Uma Abordagem de Encadeamento de Serviços para Virtualização de Funções de Rede em uma Arquitetura de Computação em Borda

Sandy F. da Costa Bezerra[1], **Jorge F. Ramos Bezerra**[2], **Atslands R. da Rocha**[1]

[1] Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Eixo de Computação
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) – Brasil

sandycosta@alu.ufc.br, jorge.fernando60@aluno.ifce.edu.br,

atslands@ufc.br

Abstract. *The Internet of Things and Edge Computing has been widely used, allowing the creation of applications and providing services that enable the interconnection between the physical and virtual worlds. However, this technology requires hardware for data storage and service execution. As a solution, the virtualization paradigm of network resources emerged, which aims to replace the use of physical devices with virtual machines. In this context, this work proposes an approach for managing and orchestrating network functions and service chains in order to reduce network overload. Based on the experiments, the CPU usage of the nodes that provide the services reached a maximum of 72.88% and the maximum memory consumption was 68.13%.*

Resumo. *A Internet das Coisas e a Computação em Borda têm sido amplamente utilizadas, permitindo a criação de diversas aplicações e fornecendo serviços que habilitam a interconexão entre o mundo físico e o virtual. Essas tecnologias necessitam de hardware para armazenamento de dados e execução dos serviços. O paradigma de virtualização de recursos de rede surge como solução para esse desafio. Nesse contexto, este trabalho propõe uma abordagem de gerenciamento e orquestração de funções de rede e encadeamento de serviços visando diminuir a sobrecarga da rede. Baseado nos experimentos realizados, o uso de CPU dos nós que fornecem os serviços obteve um máximo de 72.88% e o consumo máximo de memória foi de 68.13%.*

1. Introdução

O paradigma da Internet das Coisas (do inglês, *Internet of Things* - IoT) tem sido amplamente adotado em novas tecnologias e em diferentes áreas, desde saúde até agricultura. Segundo [Naveena Pai et al. 2020], um ecossistema de IoT é um conjunto de várias camadas, começando da camada dos objetos inteligentes até a camada da nuvem. Atualmente, os benefícios mútuos obtidos com a combinação de computação na borda da rede e na nuvem com a IoT permitem que os sistemas forneçam serviços para os usuários finais ao longo do *continuum* das coisas para a nuvem. Computação em borda é um paradigma de computação distribuído que traz a computação para um local mais próximo da fonte

de dados. A computação em borda melhora o tempo de resposta e economiza largura de banda, evitando a transferência de conjuntos de dados para um ambiente de nuvem remoto [Park et al. 2021]. No entanto, empregar a computação em borda levanta outra preocupação em relação a decisões como, por exemplo, o fornecimento de serviços nos dispositivos da borda. Existem muitos fatores que impactam nas políticas de decisão, como critérios de *offloading*, sobrecarga da rede, prioridade de tarefas, entre outros.

Basicamente, na maioria das técnicas aplicadas para fornecimento de serviços há a utilização de máquinas físicas, gerando um alto custo. Ou seja, requerem dispositivos que possuem os recursos mais amplos (por exemplo, grande capacidade de armazenamento e processamento em alta velocidade) e condições de comunicação confiáveis em termos de atraso e largura de banda. Impulsionadas pela tecnologia 5G, Internet industrial e *blockchain*, estão se formando novas tecnologias, produtos e indústrias, tornando o processo de fabricação mais inteligente, a correspondência entre oferta e demanda mais otimizada e a divisão profissional de mão de obra mais precisa, provocando grandes mudanças na estrutura econômica e impulsionando o salto geral da produtividade social [Jiao and Wang 2022]. No entanto, essas soluções enfrentam desafios significativos em relação à distribuição de carga de trabalho entre os dispositivos heterogêneos da borda, caracterizados por diferentes recursos e capacidades de computação.

