Caracterização de Funções Virtuais de Rede e Aplicação para Gerenciamento de Elasticidade de Serviços

Alexandre Heideker, Carlos A. Kamienski

Universidade Federal do ABC (UFABC)

{alexandre.heideker,cak}@ufabc.edu.br

Abstract. *Network Function Virtualization (NFV) and Service Function Chaining (SFC) provide essential tools for the widespread adoption of cloud computing, reducing CAPEX and OPEX and enabling autonomous and elastic service management. Traditionally CPU and Memory are used to promote the elasticity, which sometimes produces a misleading evaluation of the resources load. This paper characterizes the behavior of virtualized network and application functions, exploring new metrics in experiments with Firewall, DPI, Proxy, Web Servers, and IoT Context Broker, as well as their SFC compositions. The results show that the evaluation of TCP queues allows a better understanding of the behavior of these functions, clearly identifying not only the current state but the tendency of their behavior in the face of traffic growth.*

Resumo. *A Virtualização de Funções de Rede (NFV) e o Encadeamento de Funções de Serviço (SFC) fornecem ferramentas essenciais para a ampla adoção da computação em nuvem, reduzindo o CAPEX e OPEX e possibilitando o gerenciamento autônomo e elástico de serviços. Tradicionalmente as métricas de CPU e Memória são utilizadas para promover a elasticidade, por vezes produzindo equívocos na avaliação da carga destes recursos. Este artigo caracteriza o comportamento de funções de rede e aplicação virtualizadas, explorando novas métricas em experimentos com Firewall, DPI, Proxy, servidor Web e Context Broker de IoT, assim como suas respectivas composições em SFC. Os resultados obtidos mostram que a avaliação das filas TCP permitem uma melhor compreensão do comportamento destas funções, identificando claramente não só o estado atual como a tendência de seu comportamento diante do crescimento no tráfego.*

1. Introdução

Durante a década de 2010-2020 observou-se o movimento de virtualização e transferência dos recursos computacionais da indústria e academia para grandes data-centers, públicos ou privados, consolidando a computação em nuvem como padrão em infraestrutura de computação e telecomunicações. Um grande exemplo deste movimento é a tecnologia celular 5G, desenvolvida com seus alicerces na virtualização e softwarização de redes [Blanco *et al*.,2017].

Alavancada por iniciativas como as Redes Definidas por Software (SDN) [Mckeown *et al*.,2008], que transfere a inteligência dos diversos dispositivos de encaminhamento de uma rede para softwares de gestão, como também a Virtualização de Funções de Rede (NFV) [ETSI-NFVISG, 2012], que substitui equipamentos dedicados com tecnologias proprietárias por versões virtuais, contribuíram significativamente para

a adoção ampla do paradigma de computação em nuvem, reduzindo substancialmente o investimento em infraestrutura (CAPEX) como também seu consequente custo de operação e manutenção (OPEX).

A perspectiva de uma infraestrutura que pode ser configurada e administrada inteiramente por software apresenta um grande número de possibilidades e oportunidades de pesquisa, como o gerenciamento autônomo e elástico destes sistemas. Apenas considerando a tecnologia NFV, podemos destacar desafios como monitoramento, configuração, recuperação de falhas, segurança, privacidade entre outros [Chowdhury *et al*.,2009] [Mijumbi *et al*.,2015], com especial interesse na detecção de gargalos. Detectar estes gargalos ou subutilizações em sistemas, seja no nível da aplicação ou na infraestrutura, garante não só o correto funcionamento e o cumprimento de níveis de qualidade contratados (SLA), como também a redução no CAPEX e OPEX, com impacto direto na formação de preços e o consequente benefício ao usuário final. Esta tarefa não é trivial considerando a grande variedade de blocos de construção para estes sistemas.

Historicamente as métricas de CPU e memória RAM são utilizadas para identificar o nível de uso de uma máquina, real ou virtual, ou software e são amplamente utilizados em sistemas elásticos. Adequadas para diversas situações, a evolução das arquiteturas de microprocessadores e a grande diversidade de algoritmos com diferentes requisitos e impactos computacionais pode apresentar falsos positivos/negativos para os algoritmos de gestão, comprometendo o SLA ou desperdiçando recursos. Em [Heideker e Kamienski, 2016] observou-se a discrepância entre o nível de utilização de CPU/memória e o real estado do sistema.

Este artigo apresenta a caracterização do comportamento de funções virtualizadas de rede e aplicação através de experimentos em uma nuvem computacional OpenStack, aplicando um tráfego sintético em funções como *Firewall*, *Deep Packet Inspection* (DPI), Proxy, servidor web HTTP/HTTPS e *Context Broker* para *Internet of Things* (IoT), e suas respectivas composições em SFC, observando seu reflexo nas métricas de CPU, filas TCP e janela deslizante em relação ao crescimento no tráfego. Os resultados obtidos mostram que a avaliação das filas TCP permitem compreender o estado destas diversas funções, em medições diretas ou em seus reflexos nas diversas funções que compõem o SFC, identificando não só seu estado atual como também a tendência de seu comportamento diante do crescimento no tráfego. Além disso, o uso da métrica de filas TCP permite o seu uso de forma genérica, ou seja, independente do algoritmo implementado na função virtualizada. Apesar de estar fora do escopo deste trabalho é apresentado um exemplo de modelagem baseada em teoria das filas como aplicação destas métricas.

A Seção 2 discute os conceitos de infraestrutura elástica, trabalhos correlatos e métricas tradicionais. A Seção 3 apresenta a metodologia utilizada nos experimentos. Na Seção 4 são apresentados os resultados obtidos e na Seção 5 estes resultados são discutidos, com a conclusão na Seção 6.

2. Infraestrutura Elástica

O conceito de elasticidade muitas vezes é confundido com o conceito de escalabilidade [Galante e Bona, 2012]. Escalabilidade é a capacidade de um sistema se adaptar a diferentes dimensões do problema. Já a elasticidade é a capacidade do sistema de responder a mudança de dimensão do problema. Em ouras palavras, a escalabilidade é muito mais estática e a elasticidade é mais dinâmica. No contexto de SFC, a elasticidade

traduz-se na capacidade deste se reconfigurar para atender o crescimento ou redução na demanda do serviço. A resposta a esta mudança de dimensão do problema tem inicio com a identificação e quantificação desta mudança.

A quantificação desta mudança é realizada através da avaliação individual ou combinada de métricas de sistema operacional ou de aplicação. Além disso, sob o paradigma da Computação em Nuvem, a escolha de métricas para avaliar o estado do sistema em questão ainda conta com a variabilidade dos diferentes serviços de nuvem que usam diferentes sistemas de orquestração em provedores públicos e/ou privados [Simões e Kamienski, 2014]. As métricas tradicionais de sistema operacional são o tempo de CPU e o uso de Memória RAM. As métricas de aplicação são definidas pelo seu próprio algoritmo.

