisão do posicionamento dos controladores:                    */
2  y_{c,n} ← procedimento_de_posicionamento(G, DP_{n,m}, maxHops, qtdeCtrls)
   /* Definição das conexões:                                         */
3  modoDeOrdenacao ← ∅;
4  modoDeSelecao ← ∅;
5  repeat
6  |  x_{n,c} ← procedimento_de_definicao_das_conexoes(y_{c,n});
7  |  if existem_dispositivos_sem_conexao(x_{n,c}) then
8  |  |  y_{c,n} ← procedimento_de_posicionamento(G, DP_{n,m}, 0, 1)
9  |  end
10 until existem_dispositivos_sem_conexao(x_{n,c}) ;
11 return x_{n,c}, y_{c,n}
```

Procedimento de Posicionamento de Controladores

A definição da posição das instâncias de controle é obtida por um algoritmo guloso composto por dois passos: (i) determinar a ordem em que o conjunto N deve ser organizado, e (ii) decidir o método de seleção do subconjunto de nós sobre o conjunto N ordenado. O procedimento é apresentado em formato de pseudocódigo no Algoritmo 4.2. O método de ordenação (Linha 1) do conjunto N é definido da seguinte maneira: para cada nó ($n \in N$) é executada a soma de dois valores, quais sejam, número de caminhos disjuntos e *betweenness* [$\mathcal{DP}_{n,m}$ + `betweenness`(G, n)]. Após, o conjunto é ordenado de forma decrescente pelo valor resultante. O valor da propriedade *betweenness* referente a cada nó da topologia foi acrescido como critério para evitar empate em relação ao valor de entrada dado por $\mathcal{DP}_{n,m}$. Tais propriedades foram escolhidas para determinar a importância do nó para a conectividade geral da topologia.

Algoritmo 4.2: Pseudocódigo do procedimento que define posicionamento dos controladores

```
1 modoDeOrdenacao ← ordena o conjunto N de forma decrescente pelo valor
  definido por [DP_{n,m} + betweenness(G, n)] para cada n;
2 if grau_medio_topologia(G) ≤ maxHops then
3 |   modoDeSelecao ← seleciona nós evitando a formação de clusters através da
  |   escolha de nós não vizinhos;
4 else
5 |   modoDeSelecao ← seleciona nós com o maior valor, seguindo a ordem
  |   resultante do modoDeOrdenacao;
6 end
7 y_{c,n} ← ativa_instancias_de_controle(modoDeSelecao, qtdeCtrls);
8 return y_{c,n}
```

A decisão de seleção dos nós que devem receber as instâncias de controle (Linha 2–6) é guiada através da análise da propriedade grau médio, obtida a partir da análise da topologia da rede. O valor obtido do grau médio é então comparado com o valor de entrada definido para $maxHops$, resultando dois casos possíveis:

1. Quando o valor do grau médio é inferior ou igual ao valor de $maxHops$ (Linha 2), isto significa que as características topológicas da rede não oferecem uma conectividade favorável. Em outras palavras, tendem a aumentar o número de *hops* deixando de respeitar a Restrição (R5) (§4.2.1) caso as instâncias do controlador não sejam posicionadas de uma maneira mais dispersa sobre a topologia. Nessa situação, sob o conjunto ordenado N, a seleção dos nós ocorre conforme a ordem definida, porém não permite a escolha de nós vizinhos daqueles que já foram escolhidos. Este método de seleção evita o agrupamento de controladores sobre a topologia.

2. Quando o grau médio da topologia é maior que o valor de $maxHops$ (Linha 5), isso é um indício de que a rede apresenta uma boa conectividade e com uma probabilidade alta, mesmo com instâncias de controle posicionadas relativamente próximas umas das outras, será possível respeitar a Restrição (R5) (§4.2.1).

Por fim, na Linha 7 é realizada a chamada do procedimento para ativar a quantidade mínima de controladores, de acordo com o método de seleção definido sob o conjunto N previamente ordenado.

Este procedimento atua em duas etapas distintas (1 e 3) no Algoritmo 4.1. A etapa 3 utiliza uma configuração particular de parâmetros: $maxHops = 0$ e $qtdeCtrls = 1$. Essa configuração é utilizada para acrescentar apenas uma instância de controle e, conforme observado no Algoritmo 4.2, busca selecionar o nó com a maior conectividade.

Procedimento de Definição das Conexões

A estratégia heurística, no que se refere à etapa de definição das conexões dispositivo-controlador, apresenta três passos: estabelece (i) conexões com vizinhos; (ii) conexões com outros dispositivos próximos e de modo balanceado; e (iii) conexões adicionais. O Algoritmo 4.3 provê um pseudocódigo do procedimento.

Assumindo que as instâncias de controle já foram posicionadas (Algoritmo 4.2), a lógica do procedimento é descrita como segue. No início (Linhas 1–3), são computadas três variáveis utilizadas ao longo do procedimento de modo a guiar os passos seguintes.

Primeiro, é calculada a quantidade de conexões ideal ($qtdeConPorCtrls$), considerando uma distribuição uniforme de dispositivos por controlador. A seguir, é obtida a quantidade adicional de conexões ($qtdeExtraCon$) que devem ser estabelecidas, ou seja, além daquelas definidas na distribuição uniforme. Por fim, é calculado o limite de conexões que cada instância do controlador pode suportar ($qtdeMaxConPorCtrl$).

Algoritmo 4.3: Pseudocódigo do procedimento que define conexões dispositivos – controlador

```
1  qtdeConPorCtrls ← (|N|/qtdeCtrls) ;          //conexões por controlador ideal
2  qtdeExtraCon ← (|N| % qtdeCtrls) ;                          //conexões extras
3  qtdeMaxConPorCtrl ← ⌊(U_c * (1 − α_c))/R_n⌋ ;        //limite de conexões por
   controlador

   /*  Passo 1:  conexões diretas (balanceado)                              */
4  x_{n,c} ← conecta_vizinhos(y_{c,n})
   /*  Passo 2:  dispositivos ainda sem conexão (balanceado)                */
5  if existem_dispositivos_sem_conexao(x_{n,c}) then
6  |  modoDeOrdenacao ← ordena o conjunto N de forma decrescente pelo valor
   |  definido por [DP_{n,m} + betweenness(G, n)] para cada n;
7  |  modoDeSelecao ← seleciona nós com o maior valor, seguindo a ordem
   |  resultante do modoDeOrdenacao;
8  |  x_{n,c} ← conecta_dispositivos(modoDeSelecao, qtdeConPorCtrls,
   |  maxHops);
9  else
10 |  return x_{n,c}
11 end
   /*  Passo 3:  conexões adicionais                                        */
12 if qtdeExtraCon > 0 or existem_dispositivos_sem_conexao(x_{n,c}) then
13 |  modoDeOrdenacao ← ordena o conjunto N de forma crescente pelo valor
   |  definido por MC_{n,m};
14 |  modoDeSelecao ← seleciona nós com o menor valor, seguindo a ordem
   |  resultante do modoDeOrdenacao;
15 |  x_{n,c} ← conecta_dispositivos(modoDeSelecao, qtdeMaxConPorCtrl,
   |  maxHops);
16 else
17 |  return x_{n,c}
18 end
19 return x_{n,c}
```

As primeiras conexões estabelecidas (Linha 4) são dos dispositivos de encaminhamento mais próximos (nós vizinhos) de cada instância do controlador. A seguir (Linhas 6–8) é realizada uma distribuição equilibrada das conexões para cada controlador, (definida por $qtdeConPorCtrls$) priorizando dispositivos com maior conectividade e respeitando a Restrição R5.

Até este instante, a quantidade de conexões dispositivos-controlador foi limitada para uma quantidade máxima, balanceada, definida por $qtdeConPorCtrls$. Então, na situação de não ser factível a mesma quantidade de conexões para todos os controladores (ou seja,

existe valor > 0 na variável $qtdeExtraCon$, Linha 2) ou porque ainda existem dispositivos de encaminhamento sem conectividade, nas Linhas 12–15 busca-se efetuar a conexão desses dispositivos através do menor caminho para uma instância de controle. Esse procedimento respeita sempre o limite de capacidade de cada instância do controlador ($qtdeMaxConPorCtrl$) e a restrição de distância (Restrição R5). Por fim, na Linha 15 é realizada a chamada do procedimento de conexão dispositivos-controlador utilizando o parâmetro $qtdeMaxConPorCtrl$ de modo a liberar a capacidade máxima de conexões por controlador, para tentar concretizar as conexões que faltaram ser estabelecidas.

Vale relembrar que, se ainda permanecerem dispositivos desconectados após todos esses passos, o Algoritmo 4.1 irá adicionar uma nova instância de controle (Algoritmo 4.2) e reexecutar o procedimento de definição de conexões. As particularidades dessa etapa adicional foram descritas anteriormente.

4.3 Heurísticas para Definição da Lista de Controladores de Backup

Esta seção primeiro apresenta as definições (§4.3.1) utilizadas como base para os algoritmos. Em seguida, descreve-se um arcabouço (§4.3.2) para a criação de heurísticas que compõem as listas de controladores de backup, eliminando a necessidade de determinar manualmente essa lista. Ao final, apresenta-se três heurísticas (§4.3.3) como prova de conceito, uma baseada em proximidade, outra na capacidade residual e a última baseada em confiança.

4.3.1 Definições

A seguir, são detalhadas as premissas consideradas no desenvolvimento deste arcabouço e a notação utilizada para sua especificação.

Premissas

O arcabouço adota três premissas. Em primeiro lugar, as falhas do controlador são consideradas as piores ocorrências de desconexões. Em segundo lugar, assume-se que controladores falham de modo independente; isso pode ser presumido na prática, uma vez que as arquiteturas de controle atuais implementam mecanismos para ignorar instâncias de controlador quando elas falham, protegendo as restantes (KOPONEN et al., 2010; YEGANEH; GANJALI, 2012). Em terceiro lugar, o índice que representa a posição de um controlador na lista de cada dispositivo de encaminhamento é considerado independente, de forma que todos dispositivos de encaminhamento seguirão sua lista ordenada diante de um cenário de falha.

Notação

Apesar da maioria das notações utilizadas nos algoritmos que seguem serem intuitivas, algumas considerações devem ser feitas. Particularmente, os símbolos de *colchetes* ($[.]$), *estrela* ($*$), e *sinal de adição* ($+$) têm significados especiais. Os *colchetes* são usados para definir os índices em listas ordenadas, ou seja, $[i]$ indica uma posição anterior que $[j]$ para todo $i < j$. A *estrela* é usada para indicar que uma dada operação é executada para todos os índices de uma lista. Por fim, o *sinal de adição* é usado para indicar que a comparação é realizada para todos os índices de uma lista, sujeitos a pelo menos uma comparação ser verdadeira.

4.3.2 Arcabouço

O pseudo-algoritmo da estrutura é descrito no Algoritmo 4.4. Ele começa por considerar todas as instâncias de controlador que foram colocadas na topologia (Linha 3). Devido à primeira e segunda premissas, os controladores são avaliados independentemente. Em seguida, para cada índice na lista (Linha 4), é inicializada uma lista de candidatos composta por todos os controladores, exceto o que está sendo avaliado (Linha 5). Da mesma forma que os controladores, os índices são avaliados de forma independente devido à terceira premissa.

Os passos seguintes selecionam todos os dispositivos conectados ao controlador avaliado e otimizam o índice i em sua lista (Linhas 6-10). Três operações são realizadas: *selecionar o melhor local* (Linha 8), *definir o índice da lista* (Linha 9), e *atualizar lista de candidatos* (Linha 10). A operação de *seleção do melhor local* consiste em avaliar todos os candidatos possíveis, dados pelo conjunto S (isto é, $S \leftarrow K \setminus B_n[*]$), e em seguida, selecionando o melhor entre eles de acordo com uma determinada métrica (ou seja, $\varphi(n, S) = b$).

Definir o índice da lista é apenas uma operação de atribuição. Em contraste, a operação de *atualização de lista de candidatos* segue uma especificação (ou seja, δ) para atualizar as informações sobre os controladores da lista de candidatos. É importante perceber que as operações realizadas pelo procedimento de atualização são temporárias, já que elas são realizados sob o conjunto K que, por sua vez, é reinicializado na Linha 5. As operações mencionadas usam dois procedimentos genéricos: $\varphi(n, S)$ (que encontra o mínimo local) e δ (que atualiza os candidatos). Estes procedimentos têm significados específicos para cada heurística e serão exemplificados a seguir.

Algoritmo 4.4: Arcabouço Heurístico para composição de listas de controladores de backup

```
 1: B_n[*] ← ∅, ∀n ∈ N
 2: TamanhoLista ← max(|B_*|)
 3: for c ∈ C : y_{c,+} = 1 do //considera somente controladores posicionados
 4:    for i ← 1 upto TamanhoLista do
 5:       K ← C \ {c} //inicializa a lista de candidatos
 6:       for n ∈ N : x_{n,c} = 1 do //considera somente dispositivos conectados
 7:          S ← K \ B_n[*] //discarta os já utilizados
 8:          selecionar b ∈ S : φ(n, S) = b //seleciona o ótimo local
 9:          B_n[i] ← b //define b na i-ésima posição da lista n
10:          atualizar K segundo a regra δ
11:       end for
12:    end for
13: end for
14: return B_n
```

4.3.3 Heurísticas Propostas

Nesta subseção são apresentadas três heurísticas propostas para a definição da lista de controladores de backup. É detalhada cada heurística específica tomando como base o arcabouço anteriormente definido (§4.3.2).

Heurística Baseada em Proximidade

Esta heurística tenta selecionar as instâncias de controlador mais próximas para usar como backup (em termos de atraso ou saltos). Para implementar esta heurística, é necessário definir $\varphi(n, S)$ como um procedimento que leva o dispositivo n e o conjunto de controladores candidatos válidos S, e retorna a instância mais próxima. Isto pode ser obtido através da implementação de um algoritmo para encontrar o menor caminho em $\varphi(n, S)$. A regra de atualização não é necessária uma vez que os controladores não mudam de posição.

Heurística Baseada em Capacidade Residual

A heurística baseada em capacidade residual visa contabilizar o consumo de recursos ao selecionar instâncias do controlador. Mais especificamente, ela *seleciona* a instância do controlador que possui a maior capacidade residual e *atualiza* esta instância de acordo com a demanda excedente. Assim, $\varphi(n, S)$ é uma busca linear para a capacidade máxima residual, enquanto δ acrescenta a demanda do dispositivo n para o controlador n no conjunto k. Como mencionado anteriormente, as operações realizadas pelo procedimento de atualização são temporárias.

Heurística Baseada em Confiança

A proposta desta heurística é buscar as instâncias de controlador que representam maior confiança à rede em situações de falha dos equipamentos, isto é, considera a quantidade de caminhos disjuntos por vértice. De forma similar às anteriores, é preciso definir $\varphi(n, S)$ como um procedimento que leva o dispositivo n e o conjunto de controladores candidatos válidos S, e retorna a instância de controle com maior confiabilidade. Para obter isso é necessário a implementação de um algoritmo para encontrar o número de caminhos totalmente disjuntos em $\varphi(n, S)$. Neste caso, a regra de atualização não é requerida dado que o número de caminhos disjuntos entre controlador-dispositivos não muda.

5 AVALIAÇÃO

Este capítulo apresenta a avaliação de desempenho da abordagem *Survivor* com o estado-da-arte em posicionamento sobrevivente de controladores. Os experimentos realizados visam determinar: (i) a resistência a disrupções; (ii) a distribuição da carga das instâncias de controlador; e (iii) o tempo de computação. Os experimentos foram realizados em um ambiente simulado, utilizando recursos computacionais da nuvem Microsoft Azure[1]. Foram utilizadas duas instâncias A9 (16 cores Intel Xeon E5-2670 @ 2.60GHz, 112GB de RAM, 6TB de disco).

O restante deste capítulo é organizado em três seções, como segue. Primeiro, na Seção 5.1 apresenta-se a metodologia utilizada na avaliação. Após, na Seção 5.2 descreve-se o simulador implementado. Por fim, na Seção 5.3 discutem-se os experimentos e seus resultados.

5.1 Metodologia

Esta seção apresenta a metodologia empregada para comparar a abordagem *Survivor* com estado-da-arte no posicionamento sobrevivente do controlador. Esta seção foi separada em três partes. Primeiro, na Seção 5.1.1 apresenta-se a configuração da carga de trabalho, após, na Seção 5.1.2 é descrito o método de comparação com o estado-da-arte e, por fim, na Seção 5.1.3 as métricas utilizadas.