A virtualização de funções de rede (do inglês, *Network Functions Virtualization* - NFV) foi proposta para enfrentar esses desafios, aproveitando a tecnologia de virtualização para oferecer uma nova maneira de projetar, implantar e gerenciar serviços de rede [Mijumbi et al. 2016]. Nesse cenário, a NFV surgiu como uma nova estratégia em que as funções de rede, que tradicionalmente usavam hardwares dedicados, agora são implementadas em softwares executados em dispositivos de uso geral. A NFV veio com o objetivo de aumentar a flexibilidade de implantação de novos serviços de rede e realizar a alocação de recursos de maneira eficiente, tendo que ser capaz de migrar funções de um servidor a outro com o objetivo de realizar serviços como, por exemplo, balanceamento de carga. Esses serviços podem ser representados por um conjunto de funções que serão executadas em uma determinada ordem, uma cadeia de funções de serviços (do inglês, *Service Function Chaining* - SFC). O encadeamento de funções de serviço desempenha um importante papel que pode atender a esses requisitos. Ele é capaz de definir um conjunto ordenado de VNFs (como *gateway*, *firewall*, balanceador de carga, etc.) que são unidos em uma rede para criar uma cadeia de serviços complexa com funcionalidades específicas [Pattaranantakul et al. 2021]. Em uma rede NFV, o SFC desempenha um papel importante para implementar serviços de rede em uma sequência especificada.

Nesse contexto, apresentamos uma abordagem de cadeia de serviços em uma arquitetura de computação em borda tendo como base conceitos de virtualização de funções de rede para gerenciar e orquestrar a disponibilização dos recursos necessários para realização de tarefas em uma dada aplicação. As principais contribuições apresentadas nesse trabalho são:

- Desenvolvimento de uma cadeia de serviços para fornecer recursos de rede virtualizados.
- Gerenciamento do encaminhamento de tarefas utilizando um orquestrador de tráfego para virtualização de estrutura de rede.
- Desenvolvimento de uma abordagem de encadeamento coordenado e orientado a

aplicação.

2. Trabalhos Relacionados

Embora a indústria e a academia estejam buscando o NFV com mais frequência, o desenvolvimento ainda está em um estágio inicial, com muitas questões em aberto. Muitos esforços da comunidade acadêmica estão sendo direcionados à virtualização de funções de rede, especialmente no que diz respeito ao encadeamento de serviços. Baseando-se nesses esforços, um progresso considerável já foi alcançado.

O trabalho apresentado em [Taniguchi and Shinomiya 2021] propõe e avalia uma solução para um problema de configuração em um encadeamento de serviços. Essa abordagem visa garantir a operabilidade, evitar falhas de funções virtualizadas e minimizar o custo com base na computação e no uso de recursos de rede por meio de experimentos de simulação. Para alcançar esse feito, eles criam um método de configuração de serviços que leva em consideração o menor caminho entre a aplicação e o recurso disponibilizado. Dessa maneira, o modelo faz um *placement* dos nós virtualizados de modo que as funções virtuais estejam ordenadas e em funcionamento sempre que precisarem ser acessadas.

Em [Xiao et al. 2019] é proposta uma abordagem de aprendizado por reforço adaptativo e *on-line* e para implantar automaticamente SFCs para solicitações com diferentes requisitos de Qualidade de Serviço (do inglês, *Quality of Service* - QoS). Nessa abordagem, adota-se um método baseado em gradiente de políticas para melhorar a eficiência do treinamento e a convergência para a otimização, ou seja, é utilizado um modelo de aprendizado de máquina para a rede entender quando precisa disponibilizar um serviço específico para uma aplicação em tempo real.

No trabalho [Ghorab and St-Hilaire 2022] os autores apresentam um *framework* SFC flexível usando um *cluster* kubernetes de vários nós. Além do *cluster*, é implementado um controlador de rede definida por software (SDN) e um *switch* virtual habilitado para o *OpenFlow* conhecido como *Open vSwitch (OvS)* para implantar e direcionar o tráfego do usuário em um ambiente SFC distribuído. O modelo é composto por três funções bem específicas: localizador de funções virtuais, gerenciador de rede e gerenciador de conectividade. O localizador direciona o tráfego para os serviços disponíveis enquanto o gerenciador de rede configura os nós da rede associando-os a um nó mestre. Por fim, o gerenciador de conectividade conecta os nós que são adicionados a rede.

