# SBRC 2019

Gramado | RS

XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

6 a 10 de maio

Gramado | RS

## ANAIS DO WSlice 2019

sbrc2019.sbc.org.br

# SBRC 2019

## Gramado | RS

XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

6 a 10 de maio

Gramado | RS

# ANAIS DO WSlice 2019

sbrc2019.sbc.org.br

EDITORA
Sociedade Brasileira de Computação (SBC)

ORGANIZAÇÃO
Alberto Egon Schaeffer Filho (UFRGS)
Weverton Luis da Costa Cordeiro (UFRGS)
Stênio Fernandes (UFPE)
David Moura (UNICAMP)
Paulo Ditarso Maciel (IFPB)
André Beltrami Rocha (UFSCAR)
Fábio Verdi (UFSCAR)
Christian Esteve Rothenberg (UNICAMP)

REALIZAÇÃO
Sociedade Brasileira de Computação (SBC)
Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)

Laboratório Nacional de Redes de Computadores (LARC)

Copyright ©2019 da Sociedade Brasileira de Computação
Todos os direitos reservados

**Capa**: Dórian Fogliatto
**Produção Editorial**: Carlos Raniery P. dos Santos (UFSM)

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500- Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

I Workshop de Teoria, Tecnologias e Aplicações de Slicing para Infraestruturas Softwarizadas (1: 2019: Gramado, RS).

Anais / I Workshop de Teoria, Tecnologias e Aplicações de Slicing para Infraestruturas Softwarizadas - WSlice; organizado por Alberto Egon Schaeffer Filho, Weverton Luis da Costa Cordeiro, Stênio Fernandes, David Moura, Paulo Ditarso Maciel, André Beltrami Rocha, Fábio Verdi, Christian Esteve Rothenberg - Porto Alegre: SBC, 2019

84 p. il. 21 cm.
Vários autores
Inclui bibliografias

1. Redes de Computadores. 2. Sistemas Distribuídos. I. Schaeffer-Filho, Alberto Egon II. Cordeiro, Weverton Luis da Costa III. Fernandes, Stênio IV. Moura, David V. Maciel, Paulo Ditarso VI.Rocha, André Beltrami V. Verdi, Fábio VI. Rothenberg, Christian Esteve VII Título.

## Sociedade Brasileira de Computação

**Presidência**
Lisandro Zambenedetti Granville (UFRGS), Presidente
Thais Vasconcelos Batista (UFRN),Vice-Presidente

**Diretorias**
Renata de Matos Galante (UFGRS), Diretora Administrativa
Carlos André Guimarães Ferraz (UFPE), Diretor de Finanças
Antônio Jorge Gomes Abelém (UFPA), Diretor de Eventos e Comissões Especiais
Renata Mendes de Araujo (UPM), Diretora de Educação
José Viterbo Filho (UFF), Diretor de Publicações
Claudia Lage Rebello da Motta (UFRJ), Diretora de Planejamento e Programas Especiais
Marcelo Duduchi Feitosa (CEETEPS), Diretor de Secretarias Regionais
Eliana Almeida (UFAL), Diretora de Divulgação e Marketing
Ricardo de Oliveira Anido (UNICAMP), Diretor de Relações Profissionais
Esther Colombini (UNICAMP), Diretora de Competições Científicas
Raimundo José de Araújo Macêdo (UFBA), Diretor de Cooperação com Sociedades Científicas
Cláudia Cappeli (UNIRIO), Diretora de Articulação com Empresas

**Contato**
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS
CNPJ: 29.532.264/0001-78
http://www.sbc.org.br

**Laboratório Nacional de Redes de Computadores (LARC)**

**Diretor do Conselho Técnico-Científico**
Paulo André da Silva Gonçalves (UFPE)

**Vice-Diretora do Conselho Técnico-Científico**
Rossana Maria de Castro Andrade (UFC)

**Diretor Executivo**
Ronaldo Alves Ferreira (UFMS)

**Vice-Diretor Executivo**
Danielo Gonçalves Gomes (UFC)

**Membros Institucionais**
SESU/MEC, INPE/MCT, UFRGS, UFMG, UFPE, UFCG (ex-UFPB Campus Campina Grande), UFRJ, USP, PUC-Rio, UNICAMP, LNCC, IME, UFSC, UTFPR, UFC, UFF, UFSCar, IFCE (CEFET-CE), UFRN, UFES, UFBA, UNIFACS, UECE, UFPR, UFPA, UFAM, UFABC, PUCPR, UFMS, UnB, PUC-RS, PUCMG, UNIRIO, UFS e UFU.

**Contato**
Universidade Federal de Mato Grosso do Sul (UFMS)
Faculdade de Computação
Caixa Postal 549
CEP 79.070-900 Campo Grande - MS - Brasil
http://www.larc.org.br

## Organização do SBRC 2019

**Coordenadores Gerais**
Alberto Egon Schaeffer Filho (UFRGS)
Weverton Luis da Costa Cordeiro (UFRGS)

**Coordenadores do Comitê de Programa**
Antônio Jorge Gomes Abelém (UFPA)
Fabíola Gonçalves Pereira Greve (UFBA)

**Coordenador de Palestras e Tutoriais**
Ítalo Cunha (UFMG)

**Coordenador de Painéis e Debates**
Artur Ziviani (LNCC)

**Coordenador de Minicursos**
Miguel Elias Mitre Campista (UFRJ)

**Coordenador de Workshops**
Stênio Fernandes (UFPE)

**Coordenador do Salão de Ferramentas**
Leandro Villas (UNICAMP)

**Coordenador do Concurso de Teses e Dissertações**
Daniel Fernandes Macedo (UFMG)

**Coordenadores do Hackaton**
Raquel Lopes (UFMG)
Luis Carlos de Bona (UFPR)

**Comitê de Organização Local**
Avelino Zorzo (PUCRS)
Carlos Raniery Paula dos Santos (UFSM)
Cristiano Bonato Both (Unisinos)
Guilherme Rodrigues (IFSUL Charqueadas)
Jéferson Campos Nobre (UFRGS)
Juliano Wickboldt (UFRGS)
Marcelo Caggiani Luizelli (Unipampa)
Marcelo da Silva Conterato (PUCRS)
Rafael Pereira Esteves (IFRS)
Rodrigo Mansilha (Unipampa)
Tiago Ferreto (PUCRS)
Vinícius Guimarães (IFSUL Charquadas)

**Comite Consultivo**
Fabíola Gonçalves Pereira Greve (UFBA)
Paulo André da Silva Gonçalves (UFPE)
Fábio Luciano Verdi (UFSCAR)
Jó Ueyama (USP)
Antônio Jorge Gomes Abelém (UFPA)
Eduardo Cerqueira (UFPA)
Luiz Fernando Bittencourt (Unicamp)
Rossana Andrade (UFC)
Michele Nogueira (UFPR)
Edmundo Madeira (UNICAMP)

## Mensagem dos Coordenadores Gerais

É com grande alegria e orgulho que, após 9 anos, estamos trazendo o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) novamente para a cidade de Gramado. O SBRC 2019 acontece em um ano que é especial por várias razões: primeiro, o aniversário de 50 anos da ARPANET, a primeira rede a implementar o conjunto de protocolos TCP/IP, que acabaram se estabelecendo mais tarde como os fundamentos da Internet; os 30 anos da invenção da World Wide Web pelo cientista Inglês Tim Berners-Lee; a comemoração dos 30 anos do .br (country code top-level domain), o qual tem atualmente mais de 4 milhões de domínios registrados; por fim, nesse ano comemora-se os 30 anos da Rede Nacional de Ensino e Pesquisa (RNP), que iniciou em 1989 como um projeto de pesquisa. É, portanto, um imenso privilégio termos a oportunidade de sediar o SBRC em um ano que carrega tanto significado para a área de redes de computadores e sistemas distribuídos.

A 37a edição do evento se apoia no histórico de sucesso do SBRC, que tradicionalmente inclui sessões técnicas, minicursos, painéis e debates, workshops, salão de ferramentas e palestras. Além de contar novamente com um Hackathon e com um Concurso de Teses e Dissertações, teremos muitas novidades no SBRC 2019, incluindo as reuniões de mentoria, o evento MUSAS e o workshop de estudantes latino-americanos. As reuniões de mentoria permitem oportunizar conversas 1-a-1 entre estudantes de pós-graduação e pesquisadores de excelência do Brasil e do exterior. O MUSAS (MUlheres em redeS de computadores e sistemAs diStribuídos) visa fomentar conexões entre mulheres atuando nas áreas de redes de computadores e de sistemas distribuídos, no Brasil e no exterior. Já o workshop de estudantes latino-americanos (LANCOMM) serve como um ponto de encontro para estudantes da região apresentarem e discutirem o andamento de suas pesquisas, e obterem feedback construtivo de pesquisadores estrangeiros de grande prestígio.

Neste ano, a trilha principal do SBRC recebeu 228 submissões completas e que entraram no processo de avaliação. Todos os artigos receberam pelo menos 3 revisões, e após um rigoroso processo seletivo, 80 artigos foram aceitos para publicação e organizados em 23 sessões técnicas. O Salão de Ferramentas selecionou 12 trabalhos que a serem demonstrados ao longo do SBRC 2019. Por sua vez, o Concurso de Teses e Dissertações selecionou 8 dissertações de mestrado e 8 teses de doutorado para apresentação durante o evento. Além disso, esse ano o evento conta com 4 palestrantes internacionais, 3 painéis, 1 tutorial, 5 minicursos e 11 workshops.

A organização de um evento com o porte do SBRC é um processo longo, que demanda muita energia e empenho. O evento desse ano só foi possível graças ao apoio e suporte incondicional de muitos grupos de pessoas e instituições. Agradecemos em particular o apoio da SBC, do LARC, do Comitê Consultivo do SBRC e da Comissão Especial de Redes de Computadores e Sistemas Distribuídos da SBC. O evento contou com o apoio do Comitê Gestor da Internet no Brasil (CGI.br) e do Núcleo de Informação e Coordenação do Ponto BR (NIC.br), da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), do Conselho Nacional de Desenvolvimento

Científico e Tecnológico (CNPq), da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) e da ACM SIGCOMM. Contamos novamente também com o apoio institucional da Rede Nacional de Ensino e Pesquisa (RNP). Além disso, o SBRC 2019 recebeu o apoio de importantes patrocinadores, incluindo SAP, HUAWEI, DATACOM, Google, Bedu.tech, BRDigital e AdylNet. Também agradecemos à Universidade Federal do Rio Grande do Sul (UFRGS), e em especial ao Instituto de Informática (INF), pelo suporte incondicional.

Agradecemos a todas as pessoas que trabalharam diretamente na organização do evento. Obrigado pelo excelente trabalho de todos os membros do comitê de organização: Antônio Jorge Gomes Abelém/UFPA e Fabíola Gonçalves Pereira Greve/UFBA (Coordenadores do Comitê de Programa), Stênio Fernandes/UFPE (Coordenador de Workshops), Ítalo Cunha/UFMG (Coordenador de Palestras, Tutoriais e Mentoria), Artur Ziviani/LNCC (Coordenador de Painéis), Miguel Elias Mitre Campista/UFRJ (Coordenador de Minicursos), Leandro Villas/UNICAMP (Coordenador do Salão de Ferramentas), Daniel Fernandes Macedo/UFMG (Coordenador do Concurso de Teses e Dissertações), e Raquel Lopes/UFCG e Luis Carlos De Bona/UFPR (Coordenadores do Hackathon). Agradecemos também aos membros do nosso comitê local, que nos ajudaram na operacionalização de diversas tarefas relacionadas à organização: Avelino Zorzo/PUCRS, Carlos Raniery Paula dos Santos/UFSM, Cristiano Bonato Both/Unisinos, Guilherme Rodrigues/IFSUL Charqueadas, Jéferson Campos Nobre/UFRGS, Juliano Wickboldt/UFRGS, Marcelo Caggiani Luizelli/Unipampa, Marcelo da Silva Conterato/PUCRS, Rafael Esteves/IFRS, Rodrigo Mansilha/Unipampa, Tiago Ferreto/PUCRS, e Vinícius Guimarães/IFSUL Charqueadas. Agradecemos ao Luis Otávio Luz Soares, técnico administrativo do INF/UFRGS, pela imensa ajuda nos mais variados aspectos relacionados à organização do evento, assim como pelo apoio de Leandro Disconzi Vieira e Carlos Alberto da Silveira Junior, também técnicos administrativos da UFRGS. Por fim, agradecemos à nossa equipe local, formada por alunos de graduação, pós-graduação e pós-doutorando da UFRGS, pela dedicação e pronta ajuda sempre que precisamos: Arthur Selle Jacobs, Augusto Zanella Bardini, Bruno Dalmazo, Fernanda da Silva Bonetti, Guilherme Bueno de Oliveira, Guilherme Rotth Zibetti, Isadora Pedrini Possebon, Leonardo Lauryel, Libardo Andrey Quintero González, Lucas Bondan, Lucas Castanheira, Luciano Zembruzki, Mateus Saquetti, Jonatas Marques, Rafael Hengen Ribeiro, e Ricardo Parizotto.