5.1.1 Carga de Trabalho

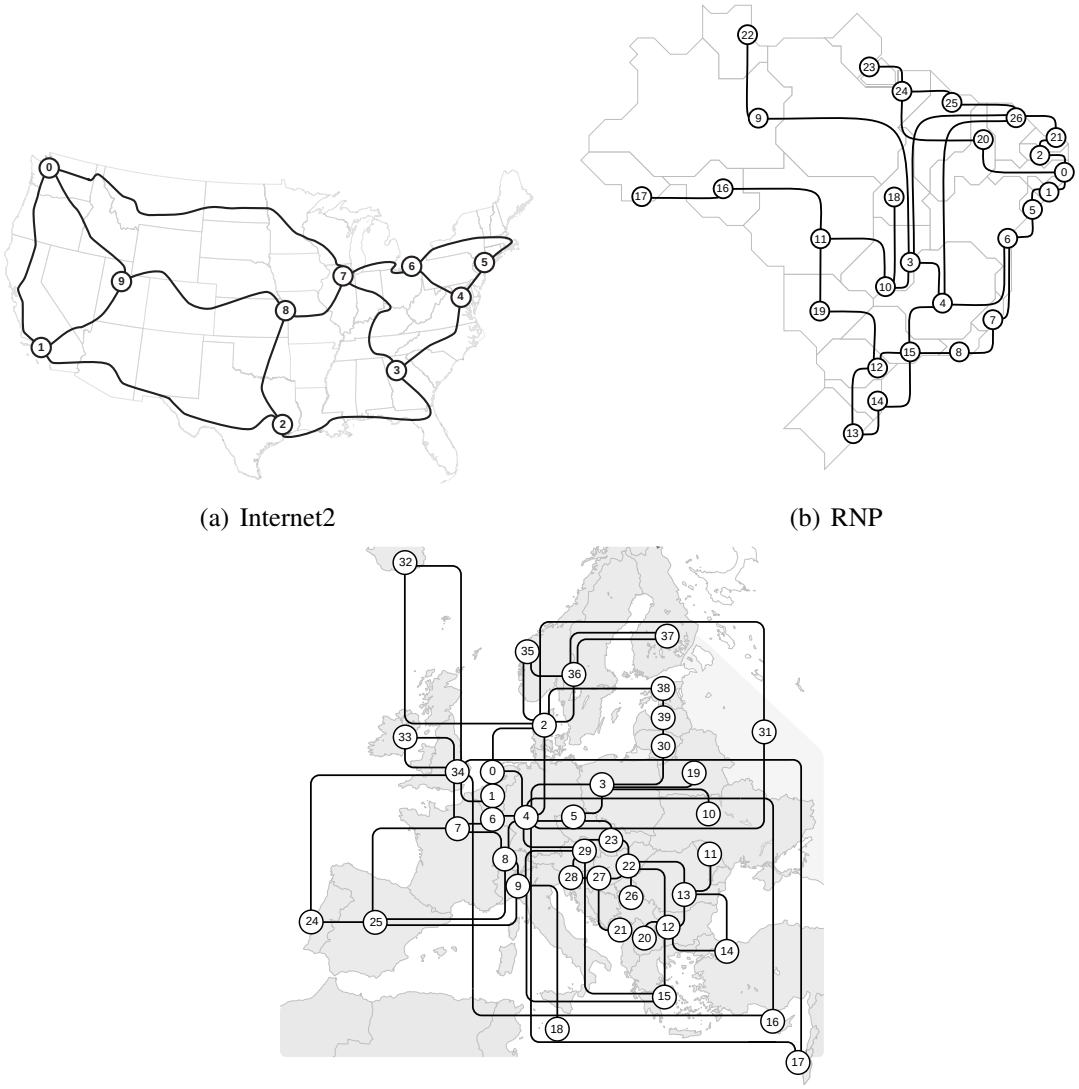
A avaliação realizada toma como entrada o projeto de rede resultante por cada estratégia de posicionamento e simula, sobre cada projeto, múltiplos cenários de falhas. Para tanto, três topologias WAN diferentes foram consideradas[2]: Internet2 (10 nós, 15 enlaces), RNP (27 nós, 33 enlaces) e GÉANT (40 nós, 61 enlaces). A Figura 5.1 ilustra as três topologias recém mencionadas.

Este ambiente pressupõe que os nós sejam dispositivos compatíveis com o protocolo OpenFlow e que os controladores podem ser posicionados sobre qualquer nó. Além disso, as requisições de instalação de novos fluxos são utilizadas para definir a capacidade de cada controlador e as demandas impostas pelos dispositivos de encaminhamento. Baseados em estudos da literatura, todos os controladores têm a mesma capacidade, 1800 K requisições/s (TOOTOONCHIAN et al., 2012), ao passo que cada dispositivo de encaminhamento, por sua vez gera 200 K requisições/s (CUNHA et al., 2009).

[1] https://azure.microsoft.com

[2] O mapeamento das topologias foi obtido em Maio, 2014, através do repositório Topology Zoo: `http://www.topology-zoo.org/`

Finalmente, as configurações da percentagem de recursos reservados como backup utilizadas foram de 20% e 30%, e a distância máxima utilizada entre dispositivos e a instância de controle foi definida em três *hops* (HELLER; SHERWOOD; MCKEOWN, 2012).



(a) Internet2

(b) RNP

(c) GÉANT

Figura 5.1: Topologias consideradas na experimentação.

5.1.2 Método de Comparação

A abordagem Survivor (estratégias SVVR-O e SVVR-H) é comparada com duas versões da estratégia desenvolvida por Zhang et al. (ZHANG; BEHESHTI; TATIPAMULA, 2011). A estratégia de Zhang et al. foi escolhida por se tratar da proposta mais próxima da abordagem Survivor. Ela busca definir um plano de controle resiliente através de uma estratégia de posicionamento do controlador. A primeira é a versão original do algoritmo, a qual é composta por três etapas: (i) definir a quantidade de partições desejada; (ii) identificar as partições na topologia da rede com cortes mínimos; e (iii) atribuir um controlador por região no ponto da rede que possui os caminhos mais curtos para todos os dispositivos na mesma região. Esta estratégia é denotada *MCC*, acrônimo de *MinCut-Centroid*.

A segunda versão trata-se de uma extensão natural desse algoritmo, denominada *MCC+*. Tal versão consiste em considerar todos os caminhos disponíveis entre controlador e dispositivos. Os passos (i), (ii) e (iii) do algoritmo são executados normalmente, mas depois que as instâncias de controlador estão posicionadas, as conexões são estabelecidas utilizando todos os caminhos disjuntos para os dispositivos.

5.1.3 Métricas

Esta seção descreve as métricas utilizadas para medir resiliência à disrupções e sobrecarga dos projetos do plano de controle resultantes de cada algoritmo, e sobre cada uma das topologias consideradas. A resiliência representa a capacidade dos projetos de rede avaliados em suportar a perda de conectividade com controladores (focado na conexão dispositivo-controlador), face à disrupções em enlaces da rede. Por sua vez, a sobrecarga indica como a carga das instâncias de controlador são distribuídas durante a operação normal e após a ocorrência de disrupções. Quatro métricas são utilizadas: as duas primeiras quantificam resiliência, enquanto as duas últimas determinam a sobrecarga.

Para equidade de comparação, a primeira avaliação considera a mesma *equação de resiliência* utilizada pelos autores de MCC (ZHANG; BEHESHTI; TATIPAMULA, 2011). A equação mede a probabilidade de perda de conectividade em um cenário de falha uniforme. Ela é dada da seguinte forma. Seja P_{ab} o(s) caminho(s) entre o dispositivo a e b, e assumindo $P_E[e]$ como a probabilidade de falha para o enlace e e $P_E[n]$ a probabilidade de falha para o dispositivo n. A probabilidade de ocorrer uma falha na comunicação entre os dispositivos a e b é definida pelo resultado da Equação 5.1. Esta equação mede a probabilidade de perda de conectividade em um cenário de falha uniforme.

$$1 - \prod_{e,n \in P_{ab}} (1 - P_E[e]) \cdot (1 - P_E[n]). \tag{5.1}$$

A segunda métrica utilizada é o cardinal de conectividade de enlaces (BAGGA et al., 1993). Ela é utilizada para calcular a porcentagem de componentes desconectados dado todos os cenários possíveis de falha. Essa métrica enumera os conjuntos de corte de arestas de cardinalidade i, além de identificar os casos onde a rede apresenta a maior vulnerabilidade, ou seja, permite identificar o pior caso de desconexões entre controlador-dispositivos.

A terceira métrica conta o *número de cenários com sobrecarga* em todos os cenários de falhas possíveis. Tais cenários são gerados a partir da falha de uma instância de controle por vez, seguida da ativação das estratégias de recuperação (§4.3.3). Finalmente, a quarta métrica mostra a *distribuição de carga* para cada uma das instâncias do controlador.

5.2 Simulador

O simulador é responsável por executar a chamada para as estratégias de planejamento do plano de controle e a avaliação do posicionamento resultante de cada uma das mesmas (§5.1.2). A maior parte do código foi implementado utilizando a linguagem Python. A única exceção é o modelo PLI, que foi criado utilizando a linguagem AMPL[3](*A Mathematical Programming Language*).

[3]http://www.ampl.com

Algoritmo 5.1: Pseudocódigo do simulador

```
   Entrada: G(N, L)                                   //grafo da topologia
   Entrada: A                                  //algoritmo de posicionamento

 1 P ← ∅;                              //posicionamento resultante do algoritmo
 2 if A is SVVR-O then
 3 |   P ← chama cplex com Alg. 4.2.1 ;          //estratégia Survivor ótimo
 4 end
 5 if A is SVVR-H then
 6 |   P ← Alg. 4.1 ;                          //estratégia Survivor heurística
 7 end
 8 if A is MCC then
 9 |   P ← Alg. (Zhang et al., 2011) ;                    //estratégia MCC
10 end
11 if A is MCC+ then
12 |   P ← Alg. (Zhang et al., 2011) modificado (§ 5.1.2) ;   //estratégia MCC+
13 end
14 Distribuição de carga(G, P) ;                          //operação normal
15 Prob ← {0, 0.1, 0.2, ..., 0.9, 1} ;   //lista com valores de probabilidade
16 for p ∈ Prob do
17 |   Calcula Equação de Resiliência(p, G, P) ;        //(Zhang et al., 2011)
18 end
19 for k ← 1 upto Número de Falhas Simultâneas do
20 |   F ← Combinações(|L|, k) ;              //falhas possíveis de enlaces
21 |   for f ∈ F do
22 |   |   Calcula Cardinal de conectividade de enlaces(f, G, P);
23 |   end
24 end
25 C ← Controladores(P);
26 F ← Combinações(|C|, 1) ;             //falhas possíveis de controladores
27 for f ∈ F do
28 |   Calcula Distribuição de carga(f, G, P);
29 |   Calcula Quantidade de controladores sobrecarregados(f, G, P);
30 end
```

Para realizar os experimentos, o simulador recebe como entrada o grafo da topologia de rede G e o identificador A do algoritmo de posicionamento de controladores a ser executado. A seguir são apresentadas as três etapas do simulador: (i) planejamento do plano de controle; (ii) avaliação da distribuição de carga; e (iii) simulação de disrupções.

O Algoritmo 5.1 mostra o pseudocódigo do simulador. A primeira etapa (Linhas 2–13) consiste na execução da estratégia de posicionamento de controladores. O simulador seleciona o algoritmo correto e executa o código em AMPL (Linha 3) ou Python (Linhas 6, 9 e 12), de acordo com a necessidade. No caso do modelo em AMPL (Linha 3), o simulador realiza uma chamada externa para o *solver CPLEX Optimization Studio* versão 12.5[4]. Ao final da execução desta etapa, o resultado do planejamento é armazenado na

[4]http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/

variável P.

Nas etapas seguintes são realizadas as avaliações sob o planejamento resultante (P) utilizando as métricas apresentadas anteriormente (§5.1.3). A segunda etapa (Linha 14) analisa a distribuição de carga em cada instância do controlador quando a rede encontra-se em operação normal, ou seja, sem falhas. Para isto, a métrica recebe como entrada o grafo da topologia G e o resultado do planejamento do plano de controle da rede P.

Por fim, na terceira etapa (Linhas 19–30) são simuladas as disrupções, em particular nos enlaces e nas instâncias de controlador. No primeiro passo (Linhas 15–18), são calculados os valores para a Equação (5.1) sob diferentes probabilidades de falha em nós e enlaces da rede. No segundo passo (Linhas 19–24), são simuladas todas as possíveis falhas de enlaces, considerando a ocorrência de falhas de até k enlaces consecutivos. O terceiro passo (Linhas 26–30) avalia o comportamento da rede na ocorrência de falhas nas instâncias do controlador. Nesta última avaliação, são computadas duas métricas: (i) distribuição de carga da rede após convergência mediante as diferentes estratégias de recuperação possíveis; e (ii) quantidade de controladores sobrecarregados em cada situação.

5.3 Resultados da Avaliação

Esta seção compara o Survivor (SVVR-O e SVVR-H) com a literatura (MCC/MCC+). Para facilitar a exposição, o SVVR-O é ilustrado com linhas contínuas em preto, o SVVR-H linhas tracejadas em azul, enquanto o MCC (ou o MCC+) com linhas pontilhadas em verde. Primeiro são discutidas as diferenças entre os posicionamentos ótimo e heurístico. Após, são analisadas as métricas de resiliência e sobrecarga apresentadas anteriormente (Seção 5.1.3).

A dificuldade do problema justifica o estudo de heurísticas

Primeiro, é realizada uma análise do desempenho das estratégias Survivor (SVVR-O e SVVR-H). Foram considerados o tempo e a quantidade de memória RAM necessários para computar a solução em cada experimento. Os resultados desta análise encontram-se na Tabela 5.1 e são discutidos a seguir.

As duas estratégias apresentam comportamentos diferentes no que diz respeito a ambas métricas. Enquanto a estratégia heurística executa em um tempo inferior a 3 segundos em todos experimentos, a estratégia ótima requer um tempo muito maior de processamento de acordo com o tamanho da topologia considerada. Além disso, se observa que o fator relacionado ao percentual de backup tem grande influência no tempo de solução do problema. Isso se deve ao fato de que a solução pode ser mais ou menos restrita, de acordo com esse fator. Observa-se também que a diferença no tempo de execução para encontrar a solução entre as estratégias acontece porque, no caso da estratégia ótima, é exigida a solução de um problema de natureza combinatória, pertencente à classe de problemas de localização de instalações, que é $\mathcal{NP}$-Difícil (ARYA et al., 2001).

Esta diferença de comportamento, entre as duas estratégias, é observada também na métrica de consumo de memória RAM. A estratégia heurística apresenta um consumo inferior a 100 MB independentemente do tamanho da topologia. Em contraste, na estratégia ótima há um crescimento no consumo da memória de acordo com o tamanho da instância de entrada. Em particular, a maior instância (GÉANT-30%) demandou mais de 16h de computação e um consumo superior a 100 GB de memória RAM para encontrar a solução do problema. Além disso, se adicionado um nó extra à topologia GÉANT o tempo de computação ultrapassa duas semanas e requer mais de 6 TB de disco para processar a

solução do problema.

Os resultados demonstram a dificuldade do problema e a necessidade da estratégia heurística proposta. O algoritmo heurístico é capaz de elaborar soluções factíveis para o problema mesmo variando o tamanho das topologias de rede consideradas, mantendo um bom desempenho em sua execução.

Tabela 5.1: Desempenho das estratégias implementadas com base na abordagem Survivor.

	Tempo (s)					
Topologias (% Bkp)	I2 (20%)	I2 (30%)	RNP (20%)	RNP (30%)	GÉANT (20%)	GÉANT (30%)
SVVR-O	3.37	4.36	94.75	125.28	12231.89	59015.29
SVVR-H	< 2	< 3	< 3	< 3	< 3	< 3
	Memória (MB)					
Topologias (% Bkp)	I2 (20%)	I2 (30%)	RNP (20%)	RNP (30%)	GÉANT (20%)	GÉANT (30%)
SVVR-O	$> 10^2$	$> 10^2$	$> 10^3$	$> 10^3$	$> 10^6$	$> 10^6$
SVVR-H	$< 10^2$	$< 10^2$	$< 10^2$	$< 10^2$	$< 10^2$	$< 10^2$

O posicionamento sub-ótimo da heurística, em alguns casos, acarreta custo adicional no número de controladores

Conforme apresentado no capítulo anterior, na Seção 4.2.2, a estratégia heurística pode eventualmente retornar uma solução que implica uma quantidade maior de instâncias do controlador em operação na rede. Tal situação decorre de decisões heurísticas implementadas no algoritmo: ao contrário da estratégia ótima, na heurística as etapas de posicionamento e de definição das conexões não preveem a exploração de todo o espaço de soluções.

Tabela 5.2: Quantidade de instâncias de controle alocadas em cada estratégia.

% Backup	20%			30%		
Topologias	I2	RNP	GÉANT	I2	RNP	GÉANT
SVVR-O	2	5	6	2	5	7
SVVR-H	2	5	7	2	6	8
MCC	2	5	6	2	5	7

A Tabela 5.2 mostra a quantidade de instâncias de controle resultantes para cada estratégia. Na avaliação são considerados dois cenários de configurações de reserva de backup (20% e 30%) para cada uma das três topologias (I2, RNP e GÉANT). Observa-se que para o cenário com 20% de reserva de backup, apenas a topologia GÉANT necessitou de um controlador adicional em relação ao valor ótimo computado. Por sua vez, no cenário com 30% de backup, identifica-se que tanto a topologia RNP quanto a GÉANT possuem instâncias extras. A diferença no número de topologias que apresentam este comportamento, entre os cenários de 20% e 30% de backup, está relacionada com a dificuldade do

problema, isto é, o aumento desse percentual torna mais restritiva a solução. Finalmente, vale notar que o custo adicional é de apenas uma instância para todos os casos analisados.

Com base nesses resultados, uma possível inferência é a de que a heurística provê um planejamento mais resiliente por apresentar um número maior de controladores em alguns casos. No entanto, conforme será demonstrado a seguir, mesmo alocando um número maior de controladores em alguns casos, as saídas geradas pela solução heurística são em geral menos resilientes do que as soluções ótimas. Isso se deve ao fato de que o posicionamento sobrevivente é influenciado por um conjunto maior de fatores.

Survivor reduz a probabilidade de perda de conectividade

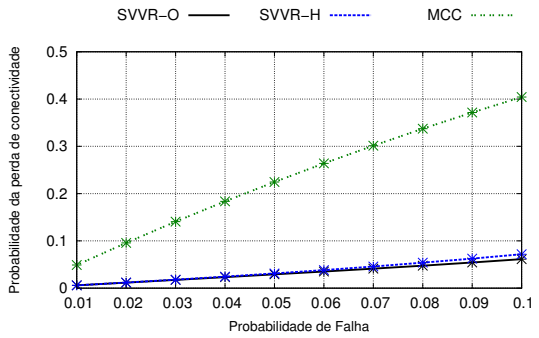
A primeira avaliação, apresentada na Figura 5.2, considera que os elementos da rede (ou seja, nós e enlaces) falham de forma independente e, em seguida, mede a probabilidade de uma conexão dispositivo-controlador ser interrompida. Os valores são obtidos utilizando a equação de resiliência (Equação 5.1, § 5.1.3). Os eixos x representam a probabilidade de falha atribuída a todos os elementos da rede (isto é, 0.01 significa que todos os nós e os enlaces têm 1% de chance de falhar); por sua vez, o eixo y mostra a probabilidade de perda de conectividade. A Figura 5.2 mostra na coluna esquerda – (a), (c) e (e) – os valores entre 1% e 10% (os mesmos utilizados por Zhang et al. (ZHANG; BEHESHTI; TATIPAMULA, 2011)), enquanto a coluna da direita – (b), (d) e (f) – extrapola a análise para toda a faixa (0% a 100% de probabilidade de falha).

