Arquitetura e Mapeamento de Serviços Virtualizados de Rede Tolerantes a Falhas e Intrusão

Vinicius Fulber-Garcia, Giovanni Venâncio, Elias P. Duarte Jr.

Universidade Federal do Paraná (UFPR) – Depto. Informática
Caixa Postal 19018 81531-990, Curitiba - PR

{vfgarcia, gvsouza, elias}@inf.ufpr.br

Abstract. *The IETF architecture for virtual network services defines Service Function Chains (SFCs) as compositions of multiple Virtualized Network Functions (VNFs). SFCs are also based on components for forwarding traffic, mainly Service Classifiers (SCs) and Service Function Forwarders (SFFs). This work presents a new strategy to replicate, map and tolerate crash and intrusion faults considering all components of the IETF SFC architecture; previous works consider only crash faults of VNFs. A federated environment with multiple candidate domains to host VNF, SC and SFF instances is assumed. The forwarding components are replicated and then mapped onto the federated environment using optimization based on connectivity criteria. A genetic heuristic is proposed, in addition to an exact solution based on Integer Linear Programming. Results show the efficiency of the heuristic and that it returns near-optimal results. In addition, a prototype of the proposed strategy was implemented and experimets show the ability to keep virtual services available under both crash faults and intrusion resulting from a man-in-the-middle attack.*

Resumo. *A arquitetura para serviços virtuais de rede da IETF define Service Function Chains (SFCs) como composições de múltiplas Virtualized Network Functions (VNFs). As SFCs são baseadas em componentes para o encaminhamento de tráfego, principalmente os classificadores Service Classifiers (SCs) e encaminhadores Service Function Forwarders (SFFs) de tráfego. Este trabalho apresenta uma nova estratégia para replicar, mapear e tolerar falhas por parada e intrusão de todos os componentes da arquitetura de SFC da IETF; trabalhos anteriores consideram apenas falhas por parada de VNFs. Assume-se um ambiente federado com múltiplos domínios candidatos a hospedarem instâncias de VNF, SC e SFF. Os componentes de encaminhamento são replicados, sendo então mapeados no ambiente federado utilizando otimização baseada em critérios de conectividade. Uma heurística genética é proposta, além de solução exata baseada em programação linear inteira. Resultados mostram a eficiência da heurística e que retorna resultados próximos ao ótimo. Além disso, um protótipo da estratégia proposta foi implementado e demostrou a capacidade de manter o serviço disponível sob falhas por parada e intrusão resultante de ataque man-in-the-middle.*

1. Introdução

O paradigma *Network Function Virtualization* (NFV) permite desacoplar as funções de rede da sua alternativa clássica de implementação baseada em hardware dedicado

[R. et al. 2016]. Assim, funções de rede passam a ser implementadas como VNFs (*Virtual Network Functions*) através de tecnologias de virtualização amplamente disponíveis e executáveis em *hardware* de prateleira [Fulber-Garcia et al. 2019]. Uma *Service Function Chain* (SFC), por sua vez, é um serviço virtual de rede definido como uma composição de VNFs para a execução de funcionalidades elaboradas [Fulber-Garcia et al. 2020]. A *Internet Engineering Task Force* (IETF) define uma arquitetura para SFC [Quinn et al. 2015], que é baseada em três componentes principais: *Service Classifier* (SC), *Service Function Forwarder* (SFF) e VNF. Ao empregar tais tecnologias, o paradigma NFV flexibiliza e reduz os custos de instalação e operação de redes, além de promover uma descentralização no mercado de desenvolvimento de funções e serviços de rede. Entretanto, é necessário garantir a alta disponibilidade das soluções baseadas em NFV, permitindo a sua adoção em grande escala em redes comerciais.

Apesar de que estratégias diversas para a construção de serviços virtuais tolerantes a falhas já foram propostos [Venâncio et al. 2021, Ghaznavi et al. 2020, Qu et al. 2016], todos consideram apenas falhas por parada (*crash*) de VNFs. Sendo assim, este trabalho propõe uma nova estratégia para a replicação, mapeamento e tolerância a falhas em SFCs que considera a possibilidade dos três componentes (VNF, SC e SFF) da arquitetura SFC da IETF sofrerem falhas *crash* ou intrusão (falhas bizantinas). A estratégia proposta considera um ambiente federado com vários pontos de presença capazes de hospedar e executar os componentes de uma SFC. Tais componentes são replicados e mapeados no ambiente federado considerando otimização baseada em critérios de conectividade, que aumentam a probabilidade de uma réplica se manter atingível depois de particionamentos da rede. Além de uma solução ótima baseada em programação linear inteira, uma heurística genética é proposta. Resultados de simulação demonstram que a heurística produz mapeamentos próximos ao ótimo em um tempo de execução viável. Além disso, uma implementação de protótipo da estratégia proposta de tolerância a falhas demonstra sua capacidade de manter serviços corretos mesmo após a ocorrência de falhas por parada ou intrusões decorrentes de ataques *man-in-the-middle*.

O restante do artigo está organizado como segue. A Seção 2 são apresenta a arquitetura SFC tolerante a falhas proposta. A Seção 3 descreve o conjunto de premissas e objetivos adotados para otimização do mapeamento da arquitetura tolerante a falhas em ambientes federados, além de apresentar uma heurística genética que implementa o mesmo. Experimentos e resultados são apresentados na Seção 4. Trabalhos relacionados são descritos na Seção 5. Finalmente, a conclusão segue na Seção 6.

2. Uma Arquitetura SFC Tolerante a Falhas e Intrusão

Diversos serviços de rede oferecem funcionalidades essenciais para manter uma infraestrutura de rede operacional. Uma falha, por exemplo, de serviços de segurança ou de engenharia de tráfego pode comprometer o funcionamento global da rede, além de implicar em consequências desastrosas. Assim, é necessário garantir a alta disponibilidade de serviços que são críticos para o bom funcionamento da rede.