Em [Kang et al. 2021] é proposto um modelo de *placement* de VNF com *backup* para evitar interrupções de serviço causadas por indisponibilidade de nós usando funções de *backup*. As funções têm um período de tempo de inicialização para preparação antes de poderem ser usadas e o número delas é limitado. O modelo proposto é formulado como um problema de programação linear para colocar os VNFs primários e os de *backup* com base na disponibilidade calculada em intervalos de tempo contínuos. O objetivo é maximizar o número mínimo de intervalos de tempo continuamente disponíveis em todas as SFCs ao longo do período de disponibilidade determinístico.

Dentre os trabalhos apresentados (Tabela 1), o que mais se assemelha a abordagem proposta é o [Xiao et al. 2019], pois utiliza uma abordagem de gerenciamento para disponibilizar os recursos necessários para execução dos serviços. Todas as outras abordagens focam apenas na disponibilização dos serviços e funções sem se preocupar com a

sobrecarga da rede ou com a validação da composição da cadeia de serviços. Além disso, ainda existem importantes desafios de pesquisa pouco explorados, como teste e validação de SFCs, realizados no presente trabalho.

Tabela 1. Trabalhos Relacionados

Trabalho	Metodologia / Algoritmo	Finalidade
[Taniguchi and Shinomiya 2021]	Algoritmo de configuração de Serviços	*Placement*
[Xiao et al. 2019]	Algoritmo de Aprendizado de Máquina e *Deep Learning*	*Placement*
[Ghorab and St-Hilaire 2022]	*Framework* de Serviços Orientados a Conexão	Gerenciamento
[Kang et al. 2021]	Algoritmo de Estimativa e *backup*	*Placement* e Alocação de Recursos
Abordagem Proposta	Composição, teste e validação de Cadeia de Serviço	Gerenciamento e Orquestração

3. Cadeia de Serviços para Virtualização de Funções de Rede

Na arquitetura NFV o componente que exerce a função de alocação de recursos é o orquestrador. O orquestrador avalia todas as condições para fazer a associação de funções nos recursos físicos, se apoiando nos gerenciadores de infraestrutura virtualizada. A ideia principal do NFV é desacoplar os equipamentos físicos da rede das funções que neles são executados. Dessa forma, na abordagem proposta nesse trabalho, a camada de virtualização de recursos é responsável por realizar a alocação de recursos dos nós. Assim, o processo de alocação dos recursos ocorre em dois estágios: (i) Composição de uma cadeia de funções que disponibiliza os serviços de rede e (ii) Escalonamento de recursos, que atua de acordo com o orquestrador que disponibiliza a virtualização do recurso necessário para a finalização da tarefa.

Na abordagem proposta o ecossistema NFV é composto por duas camadas:

1. a primeira camada é formada por serviços encadeados que fornecem funções de rede para as aplicações;
2. a segunda camada é formada pelos recursos virtuais disponibilizados para suprir a necessidade das aplicações de acordo com a demanda.

A camada de aplicação enviará tarefas para serem executadas na camada da borda. Essa tarefa será enviada por meio de uma mensagem contendo uma tupla com os dados que serão processados na execução da tarefa e o tipo da tarefa (por exemplo, armazenamento, consulta, etc). O orquestrador permite que seja adicionado um novo nó virtual que forneça o serviço e que esse nó deixe de existir no momento que finalizar a execução da tarefa para o qual ele foi criado, para não sobrecarregar a rede.

A Figura 1 mostra a proposta inserida no modelo da arquitetura NFV. O gerenciador e orquestrador estão implementados na borda (*edge*). O gerenciador é o nó responsável por armazenar as informações sobre os serviços ofertados na SFC, quais funções compõem esses serviços e qual sua infraestrutura. Já o orquestrador é responsável por posicionar os recursos virtuais disponibilizados pela rede. Esse *placement* virtual é dinâmico

e ocorre de acordo com a necessidade da rede. Desse modo, os recursos são virtualizados de acordo com a demanda das aplicações e das tarefas realizadas.