A coluna esquerda da Figura 5.2 mostra que a abordagem Survivor, tanto na estratégia SVVR-O como na SVVR-H, supera MCC de forma consistente e substancial para todas as probabilidades (no eixo x). Mais importante ainda, as curvas representando o comportamento das estratégias que implementam a abordagem Survivor têm um fator de crescimento mais lento do que o MCC. Como pode ser observado, quando os componentes da rede têm 1% de chance de falha, as estratégias se comportam de forma parecida (menos de 8% de chance de perda de conectividade). Entretanto, conforme a possibilidade de falha aumenta para 5%, a probabilidade de perda de conectividade para as estratégias SVVR-O e SVVR-H é menos da metade do que para a MCC. Neste caso, as estratégias da abordagem Survivor permaneceram abaixo de 10%, enquanto que o MCC cresceu acima de 20%.

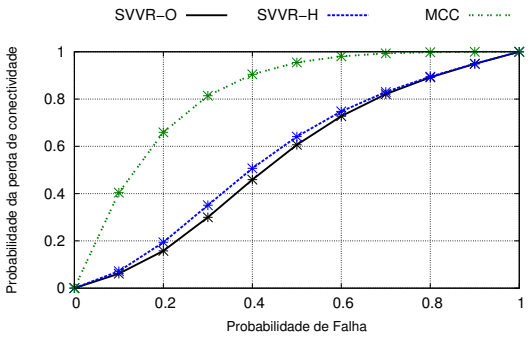
Além disso, embora a diferença relativa entre as estratégias (SVVR-O, SVVR-H x MCC) diminua, a coluna da direita na Figura 5.2 [(b), (d) e (f)] mostra que SVVR-O e SVVR-H chegam perto de 100% de perda de conectividade muito mais tarde que do MCC. Enquanto a probabilidade de perda de conectividade de SVVR-O e SVVR-H ainda esteja em torno de 80% quando a chance de falha é de 60%, a MCC já está perto de 100%.

Parte da melhora observada nas estratégias depende das propriedades topológicas da rede. Por exemplo, na estratégia MCC uma falha num único nó ou enlace pode romper uma ou múltiplas conexões dispositivo-controlador; entretanto na abordagem Survivor, é necessário pelo menos uma falha (em nó ou enlace) por conexão dispositivo-controlador. Este parece ser um efeito direto de explorar a diversidade de caminhos durante o posicionamento. Os próximos experimentos fornecem uma evidência adicional desse comportamento. Além disso, conforme será discutido, é insuficiente considerar diversidade de caminhos apenas após o posicionamento (ao invés de durante o mesmo, como na abordagem Survivor).

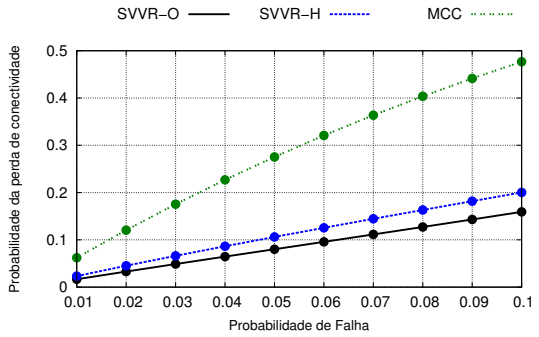
Diversidade de caminhos aumenta a capacidade de sobrevivência da rede



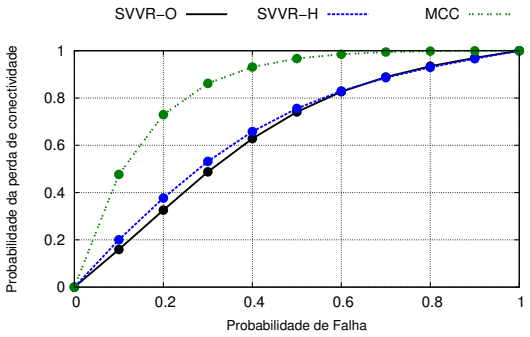
(a) INTERNET2 – 0.01% a 10%.



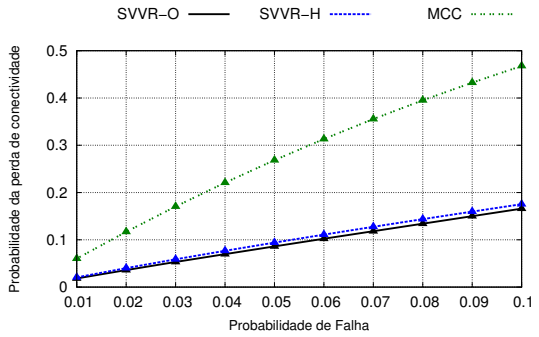
(b) INTERNET2 – 0% a 100%.



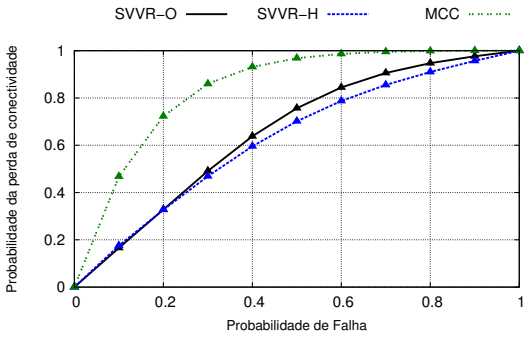
(c) RNP – 0.01% a 10%.



(d) RNP – 0% a 100%.



(e) GÉANT – 0.01% a 10%.

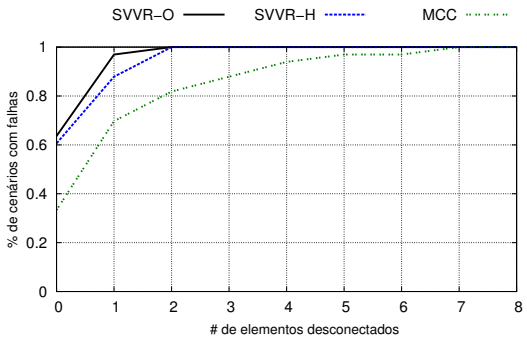


(f) GÉANT – 0% a 100%.

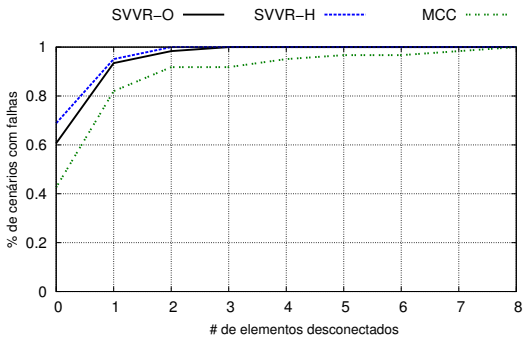
Figura 5.2: Resultados para as três topologias INTERNET2, RNP e GÉANT, com reserva de 20% de backup – SVVR-O x SVVR-H x MCC. Probabilidade da perda de conectividade controlador-dispositivo para diferentes probabilidades de falhas. *Quanto mais baixa for a curva, melhor.*

A Figura 5.3 expressa, por meio de Funções de Distribuição Acumuladas (Cumulative Distribution Functions, CDFs), o número de elementos desconexos ao enumerar todos os possíveis cenários de falhas. Três variações são avaliadas: falha única de um enlace, falha de 3 enlaces simultâneos, e falha de 6 enlaces simultâneos. Estas variações serão denominadas $k = 1$, $k = 3$, e $k = 6$, respectivamente. Os resultados seguem o mesmo comportamento para todas as topologias, e portanto foram selecionadas as análises das topologias GÉANT e RNP em um cenário com 20% de reserva de backup. A coluna da esquerda da Figura 5.3 [(a), (c) e (e)] mostra os resultados para a topologia RNP e a

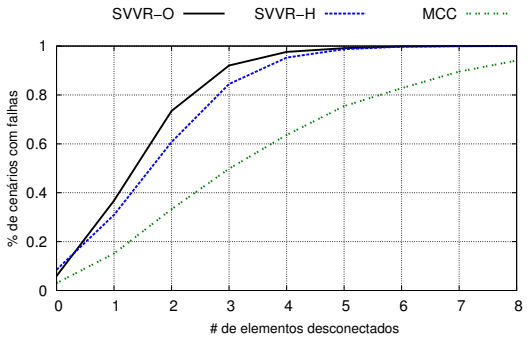
coluna da direita [(b), (d) e (f)] exibe os resultados para a topologia GÉANT para cada variação avaliada. As estratégias implementadas seguindo a abordagem Survivor superam a estratégia MCC em todas as variações.



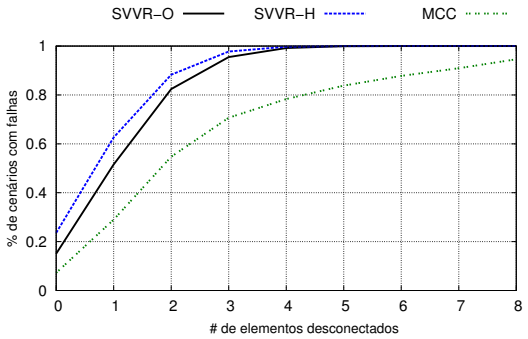
(a) RNP – falhas individuais.



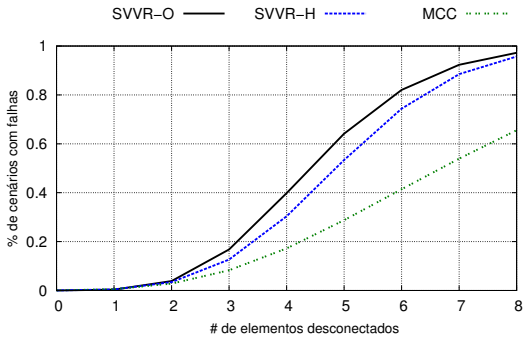
(b) GÉANT – falhas individuais.



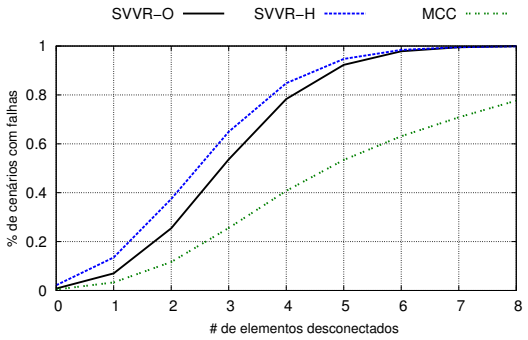
(c) RNP – 3 falhas simultâneas.



(d) GÉANT – 3 falhas simultâneas.



(e) RNP – 6 falhas simultâneas.



(f) GÉANT – 6 falhas simultâneas.

Figura 5.3: Resultados para as topologias RNP e GÉANT, com reserva de 20% de backup – SVVR-O x SVVR-H x MCC. Distribuição cumulativa de dispositivos desconectados para todos os possíveis casos de falhas 1, 3 e 6 enlaces. *Curvas mais próximas do lado superior esquerdo são melhores*.

Os resultados na Figura 5.3 [(a) e (b)] comparam as estratégias da abordagem Survivor, SVVR-O e SVVR-H com a MCC para falhas de enlaces individuais. Ao analisar os cenários sem qualquer dispositivo desconectado (X = 0) para a topologia RNP, verifica-se que as estratégias SVVR-$O_{k=1}$, SVVR-$H_{k=1}$ e $MCC_{k=1}$ apresentam proteção total em, 65%, 60% e 33% dos casos, respectivamente. Na topologia GÉANT, a proteção total (X

= 0) é identificada em 62%, 70% e 42% de todos os possíveis cenários de falhas. Em outras palavras, a estratégia SVVR-O (e respectivamente SVVR-H) alcança uma proteção 32% superior (respectivamente 27%) em relação à MCC na topologia RNP. Na topologia GÉANT, esse ganho é de 20% (respectivamente 28%). Além disso, em até 95% dos casos de falha de um enlace, o número de dispositivos desconectados é de no máximo 1 dispositivo para SVVR-O e SVVR-H, enquanto que em MCC chega a 6.

A Figura 5.3 mostra também que as estratégias da abordagem Survivor, nos cenários com 3 e 6 falhas de enlaces simultâneas, ocasionalmente superam a estratégia MCC em falhas de enlaces individuais. Como este comportamento é comum a todas topologias, é realizada a análise da maior delas, a rede GÉANT, a partir da Figura 5.3 [(b), (d) e (f)]. SVVR-$O_{k=3}$ e SVVR-$H_{k=3}$ (Figura 5.3(d)) superam $MCC_{k=1}$ (Figura 5.3(b)) nos 20% piores cenários de desconexão. Em tais situações SVVR-$O_{k=3}$ e SVVR-$H_{k=3}$ (Figura 5.3(d)) desconectam no máximo 4 dispositivos, enquanto $MCC_{k=1}$ (Figura 5.3(b)) desconecta até 8. Além disso, para SVVR-$O_{k=6}$, SVVR-$H_{k=6}$ (Figura 5.3(f)) e $MCC_{k=1}$ (Figura 5.3(b)), existe um ponto de virada em torno de 90%, ou seja, SVVR-$*_{k=6}$ superam $MCC_{k=1}$ nos 10% piores casos.

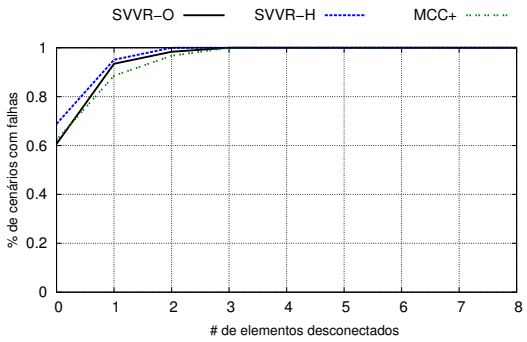
A diferença é mais perceptível ao considerar 6 falhas simultâneas para todas estratégias. Em SVVR-$O_{k=6}$ e SVVR-$H_{k=6}$ (Figura 5.3(f)), os piores 20% dos casos de desconexão têm entre 4 e 8 dispositivos desconectados; em $MCC_{k=6}$, o mesmo intervalo $[4, 8]$ ocorre em 40% dos casos (a partir de 40% até 80%). Nesse intervalo, a estratégia SVVR-O é 50% menos provável de desconectar a mesma quantidade de dispositivos ($100\% - \frac{\simeq 20\%}{\sim 40\%} \approx 50\%$).

Ainda observando a Figura 5.3, percebe-se que há uma diferença no desempenho entre as estratégias da abordagem Survivor (SVVR-O e SVVR-H). Em particular, para o cenário da topologia GÉANT na Figura 5.3 [(b), (d) e (f)] nota-se que a heurística (SVVR-H) apresenta um comportamento superior à ótima (SVVR-O) pois, neste cenário, o resultado da heurística apresenta uma instância de controle extra quando comparada a saída da estratégia ótima (Tabela 5.2). Isto demonstra que a quantidade de instâncias de controle dispostas na rede pode contribuir com a sobrevivência da mesma, entretanto, este não é o fator fundamental. A próxima análise (Figura 5.4) demonstra que o posicionamento das instâncias é determinante, mesmo quando a quantidade de instâncias de controle é superior.

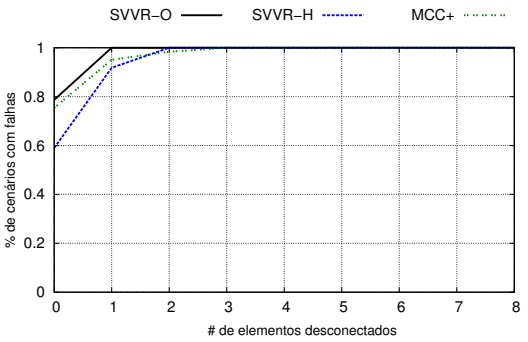
Diversidade de caminhos precisa ser explicitamente considerada durante o posicionamento para ser melhor explorada

A fim de avaliar a eficácia do posicionamento da abordagem Survivor, a diversidade de caminhos é adicionada à estratégia MCC após o posicionamento; esta modificação é denominada MCC+. A avaliação foi realizada para todas as topologias, porém foram escolhidos os resultados da maior topologia (rede GÉANT) para a discussão. Ademais, os resultados foram avaliados sobre dois cenários definidos pela variação do percentual de backup de recursos, o primeiro com 20% e o segundo com 30%. Os gráficos da Figura 5.4 tem o mesmo objetivo do anterior (Figura 5.3), isto é, expressar, por meio de CDFs, o número de elementos desconexos, ao enumerar todos os possíveis cenários de falhas.

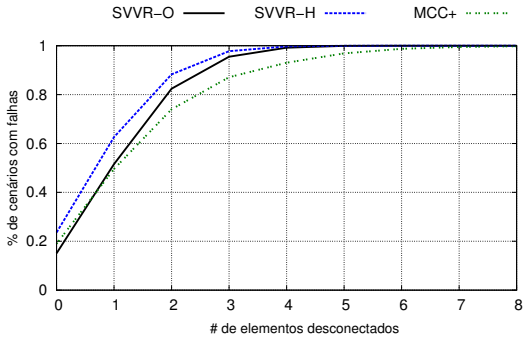
Observando a Figura 5.4, percebe-se que os gráficos exibem um comportamento em grande parte similar ao anterior, porém nota-se que a diferença entre as estratégias da abordagem Survivor e a estratégia MCC+ são mais baixas no geral. Apesar disso, a estratégia SVVR-O ainda supera MCC+. Tal se deve ao fato que, adicionar diversidade de caminhos após o posicionamento é insuficiente. Por exemplo, ao se analisar todos



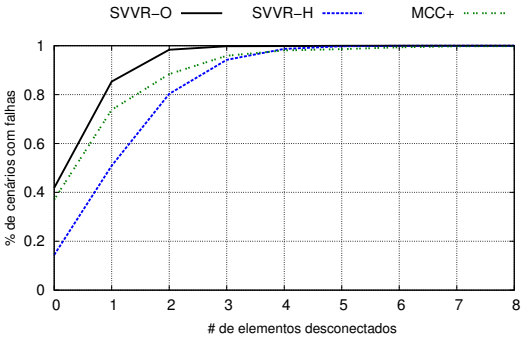
(a) GÉANT – falhas individuais, 20% backup.



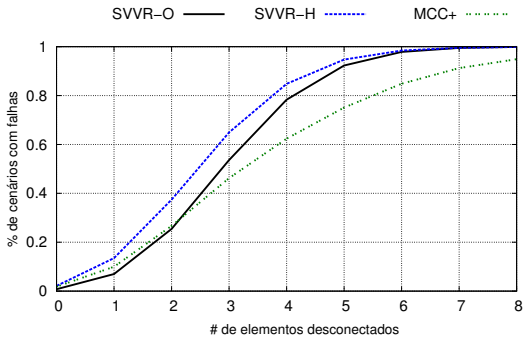
(b) GÉANT – falhas individuais, 30% backup.