Uma *Service Function Chain* (SFC) é uma composição de múltiplas VNFs que fornecem um serviço de rede [Halpern and Pignataro 2015]. Funções arbitrárias podem ser encadeadas em uma ordem predefinida através da qual o tráfego é encaminhado e processado. A *Internet Engineering Task Force* (IETF) propôs uma arquitetura para SFCs,

ilustrada na Figura 1. Além das próprias funções de rede, a arquitetura consiste em outros componentes responsáveis pela criação e operação das SFCs. Os principais componentes desta arquitetura são o *Service Classifier* (SC) e o *Service Function Forwarder* (SFF), descritos a seguir.

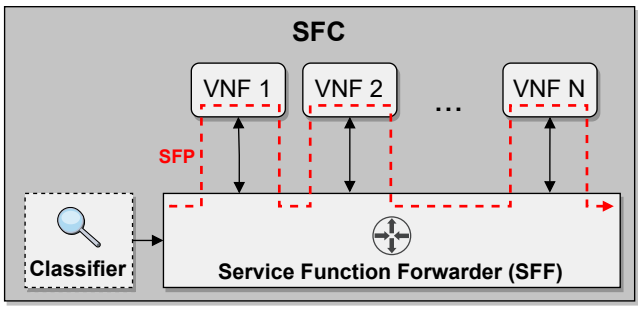

Figura 1. Componentes da arquitetura SFC da IETF.

O SC é o componente responsável por receber e classificar o tráfego de rede, *i.e.*, determinar, de acordo com um conjunto de regras, para qual SFC o tráfego deve ser encaminhado. As regras podem utilizar campos do cabeçalho do pacote, *e.g.*, endereços de origem/destino e portas, ou até mesmo o conteúdo do pacote. Depois de classificado, o SC encapsula o pacote com um conjunto de informações que formam o *Network Service Header* (NSH) [Quinn et al. 2018]. O NSH consiste em informações para encaminhamento de tráfego através da SFC, determinando o seu *Service Function Path* (SFP): sequência de VNFs para as quais o tráfego deverá ser encaminhado através da SFC. É importante notar que mais de um SFP pode ser especificado para uma mesma SFC, permitindo que restrições adicionais possam ser empregadas para determinar qual caminho o tráfego de rede tomará de acordo com condições específicas.

O SFF é o componente responsável por receber o tráfego do SC e realizar o encaminhamento dos pacotes para o SFP correspondente. Conforme ilustrado na Figura 1, inicialmente o SFF envia o tráfego recebido pelo SC para a primeira VNF da SFC. A VNF recebe e processa o tráfego, retornando os pacotes processados novamente para o SFF. O SFF recebe novamente esses pacotes e os encaminha para a próxima VNF segundo o SFP indicado no NSH. Este processo se repete até que o tráfego tenha passado por todas as VNFs do SFP. Ao final, o SFF remove o NSH dos pacotes e os encaminha aos seus destinos.

A seguir é proposta a estratégia denominada FT-SFC (*Fault-Tolerant* SFC) que prevê a resiliência dos componentes da arquitetura SFC da IETF e utiliza técnicas de replicação [Santos et al. 2004] para garantir: (*i*) que nenhum pacote seja perdido ou duplicado na presença de falhas; e (*ii*) que a arquitetura tolere f falhas *crash* ou bizantinas [Lamport et al. 2019]. Uma falha bizantina é uma falha arbitrária, que é normalmente utilizada para representar um componente que sofreu uma intrusão decorrente de um ataque. Neste trabalho um ataque consiste na modificação não autorizada de pacotes [Ziwich et al. 2005] que percorrem a SFC.

O modelo de sistema adotado é assíncrono, isto é, não há garantias temporais fortes, cada componente do sistema pode atrasar arbitrariamente para processar e enviar pacotes na SFC. Todos os componentes de uma SFC - SC, SFF ou VNF - podem sofrer

falhas bizantinas. Por decorrência, os componentes podem também sofrer falhas por parada, ou podem omitir pacotes específicos. Entretanto, para cada componente o número máximo é de f unidades falhas de cada tipo de componente. A estratégia proposta é baseada em replicação, com $3f + 1$ réplicas de cada componente.

A Figura 2 ilustra a arquitetura geral da FT-SFC. O tráfego é recebido pelo SC replicado, responsável pela classificação e encaminhamento para SF. A arquitetura prevê que o cliente que invoca a SFC é confiável, tendo sido corretamente autenticado. Ao cliente deve ser acrescentado um *wrapper*, que tem duas funcionalidades. A primeira consiste de enviar cada pacote para as $3f + 1$ réplicas do SC, ao de invés de apenas o SC único da arquitetura da IETF. A segunda consiste na marcação de cada pacote com um *timestamp* local, que pode ser implementado como um contador local de pacotes transmitidos para a SFC. O objetivo é permitir a identificação única de cada pacote, descartando cópias de pacotes já processados. Assim, cada réplica de SC encaminha uma única vez cada pacote recebido para todas as réplicas de SFF. Caso uma entidade seja maliciosa, a SFC-TF garante que todos os componentes da SFC decidem pelo mesmo pacote em cada estágio, caso haja maioria. Se não houver maioria, o encaminhamento e processamento do tráfego é interrompido (em qualquer estágio).