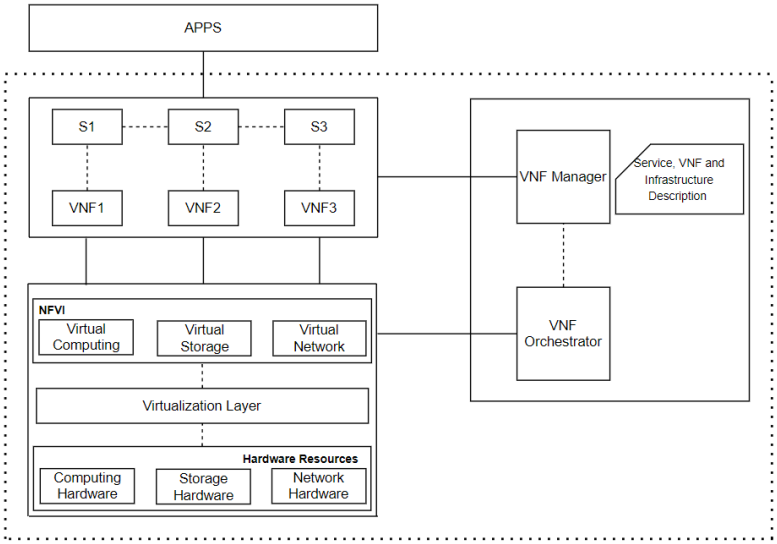


Figura 1. Arquitetura NFV (Adaptado de: [ETSI 2022])

4. Metodologia

Nesta seção é apresentada a metodologia adotada para a validação da proposta, incluindo as ferramentas utilizadas. Adicionalmente, também é apresentado o detalhamento e a composição da SFC além de todo o fluxo da arquitetura. Desde a inclusão das ferramentas nas camadas de virtualização de funções até como os serviços interagem entre si.

Para validação da abordagem proposta foram implementadas três funções que formam a cadeia de serviços de virtualização. A camada de serviço utiliza o paradigma de Função como Serviço (do inglês, *Function as a Service* - FaaS) para definir as funções que farão parte da cadeia de serviço. FaaS é uma tecnologia que permite aos usuários executar funções de maneira orientada a eventos, em um ambiente de tempo de execução leve e isolado [Kim and Roh 2021]. Na abordagem proposta, esse ambiente será composto por três serviços principais:

- **Serviço 1:** *Gateway*, que organiza a troca de informações entre os nós.
- **Serviço 2:** Consulta a informações contidas no banco de dados.
- **Serviço 3:** Armazenamento, que persiste os dados no banco de dados.

Esses serviços e funções são desenvolvidos com a utilização de três principais ferramentas: *Docker*, *Kubernetes* e MongoDB, como ilustrado na Figura 2.

Segundo [Agarwal et al. 2021], o *Docker* é uma das ferramentas mais utilizadas para conteinerização de aplicações e serviços. O Docker aumenta a produtividade e reduz os custos operacionais, permitindo que qualquer desenvolvedor em qualquer ambiente de desenvolvimento crie aplicações robustas e resilientes. Na abordagem proposta, o *docker* vai compor os nós que disponibilizam os serviços, permitindo que as aplicações sejam implementadas e replicadas de acordo com a necessidade da rede.