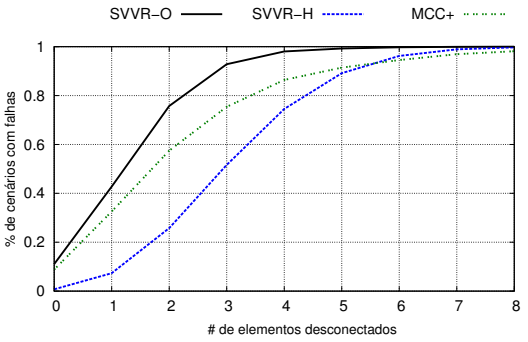
(c) GÉANT – 3 falhas simultâneas, 20% backup.



(d) GÉANT – 3 falhas simultâneas, 30% backup.



(e) GÉANT – 6 falhas simultâneas, 20% backup.



(f) GÉANT – 6 falhas simultâneas, 30% backup.

Figura 5.4: Resultados para topologia GÉANT – SVVR-O x SVVR-H x MCC+. Distribuição cumulativa de dispositivos desconectados para todos os casos possíveis na disrupção de 1, 3 e 6 enlaces. *Curvas mais próximas do lado superior esquerdo são melhores*.

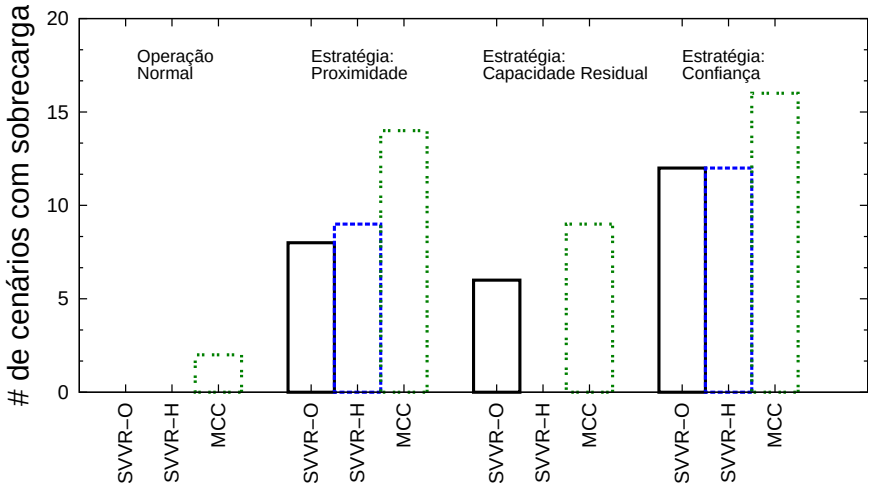
os valores obtidos do cenário GÉANT–30% temos que a estratégia MCC+, em relação à SVVR-O, tem 2-3 vezes mais dispositivos desconectados nos piores casos; $\frac{3}{1} = 3$, $\frac{19}{6} \approx 3$, e $\frac{22}{11} = 2$, para $k = 1$, $k = 3$, e $k = 6$, respectivamente ($\frac{pior\ caso\ MCC+}{pior\ caso\ Survivor}$). Por sua vez, a estratégia SVVR-H apresenta um comportamento que exige algumas observações adicionais, conforme segue.

Analisando o desempenho da estratégia SVVR-H nas colunas da direita e da esquerda na Figura 5.4, observa-se comportamentos distintos. No primeiro cenário, GÉANT–20%, a estratégia SVVR-H apresenta um desempenho superior às demais, enquanto que no segundo cenário, GÉANT–30%, o mesmo comportamento não consegue ser mantido. É

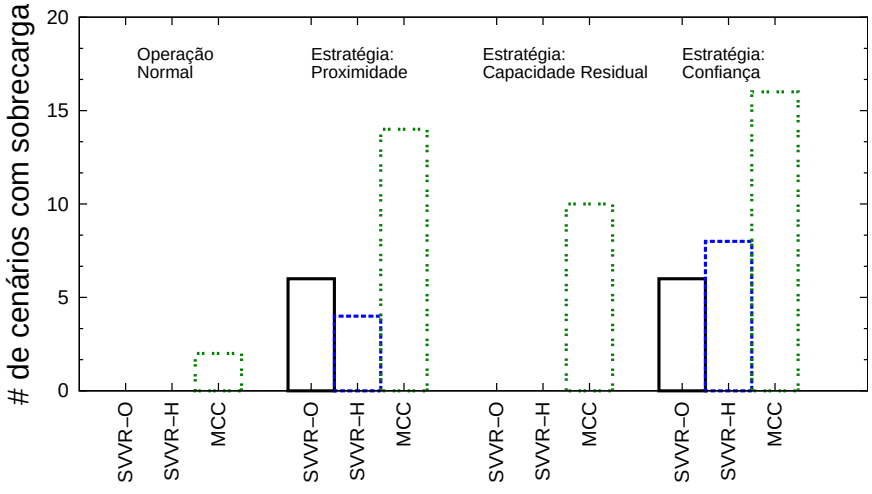
importante notar que essa diferença ocorre mesmo com ambas saídas da estratégia SVVR-H contando com uma instância de controlador adicional em relação ao valor computado por SVVR-O (Tabela 5.2). Isto é, a quantidade de instâncias de controle não é o único fator que impacta na sobrevivência. A razão pela qual esta diferença de comportamento ocorre é que o aumento do percentual de recursos de backup torna mais restritiva a solução, o que na prática exige uma análise de um espaço de soluções maior para chegar a um resultado possivelmente melhor.

Contudo, os resultados da heurística em relação à estratégia MCC+ ainda são melhores no que se refere ao gerenciamento de recursos e à recuperação da rede após disrupções. Para compreender melhor é essencial obter uma visão global da distribuição da carga e do comportamento de convergência da rede, questões investigadas nas análises seguintes.

A convergência de rede após disrupções é altamente sensível às informações pré-definidas em mecanismos de recuperação



(a) 20% backup.



(b) 30% backup.

Figura 5.5: Número de cenários com sobrecarga (carga do controlador >100%) durante operação normal e na situação após a fase de recuperação utilizando cada uma das heurísticas propostas. *Quanto menores as barras, melhor.*

A Figura 5.5 mostra a quantidade de cenários com sobrecarga em instâncias de con-

trolador nas topologias RNP e GÉANT. São considerados os cenários com 20% e 30% de reserva de backup. Cada cenário de sobrecarga conta com no mínimo uma instância de controlador com carga superior a 100%. Tal situação significa que a instância de controle apresenta perda de pacotes, atrasos no processamento das requisições e na situação de alguma falha na infraestrutura física não comporta receber o redirecionamento de conexões. A análise considera a rede em operação normal e três casos de operação pós-disrupção configurados com heurísticas de recuperação diferentes – proximidade, capacidade residual e confiança.

Durante operação normal (sem disrupção), as estratégias da abordagem Survivor não apresentam uma única instância de controlador sobrecarregada. Isso acontece porque a estratégia de posicionamento é ciente da capacidade e, portanto, evita sobrecargas na atribuição de dispositivos para controladores. Em contraste, a estratégia MCC apresenta algumas instâncias de controladores sobrecarregados mesmo durante operação normal. Este comportamento decorre do fato de que a estratégia MCC considera apenas uma heurística com base no centroide ao atribuir dispositivos aos controladores. Assim, em alguns casos, uma instância de controlador acaba por ter uma carga muito mais elevada.

Após a ocorrência das disrupções, o mecanismo de recuperação utiliza as listas criadas pelas três heurísticas propostas anteriormente (§4.3.3). A avaliação mostra que o mecanismo de recuperação tem um desempenho muito diferente dependendo da forma como as suas listas de backup foram compostas. A heurística baseada na capacidade residual superou as demais (proximidade e confiança), tanto para MCC quanto para Survivor, para ambos cenários de percentual de backup (20% e 30%).

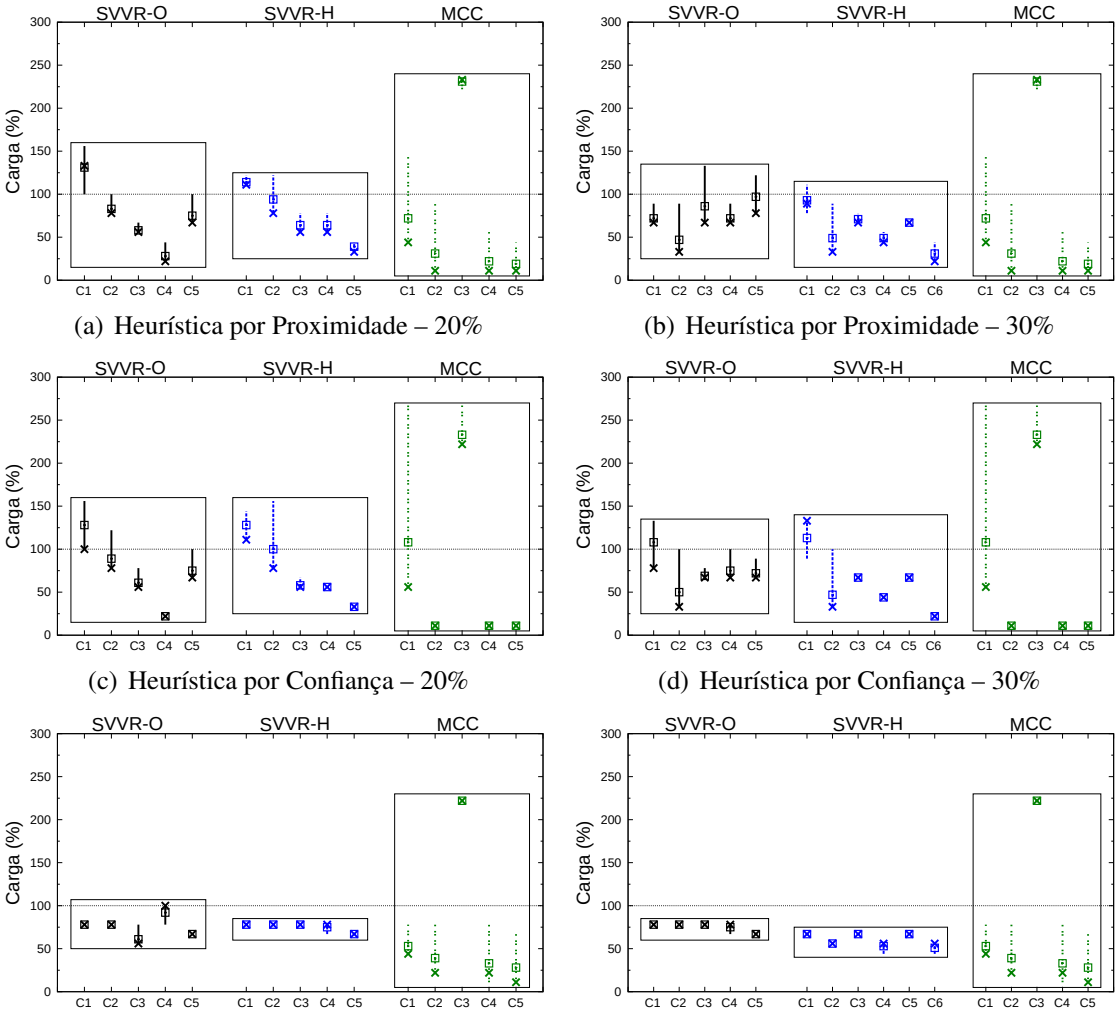
Além disso, destacam-se duas situações em que o resultado da heurística SVVR-H tem um comportamento diferente de SVVR-O em função de uma instância de controlador a mais, Figura 5.5 (a) na heurística de capacidade residual e Figura 5.5 (b) na heurística de proximidade. Apesar das estratégias na abordagem Survivor estabelecerem recursos de backup, alguns casos ainda apresentaram sobrecarga. Esse comportamento decorre da qualidade da heurística de recuperação, a qual pode gerar uma distribuição de recursos melhor ou pior, conforme observado na próxima avaliação.

Sobrecargas nos controladores podem ser evitadas de forma proativa, utilizando algoritmos cientes de capacidade e definindo recursos de backup

A avaliação anterior mostrou que diferentes tipos de informações utilizadas durante a recuperação geram um impacto substancial no balanceamento de carga entre controladores. Essa avaliação observa mais atentamente o estado da rede após a convergência para entender melhor o comportamento observado. Para esse fim, dois conjuntos de gráficos são apresentados, o primeiro na Figura 5.6 referente a topologia RNP e, logo após, na Figura 5.7 para a topologia GÉANT, ambos conjuntos com as respectivas variações do percentual de reserva de backup 20% e 30%. Esses dois conjuntos mostram a carga em cada controlador considerando todos os cenários de falha e o respectivo desempenho de cada heurística de recuperação. Os valores indicam o mínimo, o máximo, a média e a moda da carga.

Os gráficos exibem um comportamento em grande parte similar. Percebe-se que, de modo geral, o comportamento resultante das heurísticas baseadas em caminhos, isto é, heurística por proximidade e heurística por confiança, apresentam comportamentos semelhantes. Como observado, apesar do posicionamento resultante garantir capacidade livre suficiente ao final, as heurísticas de proximidade e confiança (Figura 5.6 [(a), (b), (c), (d)] e Figura 5.7 [(a), (b), (c), (d)]) tendem a sobrecarregar alguma(s) instância(s) (por

exemplo Figura 5.6 [(a) e (c)], C1 em SVVR-O; C1, C2 em SVVR-H e C1, C3 em MCC). Para estes casos (heurísticas de proximidade e confiança) as estratégias que implementam a abordagem Survivor, os valores máximos e mínimos apresentam diferenças para todos os controladores e a média varia, mas a moda em particular, mostra que as instâncias de controladores têm sua carga perto do mínimo na maioria dos casos. A razão para tal comportamento é que a maioria dos dispositivos seleciona o mesmo controlador de backup quando seu controlador primário falha, deixando assim os outros controladores com sua carga inicial. O posicionamento MCC mostra resultados semelhantes, mas a variação entre os valores tende a ser maior, porque o posicionamento inicial já está desequilibrado.



(a) Heurística por Proximidade – 20%

(b) Heurística por Proximidade – 30%

(c) Heurística por Confiança – 20%

(d) Heurística por Confiança – 30%

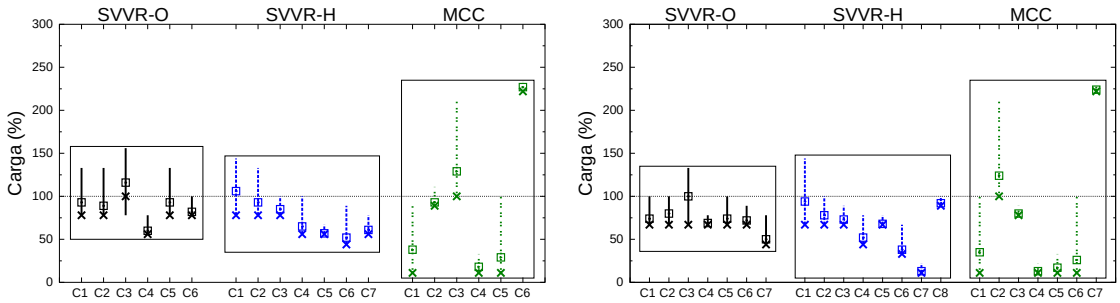
(e) Heurística por Capacidade Residual – 20%

(f) Heurística por Capacidade Residual – 30%

Figura 5.6: Resultados para topologia RNP. Mínimo, máximo, média (quadrado), e moda (estrela) para a carga em cada instância do controlador (nomeadas C1-C*n*) após recuperação. Estratégias de posicionamento (SVVR-O, SVVR-H e MCC) estão delimitadas por retângulos para facilitar a exposição e comparação.

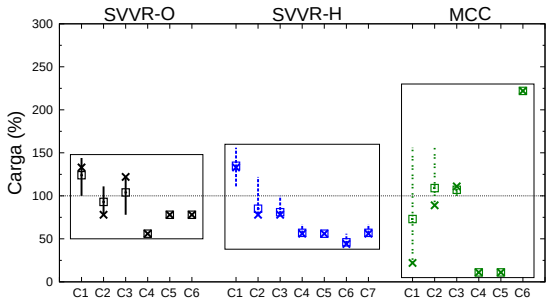
Em contraste, a heurística de capacidade residual (Figura 5.6 [(e) e (f)] e Figura 5.7 [(e) e (f)]) tende a apresentar uma carga distribuída de maneira mais uniforme. Nas estratégias que implementam a abordagem Survivor, o máximo, o mínimo, a médio e a moda têm quase o mesmo valor. Tal comportamento pode ser explicado porque os dispositivos tentam se conectar a diferentes controladores na rede, dando preferência àqueles com

maior capacidade residual. Como resultado, a carga em controladores aumenta, mas nenhum deles sobrecarrega. Neste caso, o MCC se comporta de forma diferente de SVVR-O e SVVR-H. Isto pode ser explicado pelo fato de que na estratégia MCC, controladores C3 na topologia RNP e C2, C3, C7 na topologia GÉANT foram atribuídos a uma carga maior (em comparação com os outros controladores) durante o posicionamento inicial – em particular, C3 na RNP e C7 na GÉANT já estavam sobrecarregados antes da recuperação. Como resultado, os controladores C1, C2, C4, C5 na topologia RNP e C1, C4, C5 e C6 na topologia GÉANT foram os únicos que receberam conexões durante a recuperação e suas variações de carga são semelhantes.