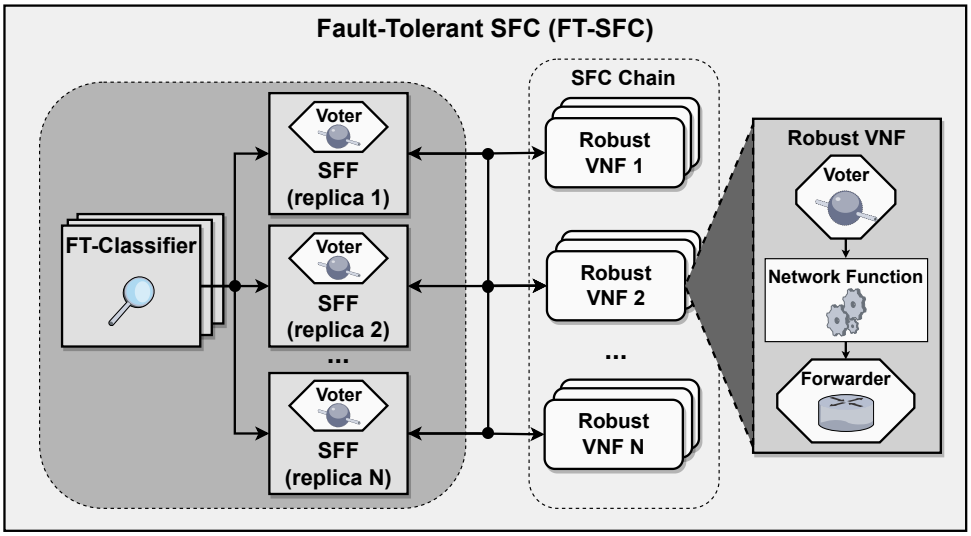

Figura 2. Arquitetura SFC tolerante a falhas.

O SFF recebe as cópias de cada pacote de todas as réplicas do SC. Podem haver até f SCs falhos e assim é necessário que o SFF faça uma votação para selecionar o pacote correto – versões diferentes de pacotes podem ser recebidas devido às falhas bizantinas. Esse processo de votação é realizado também por VNFs, que da mesma forma recebem pacotes de todas as réplicas de SFFs.

Para viabilizar esta funcionalidade, um elemento adicional é incorporado aos SFFs e VNFs, denominado *Voter*. A partir do momento em que o *Voter* recebe $f + 1$ cópias idênticas de um pacote, este pode processar e encaminhar o pacote para o destino seguinte. Este número de cópias pode ser entendido da seguinte forma: quando há o recebimento de $2f + 1$ cópias do pacote, o *Voter* deve fazer a votação, pois f componentes podem falhar por omissão e não devem ser esperados. Por outro lado, ao receber $2f+1$ cópias do pacote, na verdade é possível que até f delas tenham sido produzidas por componentes bizantinos,

assim $f + 1$ necessariamente correspondem ao pacote original correto. Sendo assim, a estratégia de replicação garante a tolerância de f falhas de cada tipo de componente dado um número de réplicas de cada componente igual a $3f + 1$. A próxima seção descreve uma estratégia de otimização do mapeamento das réplicas em ambientes federados.

3. Mapeamento Baseado em Conectividade dos Componentes da SFC

Esta seção descreve uma formulação da estratégia desenvolvida utilizando programação linear inteira, bem como uma heurística genética eficiente.

3.1. Definição do Sistema

O substrato físico considerado consiste em uma infraestrutura distribuída e federada. Essa infraestrutura é representada através de um grafo $G^{ps} = (V^{ps}, E^{ps})$, em que V^{ps} é o conjunto de pontos de presença (*Points of Presence* - PoP) e E^{ps} é o conjunto de arestas do grafo, representando as conexões entre os PoPs. A função $R(V_i^{ps}, V_j^{ps})$ retorna o número de rotas diferentes disponíveis entre os pontos de presença V_i^{ps} e V_j^{ps}, enquanto que a função $D(V_i^{ps})$ retorna o grau de um dado ponto de presença da federação.

Uma SFC é representada como um grafo direcionado $G^{sfc} = (V^{sfc}, E^{sfc})$, em que V^{sfc} contém as funções de rede (VNF) como os vértices do grafo e E^{sfc} é o conjunto de conexões lógicas entre as funções de rede, sendo as arestas do grafo. Um vértice i e uma aresta j são denotados, respectivamente, V_i^{sfc} e E_j^{sfc}.

A função $M(V_i^{sfc}, V_j^{ps})$ retorna 1 caso a função de rede V_i^{sfc} esteja mapeada no ponto de presença V_j^{ps} da federação. Caso contrário, a função retorna zero. Ademais, a função $N(V_j^{ps}, S)$ retorna um valor inteiro representando a quantidade de elementos em um conjunto genérico S, mapeados no ponto de presença V_j^{ps}. Nesse caso, o conjunto S pode ser, por exemplo, os vértices V^{sfc} (funções de rede) da SFC.

Os classificadores de serviço (SC) estão no conjunto SC. Uma instância i de SC é denotada SC_i. A função $M(SC_i, V_j^{ps})$ retorna 1 se a instância i de SC está mapeada no ponto de presença j da federação. Caso contrário, a função retorna zero. Já a função $R(SC_i, V_j^{ps})$ retorna 1 se a instância i de SC classifica o tráfego destinado ao ponto de presença V_j^{ps}. Caso contrário, retorna zero.

Os encaminhadores de tráfego estão no conjunto SFF. Uma instância i de encaminhador é denotada SFF_i. A função $M(SFF_i, V_j^{ps})$ retorna 1 se a instância i de SFF está mapeada no ponto de presença j da federação. Quando não, a função retorna zero. Já a função $R(SFF_i, V_j^{ps})$ retorna 1 se a instância i de SFF conhece as rotas para as funções de rede executando no ponto de presença j da federação. Do contrário, a função retorna zero.

3.2. Premissas e Limitações da Estratégia de Mapeamento

As premissas e limitações da estratégia de mapeamento dos componentes da arquitetura SFC são focadas em dois componentes: SC e SFF. Primeiramente, quanto ao SC, existem duas premissas básicas, estas apresentadas nas equações 1 e 2.

$$\sum_{SC_i}^{SC} 1 \geq 1 \tag{1}$$

$$\sum_{SC_i}^{SC} M(SC_i, V_j^{ps}) \leq 1 \;\; \forall V_j^{ps} \in V^{ps} \tag{2}$$

A Equação 1 define que existe pelo menos uma instância de SC atuante na federação. Por sua vez, a Equação 2 determina que cada ponto de presença da federação hospeda, no máximo, uma instância de classificador de serviço.