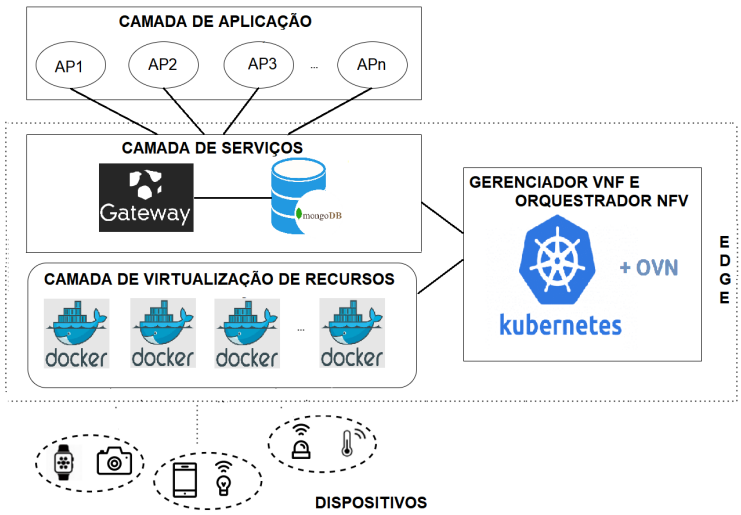


Figura 2. Arquitetura da Abordagem Proposta

Outra ferramenta aplicada no desenvolvimento da abordagem proposta é o *Kubernetes*. De acordo com [Nguyen and Kim 2022], é a tecnologia de orquestração de *containers* mais utilizada na atualidade. Essa orquestração é realizada através da criação de pods, que são os *containers* criados em um ambiente *kubernets*. Além disso, o ambiente fornece uma ampla variedade de ferramentas e serviços inovadores, como, por exemplo, *plugins* para gerenciamento de serviços. Um desses *plugins* é o OVN-Kubernetes, que é baseado no paradigma de *Open Virtual Network (OVN)* e fornece uma implementação de rede baseada em sobreposição, ou seja, permite que sejam implementados, dentro das aplicações, microserviços para gerenciamento de informações da rede.

Como modo de viabilizar a criação de um serviço de armazenamento foi utilizado o MongoDB. MongoDB é um software de banco de dados orientado a documentos, de código aberto e multiplataforma [Mok 2021]. Um campo de documento também pode conter vários documentos incorporados que são usados para representar relacionamentos entre entidades de dados. A utilização do MongoDB na abordagem proposta permite agrupar o máximo de informações em um único documento, usando o padrão JSON para consulta e armazenamento dos dados.

Através do uso das ferramentas detalhadas acima, os serviços conseguem ser disponibilizados na rede, de acordo com a necessidade da aplicação. O *gateway* será acionado pela aplicação quando houver uma requisição de processamento de uma tarefa. Após o acionamento, ele irá realizar uma requisição ao serviço de banco de dados. Essa requisição pode ser de dois tipos, consulta ou armazenamento. A consulta será realizada quando a tarefa precisar de alguma informação que está armazenada no banco de dados para seu processamento. Já o armazenamento será acionado quando a tarefa precisar que a informação persista no banco de dados, ou seja, salva no MongoDB. No processo de consulta, dependendo da aplicação, serão disponibilizados recursos virtualizados para que essa tarefa seja processada. O banco de dados repassa o dado consultado ao *gateway*.

O Banco de dados será acionado de duas maneiras: quando, para a realização da tarefa, seja necessário, além dos dados que vieram da aplicação, dados que estão armazenados. Nesse primeiro serviço, o banco de dados devolve como resposta o dado solicitado

ao *gateway*, para que a tarefa possa ser finalizada. No segundo caso, quando o serviço de armazenamento é acionado, o banco de dados salva os dados no MongoDB e envia como resposta ao *gateway* uma mensagem de confirmação do salvamento e a tarefa é finalizada.

Dentro dos serviços de *gateway* e consulta e armazenamento no banco de dados existe um serviço de *log* para que as mensagens trocadas entre os pods (nós do Kubernetes) que contém cada serviço possam ser gerenciadas dentro da rede. Esse gerenciamento é feito por um *plugin* que adiciona essa camada de gerenciamento das funções de rede, o OVN. Sempre que houver uma requisição ou uma resposta, esse serviço vai obter o dado, os parâmetros, o *tracing* (que carrega a informação de transmissor e receptor da mensagem), além da URL de comunicação com o MongoDB.

Seguindo essa metodologia, foram realizados experimentos a fim de verificar o comportamento da cadeia de serviços em funcionamento.