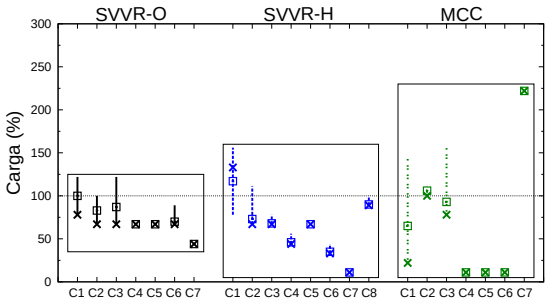


(a) Heurística por Proximidade – 20%

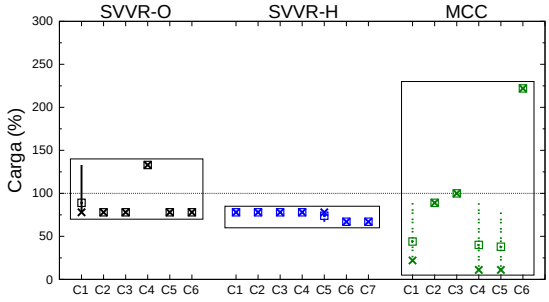
(b) Heurística por Proximidade – 30%



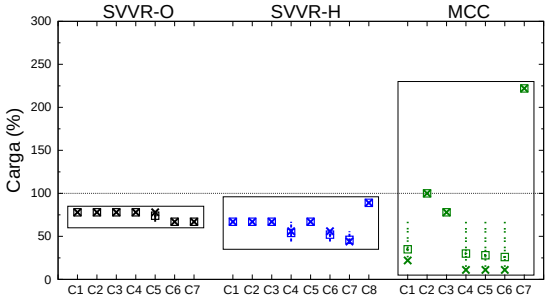
(c) Heurística por Confiança – 20%



(d) Heurística por Confiança – 30%



(e) Heurística por Capacidade Residual – 20%



(f) Heurística por Capacidade Residual – 30%

Figura 5.7: Resultados para topologia GÉANT. Mínimo, máximo, média (quadrado), e moda (estrela) para a carga em cada instância do controlador (nomeadas C1-C*n*) após recuperação. Estratégias de posicionamento (SVVR-O, SVVR-H e MCC) estão delimitadas por retângulos para facilitar a exposição e comparação.

5.4 Observações Finais

Os resultados obtidos permitem três constatações principais. Primeiro, a **quantidade de instâncias de controle não é o único fator que impacta na sobrevivência**. Um

posicionamento mais restrito com número menor de instâncias pode se comportar tão bem quanto, ou até mesmo *melhor*, do que um posicionamento flexível com um maior número de controladores.

Segundo, a **diversidade de caminhos deve ser considerada durante o posicionamento.** De modo a explorar todo o potencial das propriedades topológicas da rede identificou-se que é fundamental que seja levado em conta, desde o início do projeto do plano de controle, os atributos dessa propriedade. Tentativas de fazer uso da diversidade de caminhos após ter sido efetuado o posicionamento não permitem o melhor aproveitamento da propriedade.

E terceiro, no que se refere ao posicionamento com sobrevivência, existe um **compromisso claro entre três fatores: posicionamento, quantidade de instâncias do controlador e conectividade**. O conjunto de resultados obtidos durante a avaliação evidenciam tais fatores, além de indicar a importância do equilíbrio para obter uma solução adequada para cada topologia de rede.

6 CONCLUSÃO

Em virtude dos seus benefícios, o paradigma SDN atraiu grande interesse da indústria e da área acadêmica. No entanto, a adoção em massa de SDN esbarra em desafios decorrentes das mudanças trazidas pelo paradigma. Em particular, a sobrevivência de redes SDN precisa lidar com a manutenção da conectividade entre controlador e dispositivos de encaminhamento.

Para aumentar a sobrevivência das redes SDN, a literatura propõe estratégias de replicação e posicionamento de controladores na rede. Contudo, os trabalhos anteriores apresentam limitações. Primeiro, ignoram aspectos essenciais para a sobrevivência, tais como diversidade de caminhos, reserva de recursos e o estado de convergência da rede. Segundo, consideram que o controlador possui capacidade ilimitada, desprezando a possibilidade de sobrecarga.

Contribuições

Esta dissertação propôs *Survivor*, uma nova abordagem para o problema de posicionamento de controlador que aumenta a sobrevivência de redes SDN. Para isso, as estratégias que implementam a abordagem Survivor planejam a rede tanto para a operação normal, quanto para operação pós-disrupção. Em outras palavras, por um lado a rede deve ser configurada de forma ótima para operação normal, por outro também deve prever a reação, da melhor maneira possível, a eventos de disrupção. Para este fim, a abordagem foi dividida em duas etapas que tratam três aspectos principais – a conectividade, a capacidade e a recuperação. A primeira parte requer encontrar posições ótimas para instâncias do controlador de tal forma que a conectividade seja maximizada e as restrições de capacidade sejam satisfeitas. A segunda parte consiste na definição de uma lista ordenada de backup para cada dispositivo, tratando do aspecto recuperação. O conjunto de experimentos realizados mostrou uma série de constatações, resumidas a seguir.

Explorar diversidade de caminhos durante o posicionamento reduz significativamente a probabilidade de perda de conectividade. De modo a explorar o potencial das propriedades topológicas da rede, identificou-se que é fundamental que múltiplos caminhos sejam considerados durante o projeto do plano de controle. Tentativas de fazer uso da diversidade de caminhos após ter sido efetuado o posicionamento não permitem o melhor aproveitamento da propriedade.

A quantidade de instâncias de controle não é o único fator que impacta na sobrevivência. Identificou-se que, em alguns casos, um posicionamento com número menor de instâncias pode se comportar tão bem quanto, ou até mesmo melhor, do que um posicionamento com um maior número de controladores.

Algoritmos cientes de capacidade são essenciais. Foi constatado que as instâncias de

controlador são facilmente sobrecarregadas, mesmo quando o número de controladores é suficiente. Estratégias de posicionamento não cientes de capacidade geram controladores com demandas desbalanceadas. Ademais, tal comportamento foi identificado em ambos os cenários normal e pós-recuperação.

Por fim, *a heurística para a seleção de controladores backup tem impacto significativo sobre o estado da rede convergente*. Observou-se que a convergência da rede após disrupções é altamente sensível às informações pré-definidas em mecanismos de recuperação. Além do mais, esse comportamento se não observado pode gerar uma distribuição desequilibrada de recursos, provocando a sobrecarga em instâncias de controle.

Direções para pesquisas futuras

Embora a abordagem Survivor seja eficiente e viável, oportunidades de melhorias existem. As estratégias que compõem Survivor são um primeiro passo no estudo para mitigar disrupções em redes de grande escala. Como perspectivas de trabalhos futuros são apresentadas três possíveis frentes de pesquisa: (i) aprimorar as estratégias de posicionamento, (ii) estender a avaliação e (iii) propor e desenvolver novos mecanismos de recuperação.

Em relação às estratégias, são consideradas algumas iniciativas para melhorar mais os resultados. Primeiro, pretende-se buscar outros aspectos que podem ser explorados a fim de garantir maior capacidade de sobrevivência (por exemplo, definir regiões por meio do agrupamento de equipamentos e utilizar outros modelos de reservas de recursos sobre a infraestrutura física da rede). Segundo, a restrição do número de *hops* deverá ser aprimorada para um modelo de atrasos de comunicação, considerando o comprimento dos enlaces, capacidade e suas tecnologias (fibra ótica, Wifi, satélite, etc). Além disso, é planejado o desenvolvimento de uma meta-heurística com base na estratégia heurística proposta, possibilitando melhores resultados e uma exploração maior do espaço de soluções do problema.

Em termos da avaliação, pretende-se aplicar as estratégias da abordagem Survivor sob um conjunto maior de topologias. Além disso, busca-se analisar relações de compromisso, como (a) sobrevivência *vs* custo relativo ao número de instâncias de controle; e (b) latência dispositivo-controlador *vs* latência controlador-controlador relativas ao posicionamento.

Por fim, a abordagem Survivor pode ser estendida através do estudo e incorporação de novos mecanismos de recuperação. Após o processo de posicionamento das instâncias de controle, pode ser elaborada uma estratégia *online* para o gerenciamento ativo das conexões dispositivos-controlador aumentando o desempenho e sobrevivência no gerenciamento da rede.

REFERÊNCIAS

ARYA, V.; GARG, N.; KHANDEKAR, R.; MEYERSON, A.; MUNAGALA, K.; PANDIT, V. Local Search Heuristic for K-median and Facility Location Problems. In: THIRTY-THIRD ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING, New York, NY, USA. **Proceedings** ACM, 2001. p.21–29. (STOC '01).

BAGGA, K.; BEINEKE, L.; PIPPERT, R.; LIPMAN, M. A classification scheme for vulnerability and reliability parameters of graphs. **Mathematical and Computer Modelling**, Amsterdam, Netherlands, v.17, n.11, p.13 – 16, 1993.

BANIKAZEMI, M.; OLSHEFSKI, D.; SHAIKH, A.; TRACEY, J.; WANG, G. Meridian: an sdn platform for cloud network services. **Communications Magazine**, New York, NY, USA, v.51, n.2, p.120–127, February 2013.

BARI, M. F.; ROY, A. R.; CHOWDHURY, S. R.; ZHANG, Q.; ZHANI, M. F.; AHMED, R.; BOUTABA, R. Dynamic Controller Provisioning in Software Defined Networks. In: CNSM. **Proceedings** IEEE, 2013. p.1–8.

BENSON, T.; AKELLA, A.; MALTZ, D. Unraveling the Complexity of Network Management. In: USENIX SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION, 6., Berkeley, CA, USA. **Proceedings** USENIX Association, 2009. p.335–348. (NSDI'09).

BERDE, P.; GEROLA, M.; HART, J.; HIGUCHI, Y.; KOBAYASHI, M.; KOIDE, T.; LANTZ, B.; O'CONNOR, B.; RADOSLAVOV, P.; SNOW, W.; PARULKAR, G. ONOS: towards an open, distributed sdn os. In: THIRD WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2014. p.1–6. (HotSDN '14).

BIG SWITCH NETWORKS. **Project Floodlight**. Disponível em <`http://www.projectfloodlight.org`>.

CAI, Z.; COX, A.; NG, E. **Maestro**: a system for scalable openflow control. Technical Report TR10-08, Rice University.

CASADO, M.; KOPONEN, T.; RAMANATHAN, R.; SHENKER, S. Virtualizing the network forwarding plane. In: WORKSHOP ON PROGRAMMABLE ROUTERS FOR EXTENSIBLE SERVICES OF TOMORROW, New York, NY, USA. **Proceedings** ACM, 2010. p.8:1–8:6. (PRESTO '10).

CUNHA, I.; SILVEIRA, F.; OLIVEIRA, R.; TEIXEIRA, R.; DIOT, C. Uncovering Artifacts of Flow Measurement Tools. In: PAM. **Proceedings** Springer-Verlag, 2009. p.187–196.

DIXIT, A.; HAO, F.; MUKHERJEE, S.; LAKSHMAN, T.; KOMPELLA, R. Towards an elastic distributed SDN controller. In: ACM SIGCOMM WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2013. p.7–12. (HotSDN '13).

ERICKSON, D. The Beacon Openflow Controller. In: SECOND ACM SIGCOMM WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2013. p.13–18. (HotSDN '13).

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The Road to SDN. **Queue**, New York, NY, USA, v.11, n.12, p.20:20–20:40, Dec. 2013.

HELLER, B.; SHERWOOD, R.; MCKEOWN, N. The controller placement problem. In: SIGCOMM HOTSDN WORKSHOP. **Proceedings** ACM, 2012. p.7–12.

HOCK, D.; HARTMANN, M.; GEBERT, S.; JARSCHEL, M.; ZINNER, T.; TRAN-GIA, P. Pareto-optimal resilient controller placement in SDN-based core networks. In: TELETRAFFIC CONGRESS (ITC), 2013 25TH INTERNATIONAL. **Proceedings** IEEE, 2013. p.1–9.

HU, Y.; WANG, W.; GONG, X.; QUE, X.; CHENG, S. On the placement of controllers in software-defined networks. **The Journal of China Universities of Posts and Telecommunications**, Amsterdam, Netherlands, v.19, Supplement 2, n.0, p.92 – 171, 2012.

HU, Y.; WENDONG, W.; GONG, X.; QUE, X.; SHIDUAN, C. Reliability-aware controller placement for Software-Defined Networks. In: IM. **Proceedings** IEEE, 2013. p.672–675.

JAIN, S.; KUMAR, A.; MANDAL, S.; ONG, J.; POUTIEVSKI, L.; SINGH, A.; VENKATA, S.; WANDERER, J.; ZHOU, J.; ZHU, M.; ZOLLA, J.; HöLZLE, U.; STUART, S.; VAHDAT, A. B4: experience with a globally-deployed software defined wan. In: ACM SIGCOMM 2013 CONFERENCE ON SIGCOMM, New York, NY, USA. **Proceedings** ACM, 2013. p.3–14. (SIGCOMM '13).

JARRAYA, Y.; MADI, T.; DEBBABI, M. A Survey and a Layered Taxonomy of Software-Defined Networking. **Communications Surveys Tutorials**, New York, NY, USA, n.99, p.1–1, 2014.

KOPONEN, T.; CASADO, M.; GUDE, N.; STRIBLING, J.; POUTIEVSKI, L.; ZHU, M.; RAMANATHAN, R.; IWATA, Y.; INOUE, H.; HAMA, T.; SHENKER, S. Onix: a distributed control platform for large-scale production networks. In: OSDI. **Proceedings** USENIX, 2010. p.1–6.

KREUTZ, D.; RAMOS, F.; VERISSIMO, P.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. **Software-Defined Networking**: a comprehensive survey. Technical Report. Disponível em: http://arxiv.org/abs/1406.0440.

KRISHNAMURTHY, A.; CHANDRABOSE, S. P.; GEMBER J., A. Pratyaastha: an efficient elastic distributed sdn control plane. In: THIRD WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2014. p.133–138. (HotSDN '14).

LEVIN, D.; WUNDSAM, A.; HELLER, B.; HANDIGOL, N.; FELDMANN, A. Logically centralized?: state distribution trade-offs in software defined networks. In: HOT TOPICS IN SOFTWARE DEFINED NETWORKS, New York, NY, USA. **Proceedings** ACM, 2012. p.1–6. (HotSDN '12).

LIU, J.; PANDA, A.; SINGLA, A.; GODFREY, B.; SCHAPIRA, M.; SHENKER, S. Ensuring Connectivity via Data Plane Mechanisms. In: NSDI. **Proceedings** USENIX, 2013. p.113–126.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, New York, NY, USA, v.38, n.2, p.69–74, Mar. 2008.

MOSHREF, M.; YU, M.; SHARMA, A.; GOVINDAN, R. Scalable Rule Management for Data Centers. In: USENIX CONFERENCE ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION, 10., Berkeley, CA, USA. **Proceedings** USENIX Association, 2013. p.157–170. (nsdi'13).

NUNES, B. A. A.; MENDONCA, M.; NGUYEN, X.-N.; OBRACZKA, K.; TURLETTI, T. A Survey of Software-Defined Networking: past, present, and future of programmable networks. **Communications Surveys Tutorials**, New York, NY, USA, v.16, n.3, p.1617–1634, Third 2014.

ONF. **OpenFlow Switch Specification 1.4.0**. Disponível em <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf>.

OPENDAYLIGHT. **OpenDaylight**: a linux foundation collaborative project. Disponível em <http://www.opendaylight.org>.

OSYCZKA, A. An approach to multicriterion optimization for structural design. In: INTERNATIONAL SYMPOSIUM ON OPTIMAL STRUCTURAL DESIGN (UNIVERSITY OF ARIZONA), Tucson, AZ. **Proceedings** University of Arizona, 1981.

PANDA, A.; SCOTT, C.; GHODSI, A.; KOPONEN, T.; SHENKER, S. CAP for Networks. In: SECOND ACM SIGCOMM WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2013. p.91–96. (HotSDN '13).

REITBLATT, M.; CANINI, M.; GUHA, A.; FOSTER, N. FatTire: declarative fault tolerance for software-defined networks. In: SECOND ACM SIGCOMM WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2013. p.109–114. (HotSDN '13).

ROS, F. J.; RUIZ, P. M. Five Nines of Southbound Reliability in Software-defined Networks. In: THIRD WORKSHOP ON HOT TOPICS IN SOFTWARE DEFINED NETWORKING, New York, NY, USA. **Proceedings** ACM, 2014. p.31–36. (HotSDN '14).

SEZER, S.; SCOTT-HAYWARD, S.; CHOUHAN, P.; FRASER, B.; LAKE, D.; FINNEGAN, J.; VILJOEN, N.; MILLER, M.; RAO, N. Are we ready for SDN? Implementation challenges for software-defined networks. **Communications Magazine**, New York, NY, USA, v.51, n.7, p.36–43, July 2013.

SHIN, M.-K.; NAM, K.-H.; KANG, M.; CHOI, J.-Y. **Formal Specification Framework for Software-Defined Networks (SDN) draft-shin-sdn-formal-specification-03**. Disponível em <http://tools.ietf.org/html/draft-shin-sdn-formal-specification-03>.

STEPHENS, B.; COX, A. L.; RIXNER, S. Plinko: building provably resilient forwarding tables. In: TWELFTH ACM WORKSHOP ON HOT TOPICS IN NETWORKS, New York, NY, USA. **Proceedings** ACM, 2013. p.26:1–26:7. (HotNets-XII).

TELEGRAPH, N.; CORPORATION, T. **RyU**. Disponível em <http://osrg.github.io/ryu/>.

TOOTOONCHIAN, A.; GANJALI, Y. HyperFlow: a distributed control plane for openflow. In: RESEARCH ON ENTERPRISE NETWORKING, 2010., Berkeley, CA, USA. **Proceedings** USENIX Association, 2010. p.3–3. (INM/WREN'10).

TOOTOONCHIAN, A.; GORBUNOV, S.; GANJALI, Y.; CASADO, M.; SHERWOOD, R. On controller performance in software-defined networks. In: HOT-ICE WORKSHOP. **Proceedings** USENIX, 2012. p.10–10.

TREMA. **Trema Full-Stack OpenFlow Framework in Ruby and C**. Disponível em <http://trema.github.io/trema/>.

TRESTIAN, R.; MUNTEAN, G.-M.; KATRINIS, K. MiceTrap: scalable traffic engineering of datacenter mice flows using openflow. In: INTEGRATED NETWORK MANAGEMENT (IM 2013), 2013 IFIP/IEEE INTERNATIONAL SYMPOSIUM ON. **Proceedings** IEEE, 2013. p.904–907.

VISSICCHIO, S.; VANBEVER, L.; BONAVENTURE, O. Opportunities and Research Challenges of Hybrid Software Defined Networks. **ACM Computer Comm. Review**, New York, NY, USA, v.44, n.2, April 2014.