2.1. Trabalhos Relacionados

Um importante aspecto do problema de gerenciamento de Funções de Rede Virtualizadas (VNF) é tratado por Khalid et al em [Khalid *et al.,* 2016], onde o autor propõe uma padronização da *Southbound API*, responsável por configurar e obter informações em tempo real da condição de uma VNF. Já Wang et al [Wang *et al*., 2017] tratam o problema de escalonamento dinâmico de funções de rede atribuindo a cada VNF uma porção fixa de CPU, memória, disco e banda de rede. As máquinas físicas também são modeladas da mesma forma e o algoritmo busca o melhor posicionamento das VNFs nas máquinas físicas em um instante de tempo.

Problemas de balanceamento de tráfego também requerem a correta utilização de métricas para avaliação do estado das funções, já que a qualidade de experiência do usuário, como a utilizada em [dos Passos e Fiorese, 2019] para avaliar o balanceamento de carga em servidores de video sob demanda, não são acessiveis ao orquestrador. Em [Heideker e Kamienski, 2019], é apresentado um orquestrador para SFC utilizando NFV e SDN onde a avaliação da carga de trabalho sobre as funções é realizada por métricas de aplicação, definidas por uma API que é acessada pelo orquestrador para monitorar a presença de gargalos ou subutilização de recursos. A modelagem destas métricas também são exploradas em um modelo baseado em filas em [Heideker, Zyrianoff e Kamienski, 2018].

Alinhado com os objetivos deste trabalho, em [Pfitscher *et al*., 2018] os autores apresentam um modelo para quantificar a responsabilidade de uma VNF sobre o desempenho geral de um SFC através do mapeamento de diversas métricas para criar um modelo baseado em filas para o sistema, com o uso de algoritmos de regressão linear e redes neurais. Este trabalho diferencia-se por examinar as métricas individualmente e apresentar seus respectivos impactos nas diferentes funções de rede e aplicações virtualizadas de forma analítica, assim como de uma maior variedade de funções, contudo, sem avaliar o processo de elasticidade. Além disso, os autores modificam as condições das Máquinas Virtuais (VM) para produzir gargalos artificialmente no SFC.

Finalmente, em [Naik, Shaw e Vutukuru, 2017], os autores apresentam a proposta de um *middleware* para detectar problemas de desempenho em VNFs através da avaliação de métricas de sistema e inspeção do tráfego, o *NFVPerf*. Os resultados deste trabalho podem ser utilizados para ajustes e configurações no *NFVPref,* apresentando a melhor escolha de métricas em cada sistema onde o *NFVPerf* for aplicado.

3. Metodologia

As funções avaliadas neste trabalho são divididas em três grupos: funções de rede, aplicações e IoT. As funções de rede estão relacionadas tipicamente, mas não limitadas, com a manipulação do tráfego nas camadas de rede e transporte (camadas 3 e 4 do modelo OSI). Já as aplicações possuem funcionalidades relacionadas direta ou indiretamente ao usuário. Finalmente o grupo de IoT também apresenta características de aplicação utilizando protocolos específicos deste novo paradigma.

A avaliação de desempenho foi realizada através da geração de tráfego sintético em uma nuvem OpenStack. Para gerar este tráfego foram desenvolvidos dois programas em linguagem c++, um para o tráfego HTTP/HTTPS[1] (rede e aplicação) e um para o tráfego HTTP-NGSI (IoT). As máquinas virtuais geradoras de tráfego estão isoladas em máquinas físicas distintas, evitando desta forma que sua carga de trabalho interfira nas métricas avaliadas. O objetivo das avaliações é identificar o seu comportamento diante do crescimento do tráfego e não o desempenho específico destas funções – isso significa que nenhum tipo de otimização de foi realizada nestas funções para melhoria de seu desempenho, limitando sua configuração ao que é oferecido como padrão em suas respectivas distribuições. Além disso, as funções foram instaladas em VMs com especificações iguais em todos os cenários avaliados. Os cenários foram avaliados em 15 baterias de trafego sintético por 1600 segundos cada e os resultados apresentados com um intervalo de confiança de 99%.

Finalmente, os experimentos utilizam em sua totalidade o protocolo de transporte TCP, ou seja, há um compromisso de conexão fim a fim e este fato faz com que o tráfego ao qual o sistema está submetido dependa do estado da rede, o que em cenários de sobrecarga significa um aumento do tamanho da janela TCP e consequente atraso na conexão. Para monitorar as métricas das máquinas foi desenvolvido um programa em C++ que as registra em intervalos de 1 segundo.

3.1. Funções de Rede

Para explorar as métricas de desempenho de funções de rede, foi utilizado um tráfego web sintético caracterizado pela transferência de arquivos com tamanhos definidos por uma distribuição Log-normal, com média de 400KB e σ de $3x10^{-7}$. O gerador de tráfego efetua uma requisição a um ou mais servidores HTTP requisitando um arquivo selecionado aleatoriamente por uma distribuição uniforme, em intervalos exponencialmente espaçados com média de 5ms. Este tráfego é direcionado através de rotas estáticas para a função de rede avaliada que é responsável por encaminhar a requisição ao destino apropriado. Finalmente, ao concluir a transferência do arquivo, o gerador registra a taxa de transferência e calcula o tempo necessário para realizar a transferência de 1 MB, normalizando desta forma todas as transferências efetuadas.

A Figura 1 mostra o cenário base (R0-NoOp) e os três cenários de funções de rede avaliados. As VMs utilizadas nos experimentos foram modeladas de acordo com o padrão utilizado pela Amazon Web Services (AWS). A geração de tráfego do experimento contou com quatro máquinas virtuais do tipo `t2.large` (2 vCPUs, 8GB de RAM). Para atender ao tráfego gerado foram utilizados três servidores web Apache 2, configurados em VMs do tipo `t2.small` (1 VCPU, 2GB de RAM).

[1] Disponível em https://github.com/heideker/GeradorHTTP, acesso em 18/12/2019.

O cenário R0-NoOp é utilizado como base (*baseline*) para os experimentos de funções de rede e é composto por uma VM idêntica à utilizada pelas funções de rede, apenas encaminhando os pacotes. Ou seja, a comparação deste cenário com as funções de rede permite abstrair dos resultados todo o overhead causado pelos enlaces entre as funções e o processo de encaminhamento (*Forwarding*).