Quanto ao mapeamento de encaminhadores de tráfego, existem seis premissas que devem ser consideradas, estas definidas na Equação 3, 4, 5, 6, 7 e 8.

$$\sum_{V_j^{ps}}^{V^{ps}} M(SFF_i, V_j^{ps}) = 1 \;\; \forall SFF_i \in SFF \tag{3}$$

$$\sum_{SFF_i}^{SFF} M(SFF_i, V_j^{ps}) \leq 1 \;\; \forall V_j^{ps} \in V^{ps} \tag{4}$$

$$\sum_{SFF_i}^{SFF} M(SFF_i, V_j^{ps}) = 1 \;\; \forall V_j^{ps} \in V^{ps} \mid N(V_j^{ps}, V^{sfc}) > 0 \tag{5}$$

$$\sum_{V_j^{ps}}^{V^{ps}} R(SFF_i, V_j^{ps}) \geq 1 \;\; \forall SFF_i \in SFF \tag{6}$$

$$N(V_j^{ps}, V^{sfc}) > 0 \rightarrow \exists SFF_i \in SFF \mid M(SFF_i, V_j^{ps}) = 1 \wedge R(SFF_i, V_j^{ps}) = 1 \;\; \forall V_j^{ps} \in V^{ps} \tag{7}$$

$$\sum_{SFF_i}^{SFF} R(SFF_i, V_j^{ps}) \geq 1 \;\; \forall V_j^{ps} \in V^{ps} \mid N(V_j^{ps}, V^{sfc}) > 0 \tag{8}$$

A Equação 3 dita que cada instância de SFF está mapeada em exatamente um ponto de presença da federação. A Equação 4 indica que cada ponto de presença da federação hospeda, no máximo, uma instância de SFF. A Equação 5 define que qualquer ponto de presença que contenha uma ou mais funções de rede, contém também uma instância de SFF. A Equação 6 determina que cada instância de SFF reconhece as rotas de um ou mais pontos de presença da federação. A Equação 7 dita que funções de rede executam em um ponto de presença da federação se neste ponto existe uma instância de SFF que reconheça todas as rotas para essas funções de redes. Finalmente, a Equação 8 dita que todo o ponto de presença que hospede uma ou mais funções de rede tem, ao menos, duas instâncias de SFF que conhecem suas rotas.

3.3. Candidatura ao Mapeamento

O mapeamento de instâncias de SFF e SC em pontos de presença da federação só pode ser realizado considerando algumas condições fundamentais. De início, seja V_j^{ps} um ponto de presença candidato a hospedar a instância SFF_j, então as seguintes condições devem ser verdadeiras.

$$\sum_{SFF_i}^{SFF} M(SFF_i, V_j^{ps}) = 0 \tag{9}$$

$$R(V_i^{ps}, V_j^{ps}) = 1 \ \ \forall V_i^{ps} \in V^{ps} \mid R(SFF_j, V_i^{ps}) = 1 \tag{10}$$

A condição na Equação 9 indica que um ponto de presença candidato não hospeda instâncias de SFF. Por outro lado, a Equação 10 determina que o ponto de presença será candidato se possuir rotas que alcancem todos os pontos de presença que executem as funções de rede para as quais o tráfego de rede será encaminhado pelo SFF_j. De maneira similar, seja V_j^{ps} um ponto de presença candidato a hospedar a instância SC_j, as condições descritas nas Equações 11 e 12 devem ser atendidas.

$$\sum_{SC_i}^{SC} M(SC_i, V_j^{ps}) = 0 \tag{11}$$

$$R(V_i^{ps}, V_j^{ps}) = 1 \ \ \forall V_i^{ps} \in V^{ps} \mid R(SC_j, V_i^{ps}) = 1 \tag{12}$$

A condição na Equação 11 dita que um ponto de presença candidato não hospeda nenhuma instância de SC. A Equação 12, por sua vez, indica que um ponto de presença só é candidato caso possua rotas alcançando todos os pontos de presença para os quais este deve encaminhar tráfego classificado.

Assume-se que as réplicas das VNFs já estão sendo disponibilizadas por PoPs específicos. É feito primeiro o mapeamento dos SFFs e depois dos SCs. Além disso, cabe ressaltar que um ponto de presença que já hospeda uma instância de SFF ou SC pode ser utilizado, desde que: (*i*) a mesma instância já em execução é utilizada no novo SFC e; (*ii*) sendo então configuradas as rotas de encaminhamento e regras de classificação na instância já operante. Porém, as condições apresentadas nas Equações 10 e 12 ainda devem ser observadas.

3.4. Estratégia de Otimização de Mapeamento

A estratégia de otimização adotada neste trabalho é aplicada às réplicas do SFF e do SC. Lembrando que o SC só é utilizado no início, enquanto há SFFs ao longo de toda a SFC. Essa estratégia tem como objetivo maximizar a conectividade dos pontos de presença onde são alocadas tais instâncias. Há diversos critérios de conectividade que podem ser usados para ordenar nodos de uma rede [Cohen et al. 2011, Duarte Jr et al. 2004], na estratégia proposta é maximizado o somatório do grau dos pontos de presença utilizados no mapeamento, assim como o somatório do grau médio dos pontos de presença das rotas, que têm tamanho máximo co.