Desejamos a todos um ótimo evento e uma semana produtiva e com muitas trocas de ideias em Gramado.

Alberto Egon Schaeffer Filho e Weverton Luis da Costa Cordeiro
Coordenadores Gerais do SBRC 2019

## Mensagem do Coordenador de Workshops

Os Workshops do SBRC são uma importante oportunidade para aprofundamento no conhecimento de temas especializados ou emergentes na nossa comunidade científica. Como em anos anteriores, a coordenação manteve a *Chamada de Propostas de Workshops do SBRC*, que tem estimulado a comunidade brasileira de Redes e Sistemas Distribuídos a discutir a viabilidade de workshops com temas de pesquisa mais tradicionais, bem como estimular o debate para adoção de eventos que tratam de temas mais emergentes.

Tivemos uma diversidade salutar no Comitê de Avaliação em termos de localização geográfica e experiência, gerando, portanto, opiniões complementares sobre as propostas submetidas. Consideramos todas as propostas de altíssima qualidade, com temas, focos e escopo diversos. Nesta edição de 2019, buscamos equilibrar os benefícios trazidos aos participantes com a capacidade de alocação de espaços no evento. Desta forma, foram selecionados 10 Workshops de alta qualidade, além do tradicional Workshop da RNP (*WRNP*). Dentre as propostas aceitas, oito são reedições de workshops tradicionais do SBRC, a saber: **Gerência e Operação de Redes e Serviços** (*WGRS*), **Testes e Tolerância a Falhas** (*WTF*), **Pesquisa Experimental da Internet do Futuro** (*WPEIF*), **Segurança Cibernética em Dispositivos Conectados** (*WSCDC*), **Trabalhos de Iniciação Científica e Graduação** (*WTICG*), **Blockchain: Teoria, Tecnologias e Aplicações** (*WBlockchain*), **Clouds e Aplicações** (*WCGA*), e **Computação Urbana** (*CoUrb*). Como novidade, teremos dois novos workshops, a saber: o Workshop de **Teoria, Tecnologias e Aplicações de Slicing para Infraestruturas Softwarizadas** (*WSlice*) e o **Latin American Student Workshop on Data Communication Networks** (*LANCOMM Student Workshop*), que tem como público-alvo principal os alunos de pós-graduação e pesquisadores da América Latina, além de estreitar os relacionamentos da nossa comunidade com a Association for Computing Machinery (*ACM*).

Como coordenador dos Workshops do SBRC 2019, gostaria de agradecer a todos os envolvidos na seleção das propostas. Primeiramente aos coordenadores gerais do SBRC 2019, Weverton Cordeiro (UFRGS) e Alberto Egon Schaeffer Filho (UFRGS), pelo convite para a coordenação desta chamada de trabalhos, além de todo o apoio necessário para sua condução apropriada. Agradeço também a todos os membros do comitê de avaliação pelo imenso esforço nas revisões e discussões de alta qualidade de todas as propostas submetidas. Por fim, agradeço aos coordenadores dos workshops aceitos, pela dedicação no cumprimento dos prazos e na condução dos trabalhos internos de seleção dos artigos para seus respectivos eventos, mantendo a alta qualidade geral do SBRC. Os inscritos nos diversos workshops esperam ansiosamente pelas apresentações dos trabalhos e pelas frutíferas discussões que serão geradas.

Stênio Fernandes
Coordenador dos Workshops do SBRC 2019

## Comitê de Programa

- Antônio Abelém (Universidade Federal do Pará - UFPA)
- Augusto Neto (Universidade Federal do Rio Grande do Norte - UFRN)
- Billy Pinheiro (Universidade Federal do Pará)
- Cesar Marcondes (Instituto Tecnológico da Aeronáutica)
- Cristiano Both (Universidade do Vale do Rio dos Sinos (UNISINOS))
- Douglas Viroel (FIT - Instituto de Tecnologia)
- Edmundo Madeira (UNICAMP)
- Joan Serrat (Universitat Politècnica de Catalunya)
- Kleber Cardoso (Universidade Federal de Goiás)
- Lefteris Mamatas (University of Macedonia)
- Luis Carlos De Bona (University Federal of Parana)
- Luis Miguel Contreras Murillo (Telefonica I+D)
- Luiz Fernando Bittencourt (UNICAMP)
- Rafael Pasquini (Federal University of Uberlândia (UFU))
- Sand Correa (Universidade Federal de Goiás - UFG)

# Sumário

**Sessão Técnica 1 ..... 1**

**Proposta de VIM On-Demand para Fatiamento de Nuvem ..... 2**
Eduardo Zanela (UFSCar), Cesar Marcondes (ITA), Fábio Verdi (UFSCar), Emerson Barea (UFSCar), Alex Marino de Almeida (FATEC)

**Slices como serviço sobre um centro de dados itinerante aplicado ao cenário amazônico ..... 16**
Josiane Ribeiro (GERCOM), Lucas Borges (GERCOM), Paulo Sena (GERCOM), Billy Pinheiro (UFPA), Antônio Abelém (UFPA)

**Sessão Técnica 2**

**Arcabouço para Orquestração Osmótica de Cloud Slices ..... 30**
Aryadne Guardieiro (UFU), Gustavo Marques (UFU), Raquel Fialho (UFU), Rafael Pasquini (UFU)

**Uma Proposta de Arquitetura para o Monitoramento Multidomínio de Cloud Network Slices ..... 44**
André Beltrami (UFSCar), Matheus Nadaleti (UFSCar), Vinicius Furukawa (UFSCar), Paulo Ditarso Maciel Jr. (IFPB), Fábio Verdi (UFSCar)

**Arcabouço de um Sistema Inteligente de Monitoramento para Cloud Slices ..... 58**
Gustavo Marques (UFU), Aryadne Guardineiro (UFU), Ian Cunha (UFU), Raquel Fialho (UFU), Rafael Pasquini (UFU)

**Virtualização em Redes Terrestre-Satélite 5G ..... 71**
Karine Costa (INATEL), Tibério Tavares Rezende (INATEL), Antonio Marcos Alberti (INATEL)

# I Workshop de Teoria, Tecnologias e Aplicações de Slicing para Infraestruturas Softwarizadas

# SBRC 2019
# Sessão Técnica 1

# Proposta de Vim *On-Demand* para fatiamento de Nuvem

**Eduardo H. R. Zanela[1], Cesar A. C. Marcondes[2], Fábio L. Verdi[1]**
**Alex M. G. de Almeida[3], Emerson R. A. Barea[1]**

[1]Departamento de Computação – UFSCar

[2]Instituto Tecnológico da Aeronáutica – ITA

[3]Faculdade de Tecnologia de Ourinhos – FATEC

eduardo-hnq@hotmail.com

{marcondes,verdi,emerson.barea}@ufscar.br

alex.marino@fatecourinhos.edu.br

***Abstract.*** *This paper presents a proposal for a Virtualized Infrastructure Manager On-Demand for cloud slicing that allows bare metal control and network services with focus in on-demand creation. For this, a management system was implemented using Openstack in conjunction with virtualization tools allowing the instantiation of a physical machine. In this paper, we present 3 options of installation: complete instantiation bare metal, complete instantiation with containers and complete instantiation with preloaded image. Our results indicate that the proposed, implemented, orchestrated and tested architecture meets the requirements of VIM On-Demand, and points out the main prevailing points in VIM.*

***Resumo.*** *Este trabalho apresenta uma proposta de VIM Sob-Demanda para fatiamento de nuvens que permita controle bare metal e de serviços de redes, com enfoque na sua criação sob demanda. Para isto, foi implementado um sistema de gerenciamento utilizando o Openstack em conjunto com ferramentas de virtualização, permitindo a instanciação de uma máquina física. Neste artigo, apresentamos 3 opções de instalação: instanciação completa bare metal, instanciação completa com containers e instanciação completa com imagem pré-carregada. Nossos resultados indicam que a arquitetura proposta, implementada, orquestrada e testada atende os requisitos do VIM On-Demand, e apontam os principais pontos preponderantes no VIM.*

## 1. Introdução

Um dos maiores desafios enfrentados no desenvolvimento da rede 5G é atender às diversas demandas de consumidores por novos serviços, modificando a forma de implantação de redes de tamanhos fixos para tamanhos variáveis, possibilitando atender aos requisitos de desempenho específicos quanto à latência, escalabilidade, disponibilidade e confiabilidade, impostos por cada caso de uso.

Para tal, é necessária a introdução do fatiamento de rede baseado em demandas, que permitirá que cada fatia de rede, denominado como *slice*, seja alocada conforme a demanda e requisitos do locatário em função de suas aplicações. O ponto central da

implantação do fatiamento é dependente da orquestração de atividades que possibilitem a instalação, preparação do ambiente sucedida de suas configurações e instalação dos serviços requeridos. A conjuminação destas tarefas visa, por fim, a elaboração de um aparato de gerenciamento de fatias de redes virtualizadas, as quais adotam anglicismo *Virtualized Infrastructure Manager* (VIM).

A definição de *slice* de rede pode ser dado como redes lógicas executadas em rede física ou virtual, mutuamente isoladas, dotadas de controle e gerenciamento independentes, que podem ser criados sob demanda [Clayman 2017].

Uma das alternativas que vem sendo utilizada para demonstração do conceito apresentado é a virtualização da rede, que possibilita transformar redes reais em redes virtuais, utilizando soluções tecnológicas baseadas, principalmente, em software tais como: *Software Defined Networking* (SDN), *Network Functions Virtualization* (NFV) e *Virtualized Network Function* (VNF). Estas tecnologias concedem capacidade programática à solução, flexibilidade e modularidade para o desenvolvimento de diferentes redes virtuais para diferentes casos de uso.

Dentre os aspectos importantes correlacionados ao desenvolvimento do VIM, são as verificações dos pontos preponderantes relativos à escalabilidade, aferições pertinentes à resiliência do ambiente e constatações dos tempos gastos durante o processamento de cada etapa da instalação. A proposta apresentada neste trabalho aponta para o desafio de um VIM de fatiamento de rede que criem *slices* sob demanda. Espera-se que os *slices* caracterizem leveza, praticidade, facilidade de instalação e configuração, somando-se ainda à capacidade de gerenciamento dos componentes de infraestruturas virtualizadas. Ao término deste trabalho, espera-se responder duas questões: (*i*) Os recursos computacionais, tais como memória principal e quantidade de núcleos são determinantes para a eficiência do processo? (*ii*) É possível aferir vantagem em desempenho da abordagem VIM *on-demand* em relação à tradicional?

No melhor dos nossos esforços, este trabalho apresenta uma proposta de desenvolvimento de um VIM sob demanda (VIM *on-Demand*), no qual será apresentado o desenvolvimento de forma virtualizada e dividido em 3 opções de instalações: Instanciação Completa *Bare Metal*; Instanciação Completa com *Containers* e Instanciação Completa com Imagem Pré-Carregada. Atribui-se como aspecto inovador desta proposta (*i*) o uso de equipamentos de baixo custo, (*ii*) uso exclusivo de tecnologias de código aberto, (*iii*) orquestração total da implementação e (*iv*) instanciação de diferentes casos de uso. Ressalta-se a escolha da utilização do Openstack em função dos seus serviços aderentes ao propósito deste trabalho, que são a existência do *bare metal* e serviços de controle de computação.