Firewall: O cenário R1-FW avalia os efeitos de um Firewall no tráfego. Para implementar o Firewall utilizou-se o iptables, em uma distribuição Ubuntu 18.04. Foram inseridas 10.000 (dez mil) regras de bloqueio de endereços IP obtidos em uma lista pública (*black list*), além de regras para prevenção de ataque de Negação de Serviço (DoS) e inundação por ping (*Smurf Attack*) e ping da morte (*Ping of Death*).

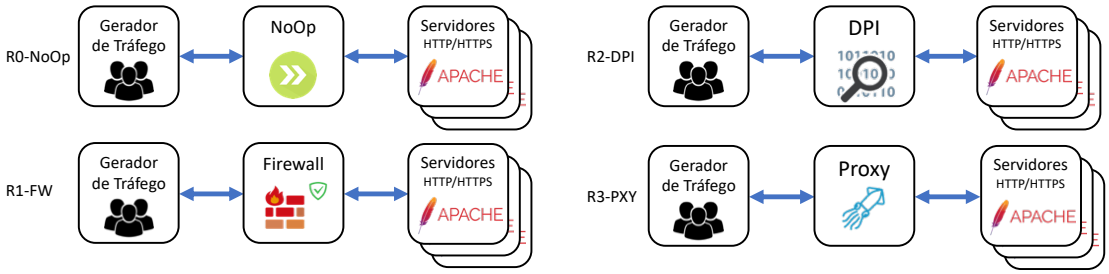

Figura 1: Cenários para Avaliação de Funções de Rede.

DPI: No cenário R2-DPI, é avaliado a Inspeção Profunda de Pacotes (DPI). Utilizando a biblioteca nDPI, parte integrante do projeto nTop, esta função de rede classifica o tráfego pela análise do conteúdo dos pacotes e não do endereço de destino. Foram utilizadas 235 classificações diferentes e, caso uma destas classificações seja reconhecida, o pacote é eliminado.

Proxy: Finalmente o cenário R3-PXY avalia o servidor Proxy, implementado com o Squid [Squid SF, 2011] , um popular servidor Proxy de código aberto. Apesar de ser considerada uma função de rede, o Proxy possui um comportamento ambíguo que pode ser considerado tanto como elemento de rede como de aplicação, já que o mesmo se comporta similarmente a um servidor web respondendo às solicitações no lugar do servidor original.

3.2. Aplicações

Para avaliar as funções de aplicação utilizou-se o servidor HTTP Apache. Além de ser utilizado para a avaliação dos cenários de funções de rede, a escolha do servidor HTTP foi motivada por representar não só o tráfego típico de páginas e aplicações web, como também é adotado como protocolo de uma grade variedade de aplicações, como APIs, dispositivos de IoT e até mesmo áudio e vídeo. A Figura 2 apresenta os dois cenários avaliados. Diferente dos cenários de funções de rede, aqui apenas um servidor é responsável por atender ao tráfego.

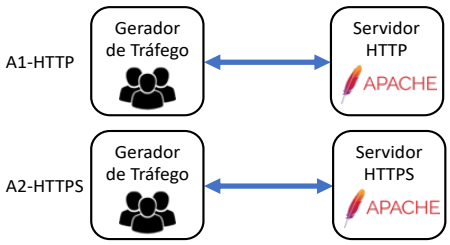

Figura 2: Cenários para Avaliação de Funções de Aplicação.

A1-HTTP: O protocolo é um dos primeiros protocolos de aplicação utilizados na Internet, e um dos seus mais notórios servidores é sem dúvida o Apache. Apesar de estar em processo de substituição por versões com criptografia, por exemplo o HTTPS, o protocolo HTTP ainda é amplamente utilizado, seja em web sites ou em aplicações web, muitas vezes servindo como *backend* de sofisticadas APIs. O cenário A1-HTTP explora do comportamento do servidor web Apache 2 sobre Ubuntu 18.04, além de ser utilizado também nos cenários de funções de rede como resposta ao tráfego.

A2-HTTPS: Diferente do protocolo HTTP puro, o HTTPS utiliza criptografia fim a fim para prover privacidade aos sites e aplicações web. No cenário A2-HTTPS, o comportamento do Apache exige a troca de chaves entre o cliente e o servidor durante a conexão (criptografia de chave pública) e o consequente processo de criptografia dos pacotes durante a comunicação, o que intuitivamente já representa um *overhead* neste processo.

3.3. IoT

A pesar da grande variedade de dispositivos e protocolos envolvidos com o ambiente de IoT, uma figura frequente nestes cenários é o *Context Broker*, responsável por armazenar e distribuir o estado destes objetos em escala mundial. Uma das iniciativas de sucesso no universo de IoT é o projeto FIWARE [FIWARE, 2016], um ecossistema de ferramentas, plataformas e iniciativas públicas e privadas para a ampla adoção da tecnologia. Pedra fundamental do FIWARE, o Orion Context Broker figura como uma das alternativas de destaque.

Dois cenários são avaliados no contexto de IoT, sendo que sua escolha está diretamente relacionada com o processo de *deployment* (instalação e configuração) da aplicação. No cenário T1-ORI, o Orion é implantado juntamente com o seu banco de dados utilizado para a persistência, o MongoDB [MongoDB, 2009], um SGBD orientado a documentos. Sob o olhar da escalabilidade do serviço, as boas práticas sugerem a separação do *Context Broker* de seu respectivo banco de dados, de acordo com o cenário T2-ORIMG, permitindo desta forma avaliar qual o elemento mais sensível ao crescimento do tráfego. A Figura 3 apresenta os cenários de IoT avaliados.

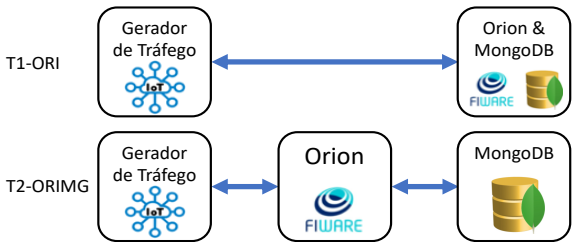

Figura 3: Cenários para Avaliação de Funções para IoT.

Diferente dos cenários anteriores, as requisições ao *Context Broker* foram modeladas de acordo com um caso de uso típico de IoT, onde uma grande quantidade de sensores envia dados periodicamente atualizando suas respectivas leituras, como em ambientes de Cidades Inteligentes ou Agricultura Inteligente [Kamienski *et al.*, 2016]. Este tráfego é composto de requisições HTTP com seu conteúdo de acordo com o protocolo NGSI, padrão do FIWARE. Os experimentos simularam a atualização de 10.000 sensores em intervalos de tempo decrescente. Em todos os cenários, além das métricas específicas como o tempo para transferência de 1MB(cenários de funções de

rede e aplicação) como o tempo para registro do valor obtido no sensor (cenários de IoT), foram coletadas as métricas de CPU, Memória, Tamanho das Filas e Janela TCP das VMs.