Para alcançar tais objetivos, uma nova função é exigida: $P(V_i^{ps}, V_j^{ps}, co)$, que retorna os subgrafos $G^{path} = (V^{path}, E^{path})$, que levam do ponto de presença V_i^{ps} ao ponto de presença V_j^{ps} (com tamanho máximo co). Com isso, os objetivos de otimização de mapeamento dos componentes da arquitetura IETF SFC são descritos a seguir.

$$max \sum_{V_i^{ps} \mid N(V_i^{ps}, SFF) \geq 0}^{V^{ps}} D(V_i^{ps}) \tag{13}$$

$$max \sum_{V^{path}}^{P(V_i^{ps},V_j^{ps},co)} \frac{\sum_{V_i^{path}}^{V^{path}} D(V_i^{path})}{\sum_{V_i^{path}}^{V^{path}} 1}$$

$$\forall V_i^{ps}, V_j^{ps} \in V^{ps} \mid \exists SFF_i \in SFF,\ M(SFF_i, V_i^{ps}) = 1 \wedge R(SFF_i, V_j^{ps}) = 1 \quad (14)$$

A Equação 13 maximiza o grau dos pontos de presença onde serão alocadas instâncias de SFF. Já a Equação 14 maximiza o somatório do grau dos pontos de presença nos subgrafos de tamanho até co com rotas entre dois componentes. As mesmas funções de otimização se aplicam para as instâncias de SC. Para isso, basta substituir as ocorrências de "SFF" por "SC" em ambas as equações.

3.5. Heurística Genética

É descrita a seguir uma heurística para o mapeamento baseada em algoritmos genéticos [Deb et al. 2002, Nassu et al. 2005]. Ressalta-se que a estratégia genética é aplicada apenas ao mapeamento das réplicas do SFF. Para o mapeamento das instâncias de SC, a estratégia exata é utilizada. Essa decisão decorre do fato de uma instância de SC ser independente das demais. Portanto, para realizar o mapeamento das mesmas, basta ranquear todos os pontos de presença capazes de hospedá-las, atribuindo-as aos pontos melhor ranqueados. Os componentes principais da heurística são apresentados a seguir.

Os indivíduos utilizados no contexto da heurística genética são criados a partir de um gerador. Esse gerador é responsável por garantir a viabilidade dos indivíduos criados em relação às premissas do modelo de otimização previamente apresentadas. Cada indivíduo apresenta um cromossomo modelado através de um vetor de tamanho N, em que $N > 1$ equivale à quantidade de instâncias de SFF a serem mapeadas (*i.e.*, cada instância de SFF é representada por uma posição do vetor). Os genes consistem em números inteiros com valor (*i.e.*, alelo) presente no intervalo de indexadores $[0, M-1]$, sendo $M > 0$ o número de pontos de presença em uma determinada topologia de rede.

A população inicial é criada de maneira aleatória pelo gerador de indivíduos, garantindo que todos os indivíduos gerados apresentam caminhos para os pontos de presença aos quais devem encaminhar tráfego (*i.e.*, para as instâncias de funções de rede previamente mapeadas e para as instâncias de SFF vizinhas). O tamanho da população é um parâmetro definido pelo usuário.

A heurística genética proposta adota um operador de cruzamento próprio para garantir que os resultados sejam válidos em relação às premissas do modelo. Sendo assim, dados dois indivíduos, o operador ranqueia os melhores pontos de fragmentação binária dos seus genótipos. Essa fragmentação deve ser complementar entre os indivíduos, de forma que a parte fragmentada inicial de um possa ser combinada com a parte fragmentada final do outro.

O ranqueamento das fragmentações considera a criação de um novo indivíduo (filho) com o maior valor para o somatório do grau dos pontos de presença da rede representados pelos alelos dos genes no novo genótipo. O segundo indivíduo gerado pelo cruzamento não é avaliado. As fragmentações são validadas da melhor para a pior até

se encontrar um cruzamento que gere dois indivíduos válidos em relação às premissas do modelo. Caso nenhuma das fragmentações forneça indivíduos filhos válidos, o cruzamento entre os indivíduos pais não é realizado.

O operador de mutação da heurística proposta, assim como o operador de cruzamento, foi desenvolvido especialmente para a mesma. O operador indica um gene aleatório de um genoma para sofrer a mutação. Dado o gene, o operador de mutação escolhe, também aleatoriamente, um novo alelo (ponto de presença) para o gene em mutação. Então, o genótipo é testado quanto a sua validade. Se válido, o indivíduo é retornado. Caso inválido, uma nova tentativa de mutação é realizada. Esse processo é realizado n vezes, sendo n definido pelo usuário.

O algoritmo base para o desenvolvimento da heurística genética é o *Nondominated Sorting Genetic Algorithm II* (NSGAII) [Deb et al. 2002]. Este é um algoritmo genético com características elitistas, ou seja, sempre busca preservar os melhores indivíduos na população. Isso, em geral, acelera a convergência do algoritmo, porém gera maior chance de travamento em ótimos locais. Também, o NSGAII consegue processar múltiplos objetivos avaliados através de fronteiras de Pareto, sendo uma técnica adequada para o problema de otimização abordado neste trabalho.

A seleção de indivíduos para cruzamentos e mutação é feita através de torneio binário. Isto é, dois indivíduos são selecionados aleatoriamente na população, sendo aquele com melhores valores para os objetivos (*fitness*) selecionado para passar pelos processos mencionados. Por fim, a taxa de mutação (probabilidade de uma mutação acontecer), taxa de cruzamento (probabilidade de um cruzamento acontecer) e o número de gerações a ser avaliada (critério de parada do algoritmo) são determinados pelo usuário.

4. Avaliação Experimental

Esta seção descreve a implementação e avaliação tanto do (*i*) mapeamento da arquitetura SFC (Seção 4.1), bem como da (*ii*) FT-SFC (Seção 4.2). Todos os experimentos relatados foram executados 50 vezes em uma máquina com processador Intel Core I5-3330 @ 3.0GHz, 8GB RAM DDR3 e Ubuntu 16.04. As subseções a seguir apresentam os detalhes de implementação, ambiente de teste, e dos experimentos realizados, além de discutirem os resultados obtidos.