O restante desse trabalho está organizado da seguinte maneira: a Seção 2 relaciona os trabalhos correlatos a este. Na Seção 3 são apresentados os principais conceitos envolvidos no trabalho, o aparato tecnológico e a metodologia empregada na execução dos experimento. Na Seção 4 são discutidos os resultados obtidos no rol de experimentos realizados e, por fim, na Seção 5 conclui este trabalho e apresenta sugestões de trabalhos futuros.

## 2. Trabalhos Relacionados

Apesar do conceito de 5G ser considerado novo, existem diversos trabalhos acadêmicos que colaboram com o seu desenvolvimento, porém o único que utiliza o conceito de *Slice as a Service*, fator central para o provisionamento VIM *on-demand*, é o projeto NECOS[1], no qual este trabalho faz parte [Silva et al. 2018].

Como outros projetos e trabalhos acadêmicos, destacam o 3GPP, o qual apresentou a proposta de padronização do sistema de rede de acesso 5G e do núcleo de sistema de rede 5G ao mesmo tempo [Kim et al. 2017]; o 5G NORMA, que propôs-se uma arquitetura capaz de lidar eficientemente com diversos requisitos e flutuações na procura de tráfego resultantes de conjunto de serviços heterogêneos e suspensos [Rost et al. 2017]; e o SONATA abordou os desafios significativos associados ao desenvolvimento e à implantação dos serviços complexos previstos para as redes 5G [Dräxler et al. 2017].

O trabalho [Freitas et al. 2018], apresenta um sistema capaz de implantar e fornecer VIMs operacionais completos através de um Data Center (DC) *slice Controller*, com base em uma arquitetura em que as fatias do DC são criadas sobre recursos transformáveis (computação e armazenamento). O presente trabalho tem como diferenças do trabalho de [Freitas et al. 2018], a instalação total do ambiente de controle sob demanda analisando diferentes abordagens para carregamento do ambiente. Além disso, entende-se que ao realizar a instalação do ambiente ao invés de importar imagens prontas, mais flexibilidade e opções de customização de tal ambiente podem ser exploradas.

Existem diversos trabalhos voltados à indústria, e para o presente trabalho foram analisados o Kayobe[2], que é um projeto Openstack com finalidade de automatizar a implantação do Openstack em um conjunto de servidores *bare metal*, utilizando *containers*; e o TripleO (Openstack On Openstack)[3] que é um projeto do Openstack que permite implantar duas nuvens, com a finalidade de instalar, atualizar e operar nuvens Openstack utilizando as próprias instalações de nuvem do Openstack como base.

Em relação à característica inovadora nessa criação de *slice*, comparado com a utilização de recursos no OpenStack Nova, compreende-se que o mesmo permite assimilar uma mini-nuvem para o locatário.

## 3. Materiais e Métodos

Nesta seção tem-se descritos os materiais e métodos utilizados durante o desenvolvimento da proposta; na Subseção 3.1 são apresentados os principais conceitos e definições envolvidos neste trabalho; na Subseção 3.3 o aparato computacional e tecnológico utilizado para realização deste trabalho; na Subseção 3.2 os procedimentos adotados que possibilitaram perfazer as atividades e, por fim, na Subseção 3.4 apresentamos a descrição dos experimentos.

### 3.1. Conceitos e Definições

Para efetivar o desenvolvimento da proposta de VIM *on-demand* nas condições propostas, alguns requisitos devem ser previamente elencados e discutidos. Os requisitos que

[1] http://www.h2020-necos.eu/.
[2] https://kayobe.readthedocs.io.
[3] https://wiki.openstack.org/wiki/TripleO.

nortearam o processo de implementação, configuração e instanciação do ambiente e experimentos, seguem descritos:

### 3.1.1. Virtualização

Permite criar uma versão similar de algo, um software adaptável como sistemas operacionais, dispositivos, hardware ou recursos de redes. A computação em nuvem, faz intensa utilização do conceito da virtualização, uma vez que a mesma proporciona uma redução de custo e melhor controle de seu ambiente, segregando a aplicação e sistema operacional dos componentes físicos de origem.

A virtualização dos serviços de rede implica na transformação de suas funções de redes estáticas em funções virtuais. Para [Han et al. 2015], a virtualização das funções de redes propõem melhorar o provisionamento dos serviços de redes. A transformação de redes reais em redes virtuais pressupõe a aplicação de tecnologias baseadas em software, conferindo capacidade de controle, flexibilidade e programação, além de colaborar com o desenvolvimento de diferentes redes virtuais para diferentes casos de uso.

### 3.1.2. Fatiamento de Rede

Trata-se de uma técnica específica de virtualização, a qual proporciona a implementação de várias redes lógicas sob uma infraestrutura de rede física compartilhada [Mademann 2017]. As redes lógicas dos *slices* atuam de ponta a ponta e são formadas por funções de rede, armazenamento e computação [Freitas et al. 2018].

No fatiamento, a infraestrutura de uma rede é subdividida em partes denominadas *slices*. A *slice* é reservada à uma aplicação de um usuário ou locatário e é considerada como um conjunto de recursos que, combinados com a orquestração, atendem a todos os requisitos de um caso de uso específico [Clayman et al. 2018].

A segregação da infraestrutura em múltiplas partes confere benefícios econômicos, tais como: redução de gastos, remoção das limitações de infraestrutura e tecnologias, expansão do cenário para vários clientes diferentes e menor transmissão de dados na rede, isto porque o tráfego de dados móveis é crescente [Silva et al. 2018].

Um caso de uso que tem como alvo a utilização de nuvens para oferecer serviços e infraestrutura para propósito geral, é composto por três tipos de agentes:

- **Provedor de infraestrutura**: é o provedor que possui e gerencia uma determinada rede física e recursos computacionais, armazenamento e comunicação;
- **Locatário**: aluga os serviços de infraestrutura de rede de um ou mais provedores, sob a forma de rede virtual;
- **Usuário final**: aluga os serviços tanto de infraestrutura de redes ofertados pelos provedores, quanto os serviços fornecidos pelos locatários, porém não consegue fornecer recurso para outros usuários.

Nos *slices*, os recursos são unidades gerenciáveis e são divididos em VNFs e recursos de infraestrutura. Para cada *slice* é necessário que exista um VIM gestor do controle dos recursos, suportando diferentes demandas, denominado como VIM *on-Demand* [Silva et al. 2018].

### 3.1.3. Slice as a Service

Refere-se a um novo conceito de serviço de rede, cujos recursos são acoplados internamente à *slice* e gerenciados como um todo, podendo ainda conter componentes de serviços independentes de outros *slices* [Ordonez-Lucena et al. 2017].

O *slice* deve ser criada diretamente em ambiente de nuvens, o que proporcionará menor tempo de resposta nas alterações das demandas de serviços. O *Slice as a Service* possui o conceito de princípio leve, o qual exaure ao limite as configurações dos *slices*, recursos físicos e capacidade de gerenciamento de recursos e desempenho por software.

Em uma situação onde os recursos de infraestrutura ficam inutilizados devido ao excesso de recursos, o conceito do *Slice as a Service* torna possível eliminar a ociosidade da rede, criando novos *slices* conforme requisitados pelos clientes, proporcionando diferentes VIMs e funções virtualizadas de rede. Para [Freitas et al. 2018], os *slices* permitem uma visão geral dos mecanismos, componentes, e abstrações que podem ser utilizados para fornecer um modelo de *Slice as a Service* baseado na instanciação VIM dinâmica.

O problema da alocação dos slices possui um forte componente de otimização. Em um sistema com escala maior e com caminhos de rede disjuntos (Figura 1), a escolha de enlaces para serem alocados para cada slice pode definir a quantidade de clientes suportados pela rede, bem como o desempenho da rede para tais clientes. O VIM é um dos principais componentes de uma estrutura virtualizada, cabe-lhe a responsabilidade de manter o controle e gerenciamento dos recursos de computação, armazenamento e infraestrutura de rede [Freitas et al. 2018].

Oposto às tradicionais configurações de um único VIM para toda infraestrutura da computação em nuvem, o VIM *on-Demand* é implantado dinamicamente para cada *slice*, cabendo a ele permitir que a *slice* de rede esteja apta ao atendimento de diferentes casos de uso, independentemente da necessidade de execução de qualquer tipo de tecnologia.

## 3.2. Procedimentos e Técnicas

Nesta subseção será apresentado um conjunto de tarefas necessários à implantação de um VIM *on-Demand*. A Figura 1 permite visualizar este conjunto de tarefas e suas dependências entre si:

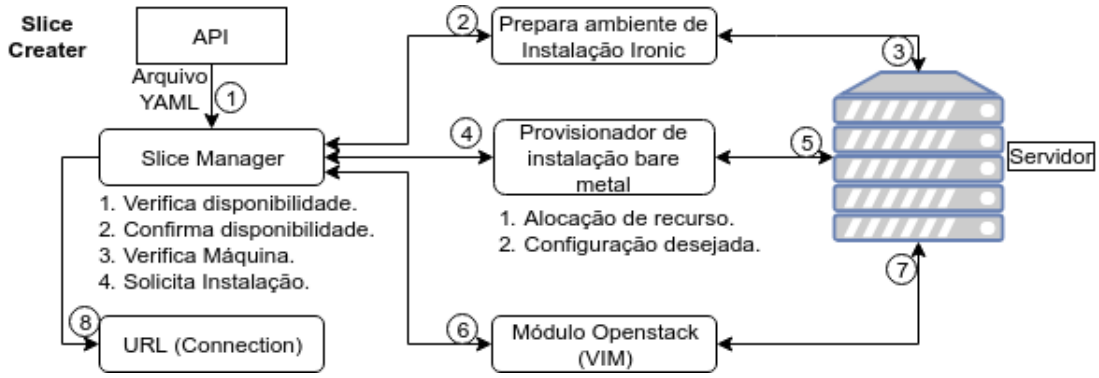



Figura 1. Infraestrutura VIM *On-Demand*.

A seguir serão apresentadas três opções de instalação VIM *on-Demand* proposta

nesse trabalho: Instanciação Completa *Bare Metal*, Instanciação Completa com Imagem Pré-Carregada e Instanciação Completa com *Containers*.

### 3.2.1. Instanciação Completa *Bare Metal*

Uma completa instanciação de um VIM *on-demand* pode ser considerada uma instalação completa (Figura 1), exigindo que seja realizado todo processo de instalação, sem procedimentos que reduzam o tempo de instalação, necessitando da execução de todos os passos.

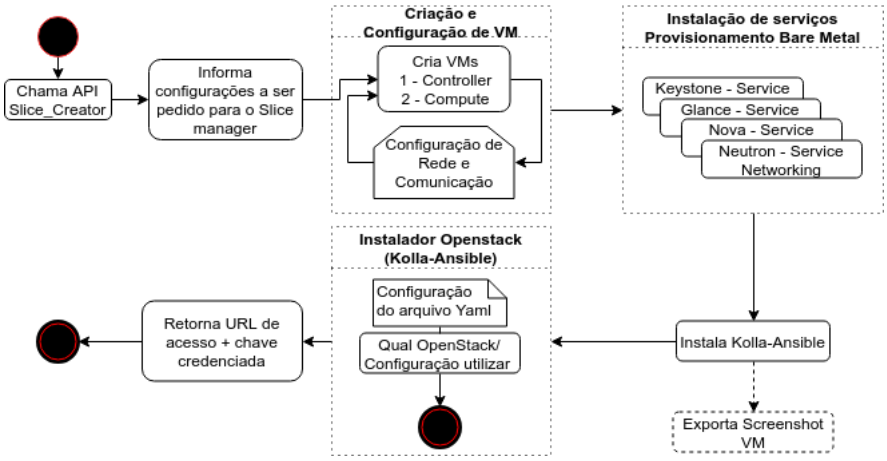



Figura 2. VIM *On-Demand Full Deployment*.

As etapas de instalação são divididas em: Criação de Virtual Machines (VMs), Configuração de rede, Comunicação entre VMs; Instalação Openstack; Instalação do serviço *bare metal* (Ironic); Provisionamento da máquina *bare metal*; Instalação Kolla-Ansible; Provisionamento de serviço Openstack e retorno da URL de acesso.