4. Experimentos

Os experimentos foram realizados em uma nuvem OpenStack versão Rocky, composta por 6 servidores equipados com processadores Intel(R) Xeon(R) CPU E3-1240 V2 @ 3.40GHz com 8 núcleos e 16GBytes de RAM e 2 servidores equipados com dois processadores Intel(R) Xeon(R) CPU 5620 @ 2.40GHz totalizando 16 núcleos e 32GBytes de RAM, utilizando um Switch Gigabit SDN Pica8 para interligação das máquinas. Todas as VMs utilizam o sistema operacional Ubuntu 18.04 exceto as VMs com o Orion (cenário T1-ORI e T2-ORIMG) que utilizaram o CentOS 7.

4.1 Cenário R1-FW, R2-DPI e R3-PXY

A Figura 4 mostra o tempo necessário para a transferência de 1 MB (eixo da esquerda) em relação ao número de conexões por segundo (eixo da direta). O cenário R0-NoOp na Figura 4(i) apresenta uma taxa média de 0,33 s/MB ao longo de todo o experimento, independente do tráfego aplicado. Na Figura 4(ii) tem-se o desempenho obtido pelo cenário R1-FW com um aumento no tempo de download a partir de 170 conexões por segundo, reduzindo o número de novas conexões admitidas para uma taxa de no máximo 205 conexões por segundo. A Figura 4(iii) apresenta a taxa de download para o cenário R2-DPI onde o mesmo aumento no tempo de download é observado a parti de 490 conexões por segundo, porém, admitindo uma taxa muito maior de conexões, atingindo um máximo de 552 conexões por segundo.

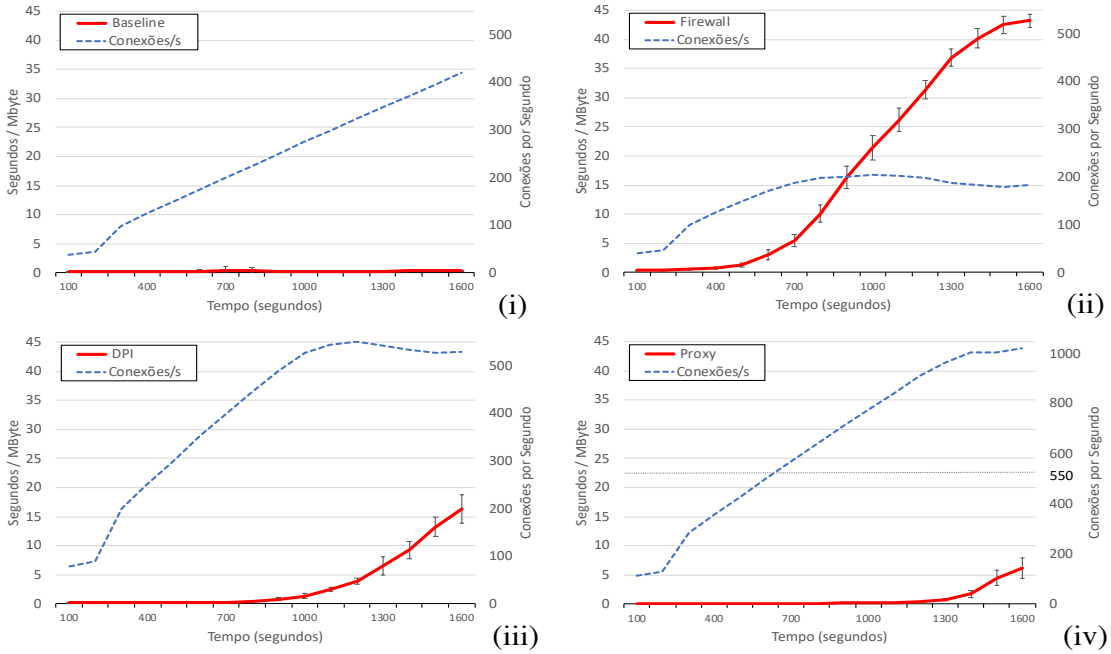

Figura 4: Tempo para download de 1 MB em relação ao número de conexões por segundo. (i) Cenário R0-NoOp(*baseline*). (ii) Cenário R1-FW - Firewall. (iii) Cenário R2-DPI - DPI. (iv) Cenário R3-PXY - Proxy.

Finalmente, na Figura 4(iv), tem-se o desempenho do cenário R3-PXY. Neste cenário a taxa de download permanece, na média, em 0,17 s/MB (abaixo do *baseline*) até

843 conexões por segundo, atingindo o máximo de 1022 conexões por segundo redução significativa no desempenho.

A Figura 5 mostra o uso de CPU nos diferentes cenários de acordo com o número de conexões simultâneas. Em Figura 5(i) tem-se o uso de CPU pelo cenário R0-NoOp, onde os pacotes são apenas encaminhados ocupando em média 0,8% do tempo de CPU. Além da redução no número de conexões admitidas, o cenário R1-FW, visto no Figura 5(ii) mostra que a ocupação total da CPU ocorre rapidamente, estando completamente ocupada com aproximadamente 200 conexões por segundo.

O cenário R2-DPI, exibido na Figura 5(iii), apresenta o uso de CPU pelo DPI – apesar do aumento observado na Figura 4(iii) no tempo de download a partir de 490 conexões por segundo, o uso de CPU pelo DPI permanece abaixo de 2% durante todo o experimento. Finalmente na Figura 5(iv) observa-se o comportamento da métrica de CPU no cenário R3-PXY – há uma clara correspondência entre o número de conexões por segundo atendidas e a carga de CPU, em um comportamento aproximadamente linear.