4.1. Otimização do Mapeamento: Resultados

As estratégias de otimização para mapeamento da infraestrutura SFC foram implementadas através de simulação utilizando a linguagem Python 3.9.1. A solução recebe como entrada: o grafo que representa a rede; as instâncias de VNFs hospedadas em cada vértice; as VNFs disponíveis; a especificação das SFCs; o número de classificadores (SCs) que devem ser mapeados e a descrição dos SFFs que devem ser mapeadas utilizando a solução proposta. Por fim, a estratégia recebe como entrada o peso de cada métrica que compõe a função objetivo da otimização e também o tamanho máximo considerado para o cálculo do grau médio dos pontos de presença presentes nas rotas entre dois pontos extremos de cada SFC. Duas versões da solução proposta foram implementadas: o algoritmo exato ILP e a heurística genética[1]. A heurística genética ainda recebe os valores desejados para

[1]O código das soluções implementadas está disponível em `https://github.com/ViniGarcia/NIEP/tree/SBRC-2022/SBRC-2022/SFC-FT_Mapper`

o tamanho da população, taxa de cruzamento, taxa de mutação e número de gerações.

Foi avaliado o mapeamento de dois serviços virtualizados (SFCs) em um ambiente com múltiplos domínios. Uma topologia representativa foi utilizada: a rede *Cogent Network*, contendo 186 pontos de presença (vértices) e 214 arestas[2]. As SFCs contavam com 5 funções dispostas em uma estrutura linear. O número de classificadores (SCs) foi fixado em 3 para todos os experimentos. Da mesma forma, o peso das métricas de otimização foi definido em 0,5. A heurística genética ainda teve os seguintes parâmetros fixados: tamanho da população igual a 50, taxa de cruzamento igual a 0,6, e taxa de mutação igual a 0,1. Estes valores foram definidos a partir da observação de resultados de testes empíricos preliminares.

Os experimentos foram realizados considerando dois conjuntos de SFFs. O primeiro conjunto (Cenário 1) conta com 7 instâncias de SFF, sendo 5 delas mapeadas em vértices em que existem funções de rede hospedadas e 2 funcionando como réplicas. Já o segundo conjunto (Cenário 2) considera 8 instâncias de SFFs nos mesmos moldes: 5 em vértices que também executam funções de rede e 3 réplicas.

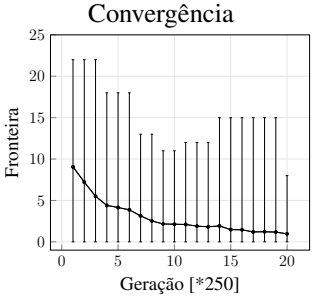

Figura 3. Mapeamento: Cenário 1

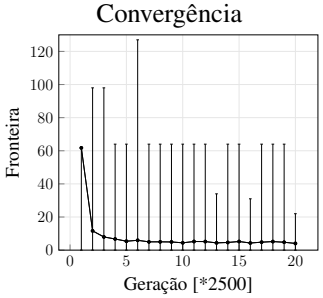

Figura 4. Mapeamento: Cenário 2

O primeiro experimento avalia a convergência e qualidade dos resultados obtidos pela heurística. As Figuras 3 e 4 exibem, respectivamente para os Cenários 1 e 2, a média das fronteiras dos mapeamentos retornados após o processamento de um determinado número de gerações pela heurística genética. As barras de erro, nesse caso, indicam a fronteira do melhor e do pior mapeamento incluído no conjunto de resultados da heurística genética. A primeira barra de erro superior da Figura 4 foi ocultada para aprimorar a visualização do gráfico, entretanto destaca-se que o valor referente a mesma é 646. Os resultados do primeiro experimento demonstram que a heurística genética apresenta boa capacidade de convergência.

As Figuras 5 e 6 apresentam, respectivamente para os Cenários 1 e 2 do experimento, o tempo de execução da solução ótima (força bruta, denotada FB no eixo x) e para a heurística genética segundo o número de gerações processadas (números no eixo x). Assim, é possível perceber a diferença significativa no tempo de execução entre ambas as soluções. Em comparação com o menor número de gerações processadas, a heurística genética representa apenas 6,8% e 0,4%, respectivamente para os Cenários 1 e 2, do tempo total necessário para a execução da solução ótima. Já considerando o maior número de gerações processadas, a heurística genética para o Cenário 1 representa 42.6% do tempo

[2]Disponível em http://topology-zoo.org

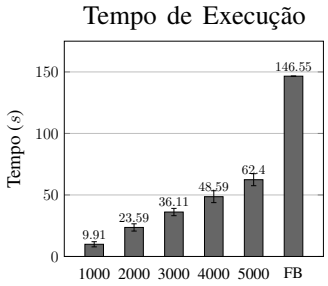

Figura 5. Mapeamento: Cenário 1

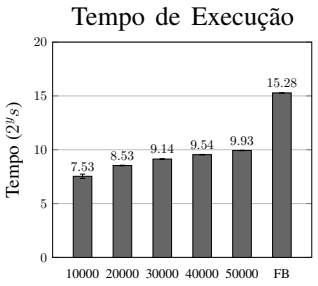

Figura 6. Mapeamento: Cenário 2

de execução em relação à solução força bruta, enquanto o Cenário 2 exibe uma relação de 2,4% nos mesmos critérios.