### 3.2.2. Instanciação Completa com Imagem pré-carregada

A instanciação de um VIM *on-Demand* com uma imagem pré-carregada, é uma instanciação, na qual parte das etapas encontram-se pré-executadas ou provisionadas da instanciação completa, porém com a máquina em um *snapshot*, necessitando ser importado e reconfigurado. O diagrama representado na Figura 3 traz à luz o arcabouço de tarefas compreendidas.

Ao invés de criar uma nova VM para instalar os controladores de cada *slice*, realizamos o *import* do *snapshot* da VM, que é uma tarefa menos custosa computacionalmente. Com isso, o oneroso trabalho de criação e configuração de VMs, instalação e configuração de serviços de provisionamento *bare metal* é evitado, restando apenas a configuração do serviço de rede da nova VM e a configuração para as próximas máquinas *bare metal* a serem criadas. As etapas restantes para conclusão de uma instalação completa de uma *slice* com seus serviços são a instalação do kolla-ansible e a instalação e configuração do controlador Openstack que serão utilizados.

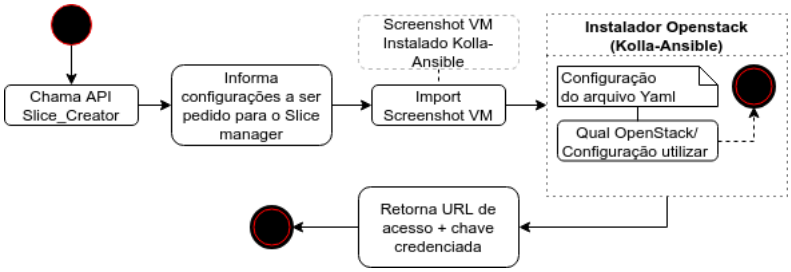



Figura 3. VIM *on-Demand* pré-carregada.

A Figura 3 apresenta passos necessários para a finalização da instalação. A eliminação do grupo de Criação e Configuração de VM e *Install services for bare metal Provisioners* são tarefas que diferenciam os grupos de atividades visualizadas na Figura 2, restando o grupo de instalação Openstack diretamente no Kolla-Ansible.

### 3.2.3. Instanciação Completa com *containers*

A última instanciação elimina todos os processos de instalação dos subgrupos anteriores: Criação de VM base para os serviços, instalação do Openstack Ironic (*Bare Metal*) e instalação do Openstack Kolla-ansible (Controlador Openstack).

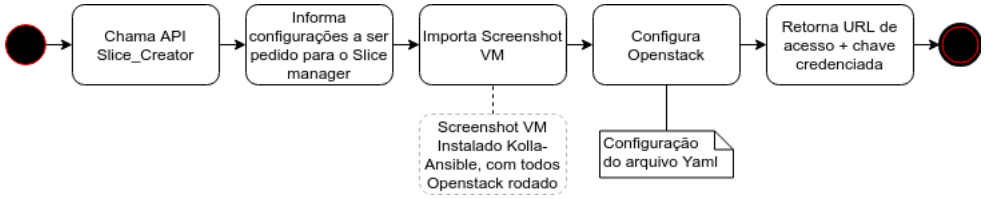



Figura 4. VIM *on-Demand* com *containers*.

Embora as etapas anteriores que pudessem tornar mais confusa e onerosa durante a instalação, esta instanciação simplificada não envolve tarefas de instalação de pacotes ou serviços. Neste método é necessário a importação do *snapshot* pronto, conforme demonstra a Figura 4, e configuração.

## 3.3. Aparato Computacional

Para realizar a implantação do ambiente foi utilizado o software de virtualização Virtualbox em conjunto com Vagrant, os quais são softwares com finalidade de criação e manutenção de ambientes de desenvolvimento virtualizados . A fim de controlar as máquinas virtuais em nível físico foi utilizado o Virtual BMC.

Com o objetivo de gerenciamento e provisionamento das máquinas físicas foi utilizando o Openstack Ironic. Para instalação de controladores que permitam operar nuvens utilizamos o Openstack Kolla-Ansible, pois este é composto por *containers* prontos para produção e ferramentas de implantação de Openstacks. Para a implementação do VIM *on-Demand* utilizamos o servidor com as especificações descritos na Tabela 1.

Para criação e controle das máquinas virtuais foi utilizado o software Vagrant, versão 2.2.3. Para exportação e importação do ambiente virtualizado foi utilizado o software

Tabela 1. Quadro de configurações do servidor utilizado.

| Componente | Dimensão |
|---|---|
| Processador | i5 3.30GHz 12 núcleos |
| Disco Rígido | 1 terabytes |
| Memória RAM: | 32 Gigabytes |
| Sistema Operacional | Linux Ubuntu Server 64 bit |

Virtualbox, versão 5.2.18. A utilização do software de virtualização Vagrant deve-se às suas características de fácil controle das máquinas, tais como a criação, o controle e a remoção do ambiente virtualizado. O Virtual BMC servi-se para gerenciar e monitorar as VMs, permitindo executar comandos a nível de máquinas *bare metal* reais, utilizando protocolo IPMI. Com o aparato computacional preparado o processo de criação do VIM *on-Demand* pode ser dividido em três partes: *bare metal*, Kolla-Ansible e a Versão do *Controller* Openstack.

### 3.4. Descrição dos experimentos

Para atender aos requisitos do ambiente proposto, definiu-se um modelo de arquitetura conforme apresentada na Figura 5, onde é possível visualizar as etapas de execução da implantação das tecnologias utilizadas em conjunto com as tarefas de importação e exportação de imagens. É possível observar que as atividades de instalações e configurações foram divididas em 3 ambientes, sejam eles:

- ***Machine Server*:** Máquina Física que recebe a instalação de um sistema e dos softwares responsáveis por criar e controlar um ambiente virtualizado;
- ***Machine Bare Metal*:** Máquina Virtual composta pelos serviços Openstack, e responsável pela instanciação da Máquina Física *Bare Metal*;
- ***Machine Full*:** Máquina Virtual anterior composta também com o Kolla-Ansible e responsável pela instalação dos serviços Openstack na Máquina Física *Bare Metal*.

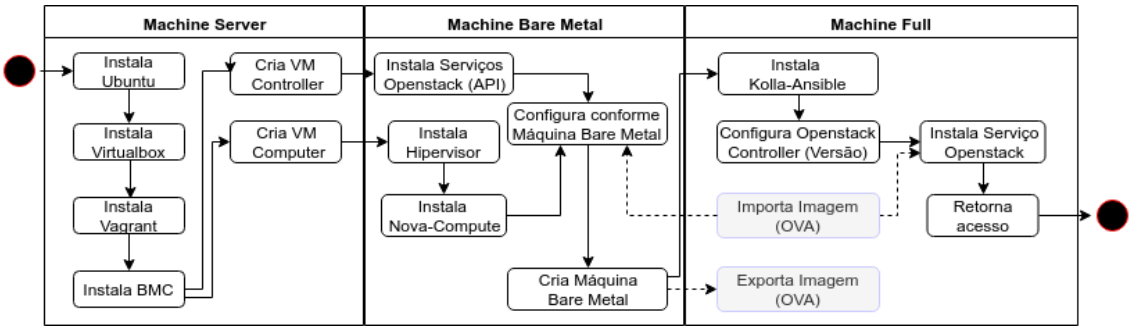



Figura 5. Diagrama de atividades em conjunto com as tecnologias utilizadas em cada VM.

Para a exportação e importação das imagens durante a instanciação com imagem pré-carregada e instanciação completa com *containers*, utilizou-se o comando VBoxManager, o qual é uma interface de linha de comando utilizada para controle do VirtualBox, que permite acessar todos os recursos presentes na interface gráfica do usuário.

A execução do experimentos contemplaram 27 repetições para cada aplicação de VIM *on-Demand* proposto, utilizando cálculos de tomada de tempo para verificação de desempenhos durante sua instanciação. Cada execução completa foi dividida em partes e os seus tempos de execução foram capturados. A execução completa para criação de um VIM *on-Demand* foi subdividida em:

- ***BM***: Representa o tempo dos procedimentos de criação e configuração de VMs (Controller e Computer), instalação do Openstack Ironic, configuração e instanciação da máquina *bare metal*;
- ***KA*** (Kolla-Ansible): Representa o tempo dos procedimentos inicializados após a finalização da criação da máquina *bare metal*, a instalação do Openstack Kolla-Ansible e configuração de ambiente para instalação do controlador Openstack na máquina *bare metal*;
- ***OS*** (Openstack): Representa o tempo do processo de configuração e instanciação da máquina para instalação do serviço Openstack;
- ***TOTAL***: Combinação dos períodos do *Bare Metal* (BAREMETAL) + Kolla-ansible (KA) + Openstack (OS);
- ***ExpVMBM*** e ***ExpVMKA*** (Export VM): Tempo correspondente a exportação de máquina virtual;
- ***ImpVMBM*** e ***ImpVMKA*** (Import VM): Tempo correspondente a importação da máquina virtual;

Tabela 2. Configuração das Máquinas Virtuais

| **CPU/Mem** | **4 GB** | **9 GB** | **14 GB** |
|---|---|---|---|
| **2 Núcleos** | 2N4G | 2N9G | 2N14G |
| **4 Núcleos** | 4N4G | 4N9G | 4N14G |
| **6 Núcleos** | 6N4G | 6N9G | 6N14G |

Uma execução completa pode ser considerada como a execução de todos os procedimentos de instalação e a configuração até o resultado final de criação da fatia com o serviço Openstack Tacker. Foram realizados três testes (execuções completas) para cada etapa da criação do *slice*. As configurações das VMs foram separadas por núcleos e memórias, sendo criadas conforme a Tabela 2.

## 4. Discussão dos Resultados

Após a captura do tempo de cada etapa durante o processamento, visando realizar cálculos de tomada de tempo para verificação de desempenhos, extraiu-se os menores valores de tempo compilados na Tabela 4. Observando os valores dos tempos para cada configuração de máquina virtual, conclui-se que o melhor computo de tempo foi 3,65 minutos, se refere ao teste ***OS*** seguido do teste ***KA*** que computou 9,52 minutos. A Tabela 3 apresenta a média das execuções de cada configuração de máquina, onde é possível observar novamente que as etapas ***OS*** e ***KA*** computaram os menores tempos médios, respectivamente 4,10 e 11,50 minutos. É possível observar também que as etapas ***KA*** e ***OS*** sucedem em ordem de tempo, tanto para a tomada de menor tempo quanto para a de tempo médio.