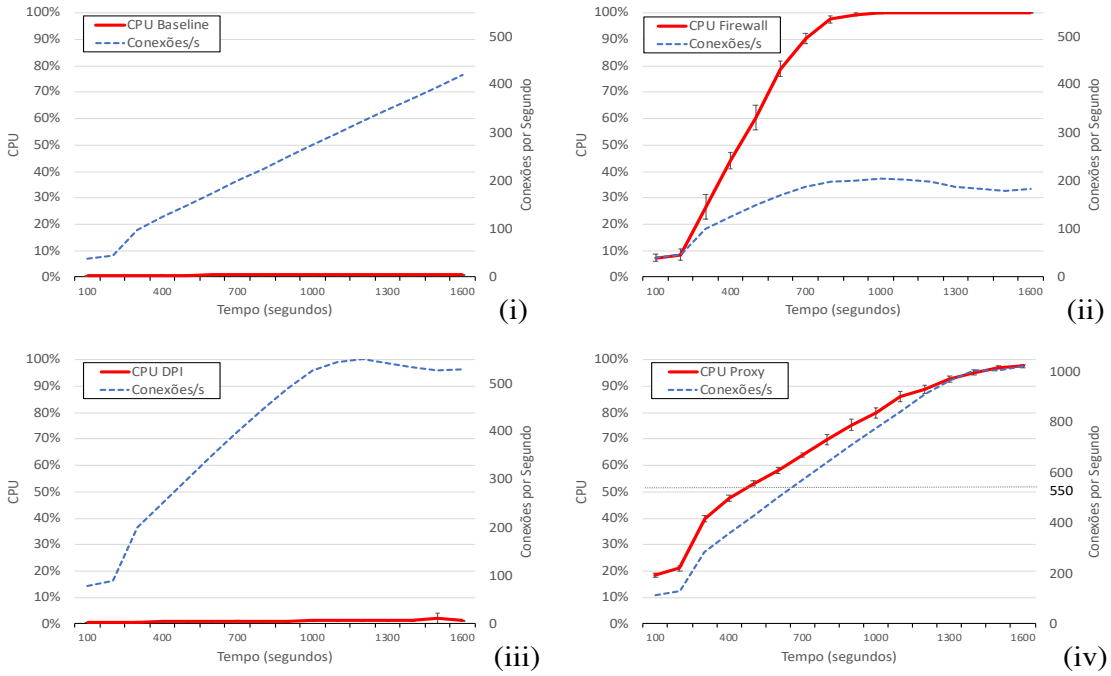

Figura 5: Uso de CPU em relação ao número de conexões por segundo. (i) Cenário R0-NoOp – apenas encaminhamento de pacotes (*baseline*). (ii) Cenário R1-FW - Firewall. (iii) Cenário R2-DPI - DPI. (iv) Cenário R3-PXY - Proxy.

Apesar do comportamento observado na Figura 5(iv) para o Proxy, a Figura 6 apresenta duas métricas que alertam para o estado da função avaliada. Na Figura 6(i) observa-se o comportamento das filas do protocolo TCP, apresentando um crescimento acentuado a partir de 780 conexões por segundo, demonstrando uma possível sobrecarga da função. Neste mesmo instante a carga de CPU é de 79%. Na Figura 6(ii) é possível observar também o aumento da janela deslizante do protocolo TCP ao longo do aumento no número de conexões por segundo. Não foram observadas variações nas métricas de fila e janela TCP nos cenários R1-FW e R2-DPI.

Mesmo não havendo enfileiramento no cenário R1-FW e no cenário R2-DPI, este ocorre nos servidores web que estão respondendo ao gerador de tráfego. A Figura 7(i)

mostra o enfileiramento no servidor web no cenário R1-FW e a Figura 7(ii) mostra este mesmo enfileiramento no cenário R2-DPI – a diferença no tamanho do enfileiramento deve-se ao fato que o Firewall admitiu um número inferior de conexões, o que reflete também no tamanho da fila.

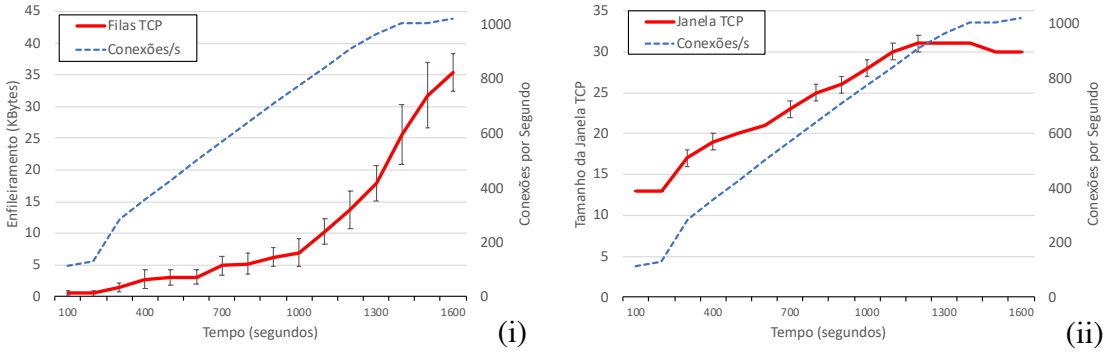

Figura 6: Cenário R3-PXY. (i) Tamanho das filas TCP em relação ao tráfego para a função de rede Proxy. (ii) Tamanho da janela TCP em relação ao tráfego para a função de rede Proxy.

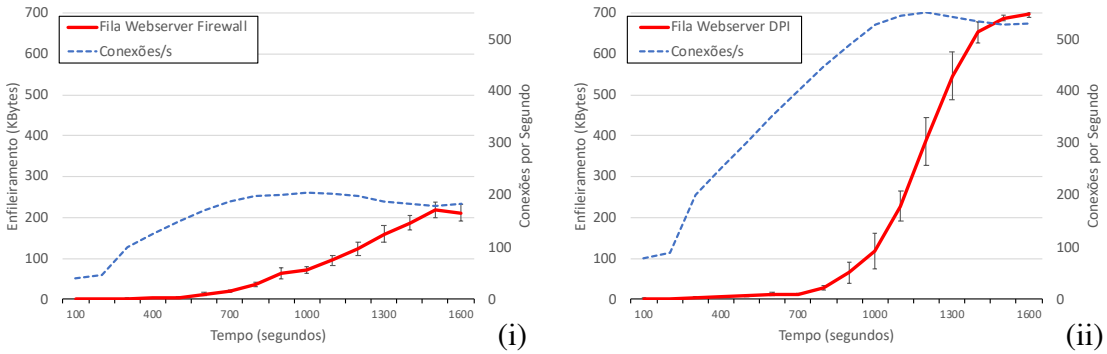

Figura 7: (i) Estado da fila de saída do servidor web no cenário R1-FW. (ii) Estado da fila de saída do servidor web no cenário R2-DPI.

4.2 Cenário A1-HTTP e A2-HTTPS

A Figura 8 mostra o tempo necessário para a transferência de 1MB (eixo da esquerda) em relação ao número de conexões por segundo (eixo da direta). O comportamento do servidor HTTP, exibido na Figura 8(i), e o comportamento do servidor HTTPS, exibido na Figura 8(ii) são análogos, com uma esperada pequena vantagem para o HTTP, que tem seu desempenho degradado a partir de 180 conexões por segundo, devido ao overhead gerado pela criptografia do protocolo HTTPS, que tem seu desempenho degradado a partir de 154 conexões por segundo.