5. Experimentos

Os experimentos foram conduzidos em uma máquina física, com sistema operacional Linux, processador i5-1135G7 e 8 GB de memória RAM. Inicialmente foi criado um cluster *kubertnetes*, utilizando *containers Docker*, conectados a uma rede local via Wi-Fi. Esse cluster é composto pelos pods contendo os serviços, um orquestrador e um gerenciador. Os serviços foram implementados como APIs e suas imagens foram inseridas nos *containers*.

Os serviços, além de sua implementação de funcionamento, continham um *script* em *javascript* composto por informações de controle. Essas informações são: o dado, o tipo de serviço, a origem, o destino, o *status* e o id. Os serviços interagem de acordo com a necessidade da tarefa a ser realizada, como ilustrado na Figura 3.

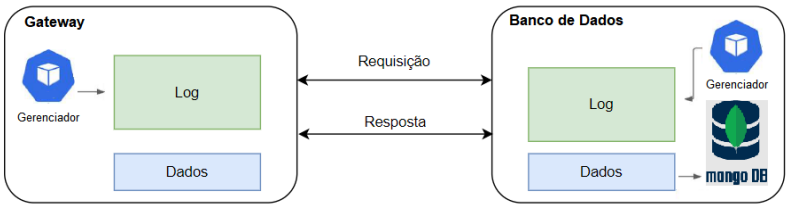


Figura 3. Comunicação entre Serviços

Essas informações servem para, além do envio do dado e do tipo de serviço, gerenciamento e orquestração dos recursos criados para processamento da tarefa. No experimento, o funcionamento da rede simulada seguiu um fluxo de acordo com os passos abaixo:

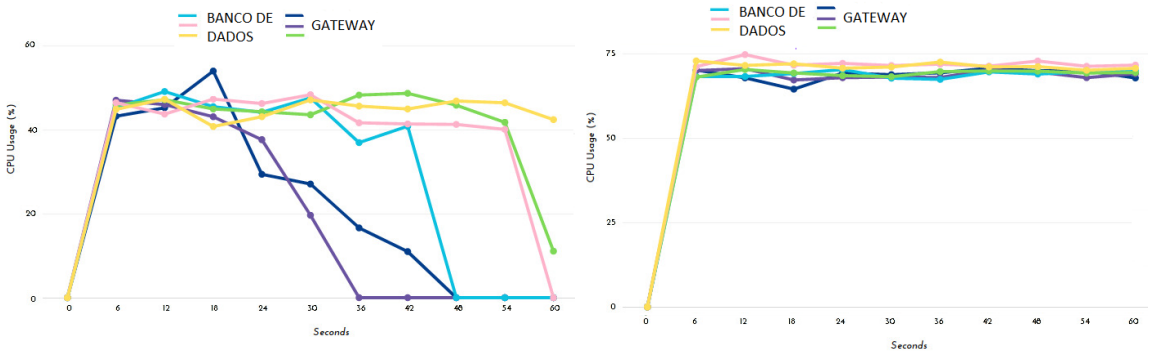
1. Um fluxo de requisições é enviado pelas aplicações.
2. O orquestrador verifica o tipo de serviço e cria os pods contendo os serviços de *gateway* e banco de dados necessários para processamento da tarefa.
3. o gerenciador inicia o serviço de *log* dentro de cada pod gerado adicionando as informações de *tracing* do *script* detalhado acima.
4. quando o identificador *status* estiver em ”COMPLETO”o orquestrador finaliza o funcionamento do pod, evitando a sobrecarga da rede.

Foram definidos fluxos de 250 e 500 requisições para verificar o comportamento da cadeia de serviços de acordo com a quantidade de requisições.