A etapa ***BM*** apresentou os maiores valores de tempos para todas configurações de máquina virtual quando comparado com as etapas ***KA*** e ***OS***, tendo como menor tempo o

Tabela 3. Média dos Tempos (Min)

| VM | BM | KA | OS | TOTAL |
|---|---|---|---|---|
| **2N 4G** | 67,84 | 20,10 | 5,19 | 94,27 |
| **2N 9G** | 59,31 | 14,85 | 4,33 | 78,95 |
| **2N 14G** | 47,83 | 10,35 | 3,99 | 62,51 |
| **4N 4G** | 50,93 | 18,29 | 4,77 | 73,80 |
| **4N 9G** | 42,82 | 11,50 | 4,10 | 58,51 |
| **4N 14G** | 40,97 | 10,35 | 3,74 | 55,46 |
| **6N 4G** | 46,07 | 14,62 | 4,46 | 64,84 |
| **6N 9G** | 40,84 | 10,87 | 3,99 | 55,38 |
| **6N 14G** | 39,27 | 9,52 | 3,65 | 52,62 |
| **Média** | 46,07 | 11,50 | 4,10 | 62,51 |

Tabela 4. Menores Tempos (Min)

| VM | BM | KA | OS | TOTAL |
|---|---|---|---|---|
| **2N 4G** | 58,89 | 20,10 | 5,19 | 94,27 |
| **2N 9G** | 56,61 | 14,85 | 4,33 | 78,95 |
| **2N 14G** | 42,68 | 10,35 | 3,99 | 62,51 |
| **4N 4G** | 48,28 | 18,29 | 4,77 | 73,80 |
| **4N 9G** | 41,83 | 11,50 | 4,10 | 58,51 |
| **4N 14G** | 39,34 | 10,35 | 3,74 | 55,46 |
| **6N 4G** | 43,66 | 14,62 | 4,46 | 64,84 |
| **6N 9G** | 39,80 | 10,87 | 3,99 | 55,38 |
| **6N 14G** | 38,44 | 9,52 | 3,65 | 52,62 |
| **Menor** | 38,44 | 9,52 | 3,65 | 52,62 |

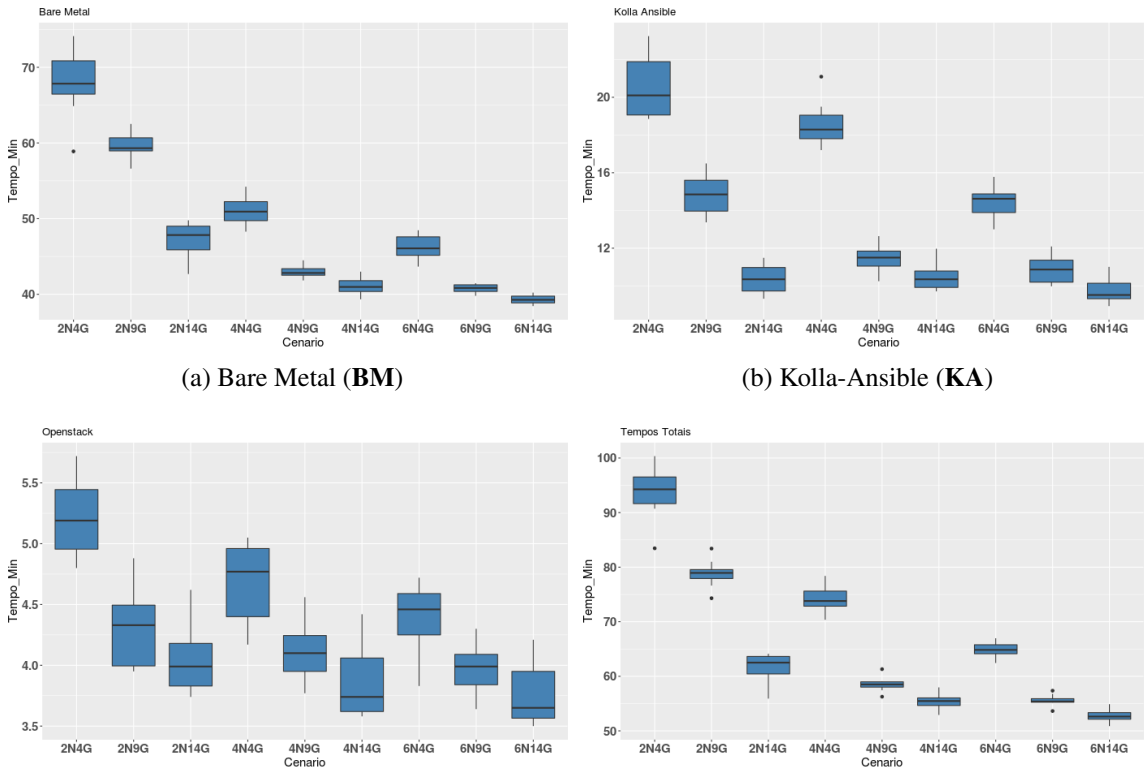
valor de 38,44 minutos. Ao observar mais atentamente, é possível notar que independentemente das configurações dos números de núcleos e memória, a ordem de relevância em função da tomada de tempo obedece sempre a mesma ordem, ou seja, na sequência ***BM***, ***KA*** e ***OS***. Para a representatividade de uma instalação completa, realiza-se a computação de todas as etapas para cada configuração de máquinas virtuais, tendo como o valor médio 62,51 minutos e como menor valor o tempo de 52,62 minutos.

De modo a observar o quão relevante se faz a capacidade computacional diante das tarefas propostas, observa-se que na Figura 6a, para o cenário do passo ***BM*** a quantidade de núcleos e memória interferem positivamente na tomada de tempo. A execução combinando 6 núcleos de processamento, com 14 e 9 gigabytes de memória RAM, aferiram desempenhos similares. Entretanto, com 14 gigabytes há uma distribuição de tempo mais estável, podendo ser observado pela menor distância entre o sétimo e o nono quartil. Desempenho semelhante ocorre para configuração com 4 núcleos e com 14 e 9 gigabytes de memória.

As tomadas de tempo para o passo ***KA***, observadas na Figura 6b, assemelham-se aos cômputos do passo ***OS***, exceto pela escala de tempo na qual a ordem de tempo do ***OS*** é menor. A similaridade do comportamento repete-se inclusive entre os *boxplots* das configurações de 6 núcleos combinados com 14 e 9 gigabytes, sinalizando homogeneidade das tomadas de tempo.

A Figura 6d apresenta a somatória dos resultados das etapas ***BM***, ***KA*** e ***OS***, onde demonstram a representatividade total da execução durante os testes, permitindo visualizar alguns pontos flutuantes.

Dos experimentos realizados com ***OS***, observa-se comportamento normal, o que pode ser visualizado na Figura 6c. Ao observar as tomadas por núcleos, analisamos que os grupos com 2, 4 e 6 núcleos, são decrescentes e para cada grupo há também uma variação decrescente conforme aumenta a quantidade de memória. Entretanto, ao se observar o comportamento entre os experimentos com 14 gigabytes de memória combinados com 6 e 4 núcleos, nota-se resultados similares em termos de estabilidade, ou seja, o conjunto de tomada de tempos possui igualdade na distribuição normal, mesmo advindo da configuração com menor número de núcleos, sugerindo que neste caso, a quantidade de memória é preponderante.




(a) Bare Metal (**BM**)

(b) Kolla-Ansible (**KA**)

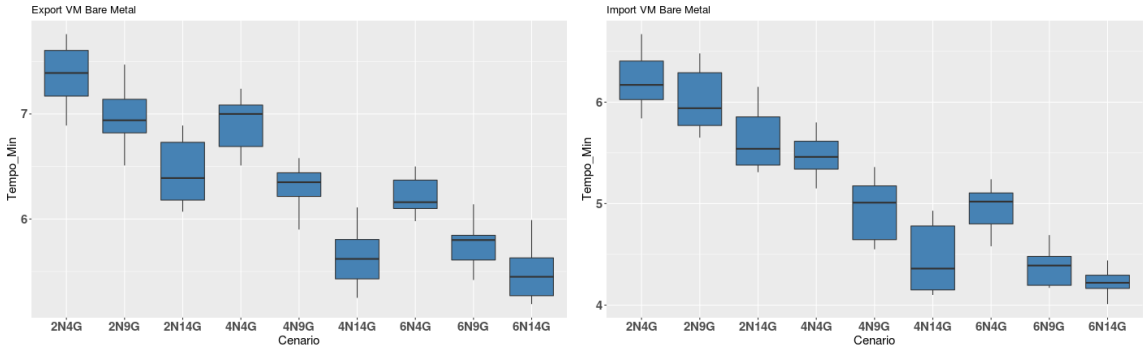
(c) Openstack (**OS**)

(d) Tempos Totais (**TOTAL**)

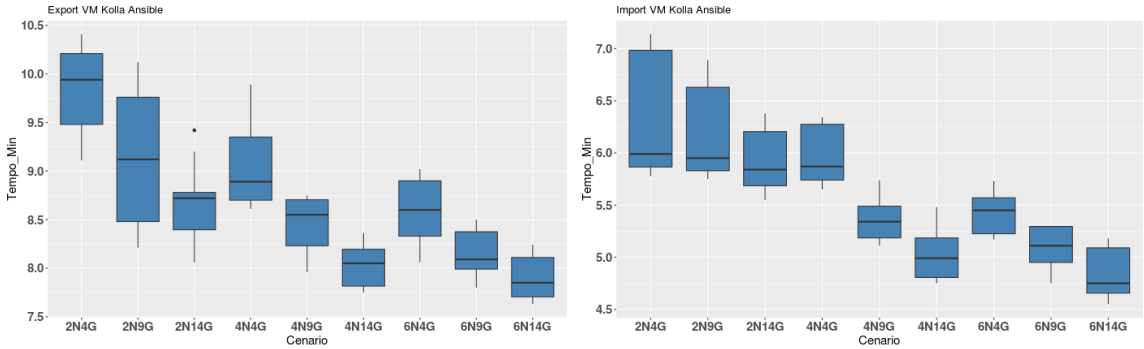
Figura 6. Execuções das etapas por configurações de máquinas virtuais.

Considerando as possíveis opções de instalações: Instanciação Completa com Imagem pré-carregada (Subseção 3.2.2) e Instanciação Completa com *containers* (Subseção 3.2.3), averiguou-se os tempos para exportação e importação de cada *snapshot*, conforme a Figura 7. As Figuras 7a e 7b representam, respectivamente a exportação e importação do *snapshot* de uma máquina virtual nas quais já foram realizadas as configurações das suas interfaces, instalação e configuração de serviços de provisionamento *bare metal*; enquanto as Figuras 7c e 7d reproduzem, respectivamente, a exportação e importação do *snapshot* de uma máquina virtual onde já foram realizadas a instalação e configuração do Openstack Ironic (*bare metal*) e do Openstack Kolla-ansible (Controlador Openstack).

Os tempos apresentados pelas Figuras 7c e 7d, foram maiores que os das Figuras 7a e 7b já que os *snapshots* utilizados possuem maior tamanho correspondente a quantidade de mais instalações realizadas no mesmo. Apesar deste fato, pode-se perceber que durante a realização das exportações para ambas situações (*bare metal* e kolla-ansible), suas realizações foram similares, assim como as execuções das importações. O principal fator de diferencial entre as exportações e importações foram os altos tempos de processamento apresentados durante as importações dos *snapshots* com o cenário de 2 núcleos. Quando comparado com todos os testes realizados em etapas anteriores, mesmo a situação com a menor quantidade de núcleos (2) e com a maior quantidade de memórias (14GB), acabou sendo tão eficientes quanto a uma situação de quantidade de 4 núcleos e com 4GB de memória.




(a) Export VM Bare Metal (**ExpVMBM**) (b) Import VM Bare Metal (**ImpVMBM**)




(c) Export VM Kolla-Ansible (**ExpVMKA**) (d) Import VM Kolla-Ansible (**ImpVMKA**)

Figura 7. Exportação e importação de máquinas virtuais.

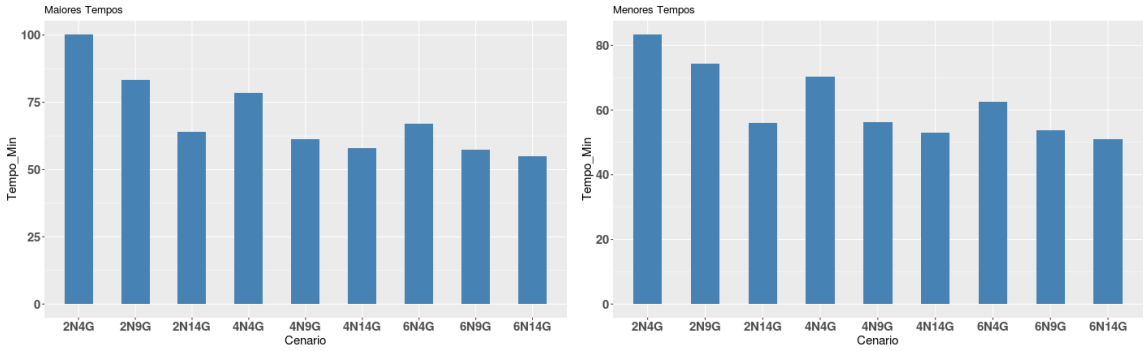
Corroborando para a constatação da prevalência da quantidade de memória em detrimento do número de núcleos, notamos nas Figuras 8a e 8b, uma ligeira superioridade quanto a estabilidade, ou seja, as barras relativas às configurações com 14 gigabytes de memória sobre as configurações com 6 núcleos.