O overhead esperado para o protocolo HTTPS pode ser observado na Figura 9, onde é demonstrado o uso de quase o dobro de CPU pelo protocolo HTTPS, na Figura 9(ii), em relação o protocolo HTTP, visto na Figura 9(i). Apesar de apresentarem 24%(HTTP) e 54%(HTTPS) de uso máximo de CPU, é possível observar na Figura 10(i) e Figura 10 (ii) que o comportamento da fila TCP demonstra a sobrecarga dos respectivos servidores, nos mesmos níveis de carga (conexões por segundo) observados na Figura 8.

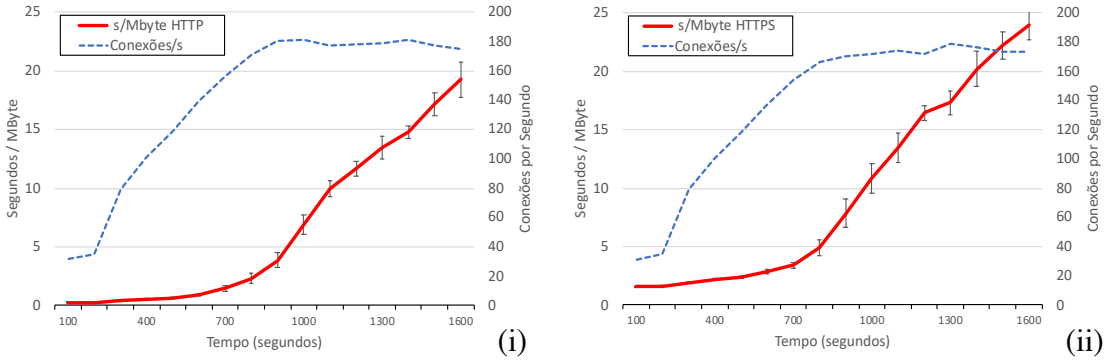

Figura 8: Tempo para download de 1 MB em relação ao número de conexões por segundo. (i) Cenário A1-HTTP – Apache 2 servindo HTTP. (ii) Cenário A2-HTTPS – Apache 2 servindo HTTPS.

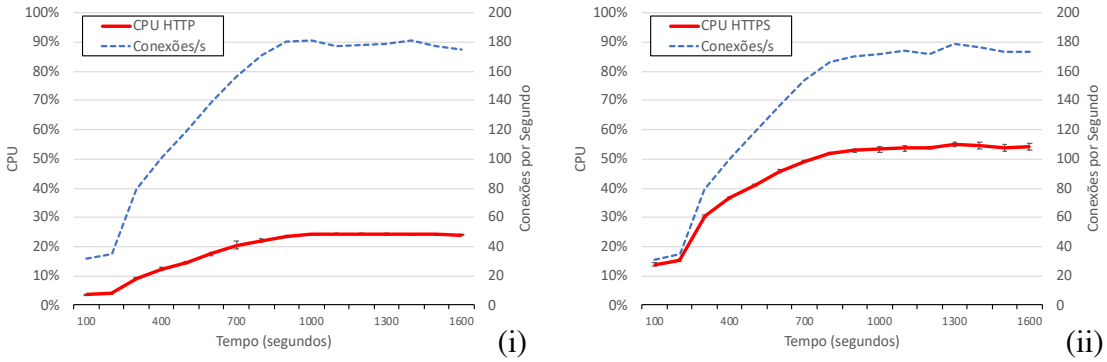

Figura 9: Uso de CPU em relação ao número de conexões por segundo. (i) Cenário A1-HTTP – Apache 2 servindo HTTP. (ii) Cenário A2-HTTPS – Apache 2 servindo HTTPS.

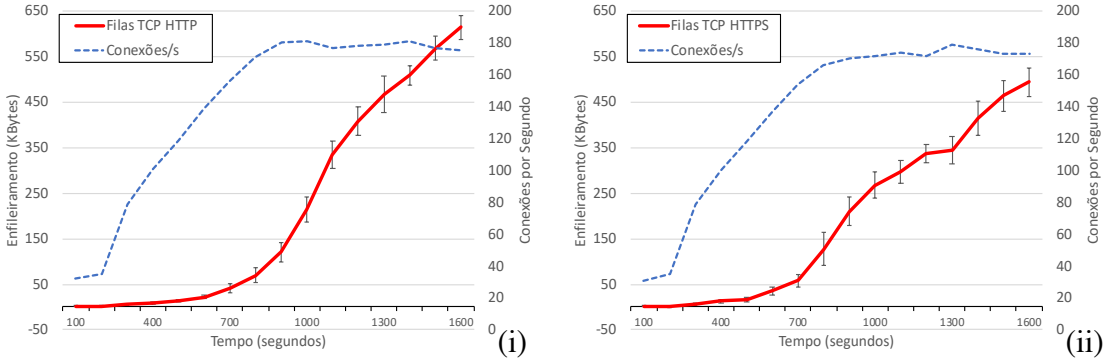

Figura 10: Tamanho da fila TCP em relação ao número de conexões por segundo. (i) Cenário A1-HTTP – Apache 2 servindo HTTP. (ii) Cenário A2-HTTPS – Apache 2 servindo HTTPS.

4.3 Cenário T1 e T2

A Figura 11 apresenta o atraso produzido pelo aumento de tráfego do registro de um atributo de um sensor no Orion *Context Broker*. Apesar do número máximo de conexões por segundo para o cenário T1-ORI de 21 conexões por segundo, visto na Figura 11(i), e de 27 conexões por segundo para o cenário T2-ORIMG, visto na Figura 11(ii), os sistemas foram submetidos a taxas muito superiores durante a avaliação. O fato desta taxa superior

não ter sido admitida deve-se à arquitetura para atendimento às conexões implementada pelo Orion.

O uso de CPU no cenário T1-ORI é apresentado na Figura 12(i) – logo que o sistema atinge a carga de 21 conexões por segundo o nível de CPU atinge 100%. Na Figura 12(ii) fica clara a divisão desta carga de CPU entre os dois módulos de software, onde observa-se uma carga de aproximadamente 1% de uso pela VM com Orion, visto na Figura 12(ii) e em média 95% de uso na VM com o MongoDB, visto na Figura 12(iii).