4.2. Implementação e Avaliação da FT-SFC

No protótipo, os componentes da arquitetura FT-SFC (VNF, SFF e SC) foram implementados utilizando a linguagem Python 3.9.1. Os componentes realizam todas as trocas de mensagens previstas na Seção 2, permitindo a detecção e mitigação de até f falhas bizantinas. Além da configuração de rotas e fluxos tradicionais da arquitetura SFC da IETF, os SCs da arquitetura FT-SFC são configurados com informações das suas réplicas em execução, permitindo a troca de mensagens entre elas. Os SFFs recebem mensagens redundantes de todos os SCs e VNFs, portanto os primeiros também são configurados para reconhecer todas as réplicas de SC disponíveis, assim como todas as réplicas das VNFs dos serviços cadastrados. Por fim, as VNFs devem ter conhecimento de todas as réplicas de SFFs aos quais elas se relacionam, permitindo o envio e recebimento de tráfego entre tais componentes.

Os cenários de teste da arquitetura FT-SFC foram construídos utilizando o emulador NIEP [Tavares et al. 2018]. Dois cenários foram considerados em um primeiro experimento: o primeiro cenário (C1) executa os componentes da arquitetura FT-SFC, porém utiliza apenas uma instância de cada elemento, não tolerando falhas; já o segundo cenário (C2) executa quatro réplicas de cada componente da arquitetura FT-SFC, tolerando o total de uma falha. O serviço de rede empregado em todos os cenários consiste em uma cadeia linear com duas SFs, cada uma executando um encaminhador de tráfego.

O primeiro experimento avalia a progressão do tempo total de processamento de um pacote do momento da emissão do mesmo pelo cliente até o seu recebimento pelo servidor ao qual foi destinado, passando antes por todos os componentes da arquitetura FT-SFC previamente descritos. A Figura 7 mostra os resultados obtidos para os cenários estabelecidos, sendo as barras a média do tempo descrito e as barras de erro o desvio padrão.

Naturalmente, existe um aumento no tempo total de processamento dos pacotes conforme a quantidade de réplicas aumenta. Isso ocorre devido a dois fatores principais: (*i*) quanto maior a quantidade de falhas a serem toleradas, maior é a quantidade de pacotes redundantes que precisam ser recebidos e analisados; (*ii*) quanto maior o número de réplicas, maior é o número de pacotes transitando na rede. Observa-se um aumento sublinear do tempo entre o C1 e C2. Esse fenômeno ocorre visto que existe um conjunto de tarefas realizadas com um tempo similar independentemente do número de réplicas

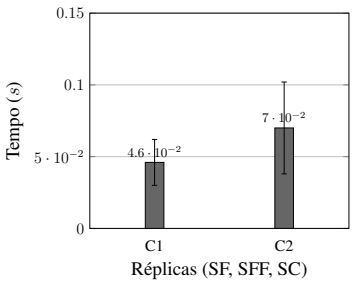

Figura 7. Atraso Decorrente da Arquitetura FT-SFC.

utilizadas, como a alocação do NSH nos pacotes pelo SC e a sua posterior manipulação pelas instâncias de SFF e SF.

O segundo experimento utiliza a arquitetura FT-SFC com quatro réplicas de cada componente necessário para a execução do serviço. Nesse caso, o serviço de rede empregado é o mesmo que foi descrito no primeiro experimento. O segundo experimento considera dois casos de testes: componentes param de responder devido a falhas *crash* (caso *Crash*); e ataques *man-in-the-middle* modificando o conteúdo de pacotes encaminhados entre os componentes (caso *Intrusão*). Consideramos, ainda, cinco cenários onde progressivamente as réplicas dos componentes falham ou são atacadas: todos os componentes corretos (C1); uma instância da primeira SF do serviço falha/é atacada (C2); além de C2, uma instância de SC falha/é atacada (C3); além de C3, uma instância de SFF falha/é atacada (C4) e; além de C4, uma instância da segunda SF do serviço falha/é atacada (C5).

Para os cenários do segundo experimento, os resultados são apresentados na Figura 8. Nessa figura, as barras representam o tempo médio de processamento dos pacotes pela arquitetura e as barras de erro representam o desvio padrão.

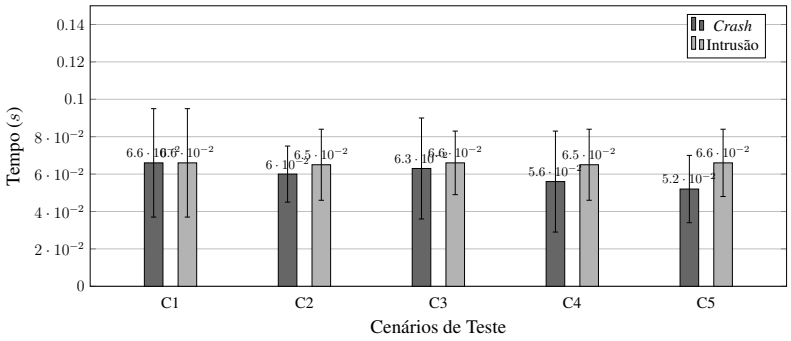

Figura 8. Atraso em cenários com falhas inclusive intrusão.

No caso de falhas *crash* do segundo experimento, é possível notar que existe uma tendência de queda na média do tempo de processamento de pacotes pela FT-SFC. Isso acontece porque quando uma réplica para de responder, a sobrecarga na rede diminui e menos pacotes são recebidos por cada uma das demais réplicas. Por outro lado, no caso dos ataques, esse fenômeno da redução no tempo de execução não acontece, pois o fluxo de pacotes dessas réplicas não é alterado. Por fim, é importante ressaltar que, no segundo experimento da arquitetura FT-SFC, todas as falhas foram toleradas.

5. Trabalhos Relacionados

Em [Ghaznavi et al. 2020] os autores propõem FTC (*Fault Tolerant Chaining*), um sistema para prover tolerância a falhas *crash* para SFCs. Através do próprio processamento dos pacotes, o sistema FTC extrai informações sobre o estado das VNFs. O FTC não utiliza réplicas propriamente ditas das VNF – cada VNF age como uma réplica para a VNF sucessora no SFP. O sistema não é compatível com a arquitetura IETF.