A Figura 8a apresenta os maiores tempos para cada configuração de máquina virtual durante o processo de execução dos testes ***BM***, ***KA*** e ***OS***, enquanto a Figura 8b, ao contrário da anterior, apresenta os menores tempos durante o processo de execução dos testes. Ambas apresentam a variação de período durante a execução para cada cenário, permitindo visualizar, embora timidamente, uma relevante diferença entre cada cenário de configurações das máquinas virtuais.

Considerando o comportamento homogêneo para os cenários apresentados até o momento é possível afirmar à questão *(i)* através dos recursos computacionais que são preponderantes na eficiência do processo e dadas nossas observações a memória principal, prevalece sobre a quantidade de núcleos.

Com relação às médias das tomadas de tempo para os experimentos, a Figura 6d exibe as médias e respectivas marcações de desvio padrão. É possível observar que obstante a onerosidade do tempo de instalação do *bare metal* (Figura 6a), os tempos dos passos de Exportação e Importação de VMs (Figura 7) são extremamente vantajosos em relação aos tempos de instalação do *Kolla-Ansible* (Figura 6b) e *Openstack* (Figura 6c).

É notória a diferença estatística das atividades de não instalação, ou seja, importa-




(a) Maiores Tempos

(b) Menores Tempos

Figura 8. Tempo MAX e MIN

ção de configurações de VMs sobre os tempos de instalação, pois os tempos das mesmas são inferiores ao tempo médio de ambas e subtraídas do desvio padrão. Deste modo, pode-se afirmar que a resposta para a questão *(ii)* aponta no sentido de que a abordagem VIM *on-demad* é fortemente vantajosa, implicando na eficiência de utilização de recursos computacionais sob esta abordagem.

## 5. Considerações finais

Este trabalho apresentou um modelo de VIM *on-demand*, a qual possibilita instanciar máquinas a nível *bare metal* e serviços de computação, proporcionando, ao final, o seu controle e gerenciamento. Além disso, também foi possivel dividir o processo de instalação, configuração e instanciação em 3 etapas e exportar um *snapshot* para cada ambiente: (*I*) Instanciação Completa *Bare Metal*, (*II*) Instanciação Completa com *Containers* e (*III*) Instanciação Completa com Imagem Pré-Carregada.

Ao concluir os experimentos, observa-se que os melhores resultados durante os experimentos foram os que usaram memória RAM com 14 GB (G14), a maior quantidade de memória utilizada durante os testes. Diante disso, conclui-se que a quantidade de memória é aspecto preponderante no VIM *on-Demand*, pois os valores que apresentaram melhor desempenho foram em situações de maior quantidade de memória, e não alteração de outros fatores como quantidade de núcleos. Adicionalmente à preponderância da quantidade de memória, foi possível constatar que a abordagem VIM *on-demand* confere ganho computacional no tocante a eficiência de uso dos recursos.

Como trabalhos futuros, pretende-se incluir outros controladores, como o VLSP e Kubernets, no VIM *on-Demand*, e realizar novos experimentos analisando consumo de energia, desempenho de memória e processamento com casos de usos que necessitem de uma sobrecarga.

## Agradecimentos

Esta pesquisa é financiada pela Comissão Europeia e Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP e CTIC no contexto da 4º chamada conjunta EU-BR, acordo #777067 (*NECOS - Novel Enablers for Cloud Slicing*). Este estudo foi também parcialmente financiado pela CAPES e CNPq.

## Referências

Clayman, S. (2017). Network slicing supported by dynamic vim instantatiation.

Clayman, S., Tusa, F., and Galis, A. (2018). Extending slices into data centers: the vim on-demand model. In *2018 9th International Conference on the Network of the Future (NOF)*, pages 31–38. IEEE.

Dräxler, S., Karl, H., Peuster, M., Kouchaksaraei, H. R., Bredel, M., Lessmann, J., Soenen, T., Tavernier, W., Mendel-Brin, S., and Xilouris, G. (2017). Sonata: Service programming and orchestration for virtualized software networks. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 973–978. IEEE.

Freitas, L. A., Braga, V. G., Corrêa, S. L., Mamatas, L., Rothenberg, C. E., Clayman, S., and Cardoso, K. V. (2018). Slicing and allocation of transformable resources for the deployment of multiple virtualized infrastructure managers (vims). In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 424–432. IEEE.

Han, B., Gopalakrishnan, V., Ji, L., and Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97.

Kim, J., Kim, D., and Choi, S. (2017). 3gpp sa2 architecture and functions for 5g mobile communication system. *ICT Express*, 3(1):1–8.

Mademann, F. (2017). System architecture milestone of 5g phase 1 is achieved. *Online verfügbar unter: http://www.3gpp.org/NEWS-EVENTS/3GPP-NEWS/1930-SYS_ARCHITECTURE, letzter Zugriffam*, 1:2018.

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J., and Folgueira, J. (2017). Network slicing for 5g with sdn/nfv: concepts, architectures and challenges. *arXiv preprint arXiv:1703.04676*.

Rost, P., Mannweiler, C., Michalopoulos, D. S., Sartori, C., Sciancalepore, V., Sastry, N., Holland, O., Tayade, S., Han, B., Bega, D., et al. (2017). Network slicing to enable scalability and flexibility in 5g mobile networks. *IEEE Communications magazine*, 55(5):72–79.

Silva, F. S. D., Lemos, M. O., Medeiros, A., Neto, A. V., Pasquini, R., Moura, D., Rothenberg, C., Mamatas, L., Correa, S. L., Cardoso, K. V., et al. (2018). Necos project: Towards lightweight slicing of cloud federated infrastructures. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 406–414. IEEE.

# Slices Como Serviço Sobre um Centro de Dados Itinerante Aplicado ao Cenário Amazônico

**Josiane Ribeiro[1], Lucas Borges [1], Paulo Senna[1], Billy Pinheiro[1], Antônio Abelém[1]**

[1]Grupo de Pesquisa em Redes de Computadores e Comunicação Multimídia (GERCOM)
Universidade Federal do Pará (UFPA)
Caixa Postal 470 – 66075-110 – Belém – PA – Brasil

{josianeribeiro796,borgeslucas873,paulosena99}@gmail.com

{billy,abelem}@ufpa.br

***Abstract.*** *Virtualization and resource slicing are becoming key approaches to automating, streamlining and cost-cutting the cloud configuration process, enabling the provision of agile and dynamic services. This paper presents the Itinerant Data Center (IDC), with the objective of bringing essential services to regions with or without infrastructure, using the NECOS platform, to provide a sustainable, low-cost and with low energy consumption, which uses cloud computing, virtualization, and slice-as-a-service.*

***Resumo.*** *A virtualização e o fatiamento de recursos vêm se tornando abordagens chaves para automatizar, tornar mais eficiente e econômico o processo de configuração de nuvens, habilitando a oferta de serviços ágeis e dinâmicos. Este artigo apresenta o Centro de Dados Itinerante (Itinerant Data Center – IDC), com o objetivo de levar serviços essenciais para regiões sem ou com pouca infraestrutura, utilizando a plataforma NECOS, para fornecer uma solução sustentável, de baixo custo e com pouco consumo de energia, que utiliza computação em nuvem, virtualização e fatiamento como serviço.*

## 1. Introdução

A maioria dos excluídos digitais vivem em zonas rurais ou regiões de ilhas de países em desenvolvimento, como nas ilhas amazônicas que apresentam uma população de baixa renda, regiões esparsamente povoadas, dependência da malha hidroviária e a falta de infraestrutura de tecnologia da informação. Tais fatores contribuem para tornar o planeta um lugar "desconectado". Para sanar este problema, muitos projetos têm sido desenvolvidos: Projeto Amazônia Conectada [Conectada 2015], balões e drones para o acesso à Internet [Vasconcelos 2017], o CoDPON [Coutinho 2009] e o CELCOM [de Castro et al. 2018].

Diante deste cenário, a conexão à Internet torna-se um elemento essencial para viabilizar os serviços computacionais baseados em nuvens. Esta tecnologia atualmente é imprescindível para os serviços que utilizamos no nosso cotidiano, porém, a computação em nuvem pode perder parte de sua eficiência devido ao advento de novas tecnologias, como a Internet das Coisas *(Internet of Things)* que geram um grande volume de dados *(Big Data)*, que precisam de atendimento prioritário, inclusive para não sobrecarregar as próprias nuvens e os meios de comunicação [Castilho and Kamienski 2018]. Adicionalmente, a computação em nuvem é vista como um modelo que permite o acesso a recursos computacionais configuráveis (como redes, servidores, armazenamento, aplicações

e serviços) por seus clientes que pagam conforme a capacidade reservada dos recursos, independente da efetiva utilização destes [Marques and Obelheiro 2017].

No entanto, o aumento da latência na computação em nuvem, devido a distância entre a nuvem e os dispositivos finais pode ser um problema para aplicações sensíveis à latência. Visando, entre outros pontos, minimizar a latência dos serviços em nuvem, a computação em névoa surgiu. A névoa *(Fog)* é alocada no ponto mais conveniente entre o usuário final e a nuvem *(Cloud)*, de modo a otimizar o processamento, armazenamento e transmissão da informação, proporcionando melhoria em todo o processo de comunicação de dados ao deixar a computação mais próximo da borda [Castilho and Kamienski 2018].

Para fornecer uma abstração única dos recursos das nuvens (processamento e armazenamento) e de redes, a plataforma NECOS[1] utiliza o conceito de *(cloud network slicing)* [Silva et al. 2018]. Este conceito ainda apresenta várias definições, mas, no geral, uma fatia da nuvem pode ser definida como uma rede lógica fim a fim (*end-to-end - E2E*) sob demanda, contando com uma infraestrutura subjacente comum, composta de recursos físicos e/ou virtuais, com controle, gerenciamento e orquestração independentes. Estas fatias devem ser mutuamente isoladas e flexíveis o suficiente para acomodar diferentes necessidades dos inquilinos *(tenants)*, fornecendo um modelo de Fatia como um Serviço (*Slice-as-a-Service*) e fornecendo a gerência virtual da infraestrutura (*Virtual Infrastructure Manager - VIM*) sob demanda [Clayman et al. 2018].

Nesse contexto, este artigo implementa a plataforma NECOS com o uso de Computação em Névoa e migração de serviços, para propor o uso de Centro de Dados Itinerante *(Itinerant Data Center - IDC)* dentro de uma fatia (*slice*). A essência da proposta é ter uma infraestrutura em um barco, que hospeda microsserviços sob demanda, permitindo a oferta de serviços nas ilhas sem acesso à Internet, ou seja, onde a população é excluída digitalmente. Esses serviços são alocados em fatias na nuvem e a migração/replicação do serviço é realizada no equipamento do IDC. Assim, além de propiciar um ambiente com serviços de redes diversos, será possível atingir maiores públicos levando até essas regiões um ambiente que dê suporte à projetos, como a educação a distância, agendamentos de consultas, atendimentos médicos à distância entre outros.

Além desta seção introdutória, este artigo é composto de mais 5 seções. A Seção 2 apresenta a arquitetura da plataforma NECOS na qual o IDC foi implementado. A Seção 3 descreve os trabalhos relacionados. A Seção 4 apresenta o IDC, seguido de sua aplicação na Seção 5. Enquanto a Seção 6 apresenta as conclusões gerais.

## 2. Arquitetura NECOS para Slicing

A arquitetura completa da plataforma NECOS define os componentes necessários para o aprovisionamento de fatiamento E2E em domínios de federação multi-tecnológicos e de multi-provedores [Silva et al. 2018]. Na Figura 1 nós apresentamos uma versão simplificada da arquitetura destacando os elementos mais relevantes no contexto do IDC.

Para facilitar a compreensão da arquitetura do NECOS, na Figura 1 nós definimos quatro áreas distintas, isto é, a área *Tenant* em vermelho, a área de Provedor de Recursos em verde, a área do *Marketplace* em amarelo e o Provedor de Slice em azul.