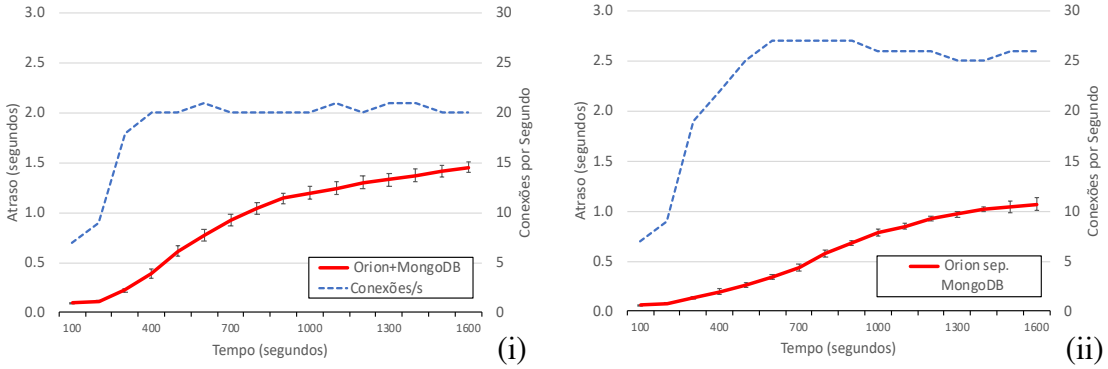

Figura 11: Atraso produzido no registro de um atributo de um sensor. (i) Cenário T1-ORI com Orion e MongoDB na mesma VM. (ii) Cenário T2-ORIMG com Orion e MongoDB em VMs distintas.

A métrica de enfileiramento TCP para o cenário T1-ORI é apresentada na Figura 13(i), onde o enfileiramento ocorre imediatamente ao atingir o nível máximo de conexões por segundo. Apesar observar este comportamento, quantitativamente este valor é pequeno devido à arquitetura de software do Orion, onde as conexões são atendidas imediatamente pelo software retirando a mesma das filas do sistema operacional e as gerenciando internamente. Da mesma forma observa-se este comportamento no cenário T2-ORIMG na Figura 13(ii), que demonstra a mesma arquitetura de software escolhida para o MongoDB.

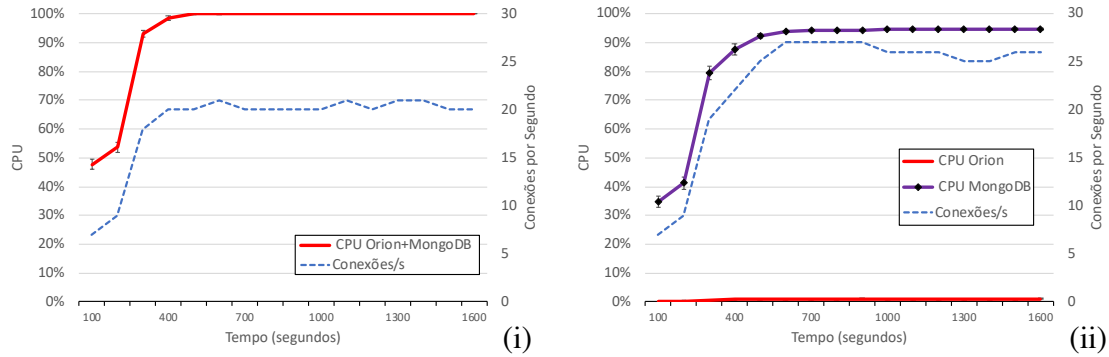

Figura 12: (i) Carga de CPU no cenário T1-ORI na VM com Orion e MongoDB. (ii) Carga de CPU no cenário T2-ORIMG na VM com Orion e na VM com MongoDB.

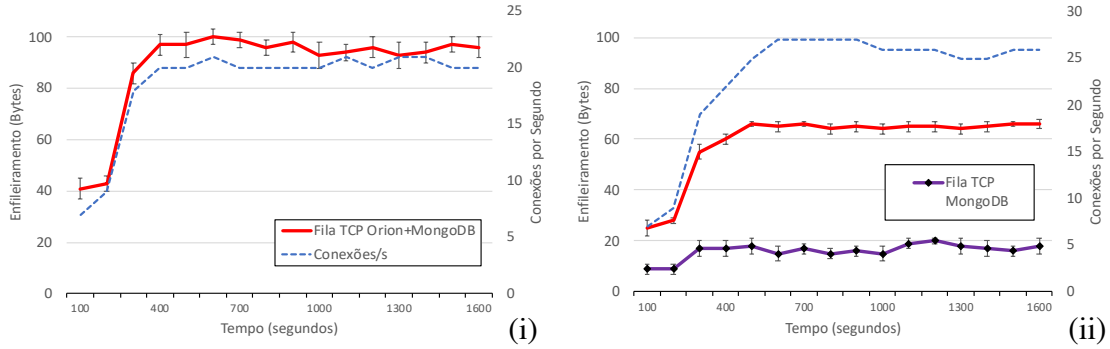

Figura 13: (i) Fila TCP para o cenário T1-ORI. (ii) Fila TCP na VM com Orion e na VM com MongoDB no cenário T2-ORIMG.

4.4. Discussões

Funções de Rede: Não há um padrão estabelecido para valores de qualidade de serviço de rede, já que o desempenho crescente nas redes força cada vez mais a redução nos valores tolerados pelos usuários. Considerando, como exemplo, uma vazão correspondente à taxa de 5 s/MB como máximo aceitável, pode-se dizer que no caso do cenário R1-FW o sistema está cometendo uma infração de SLA a partir de 170 conexões por segundo – neste instante o uso de CPU é de 79%, ou seja, um orquestrador de elasticidade pode não identificar este gargalo a depender de seus limiares. Neste caso, a identificação pode ser feita através da avaliação de outros elementos envolvidos no SFC, como por exemplo o próprio servidor web que está atendendo ao tráfego, visto na Figura 7(i) e Figura 7(ii).

Apesar de estar fora do escopo deste trabalho, um exemplo de modelagem baseada em Teoria das Filas é apresentado na Figura 14, onde podemos observar a modelagem da função não como uma única fila unidirecional e sim como duas filas, com tamanhos e taxas de serviço independentes. Para modelar um SFC como um sistema de filas, são necessárias duas informações: tamanho máximo da fila e taxa de serviço (μ). Por simplicidade, podemos desprezar o tamanho máximo das filas – resta então determinar a taxa de serviço de cada elemento.

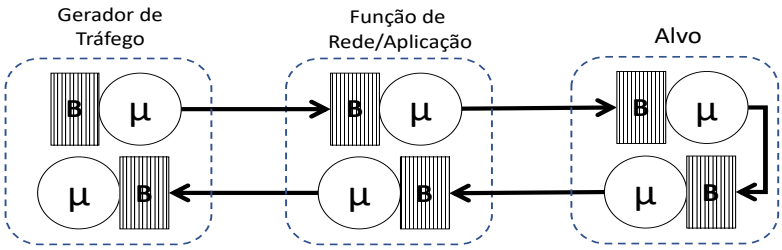

Figura 14: Modelagem do SFC com Teoria das Filas.