YEGANEH, S. H.; GANJALI, Y. Kandoo: a framework for efficient and scalable offloading of control applications. In: SIGCOMM HOTSDN WORKSHOP. **Proceedings** ACM, 2012. p.19–24.

YEGANEH, S.; TOOTOONCHIAN, A.; GANJALI, Y. On scalability of software-defined networking. **Communications Magazine**, New York, NY, USA, v.51, n.2, p.136–141, 2013.

ZHANG, Y.; BEHESHTI, N.; TATIPAMULA, M. On Resilience of Split-Architecture Networks. In: GLOBAL TELECOMMUNICATIONS CONFERENCE (GLOBECOM 2011), 2011 IEEE. **Proceedings** IEEE, 2011. p.1–6.

APPENDIX A - LISTA DE PUBLICAÇÕES

- **MULLER, L. F.**; OLIVEIRA, R. R.; LUIZELLI, M. C.; GASPARY, L. P.; BARCELLOS, M. P. Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability. In: IEEE GLOBAL TELECOMMUNICATIONS CONFERENCE, 2014. Austin, Texas. **Proceedings** IEEE GLOBECOM, 2014. pp. 1–7.

- LEHMANN, M. B.; **MULLER, L. F.**; ANTUNES, R. S.; BARCELLOS, M. P. Disconnecting to Connect: understanding Optimistic Disconnection in BitTorrent. In: IEEE INTERNATIONAL CONFERENCE ON PEER-TO-PEER COMPUTING, Tarragona, Spain. **Proceedings** IEEE P2P, 2012. pp. 1–10.

- LEHMANN, M. B.; **MULLER, L. F.**; ANTUNES, R. S.; BARCELLOS, M. P. Desvendando o Impacto da Desconexão Otimista no BitTorrent. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, XXX, Ouro Preto, MG, Brasil. **Anais** SBRC, 2012. pp. 1–14.

- BONDAN, L.; **MULLER, L. F.**; KIST, M. Multiflow: Multicast clean-slate com cálculo antecipado das rotas em redes programáveis OpenFlow. In: JOURNAL OF APPLIED COMPUTING RESEARCH (JACR), v. 1, p. 68–74, 2012.

APPENDIX B - ARTIGO PUBLICADO IEEE GLOBECOM

- Título: Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability
- Conferência: 2014 IEEE Global Telecommunications Conference (GLOBECOM 2014)
- URL: `http://globecom2014.ieee-globecom.org`
- Data: 8 – 12, dezembro, 2014
- Local: Austin, Texas

Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability

Lucas F Müller, Rodrigo R Oliveira, Marcelo C Luizelli,
Luciano P Gaspary, Marinho P Barcellos
Federal University of Rio Grande do Sul, Institute of Informatics, Porto Alegre, RS, Brazil
Email: {lfmuller, ruas.oliveira, mcluizelli, paschoal, marinho}@inf.ufrgs.br

***Abstract*—In SDN, forwarding devices can only operate correctly while connected to a logically centralized controller. To avoid single-point-of-failure, controller architectures are usually implemented as distributed systems. In this context, recent literature identified fundamental issues, such as device isolation and controller overload, and proposed controller placement strategies to tackle them. However, current proposals have crucial limitations: (i) device-controller connectivity is modeled using single paths, yet in practice multiple concurrent connections may occur; (ii) peaks in the arrival of new flows are only handled on-demand, assuming that the network itself can sustain high request rates; and (iii) failover mechanisms require predefined information, which, in turn, has been overlooked. This paper proposes Survivor, a controller placement strategy that addresses these challenges. The strategy explicitly considers path diversity, capacity, and failover mechanisms at network design. Comparisons to the state-of-the-art on survivable controller placement show that Survivor is superior because (a) path diversity increases the survivability significantly; and (b) capacity-awareness is essential to handle overload during both normal and failover states.**

I. Introduction

Software-Defined Networking defines a new architecture in which the control plane of network forwarding devices is moved to a logically centralized controller. Although beneficial for network management, SDN created an inherent dependency relationship between the forwarding devices and the controller. More precisely, devices need to remain connected to the controller in order to operate properly, otherwise they apply outdated policies and fail to deal with unspecified flows [1]. Therefore, a key challenge is to ensure the connectivity between controller and devices in face of harmful events in the network. Hereafter, this challenge will be referred to as *control plane survivability* (or *survivability*, for short).

SDN architectures avoid single-point-of-failure by implementing controllers as distributed systems [2], [3]. Although replication increases survivability, fundamental aspects make it insufficient. Particularly, two main issues must be properly addressed. First, disruptions in the network can physically isolate forwarding devices from controller instances. Second, high network demand may overload a controller replica, negatively affecting responsiveness.

In order to tackle these issues, literature has proposed different controller placement strategies [4]–[6]. However, these proposals have three main limitations. First, connections between forwarding devices and controller instances are modeled using single paths; yet, in practice multiple concurrent connections may occur. Second, changes in traffic load (such as churn in the arrival of new flows) can only be handled on-demand; this aspect assumes that the network has been planned in advance for high request rates. Lastly, failover mechanisms depend on predefined lists of backup controllers; such a list is currently composed ad-hoc, which, in turn causes these mechanisms to behave inefficiently.

This paper proposes *Survivor*, a novel controller placement strategy that overcomes the above shortcomings. Survivor has three main benefits. First, connectivity is enhanced by explicitly considering path diversity. Second, controller overload is avoided proactively by adding capacity-awareness in the controller placement. Third, failover mechanisms are improved by means of a methodology for composing their list of backups. Comparisons are performed to evaluate the performance of the proposed solution. The main contributions of this paper are threefold:

- **Significant reduction on connectivity loss.** Correctly exploring the path diversity of the network (i.e., connectivity-awareness) reduces the probability of connectivity loss in *around 66%* for single link failures. Moreover, simply adding path diversity to current solutions is not sufficient; the proposed solution still presents *between 2 and 3 times less* disconnected devices in worst case failures.
- **More realistic controller placement strategy.** Survivor is the first strategy to consider capacity-awareness proactively, since previous work handled requests churn on-demand. Experiments show that capacity planning is essential to avoid controller overload, specially during failover. Additionally, the placement strategy is implemented as an optimization model in order to generate *optimal results*.
- **Smarter recovery mechanisms.** Survivor encompasses heuristics for defining a list of backup controllers. A methodology for composing such lists, modeled as a generic heuristic framework, is developed and two heuristics are implemented. As a result, the converging state of the network can improve considerably depending on the selected heuristic.

The remainder of this paper is organized as follows: Section II introduces the fundamental concepts related to this

paper and discusses the main limitations of related work. Section III presents the proposed strategy, while experiments are evaluated in Section IV. Section V discusses the final considerations and indicates possible future work.

II. Background and Related Work

This section begins by identifying the considered set of failures, and currently available failover mechanisms.

A. Failures and Failover Mechanisms

SDN establishes a dependency relationship between the controller and the forwarding devices. More specifically, the lack of connectivity between both systems causes network malfunction, because devices get outdated and cease to receive instructions on how to forward new flows. Therefore, it is important to analyze possible failure scenarios and the available failover mechanisms[1].

Failure scenarios. In line with previous work [4], [5], the failures considered in this paper occur independently and in both the data plane and control plane. In the data plane, failures arise upon disruptions in links or forwarding devices. In turn, control plane failures happen due to software or hardware malfunction on the machine hosting the controller. Pragmatically, in both cases the result of a failure is the loss of connectivity between a set of routing devices and controller instances.

Failover mechanisms. The OpenFlow Protocol [8] provides mechanisms that allow a forwarding device to react upon loss of connectivity with the controller. Towards this end, devices maintain *auxiliary connections* and a *list of backup controllers*. Auxiliary connections with the controller can be defined, established and kept open in order for the device to avoid unnecessary delays. Moreover, they can be established over any path in the network that reaches the controller (e.g., link/node-disjoint paths, shortest-paths, widest-paths). In turn, the list of backup controllers defines the order in which the device will attempt to connect to different controller instances. This mechanism is used when all connections to the primary controller instance are lost (either due to massive link failures or controller failure).

B. Survivable Controller Placement

The Controller Placement Problem [9] is an optimization problem which consists of finding the best placement for controller instances such that a given metric is optimized. Accordingly, the *Survivable* Controller Placement consists of finding the best placement that optimizes a survivability-related metric. This subsection reviews work that attempt to maximize control plane survivability according to two metrics: network disruptions [4], [5] and controller overload [6].

In respect to disruptions, current controller placements minimize the likelihood of disconnection. Zhang et al. [4] defines a min-cut algorithm that specifies clusters of forwarding devices and then places one controller instance in each cluster's centroid. In turn, Hu et al. [5] chooses controller instances such that the chance of connectivity loss is minimized. In both strategies, connections are defined according to the shortest-path between controller instances and forwarding devices.

In contrast, Bari et al. [6] study the dynamic placement of controller instances over the network topology. The authors consider that the initial position is somehow predefined and are limited to turn on/off controllers according to a particular network load. Hence, controller-device connections are defined dynamically and on-demand.

C. Limitations with Current Proposals

Current failover mechanisms (§II-A) allow devices to recover from some link/node/controller failures. When a failure breaks the primary connection to a controller, the forwarding device may use pre-calculated auxiliary connections. If the disruption renders the controller instance unavailable, a device follows a predefined list in order to attempt to connect to other controller instances. The mechanics on how to define auxiliary connections and the list of backup controllers are currently unspecified.

In regard to controller placement strategies (§II-B), literature proposes to tackle either controller-device disconnections or controller overload. While the former is currently limited to using shortest-paths only, the latter assumes that controllers have been already optimally placed and simply determines the assignment of devices to controllers. In sum, a proper solution needs to efficiently recover from controller-device disconnections whenever feasible and optimally position controllers to reduce the chance of overload before and after failures.

III. Survivor

A. Overview

As previously discussed, Survivor deals with three main aspects: connectivity, capacity, and recovery. Towards this end, the strategy is divided in two parts. The first defines placement for controller instances, while the second specifies a list of backup controllers for each device in the network. Both parts are expressed as optimization problems.

Placing controller instances. The first part requires finding optimal placements for controller instances such that *connectivity is maximized* and *capacity constraints are satisfied*. In order to maximize connectivity, the algorithm chooses positions that yield the highest number of node-disjoint paths between the forwarding devices and the controller instance they connect to. In turn, dealing with resource constrains requires some intuition about device demands and controller capacity. Estimates for device demands can be obtained with network measurements [10], while controllers capacity can be inferred through experimentation [11]. In addition, upon disruptions in the network, some forwarding devices may be moved from one controller instance to another. This situation may cause a controller to get overloaded, leading to request

[1]This paper only considers failover mechanisms that help reestablishing the controller-device connections. For mechanisms that provide minimal functionality when connections cannot be reestablished, the interested reader may refer to the data plane connectivity literature [7] or the OpenFlow Protocol Specification [8].

delays or losses. To sustain exceeding demands, each controller has a percentage of capacity reserved as backup. This algorithm is implemented using Integer Linear Programming (ILP), which guarantees optimality. The model is presented in Section III-B.

Composing a list of backup controllers. The second part consists in defining an ordered list for each device using a given heuristic. Interestingly, several heuristics can be composed using the same principle: optimize a local metric and generate an ordered list. Accordingly, the heuristics used in this paper were implemented using a generic framework. This framework, its related assumptions, and the implemented heuristics are described later (§III-C).

B. Implementation of an Optimal Controller Placement

Input. Tuple $I = \{G(N, L);\ C;\ U_i;\ R_i;\ \mathcal{DP}_{i,j};\ \alpha_i\}$ represents the input of the ILP. The physical topology is denoted by an undirected graph $G = (N, L)$, where nodes $n \in N$ represent forwarding devices and edges $(n, m) \in L$ represent bidirectional links. The set of possible controller instances is given by C. The capacity of each controller $c \in C$ is denoted by U_c, while the request demand of each device n is represented by R_n. Additionally, $\mathcal{DP}_{n,m}$ gives the pre-calculated number of node-disjoint paths between nodes n and m. Finally, $\alpha_c : c \in C$ indicates the percentage of backup capacity set to each controller.

Output. The tuple $V = \{x_{i,j};\ y_{i,j};\ w_{i,j}\}$ represents the variables of the ILP. Device mappings are given by $x_{n,c} \in \{0, 1\}$; they indicate whether device n is mapped to controller c. Controller placements are denoted by $y_{c,n} \in \{0, 1\}$; they indicate whether controller c is placed on top of (i.e., is physically connected to) node n. Lastly, $w_{c,n} \in \mathbb{N}$ counts the number of disjoint paths between controller c and device n; this last variable holds 0 when n is not mapped to c.

Objective. The general goal of the proposed strategy is to maximize connectivity between forwarding devices and controllers instances. This goal is modeled as equation

$$\max \quad \frac{\sum_{c \in C} \sum_{n \in N} w_{c,n}}{|N|}, \tag{1}$$

which maximizes the average of disjoint paths between devices and their controller.

Constraints. The constraints of this ILP model can be divided into three categories: *placement-related*, *capacity-related*, and *connectivity-related*.

The first four constraints [Constr. (C1)–(C4)] are *placement-related*. They ensure correctness for both the placement of controller instances in the topology and the mapping of devices to controller instances. Constraint (C1) guarantees that, for all devices (i.e., $\forall n \in N$), each device n will be controlled by exactly one controller c (i.e., $\sum_{c \in C} x_{n,c} = 1$).

$$\sum_{c \in C} x_{n,c} = 1 \qquad \forall n \in N. \ \text{(C1)}$$

TABLE I
SUMMARY OF SYMBOLS FOR THE ILP MODEL.

Symbol	Definition
N	Set of network nodes (i.e., forwarding devices).
L	Set of links.
C	Set of controllers.
U_c	Maximum number of requests that controller c can handle.
R_n	Number of requests of each device n.
$\mathcal{DP}_{n,m}$	Number of disjoint paths between node pairs n and m.
α_c	Percentage of capacity reserved as backup in controller c.
$x_{n,c} \in \{0, 1\}$	Indicates whether device n is mapped to controller c.
$y_{c,n} \in \{0, 1\}$	Indicates whether controller c is placed onto node n.
$w_{c,n} \in \mathbb{N}$	Indicates the number of disjoint paths between device n and controller c. *(0 if device n is not mapped to controller c)

Constraint (C2) is twofold: (i) assigned controllers must be active (i.e., $x_{n,c} \leq \sum_{m \in N} y_{c,m}$); and (ii) a controller cannot be placed in more than one location (i.e., $\sum_{m \in N} y_{c,m} \leq 1$).

$$x_{n,c} \leq \sum_{m \in N} y_{c,m} \leq 1 \qquad \forall c \in C, \forall n \in N. \ \text{(C2)}$$

Constraint (C3) ensures that two controllers will not be placed in the same location, while Constraint (C4) guarantees that if the controller c is placed onto node n (i.e., $y_{c,n} = 1$), then the device n should be mapped to controller c (i.e., $x_{n,c} = 1$).

$$\sum_{c \in C} y_{c,n} \leq 1 \qquad \forall n \in N; \ \text{(C3)}$$

$$y_{c,n} \leq x_{n,c} \qquad \forall n \in N, \forall c \in C. \ \text{(C4)}$$

The *capacity-related* constraint [Constr. (C5)] ensures that the controller capacity will not be exceeded. It considers both normal and backup capacities in the same formulation. Specifically, for all controller instances (i.e., $\forall c \in C$), a controller normal capacity [i.e., $(1 - \alpha_c) \cdot U_c$] will not be exceeded by the sum of all demand assigned to it (i.e., $\sum_{n \in N} x_{n,c} R_n$).

$$\sum_{n \in N} x_{n,c} R_n \leq (1 - \alpha_c) \cdot U_c \qquad \forall c \in C. \ \text{(C5)}$$

The two final constraints [Constr. (C6.1) and (C6.2)] are *connectivity-related*. They are used to count the number of disjoint paths in each controller-device mapping. These constraints are trickier to interpret, as they work in conjunction and depend on the objective function. The constraints represent two cases. When device n is mapped to controller c ($x_{n,c} = 1$), $w_{c,n}$ is upper bounded by the disjoint paths between m and c's placement (n), which is given by $\mathcal{DP}_{n,m}$ [Constr. (C6.1)]. In contrast, when device n is not mapped to controller c ($x_{n,c} = 0$), $w_{c,n}$ is upper bounded by 0 [Constr. (C6.2)]. In short, Constraint (C6.1) is always set to $\mathcal{DP}_{n,m}$, while Constraint (C6.2) is either 0 or infinity, depending on $x_{n,c}$.

$$w_{c,n} \leq \sum_{m \in N} y_{c,m} \cdot \mathcal{DP}_{n,m} \qquad \forall c \in C, \forall n \in N; \ \text{(C6.1)}$$

$$w_{c,n} \leq x_{n,c} \cdot \kappa \qquad \forall c \in C, \forall n \in N : \kappa \to \infty. \ \text{(C6.2)}$$

Constraints (C6.1) and (C6.2) set upper bounds for $w_{c,n}$, which are the exact values that it should hold. In other words,

$w_{c,n}$ should always be assigned its upper bound. This is ensured by the objective function [Eq. (1)], which always tries to maximize the value for $w_{c,n}$.

C. Heuristics for Defining Lists of Backup Controllers

This subsection first introduces the assumptions and notations used as base for the algorithms. Next, it describes a generic framework for designing heuristics that compose the lists of backup controllers, eliminating the need to manually determine the list. Then, it presents two heuristics as proof of concept, one based on proximity and the other based on residual capacity.

Assumptions. The generic framework makes three assumptions. First, controller failures are considered the worst cases for disconnections; while this is not always true, it can be expected for all practical purposes since, as demonstrated in experiments (§IV), the proposed strategy greatly reduces the chance of disconnections due to other disruptions. Second, controllers are assumed to fail independently; this can also be presumed in practice, as current architectures implement mechanisms to ignore controller instances when they fail, protecting the remaining ones [2], [3]. Third, each index on the list is considered to be independent, as all instances will follow their ordered list; in other words, all devices are expected to use the same index (mostly the first) on the same failure scenario.