[1] http://www.h2020-necos.eu/

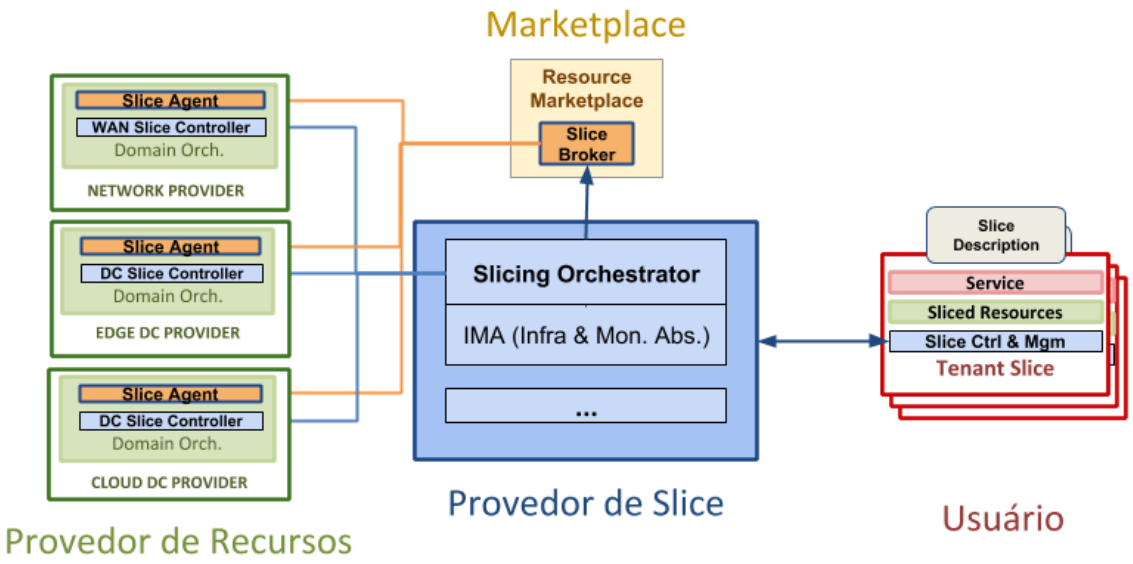



**Figura 1. Visualização abstrata da arquitetura NECOS.**

O *Slice Orchestrator* realiza manipulação de fatias, isto é, requisita ao *Marketplace* as diferentes partes da fatia que serão incluídas em uma fatia E2E através da realização de uma fase de orquestração inicial. Adicionalmente, ele supervisiona o gerenciamento do ciclo de vida dos serviços que estão sendo executados nas fatias, baseado na informação recuperada do componente *IMA*.

A Abstração de Monitoramento & Infraestrutura (Infrastructure & Monitoring Abstraction - IMA) oferece uma interface *northbound* abstrata que permite ao *Slice Orchestrator* executar suas funções enquanto os detalhes das partes das fatias permanecem agnósticos, por exemplo, coletando informações sobre topologia de recursos e informações de monitoramento de recursos para cada parte da fatia; monitorando e verificando o status dos elementos virtuais alocados a cada parte da fatia; reunindo Indicadores Chaves de Performance (*Key performace Indicators* - KPIs) voltados para recursos (como CPU, memória e armazenamento). Para alcançar isso, múltiplos adaptadores traduzem as requisições provenientes da interface *northbound* para o VIM ou o Gerenciador de Infraestrutura WAN (*WAN Infrastracture Manager* - WIM). Deste modo, eles escondem as partes das fatias heterogeneamente na interface *southbound* da *IMA*.

A descoberta de recursos que compõem uma fatia E2E é designada ao componente *Marketplace* que é um sistema distribuído responsável por localizar partes de fatia adequadas a partir de um conjunto de domínios de recursos participantes. Como mostrado na Figura 1, o *Marketplace* consiste em dois sub componentes principais do NECOS, nomeados de *Slice Broker* e um número de *Slice Agents*. O *Slice Broker* descobre as partes das fatias através da interação com um conjunto de *Slice Agents* hospedados pelos domínios de recursos envolvidos.

## 3. Trabalhos Relacionados

Baseado na tecnologia de Redes Tolerantes a Atraso *(Delay Tolerant Networks - DTN)* o CoDPON foi proposto com o intuito de viabilizar a integração de programas sociais do governo entre comunidades remotas e grandes centros urbanos, a baixo custo. Para isso

a rede CoDPON utiliza barcos que se deslocam entre as comunidades com um módulo acoplado que recebe os dados e retransmite ao se aproximar de outros barcos ou a base (cidade). Tal solução não permite que um serviço seja acessado continuamente em um período de tempo especifico, limitando a abrangência da solução [Coutinho 2009].

O projeto SeNDT utiliza redes de sensores tolerantes a atrasos a fim de possibilitar o monitoramento da qualidade da água de lagos e da poluição sonora em rodovias. Com o intuito de solucionar o problema da grande extensão geográfica do lago que foi escolhido para monitoramento, foram utilizadas tecnologias de DTNs. Onde, os sensores foram divididos em regiões e mulas de dados (nós DTN móveis com grande capacidade de armazenamento) trafegam entre essas regiões, garantindo a conectividade da rede. Para o monitoramento da poluição sonora em rodovias foram instaladas unidades de sensoriamento, equipadas com antenas IEEE 802.11, que enviam as informações coletadas mediante uma consulta do operador, que pode permanecer em seu veículo durante a operação. Essa solução mostrou-se bastante vantajosa devido ao baixo custo, robustez e tolerância a atrasos [McDonald et al. 2007].

Através de uma metodologia de comunicação assíncrona, Shivani Harnal e Jasbir Kaur (2016) desenvolveram o DAKNET com o objetivo de conectar aldeias (áreas rurais) aos grandes centros urbanos, através da infraestrutura existente de transporte. Para isso o DAKNET utiliza ônibus ou qualquer outro veículo montado com um equipamento denominado MAP (Ponto de Acesso Móvel) que movimenta-se entre as aldeias, onde são instalados dispositivos denominados FAP (Ponto de Acesso Fixo). A partir do momento em que o veículo se aproxima da aldeia, as informações são baixadas automaticamente do FAP para o MAP e posteriormente serão retransmitidos quando o veículo retorna para a cidade. [Harnal and Kaur 2016]

Uma arquitetura de Névoa foi proposta por Kniess e Ruck com o objetivo de melhorar o desempenho no acesso a páginas web requisitadas através de dispositivos conectados a uma rede sem fio. Nesta arquitetura existe um módulo de consistência que atua como um balanceador de carga formado por um número "n" de servidores de névoa (por exemplo, um centro de dados em nuvem de pequena escala *(Cloudlets)*), que têm a função de armazenar as requisições mais frequentes dos usuários. Esse armazenamento é feito através de um ranqueamento das requisições recebidas dos usuários da névoa de forma a construir um mecanismo eficiente de resposta, visto que, os servidores atuam como um cache inteligente evitando acessos desnecessários à nuvem. Assim, é o IDC, que utiliza a Computação em Névoa por apresentar características como a baixa latência, a redução de tráfego e o pré-processamento de dados, fazendo com que a computação na borda minimize o volume de dados que precisa ser armazenado [Kniess and Ruck 2016].

A Tabela 1 apresenta uma comparação das características apresentadas anteriormente nesta sessão, mostrando o diferencial entre os projetos existentes CoDPON, SeNDT, DAKNET e a proposta deste artigo. O IDC se diferencia dos demais devido suas tecnologias utilizadas, que garantem que o serviço seja oferecido em comunidades isoladas com pouca ou nenhuma infraestrutura de comunicação, através de um centro de dados itinerante que se desloca para oferecer serviços de TIC no local em que se encontra, e não apenas oferecer conectividade. Através do IDC é possível realizar serviços de rede solicitados pelo cliente, mesmo este estando desconectado do núcleo da rede, pois sucessivamente, os dados serão sincronizados com o centro de dados principal, enquanto

| Projetos | Características | | | | |
|---|---|---|---|---|---|
| | Objetivos | Tecnologia utilizada | Funcionamento | Conectividade | Vantagens |
| CoDPON | Integrar programas sociais entre locais remotos e os centros urbanos | DTN | Os barcos se deslocam da origem até o destino | Necessária | Conectividade a baixo custo. |
| SeNDT | Monitoramento da qualidade da água de lagos | DTN | Os sensores trafegam entre regiões para garantir a conectividade | Necessária | Conectividade em ambientes agressivos. |
| DAKNET | Conectar áreas rurais aos grandes centros urbanos | DTN | Os veículos se deslocam da origem até o destino | Necessária | Conectividade a baixo custo |
| IDC | Permitir o acesso de serviços de TIC nas ilhas sem acesso à Internet | Fog | Um único barco se desloca da origem até o destino | Não necessária | Levar o serviço até o cliente com baixo custo |

**Tabela 1. Comparação das propostas.**

que, tanto o CoDPON , o SeNDT e o DAKNET através da tecnologia de DTN, precisam garantir uma certa conectividade para a realização de seus serviços, diante disto, o IDC se apresenta como uma solução inovadora, sustentável, de baixo custo e com pouco consumo de energia.

## 4. Centro de Dados Itinerante (IDC)

Este artigo apresenta um Centro de Dados Itinerante de borda (*Edge* IDC) capaz de promover acesso a diversos serviços de TIC. A ideia principal é ter um centro de dados itinerante que possa hospedar microsserviços sob demanda, permitindo a cobertura de ilhas de serviço sem acesso à Internet, ilhas que normalmente não atingem o núcleo da rede para que possam acessar os serviços. Estes serviços (microsserviços) precisam de uma alocação de fatia, a implantação do serviço ocorre na fatia e a migração/replicação do serviço no equipamento do *Edge* IDC.

### 4.1. Elementos da solução IDC

O IDC é composto por 3 equipamentos chaves: um computador pequeno (*DC-slice-part*) que hospeda uma fatia de serviços do centro de dados da nuvem, um roteador sem fio (*Net-slice-part*) que hospeda uma fatia de serviços que age como provedor de serviços de rede, e um fornecimento de energia. Prevemos, o uso de serviços de vídeo (somente leitura) e serviços de captura de questionário (leitura e escrita).

A Figura 2 mostra a arquitetura do *Edge* IDC. Em um primeiro momento, o *Edge* IDC recebe uma fatia do centro de dados da nuvem enquanto está conectado a rede. Em um segundo momento, ao se deslocar para ilhas da região, o *Edge* IDC, agora desconectado, realiza serviços de redes solicitados sob demanda, e que posteriormente serão sincronizados.

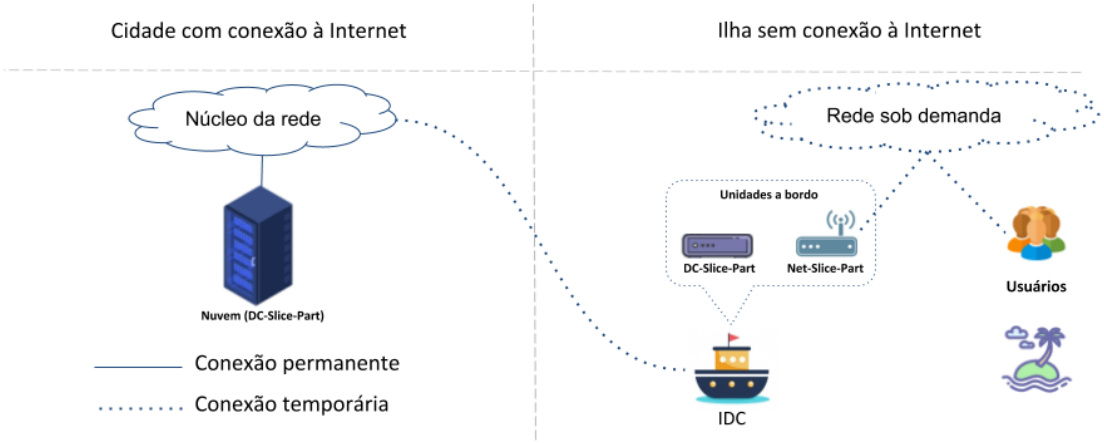



**Figura 2. A arquitetura da solução proposta é dividida em duas partes: "Núcleo da Rede" e "Unidade a Bordo".**

### 4.1.1. Unidade a Bordo

Uma fatia do centro de dados é transferida para o módulo com a criação de uma máquina virtual. Mediante a demanda do serviço de rede os recursos são alocados no módulo para realizar sua atividades. Basicamente, esta unidade apresenta duas funções: a primeira, como provisionador de VM, tendo os serviços implantados no módulo; e a segunda, com o Pré-processamento de dados, tendo os dados tratados para serem enviados para o servidor otimizando o canal de comunicação.