Outro trabalho relacionado, o REACH (*REliability-Aware service CHaining*) [Qu et al. 2016] propõe a utilização de uma solução baseada em ILP para tolerar falhas *crash* de VNFs. O REACH emprega múltiplos nodos funcionando como *backups* para atender o requisito do serviço. Além disso, uma heurística *Greedy shortest-path-based* é proposta para simplificar a ILP.

Outro trabalho relacionado propõe a utilização de instâncias em *standby* para os serviços [Wang et al. 2021]. O trabalho utiliza dois métodos de *Deep Reinforcement Learning* (DRL), cujo objetivo é decidir, de maneira eficiente, a implantação das instâncias (ativa e *standby*). Para SFCs que possuem VNFs *stateful*, um mecanismo é empregado para enviar constantemente o estado da VNF para a réplica em *standby*. A estratégia proposta para replicação de estado de VNFs *stateful* é bastante custosa. Por fim, a solução também não é compatível com a arquitetura da IETF.

Outro mecanismo para garantir a alta disponibilidade de SFCs é proposto em [Kong et al. 2017]. Os autores propõem a utilização de *backups* tanto para as rotas quanto para as VNFs. Através de um algoritmo heurístico, a solução define inicialmente a quantidade de réplicas que cada VNF necessita, utilizada para determinar a quantidade de réplicas consideram tanto a disponibilidade de enlaces quanto a disponibilidade das VNFs.

Todos os trabalhos relacionados que conhecemos voltados para a tolerância a falhas de serviços de rede virtualizados consideram somente falhas *crash* nas VNFs que compõem uma SFC. O presente trabalho inclui os demais componentes da arquitetura SFC, que podem também sofrer falhas bizantinas.

6. Conclusão

Este trabalho apresentou uma estratégia para a tolerância a falhas de serviços de rede construídos de acordo com arquitetura para SFC da IETF. A proposta, denominada FT-SFC tem duas contribuições principais para o estado da arte: é a primeira é considerar a possibilidade de falhas não apenas de VNFs mas também de componentes de classificação e encaminhamento de tráfego; além disso, a FT-SFC tolera intrusões (falhas bizantinas). Os componentes sujeitos a falhas são replicados, as réplicas são posicionadas baseando-se em critérios de conectividade. Uma solução exata para o problema de mapeamento das réplicas baseada em ILP é proposta, assim como uma heurística genética eficiente. A avaliação da proposta foi feita em tanto com simulação para a otimização do mapeamento, como um protótipo da FT-SFC, que demonstrou a resiliência a falhas *crash* e intrusões decorrentes de ataque *man-in-the-middle*. Trabalhos futuros incluem avaliar a FT-SFC em um *testbed* baseado em domínios efetivamente federados.

Referências

Cohen, J., Duarte, E. P., and Schroeder, J. (2011). Connectivity criteria for ranking network nodes. In *Complex Networks*, pages 35–45. Springer.

Deb, K. et al. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Duarte Jr, E. P., Santini, R., and Cohen, J. (2004). Delivering packets during the routing convergence latency interval through highly connected detours. In *International Conference on Dependable Systems and Networks, 2004*, pages 495–504. IEEE.

Fulber-Garcia, V., Huff, A., dos Santos, C. R. P., and Duarte Jr, E. P. (2020). Network service topology: Formalization, taxonomy and the custom specification model. *Computer Networks*, 178:107337.

Fulber-Garcia, V., Marcuzzo, L. d. C., et al. (2019). On the design of a flexible architecture for virtualized network function platforms. In *IEEE Global Communications Conference*, pages 1–6. IEEE.

Ghaznavi, M., Jalalpour, E., Wong, B., et al. (2020). Fault tolerant service function chaining. In *Annual conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 198–210.

Halpern, J. and Pignataro, C. (2015). Service Function Chaining (SFC) Architecture. RFC 7665, IETF.

Kong, J., Kim, I., Wang, X., Zhang, Q., Cankaya, H. C., Xie, W., Ikeuchi, T., and Jue, J. P. (2017). Guaranteed-availability network function virtualization with network protection and vnf replication. In *Global Communications Conference*, pages 1–6. IEEE.

Lamport, L., Shostak, R., and Pease, M. (2019). The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226.

Nassu, B. T., Duarte Jr, E. P., and Ramirez Pozo, A. T. (2005). A comparison of evolutionary algorithms for system-level diagnosis. In *Annual Conference on Genetic and Evolutionary Computation*, pages 2053–2060.

Qu, L., Assi, C., Shaban, K., and Khabbaz, M. (2016). Reliability-aware service provisioning in nfv-enabled enterprise datacenter networks. In *International Conference on Network and Service Management*, pages 153–159. IEEE.

Quinn, P. et al. (2015). Problem Statement for Service Function Chaining - RFC 7498. Technical report, Internet Engineering Task Force.

Quinn, P. et al. (2018). Network Service Header (NSH) - RFC 8300. Technical report, Internet Engineering Task Force.

R., M. et al. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1).

Santos, A. L., Duarte, E. P., and Keeni, G. M. (2004). Reliable distributed network management by replication. *Journal of Network and Systems Management*, 12(2):191–213.

Tavares, T. N. et al. (2018). Niep: Nfv infrastructure emulation platform. In *International Conference on Advanced Information Networking and Applications*, pages 173–180.

Venâncio, G. et al. (2021). Uma arquitetura de alta disponibilidade para serviços virtualizados de rede. In *Workshop de Testes e Tolerância a Falhas*, pages 85–98.

Wang, L., Mao, W., Zhao, J., and Xu, Y. (2021). Ddqp: A double deep q-learning approach to online fault-tolerant sfc placement. *IEEE Transactions on Network and Service Management*, 18(1):118–132.

Ziwich, R. P., Duarte, E., et al. (2005). Distributed integrity checking for systems with replicated data. In *International Conference on Parallel and Distributed Systems*, volume 1, pages 363–369. IEEE.