6. Resultados

Esta seção visa mostrar os resultados do nossos experimentos para validar a abordagem proposta e verificar o comportamento da rede durante o processo de orquestração e gerenciamento dos pods. A cadeia de serviços foi configurada inicialmente com 3 pods com o serviço de *Gateway*, 3 pods com o Banco de dados (onde podem ser executados os serviços de consulta e armazenamento) e o nó orquestrador, que formam a camada da borda (Figura 2). É irrelevante em qual pod o serviço será instanciado, pois ele é gerado com o serviço específico e o gerenciador. O pod é gerado de acordo com a necessidade da tarefa a ser executada. Além disso, há um *container* que simula a camada da aplicação, onde fica o *script* das requisições. Esse *script* contém tuplas com dados a serem processados, tarefa, recurso necessário e informações de log como dispositivo de origem e *timestamp*. Para validação e verificação do consumo da rede foram executados dois experimentos: 1) realizando 250 requisições e 2) realizando 500 requisições.

Inicialmente foi realizada uma análise do consumo de CPU durante a execução dos experimentos, considerando as diferentes quantidades de requisições. Quando o pod finaliza todas as tarefas, o orquestrador finaliza o seu funcionamento, como ilustrado na parte a) da Figura 4. Se o pod continuar executando o serviço até o final do execução das tarefas, ele permanece ativo, como ilustrado na parte b) da Figura 4. Além disso, foi possível verificar que a CPU teve um consumo mínimo de 10.97% durante o Experimento 1 e um consumo máximo de 72.88% no Experimento 2. Ainda no Experimento 2, vemos um comportamento parecido de todos os pods. Isso se dá devido ao atributo de balanceador de carga do orquestrador. Como nenhum pod foi desativado até o final do tempo observado nas 500 requisições, o próprio balanceador de carga realizou a distribuição das tarefas evitando a sobrecarga dos nós.

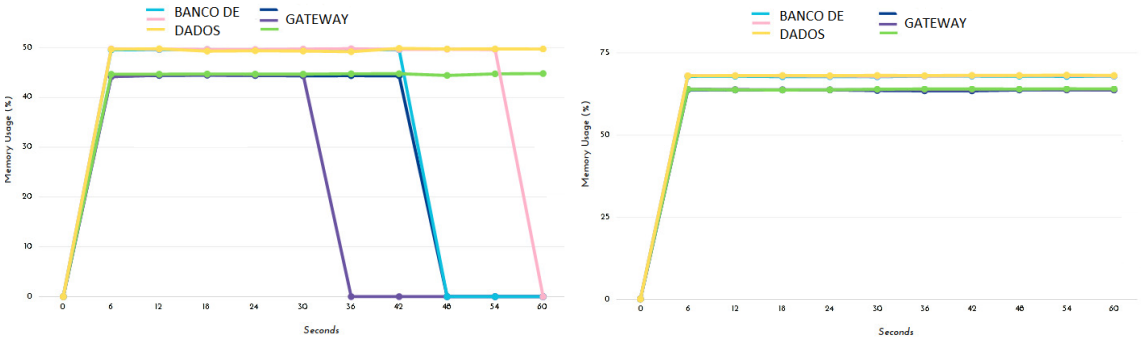


a) 250 Requisições *b) 500 Requisições*

Figura 4. Porcentagem de Uso da CPU

Após a verificação do consumo da CPU, foi mensurada a porcentagem de utilização de memória dos nós que fornecem os serviços. Nesse quesito, foi verificado que, em geral, nos dois experimentos, os nós com os serviços de armazenamento e consulta consumiram uma porcentagem maior de memória. O consumo mínimo de 44.18% foi verificado no Experimento 1 enquanto o pico máximo de 68.13% foi apresentado no Experimento 2, como ilustrado na Figura 5. Quando o consumo de memória e o uso de CPU caem para 0, significa que o pod foi desativado pelo orquestrador.

Analisando os comportamentos apresentados nas Figuras 4 e 5, pode-se observar



a) 250 Requisições *b) 500 Requisições*

Figura 5. Porcentagem de Uso de Memória

que a rede não apresenta sobrecarga de uso de CPU e de memória. Além disso, não existe desperdício de recurso, visto que quando o nó não está executando tarefas, ele é desativado, vantagem que não seria possível utilizando *hardware* físico, pois, mesmo desligado, haveria desperdício de recurso devido ao dispositivo estar disponível sem ser utilizado. O nó gerente em conjunto com o paradigma SFC permite que um mesmo nó execute um serviço, que é acessado por outro nó externo, e um microserviço que é acessado apenas internamente. Essa vantagem é habilitada pelo paradigma NFV, que permite que a proposta implemente as funções de rede como serviços.