O enfileiramento crescente em uma função indica que a taxa de chegadas, ou conexões por segundo, extrapolou a taxa de serviço, tornando o sistema inviável [Heideker, Zyrianoff e Kamienski, 2018]. Avaliando o cenário R1-FW como exemplo, nota-se que o enfileiramento no servidor web começa a crescer a partir de 170 conexões por segundo, ou seja, esta é sua taxa de serviço. No caso do cenário R2-DPI, o DPI possui uma taxa de serviço de 490 conexões por segundo.

Já no cenário R2-DPI fica claro que o bom desempenho da biblioteca nDPI não produz efeito no uso de CPU, impedindo a identificação do estado desta função de forma adequada. A mesma solução proposta ao cenário R1-FW, de avaliar a função seguinte no SFC pode identificar o gargalo no DPI neste caso. Tanto no cenário R1-FW como no cenário R2-DPI foi observado enfileiramento nos servidores web nas filas de saída das VMs, o que demonstra que, tanto a VM do Firewall como a VM do DPI, estavam congestionando o envio dos dados do servidor web em direção ao gerador de tráfego.

Finalmente o cenário R3-PXY demonstra a otimização promovida pelo Proxy, ficando evidente pelo número máximo de conexões admitidas de 1022 conexões por segundo com o início da degradação ocorrendo em 843 conexões por segundo. Dois fatores contribuem para este efeito: a eliminação do enlace intermediário, já que o cliente lida diretamente com o Proxy e a simplificação do processo de fornecer o arquivo ao cliente, já que o Proxy não precisa implementar todas as funcionalidades de um servidor HTTP. Apesar disso, a avaliação do estado das filas TCP demonstra claramente que a função está sobrecarregada antes do respectivo reflexo na métrica de CPU.

Funções de Aplicação: Nos cenários A1-HTTP e A2-HTTPS foi possível observar claramente o reflexo da sobrecarga da função nas filas TCP, corroborando não só com a adoção desta métrica para compor a identificação de gargalos nos SFC, como a sua análise no contexto geral do SFC pode apontar possíveis gargalos em funções encadeadas anteriormente, como no caso dos cenários R1-FW e R2-DPI.

Funções de IoT: Os cenários T1-ORI e T2-ORIMG mostram que não só a composição de um serviço encadeado como também o processo de instalação e configuração devem levar em conta as características do módulo de software utilizado. Além disso, o processo de enfileiramento interno do Orion impede a correta identificação de seu estado, como no caso do cenário T2-ORIMG, no qual não há sobrecarga de CPU na VM utilizada pelo Orion. Neste caso, a mesma avaliação conjunta de todo o SFC proposta para os cenários R1-FW e R2-DPI pode ser realizada para identificar sobrecarga em outras funções encadeadas, como neste caso o MongoDB.

7. Conclusão

Este trabalho procura apresentar um amplo entendimento sobre o comportamento de diversas funções de rede, aplicação e IoT, não só a partir da caracterização destas funções por suas métricas através da avaliação de desempenho apresentada, como a modelagem unificada na composição e sensoriamento de SFC, permitindo não só a manipulação de funções de rede e aplicação de forma unificada, como também a composição de serviços com algoritmos também unificados no gerenciamento de redes e serviços.

Importantes contribuições foram obtidas por este trabalho na caracterização das funções virtualizadas de rede, aplicação e IoT, identificando claramente o comportamento das métricas de CPU, filas TCP e janela deslizante de acordo com o comportamento do tráfego. Entre estas contribuições, destaca-se a identificação da influência da sobrecarga, não identificada com métricas tradicionais, de uma função de rede na subsequente função de aplicação, corroborando com a adoção de uma análise unificada do estado de funções de rede e aplicação em um SFC.

Como trabalhos futuros, espera-se realizar experimentos com uma diversidade ainda maior de funções e implementar um modelo de elasticidade baseado nestas métricas para validar o comportamento observado neste trabalho.

Referências

Blanco, Bego, et al. "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN." Computer Standards & Interfaces 54 (2017): 216-228.

Chowdhury, NM Mosharaf Kabir, and Raouf Boutaba. "Network virtualization: state of the art and research challenges." IEEE Communications magazine 47.7 (2009): 20-26.

ETSI, NFVISG. "Network functions virtualization, white paper." 2012-10-20) [2015-02-14]. http://www. esti. org/technologies cluster/technologies/nfv (2012).

FIWARE. Disponível em: <https://www.fiware.org/>. Acesso em: 9 dec. 2019.

Galante, Guilherme, and Luis Carlos E. de Bona. "A survey on cloud computing elasticity." Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing. IEEE Computer Society, 2012.

Heideker, Alexandre, Ivan Zyrianoff, and Carlos A. Kamienski. "Profiling Service Function Chaining Behavior for NFV Orchestration." 2018 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2018.

Heideker, Alexandre, and Carlos Kamienski. "Gerenciamento Flexível de Infraestrutura de Acesso Público à Internet com NFV." XXXIV SBRC, SBC, 2016.

Heideker, Alexandre, and Carlos Alberto Kamienski. "ElasticNFV: an Elasticity Manager for NFV using SDN." IEEE Latin America Transactions 17.01 (2019): 167-173.

Kamienski, Carlos, et al. "Computaçao urbana: Tecnologias e aplicaçoes para cidades inteligentes." Minicursos do XXXIV SBRC. SBC, 2016.

Khalid, Junaid, et al. "A standardized southbound API for VNF management." Proceedings of the 2016. Hot topics in Middleboxes and NFV. ACM, 2016.

McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

Mijumbi, Rashid, et al. "Network function virtualization: State-of-the-art and research challenges." IEEE Communications Surveys & Tutorials 18.1 (2015): 236-262.

MongoDB. Disponível em: https://www.mongodb.com .Acesso em: 9/12/2019.

Naik, Priyanka, et al. "NFVPerf: Online performance monitoring and bottleneck detection for NFV." 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN). IEEE, 2016.

dos Passos, Edenilson Jônatas, and Adriano Fiorese. "Balanceamento de Tráfego entre Servidores de Vídeo MPEG-DASH em Redes Definidas por Software." Anais do XXIV Workshop de Gerência e Operação de Redes e Serviços. SBC, 2019.

Pfitscher, Ricardo J., et al. "A model for quantifying performance degradation in virtual network function service chains." NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2018.

Simoes, Rhodney, and Carlos Kamienski. "Elasticity management in private and hybrid clouds." 2014 IEEE 7th International Conference on Cloud Computing. IEEE, 2014.

SQUID SF. Squid. Disponível em: http://www.squid-cache.org. Acesso em: 9/12/2019.

Wang, Xiaoke, et al. "Online VNF scaling in datacenters." 2016 IEEE 9th International Conference on Cloud Computing (CLOUD). IEEE, 2016.