Notation. Most notations used in the following algorithms are intuitive. However, some considerations must be made. Particularly, *brackets* ($[.]$), *star* ($*$), and *plus sign* ($+$) have special meanings. *Brackets* are used to define indices in ordered lists, that is, $[i]$ indicates an earlier position than $[j]$ for all $i < j$. *Star* is used to indicate that a given operation is performed to all indices of a list. Lastly, *plus sign* is used to indicate that a comparison is performed to all indices of a list, subject to at least one comparison being true.

Generic framework. The pseudo-algorithm of the framework is described in Algorithm 1. It begins by considering all controller instances that were placed in the topology (line 3). Due to the first and second assumptions, controllers are evaluated independently. Next, for each index on the list (line 4), it initializes a list of candidates, which is composed by all controllers except the one being evaluated (line 5). Similarly to controllers, indices are evaluated independently due to the third assumption.

The following steps select all devices connected to the evaluated controller and optimize the i^{th} index on their list (lines 6-10). Three operations are performed: *select local optimum* (line 8), *set list index* (line 9), and *update candidate list* (line 10). The *select local optimum* operation consists of evaluating all possible candidates, given by set S (i.e., $S \leftarrow K \setminus B_n[*]$), and then selecting the best among them according to a given metric (i.e., $\varphi(n, S) = b$).

Set list index is solely an attribution operation. In contrast, the *update candidate list* operation follows a specification (i.e., δ) to update information about the controllers of the candidate list. It is important to realize that operations performed by the update procedure are temporary, as they are performed in set K, which, in turn, is reset in line 5. The aforementioned operations use two generic procedures: $\varphi(n, S)$ (which finds the local minimum) and δ (which updates candidates). These procedures have specific meanings for each heuristic and will be exemplified in the following.

Algorithm 1: Generic heuristics framework for composing the lists of backup controllers

```
1:  B_n[*] ← ∅, ∀n ∈ N
2:  ListLength ← max(|B_*|)
3:  for c ∈ C : y_{c,+} = 1 do // only get placed controllers
4:      for i ← 1 upto ListLength do
5:          K ← C \ {c} // initialize the list of candidates
6:          for n ∈ N : x_{n,c} = 1 do // only get connected devices
7:              S ← K \ B_n[*] // discard already used
8:              select b ∈ S : φ(n, S) = b // get local optimal
9:              B_n[i] ← b // set b in the i^th index of n's list
10:             update K according to rule δ
11:         end for
12:     end for
13: end for
14: return B_n
```

Proximity-based heuristic. This heuristic attempts to select the closest controller instances to use as backup (in terms of delay or hops). To implement this heuristic, one is required to define $\varphi(n, S)$ as a procedure that takes device n and the set of valid controller candidates S, and returns the closest instance. This can be achieved by implementing a shortest-paths algorithm inside $\varphi(n, S)$. The update rule is not required since controllers do not change positions.

Residual capacity-based heuristic. This heuristic attempts to account for resource consumption while selecting controller instances. More specifically, it *selects* the controller instance that has the highest residual capacity, and *updates* this instance according to the exceeding demand. Accordingly, $\varphi(n, S)$ is a linear search for the maximum residual capacity, while δ adds the demand of device n to controller n in set k. As previously stated, operations performed by the update procedure are temporary.

IV. Evaluation

A. Methodology

This section introduces the methodology employed to compare Survivor with the state-of-the-art on survivable controller placement.

Workload. The analysis takes as input the network design given by each placement strategy and then simulates multiple failure scenarios. Three different WAN topologies were considered[2]: Internet2 (10 nodes, 15 links), RNP (27 nodes, 33 links) and GÉANT (40 nodes, 61 links). These environment assumes that nodes would be OpenFlow-capable devices and that controllers could be provisioned at any of these nodes' locations. Additionally, capacities and demands were based on studies from literature. All controllers have the same capacity,

[2]Topology maps were acquired in January 2014, from the Internet Topology Zoo: http://www.topology-zoo.org/

1800 kilorequests/s [11], while each forwarding device in turn generates 200 kilorequests/s [10]. Finally, the percentage of resources set as backup is 30%.

Comparison Method. Survivor (SVVR) is compared with two versions of the one developed by Zhang et al. [4]. The original version of the algorithm is composed by two steps: (i) identify partitions in the network topology with minimum cuts across boundaries; and (ii) assign one controller to the location which has the shortest paths to all devices in the same cluster. This strategy is denoted *MCC*, which stands for MinCut-Centroid. Additionally, a natural extension of this algorithm, denoted *MCC+*, consists of considering all available paths between controller and devices. The algorithm runs as usual, but after controller instances are placed, connections are made using all available node-disjoint paths.

Metrics. Resulting topology designs are analyzed in terms of resilience to disruptions and overload. Resilience represents the capacity of the evaluated strategies to sustain loss of connectivity upon controller or link disruptions. In its turn, overload indicates how controller instance loads are distributed during normal operation and after failures. Four metrics are used, the first two quantify resilience, while the last two measure overload. For fairness of comparison, the first evaluation considers the same *resilience equation* used by Zhang et al. [4]; this equation measures the probability of connectivity loss in a uniform failure scenario. The second metric uses *cardinal of edge-connectivity* [12] to calculate the percentage of disconnected components given all possible failure scenarios. The third metric counts the *number of overloaded controllers* on all possible failure scenarios. Finally, the fourth metric shows the *load distribution* for each of the controller instances.

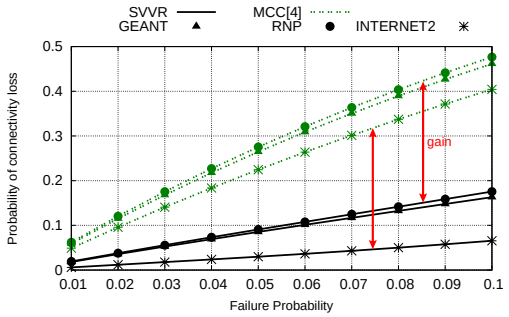
B. Results

This section compares the Survivor (SVVR) against literature (MCC/MCC+) using the four metrics presented earlier. For ease of exposure, Survivor is illustrated using full-black lines, whereas MCC (or MCC+) using dotted lines (which are green, if color is available).

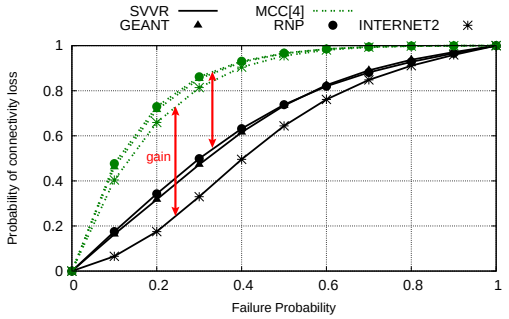
1) Survivor reduces the probability of connectivity loss: The first evaluation (shown in Fig. 1) considers that network elements (i.e., nodes and links) fail independently and then measures the probability of a controller-device connection be interrupted. Axis x represent the failure probability assigned to all network elements (that is, 0.01 means all nodes and links have 1% chance of failing); in turn, the y axis yields the probability of a connectivity loss. Fig. 1(a) shows values from 1% to 10% (the same used by Zhang et al. [4]), while Fig. 1(b) extrapolates the analysis to the entire range (0% to 100% probability of failure).

Fig. 1(a) shows that Survivor outperforms MCC consistently, and also substantially, for all probabilities (in the x axis) but the extremes 0 and 1. More importantly, the curves representing Survivor behavior have a slower growing factor than MCC. As can be observed, when elements have 1% chance of failure both strategies behave similarly (less then 8% chance of connectivity loss). As the chance of failure increases to 5%, the probability of connectivity loss for Survivor is less than half of that for MCC. In this case, Survivor has stayed bellow 10%, while MCC has increased above 20%. Moreover, although the relative difference between both strategies decrease, Fig. 1(b) shows that Survivor reaches near 100% of connectivity loss much later than MCC. While the probability of connectivity loss of Survivor is still around 80% when the chance of failure is 60%, MCC is already near 100%.

The observed improvement happens because in MCC a single element failure can break one or multiple controller-device connections; yet in Survivor, it is required at least one element failure per controller-device path. This seems to be a direct effect of exploring path diversity during placement. The next experiment provides further evidence of this behavior and shows that considering path diversity after placement is insufficient.



(a) Probability distribution 0.01% to 10%.



(b) Probability distribution 0% to 100%.

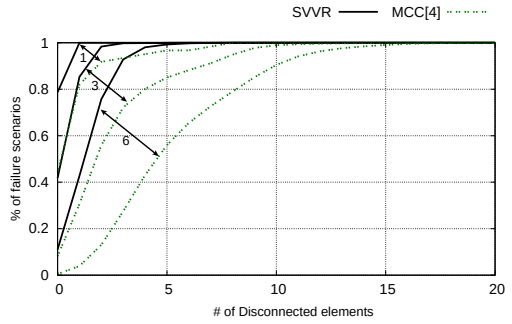
Fig. 1. Probability of controller-device connectivity loss for different failure probabilities. Values are obtained using the equation in [4]. *The lower the curve, the better.*

2) Path diversity increases the network survivability, and it requires explicit consideration to be fully explored: Fig. 2 expresses a CDF of the number of disconnected elements, when enumerating all possible failure scenarios. Three variations are evaluated: single link failure, 3 concurrent link failures, and 6 concurrent link failures. These variations will be denoted $k = 1$, $k = 3$, and $k = 6$, respectively. Moreover, results were similar for all topologies, thus, due to space constraints, only the larger one – GÉANT – is shown.

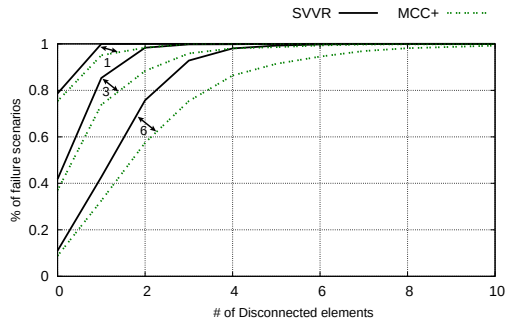
Results in Fig. 2(a) compare Survivor to MCC. For single link failures, $\text{SVVR}_{k=1}$ achieves twice the protection (80%)

than $\text{MCC}_{k=1}$. Moreover, the worst case in $\text{SVVR}_{k=1}$ is only one disconnected device, while in $\text{MCC}_{k=1}$ there may be up to 8 disconnected devices. *Finding: the probability of a single link failure affecting all connections of even one device is much smaller (*$100\% - \frac{\simeq 20\%}{\sim 60\%} \approx 66\%$ *less) than that of affecting a single connection of one or multiple devices.*

Fig. 2(a) also shows that Survivor in the scenario with 3 and 6 concurrent link failures occasionally outperforms MCC in single link failures. $\text{SVVR}_{k=3}$ and $\text{MCC}_{k=1}$ behave similarly in 80% of cases, but $\text{SVVR}_{k=3}$ outperforms $\text{MCC}_{k=1}$ in the remaining 20%. More importantly, the remaining 20% represent worst case scenarios; in such scenarios $\text{SVVR}_{k=3}$ disconnects at most 6 devices, whereas $\text{MCC}_{k=1}$ disconnects up to 8. Additionally, for $\text{SVVR}_{k=6}$ and $\text{MCC}_{k=1}$, there is a turning point around 90%. When considering 6 concurrent failures for both, this difference is more noticeable. In $\text{SVVR}_{k=6}$, the worst 20% disconnection cases have between 3 and 11 disconnected devices; in $\text{MCC}_{k=6}$, the same $[3, 11]$ interval happens in 80% of the cases (from 10% up to 90%). In this interval, Survivor is 75% less likely to disconnect the same amount of devices ($100\% - \frac{\simeq 20\%}{\sim 80\%} \approx 75\%$).



(a) Survivor x MCC.



(b) Survivor x MCC+.

Fig. 2. Cumulative distribution function of disconnected devices for all possible cases of 1, 3, and 6 link disruptions in the GÉANT topology. *Curves closer to the top-left side are better.*

Finally, in order to evaluate the effectiveness of the Survivor placement strategy, path diversity is added to MCC after placement; this modification is denoted MCC+. Fig. 2(b) shows that Survivor still outperforms MCC+ in all cases. *Finding: adding path diversity after placement in insufficient, as MCC+ has 2-3 times more disconnected devices in worst cases*; $\frac{3}{1} = 3$, $\frac{19}{6} \approx 3$, and $\frac{22}{11} = 2$, for $k = 1$, $k = 3$, and $k = 6$, respectively ($\frac{worstMCC+}{worstSurvivor}$).

3) Network convergence after disruptions is highly sensible to predefined information in failover mechanisms: Fig. 3 shows the number of overload scenarios considering all controller instances. Each overload scenario counts as a single controller instance that had a load higher than 100%. The analysis considers three cases of network operation: normal, post-failure with proximity-based failover heuristic, and post-failure with residual capacity-based failover heuristic. In normal operation Survivor has not a single overloaded controller instance. This happens because the placement strategy is capacity-aware and, thus, avoids overloads when assigning devices to controllers. In contrast, MCC has demonstrated a small number of overloaded controller instances during normal operation. This behavior arises from the fact that MCC only considers a centroid heuristic when assigning devices to controllers. Hence, in some cases, a controller instance happens to have a much higher load.

After disruptions, the failover mechanism employed the lists created by the two heuristics proposed earlier (§III-C). Evaluation shows that failover mechanisms perform very differently depending on the way their backup lists were composed. The residual capacity-based heuristic outperformed the proximity-based one for both MCC and Survivor. Even though Survivor had set backup resources, some instances were still overload in the proximity-based heuristic. This behavior becomes more evident in the next evaluation.

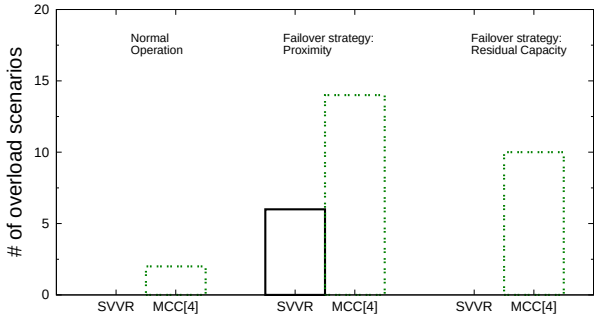


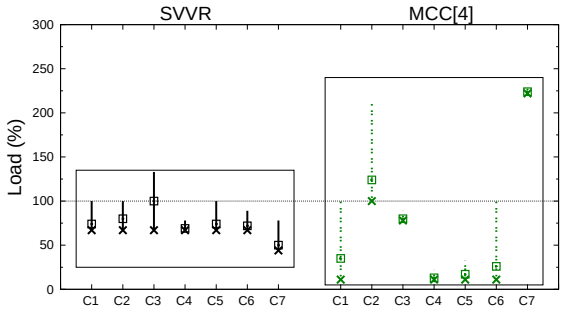
Fig. 3. Number of overload scenarios (controller load >100%) during network normal operation and two post-failover operation cases. *The lower the bars, the better.*

4) Controller overload can be handled proactively by adding capacity-awareness and setting backup resources: The previous evaluation shown that different kinds of information used during failover yield sensible performance impact. This evaluation takes a closer look in the state of the network after convergence to better understand the observed behavior. Towards this end, Fig. 4(a) shows the load on each controller considering all failure scenarios. Values indicate the minimum, maximum, average and mode loads.

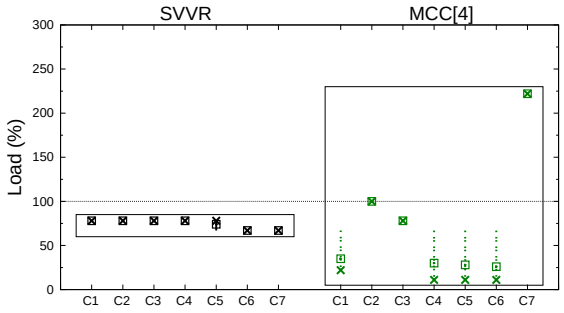
As observed, even though placements end up with enough free capacity, the proximity heuristic (Fig. 4(a)) tends to overload some instance(s) (e.g., C3 in Survivor and C1, C2, and C6 in MCC). For the Survivor placement, the maximum and minimum values have high differences for all controllers

and the average varies, but the mode in particular shows that controller instances have their load close to the minimum in most cases. This happens because most devices select the same backup controller when their primary controller fails, thus leaving other controllers with their original load. The MCC placement shows similar results, but the variation among values tends to be higher because the initial placement is already unbalanced.

In contrast, the residual capacity heuristic (Fig. 4(b)) tends to leave the load more evenly distributed. In the Survivor placement, the maximum, minimum, average, and mode have almost the same value. This happens because devices will attempt to connect to different controllers in the network, and will prefer those with higher residual capacity. As a result, the load on controllers increases, but none of them overloads. In this case, MCC behaves differently than Survivor. This can be explained by the fact that in MCC, controllers C2, C3, and C7 were assigned a higher load (in comparison to the other controllers) during the initial placement – in fact, C2 and C7 were overloaded. As a result, controllers C1, C4, C5, and C6 were the only ones assigned during failover and their load variations are similar.



(a) Proximity-based heuristic for failover mechanism.



(b) Residual Capacity-based heuristic for failover mechanism.

Fig. 4. Mininum, maximum, average (square), and mode (star) values for the load in each controller instance (labeled C1-C7) after failover. Placement strategies (Survivor and MCC) are boxed for ease of exposition and comparison. Results are shown for the GÉANT topology.

V. Conclusion and Future Work

SDN has altered the requirements for network survivability, as the main challenge became maintaining the connectivity between the controller and the forwarding devices upon disruptions in the network. Previous work has relied on strong assumptions in order to simplify the problem. This paper presented Survivor, a novel approach that is shown to improve survivability in SDN. The main improvements in comparison to previous work are: (a) considering multiple paths explicitly; (b) dealing with capacity during initial placement; and (c) developing smarter failover mechanisms.

Main findings showed that: (i) exploring path diversity during placement reduces the chance of connectivity loss; (ii) adding path diversity to current solutions was not sufficient; (iii) adding capacity-awareness to the solution was essential, otherwise controller instances might get overloaded on both normal and failover scenarios; and (iv) the heuristic for selecting backups during failover has substantial impact on the converging state of the network. In future work, the evaluation should be extended to consider specific environments, such as other topologies and failure scenarios. Additionally, more aspects could be explored in order to further improve survivability.