7. Conclusão

Neste trabalho apresentamos uma abordagem de virtualização de funções de rede e cadeia de serviços utilizando o paradigma de *Function as a Service*. Dessa forma, possibilitamos a disponibilização de serviços através de recursos virtuais que mantém a rede funcionando sem sobrecarga. O modelo de encadeamento de serviços juntamente com a utilização de recursos virtuais baseados no paradigma NFV habilitaram o gerenciamento da rede e controle dos nós, evitando desperdício de recursos. Além disso, exploramos a composição, teste e validação de SFCs, desafios de pesquisa até então pouco explorados.

Os resultados mostraram que a rede funciona sem sobrecarga de uso de CPU e memória e, além disso, desativa os recursos quando estes não estão mais sendo utilizados. Como proposta de trabalhos futuros, está prevista a avaliação da abordagem quanto a escalabilidade de disponibilização de serviços, utilizando uma cadeia maior e, consequentemente, com uma maior quantidades de nós na rede com o intuito de estressar o sistema ao máximo, para identificar os pontos de falha.

Agradecimentos

Os autores agradecem o apoio concedido pela Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) por meio do processo nº MLC-0191-00164.01.00/22.

Referências

Agarwal, S., Jain, S., and Kumar, A. (2021). Gui docker implementation: Run common graphics user applications inside docker container. In *2021 10th International*

Conference on System Modeling Advancement in Research Trends (SMART), pages 424–427.

ETSI (2022). *Network Functions Virtualisation (NFV) Release 4: Management and Orchestration and Functional requirements specification*. ETSI.

Ghorab, A. and St-Hilaire, M. (2022). Sdn-based service function chaining framework for kubernetes cluster using ovs. In *2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*, pages 347–352.

Jiao, Y. and Wang, C. (2022). A blockchain-based trusted upload scheme for the internet of things nodes. *International Journal of Crowd Science*, 6(2):92–97.

Kang, R., He, F., and Oki, E. (2021). Virtual network function allocation in service function chains using backups with availability schedule. *IEEE Transactions on Network and Service Management*, 18(4):4294–4310.

Kim, D. K. and Roh, H.-G. (2021). Scheduling containers rather than functions for function-as-a-service. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 465–474.

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

Mok, W. Y. (2021). A logical database design methodology for mongodb nosql databases. In *2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1451–1455.

Naveena Pai, G., Swathi Pai, M., Dankan Gowd, V., Shruthi, M., and Naveen K, B. (2020). Internet of things: A survey on devices, ecosystem, components and communication protocols. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 611–616.

Nguyen, N. T. and Kim, Y. (2022). A design of resource allocation structure for multitenant services in kubernetes cluster. In *2022 27th Asia Pacific Conference on Communications (APCC)*, pages 651–654.

Park, J., Choi, U., Kum, S., Moon, J., and Lee, K. (2021). Accelerator-aware kubernetes scheduler for dnn tasks on edge computing environment. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 438–440.

Pattaranantakul, M., Song, Q., Tian, Y., Wang, L., Zhang, Z., Meddahi, A., and Vorakulpipat, C. (2021). On achieving trustworthy service function chaining. *IEEE Transactions on Network and Service Management*, 18(3):3140–3153.

Taniguchi, A. and Shinomiya, N. (2021). A method of service function chain configuration to minimize computing and network resources for vnf failures. In *TENCON 2021 - 2021 IEEE Region 10 Conference (TENCON)*, pages 453–458.

Xiao, Y., Zhang, Q., Liu, F., Wang, J., Zhao, M., Zhang, Z., and Zhang, J. (2019). Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning. In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, pages 1–10.