Acknowledgment

This work has been supported by FP7/CNPq (Project SecFuNet, FP7-ICT-2011-EU-Brazil).

References

[1] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM Computer Comm. Review*, vol. 44, no. 2, April 2014.

[2] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in *Proc. OSDI*. USENIX, 2010, pp. 1–6.

[3] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proc. SIGCOMM HotSDN workshop*. ACM, 2012, pp. 19–24.

[4] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *Proc. GLOBECOM*. IEEE, 2011, pp. 1–6.

[5] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *Proc. IM*. IEEE, 2013, pp. 672–675.

[6] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *Proc. CNSM*. IEEE, 2013, pp. 1–8.

[7] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, "Ensuring connectivity via data plane mechanisms," in *Proc. NSDI*. USENIX, 2013, pp. 113–126.

[8] ONF, "Openflow switch specification 1.4.0," Available at <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf>, 2014.

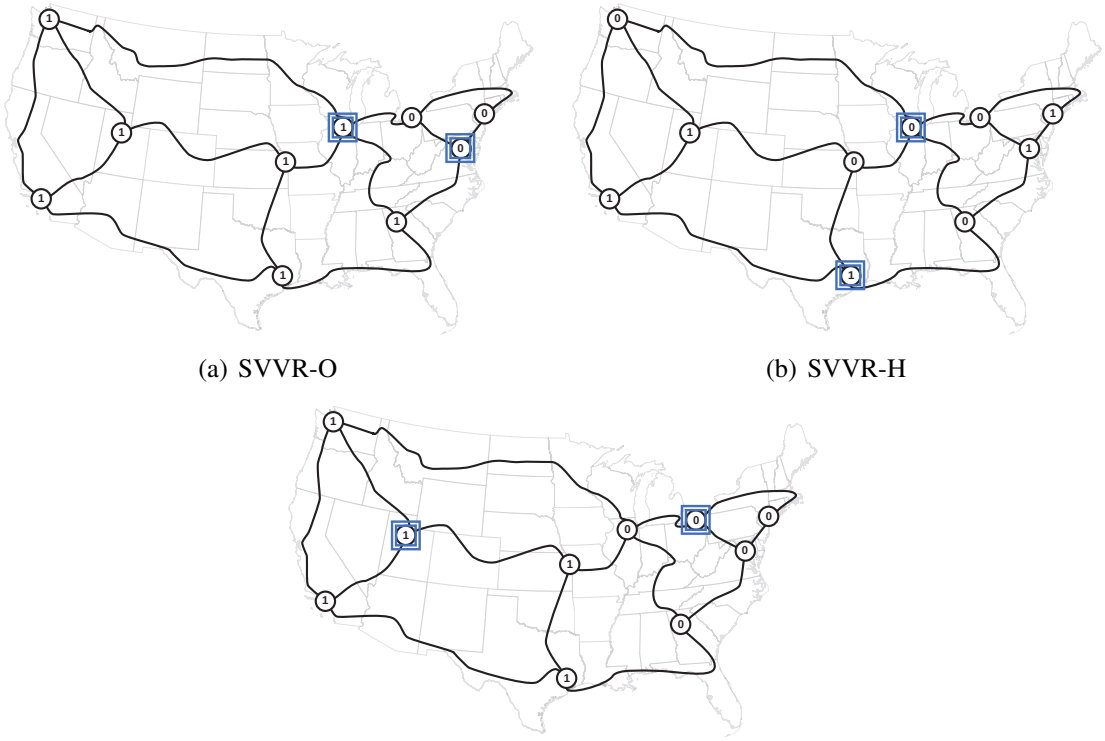
[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. SIGCOMM HotSDN Workshop*. ACM, 2012, pp. 7–12.

[10] I. Cunha, F. Silveira, R. Oliveira, R. Teixeira, and C. Diot, "Uncovering artifacts of flow measurement tools," in *Proc. PAM*. Springer-Verlag, 2009, pp. 187–196.

[11] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. Hot-ICE Workshop*. USENIX, 2012, pp. 10–10.

[12] K. Bagga, L. Beineke, R. Pippert, and M. Lipman, "A classification scheme for vulnerability and reliability parameters of graphs," *Mathematical and Computer Modelling*, vol. 17, no. 11, pp. 13 – 16, 1993.

APPENDIX C - POSICIONAMENTO RESULTANTE DAS ESTRATÉGIAS

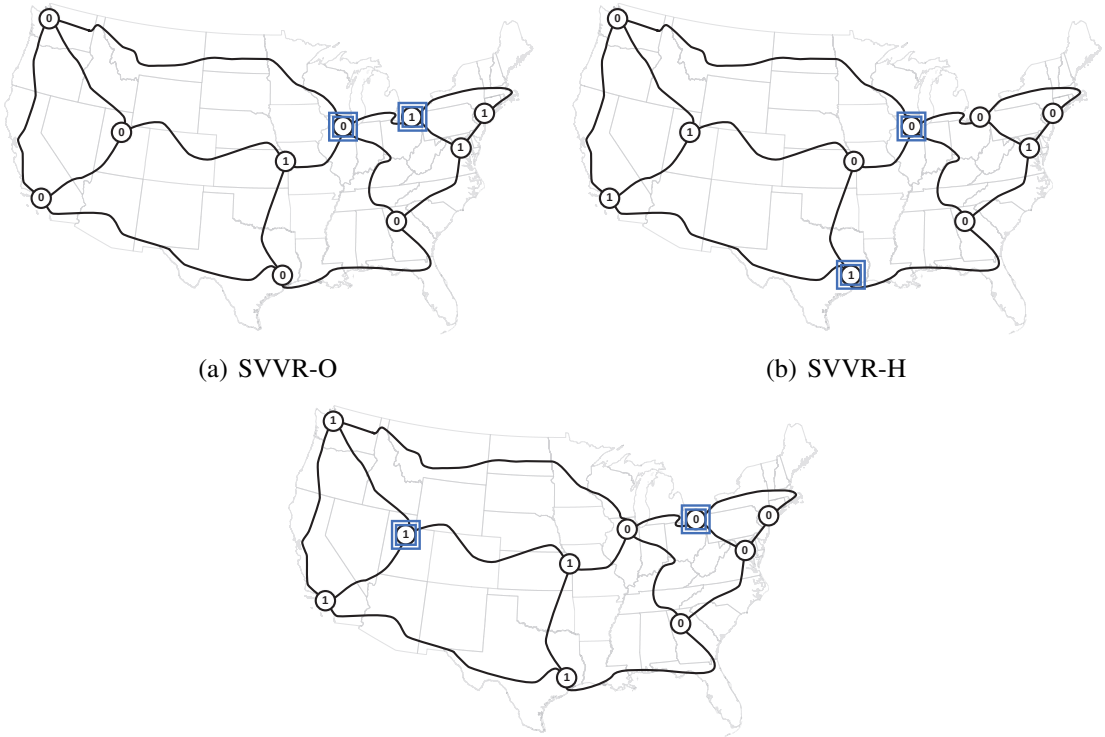


(a) SVVR-O

(b) SVVR-H

(c) MCC

Figura C.1: Internet2 – 20% de reserva de backup.

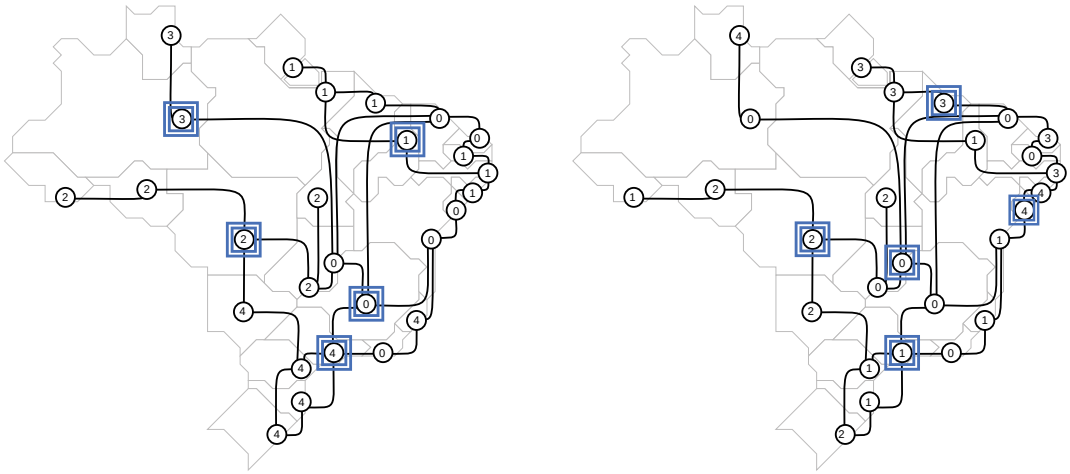


(a) SVVR-O

(b) SVVR-H

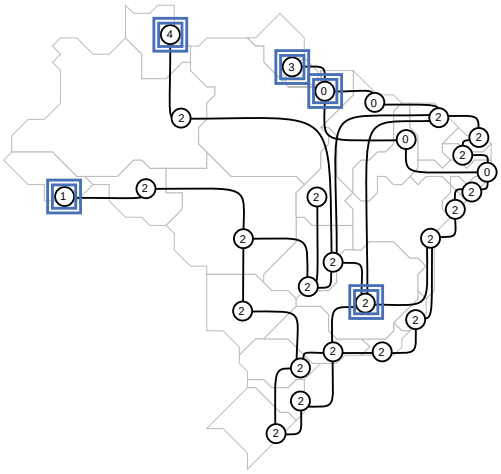
(c) MCC

Figura C.2: Internet2 – 30% de reserva de backup.



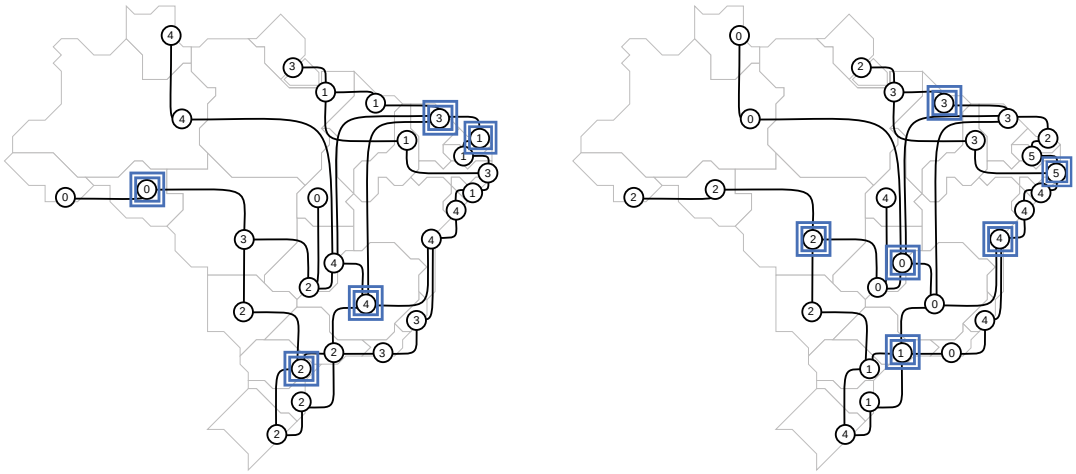
(a) SVVR-O

(b) SVVR-H



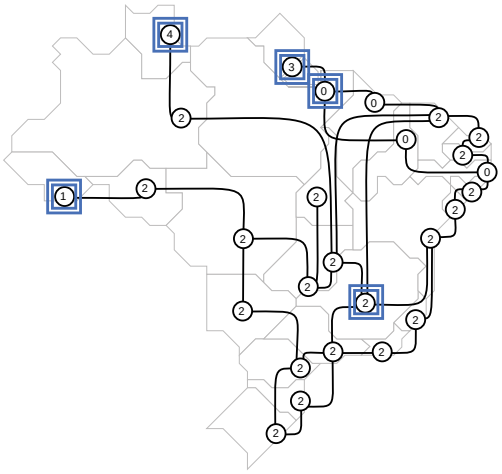
(c) MCC

Figura C.3: RNP – 20% de reserva de backup.



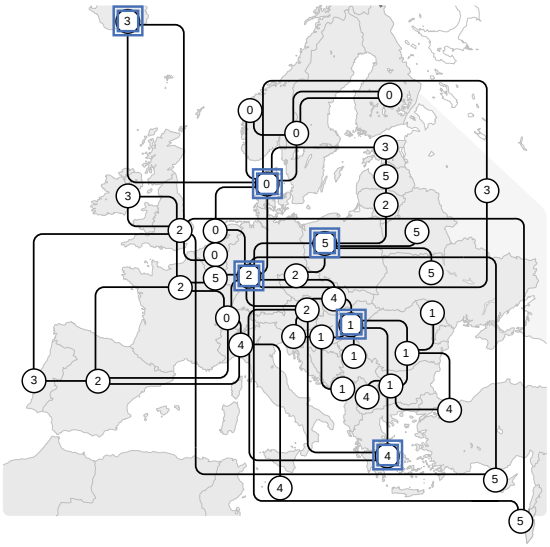
(a) SVVR-O

(b) SVVR-H

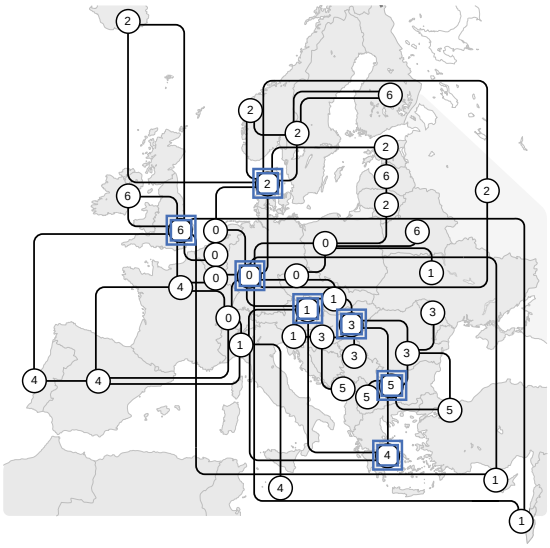


(c) MCC

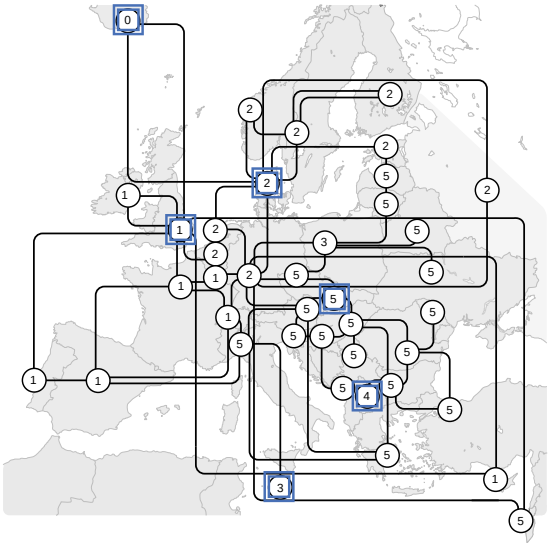
Figura C.4: RNP – 30% de reserva de backup.



(a) SVVR-O

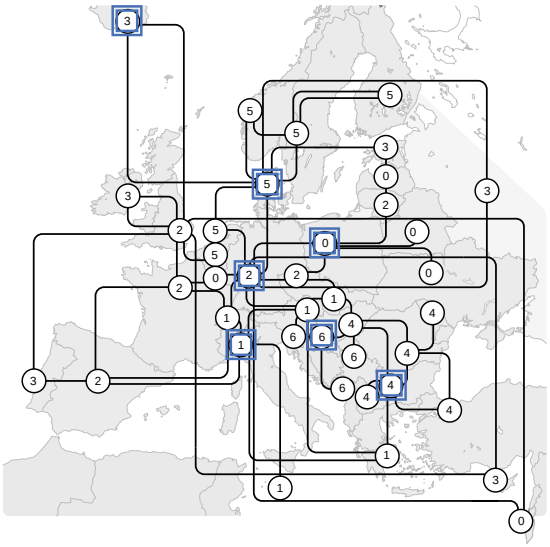


(b) SVVR-H

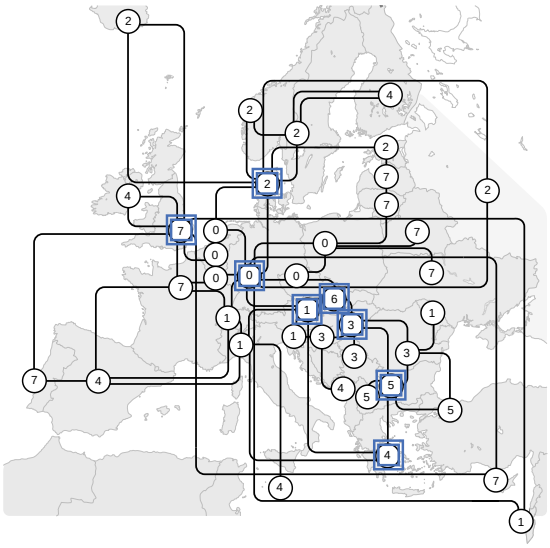


(c) MCC

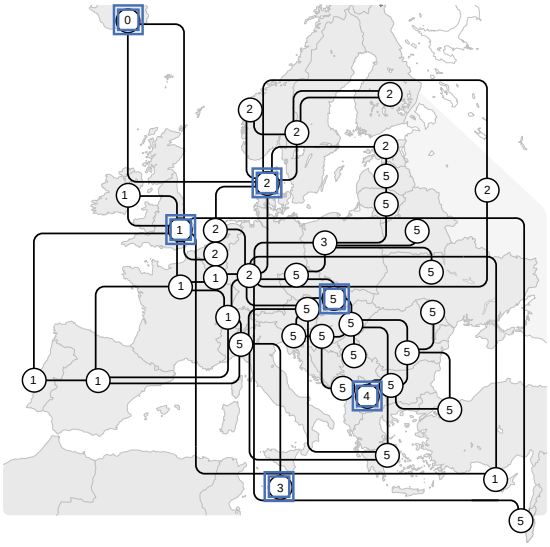
Figura C.5: GÉANT – 20% de reserva de backup.



(a) SVVR-O



(b) SVVR-H



(c) MCC

Figura C.6: GÉANT – 30% de reserva de backup.