### 4.1.2. Núcleo da Rede

Esta unidade atua com três funções: a primeira, na gerência de dispositivos de borda monitorando as atividades dos módulos, a segunda, como Provisionador de fatia implantando o serviço requerido no módulo; e a Terceira, como sincronizador de dados recebendo os dados vindos do módulo para o servidor e os atualizando.

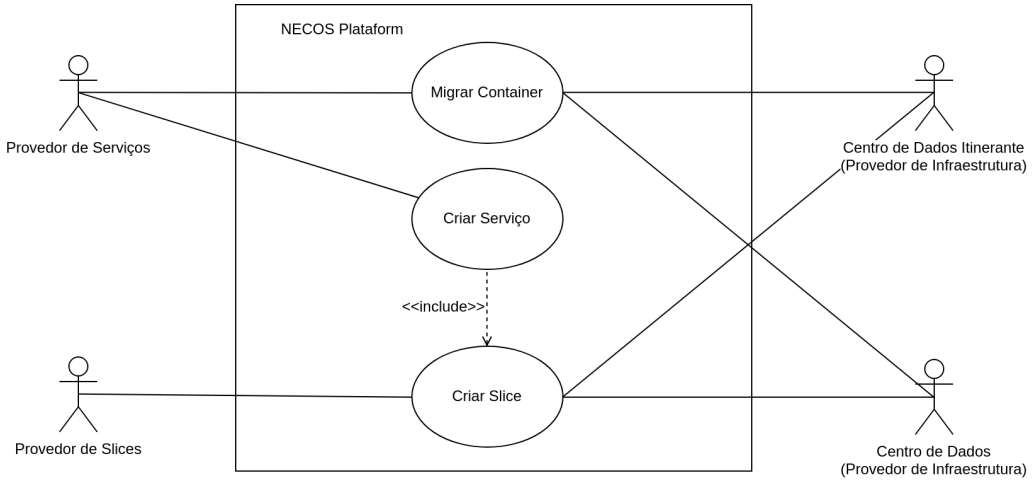



**Figura 3. Caso de uso de Criação de fatia usando o IDC.**

A Figura 3 mostra o caso de uso de criação da fatia com o IDC. Inicialmente o provedor de serviços deve solicitar a criação da fatia, que será realizada pelo provedor de fatia. O provedor de serviço deve inserir a definição de serviço a ser tratado pelo provedor de fatia para receber a definição da fatia usado para selecionar os recursos. Uma vez que a fatia é criada no núcleo do centro de dados e o serviço é instanciado, é possível migrar/replicar parte do serviço no IDC, aplicando o conceito de Computação em Névoa ativada pelas capacidades de NECOS.

### 4.2. Modelo de Negócio do IDC

A Figura 4 mostra o Modelo de Negócio do IDC. Através desta ferramenta de planejamento estratégico, é possível visualizar a forma como a proposta deste artigo irá operar e gerar valor ao mercado, definido seus principais fluxos e processos, permitindo uma análise e visualização de atuação no mercado.

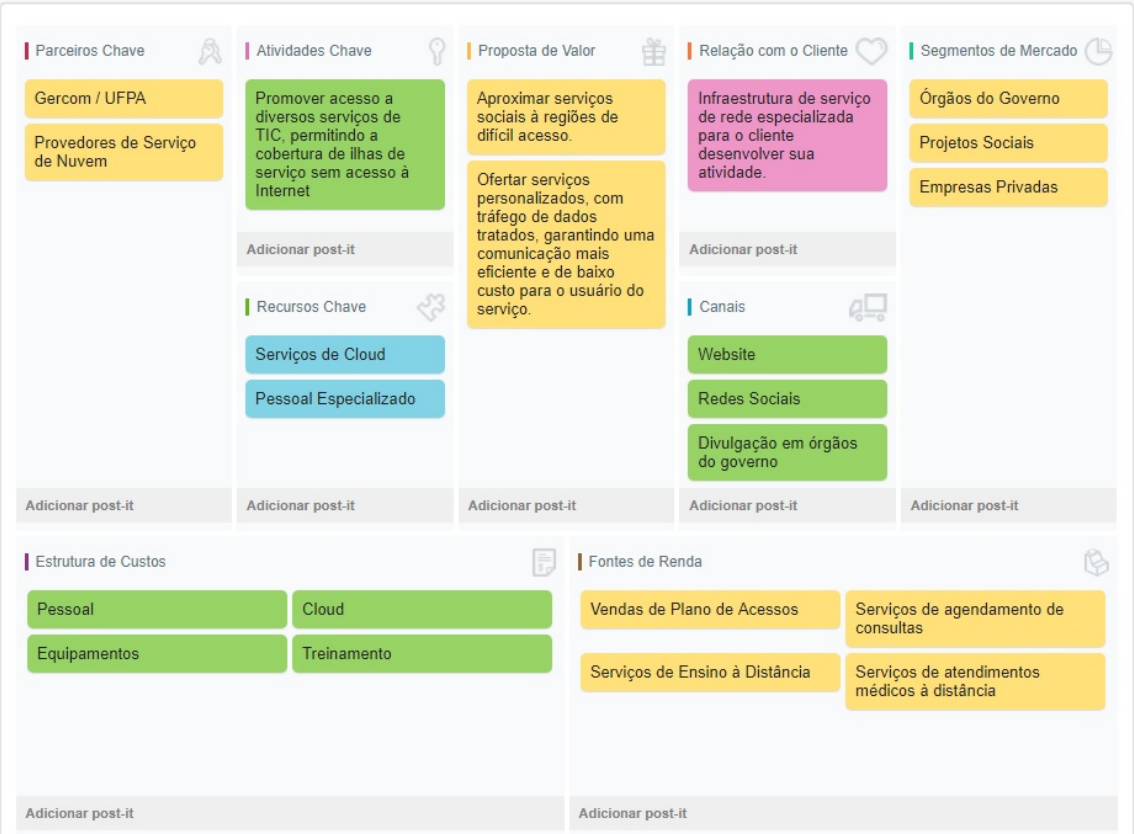



**Figura 4. Modelo de Negócio do IDC.**

A principal atividade do IDC é fornecer acesso aos diversos serviços de TIC, permitindo a cobertura de ilhas de serviço sem acesso à internet, o que o diferencia das demais propostas do mercado, que têm como objetivo oferecer apenas conectividade. O IDC leva o serviço até o cliente, ou seja, a Proposta de Valor do IDC, também se destaca, pois aproxima os serviços sociais às regiões de difícil acesso, garantindo uma comunicação mais eficiente e de baixo custo para o usuário do serviço.

Além disso, o IDC apresenta como recurso chave os serviços de Cloud (Computação em Nuvem e Computação em Névoa) tecnologias novas que permitem

alcançar tanto as Propostas de Valor quanto a Atividade Chave, visto que, as propostas de mercado utilizam a DTN como tecnologia para viabilizar seus serviços.

## 5. Prova de conceito do IDC

Para demonstrar a viabilidade da solução, nós implementamos uma prova de conceito do IDC executado sobre a ilha FIBRE da UFPA. Nela exercitamos a migração de serviço da nuvem para o IDC, simulando a aplicação do IDC para fornecer serviço para ilhas.

### 5.1. Ambiente de experimentação

Com a arquitetura do IDC em mente, o entendimento do ambiente de experimentação desenvolvido será mais simples, pois será dividido em duas partes, uma com acesso contínuo à Internet e outra que possuirá uma conexão temporária. Então, afim de construir esse ambiente é necessário a criação de uma topologia virtual de *switches*, nós *OpenFlow* e um controlador *OpenFlow* para controlar e permitir a conectividade entre os agentes envolvidos. Para alcançar este cenário e as tecnologias envolvidas, foi utilizado o ambiente de testes do FIBRE.

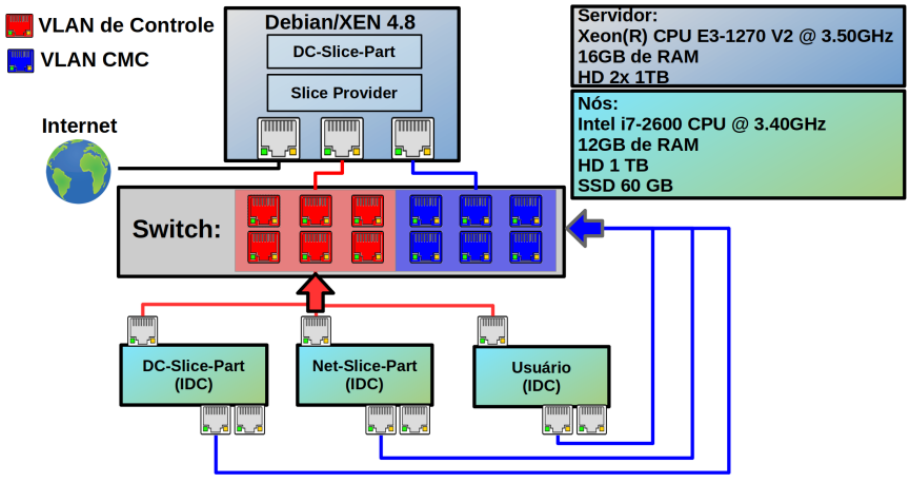



**Figura 5. Infraestrutura utilizada**

Desempenhando o papel de nuvem ou núcleo da rede, o servidor contém o *DC-Slice-Part* e *Slice Provider* responsáveis por gerir e implementar os serviços aos dispositivos que estão conectados através da *VLAN* de Controle, disposto na Figura 5. Os dispositivos são representados pelos nós Ícarus, os nós *DC-Slice-Part* (IDC) e *Net-Slice-Part* (IDC) componentes da Unidade a Bordo, estarão hospedando a fatia do serviço do centro de dados da nuvem e proporcionando serviços de rede, respectivamente. Enquanto que o nó Usuário (IDC) é o cliente final que estará sendo atendido pelo serviço IDC.

### 5.2. Componentes implementados

Nesta seção são descritos os componentes funcionais da arquitetura NECOS que foram implementados[2] para viabilizar os testes de validação da proposta, focando no provisionamento dos recursos. Embora as ideias iniciais de projeto desses componentes específicos

[2]https://gitlab.com/gercom/idc

estejam atualmente sob investigação, o design dos componentes descritos é maduro o suficiente para a implementação.

Na arquitetura NECOS, o *DC Slice Controller* é responsável por criar fatias do centro de dados, alocando os recursos necessários de computação e armazenamento para uma determinada fatia e retornando um identificador para o VIM em execução nele. O VIM pode ser uma instância separada implementada sob demanda com base na especificação do *Tenant* ou no VIM existente em execução nesse domínio de recursos, no qual um ponto de interação isolado foi criado para o *Tenant*.

O *Slice Controller* do Centro de Dados é responsável por gerenciar um conjunto de recursos, por exemplo computação e armazenamento, que devem ser alocados para o processo de fatiamento. Além disso, ele lida com solicitações de fatias e determina se é possível criar uma nova fatia com base na disponibilidade de recursos locais. Se a criação da fatia for possível, ela selecionará recursos do conjunto para serem alocados.

Em cada *Network Provider* reside um componente chamado *WAN Slice Controller* necessário para criar dinamicamente uma fatia da rede, como uma *Slice part* do fatiamento da nuvem. Uma fatia de rede é um conjunto de *links* virtuais que conecta duas fatias. Para criar uma fatia de rede, o *WAN Slice Controller* gerencia todos os recursos de rede alocados para participar do fatiamento e controla quais recursos de rede já foram alocados a cada fatia.

O *WAN Slice Controller* manipula as solicitações de fatias de rede e determina se é possível criar uma nova fatia de rede no domínio com base na disponibilidade de recursos locais (por exemplo, largura de banda). Se a criação da fatia de rede for possível, o *WAN Slice Controller* fornece o conjunto de *links* virtuais necessários para conectar as *Slice parts* fornecidas.

### 5.3. Resultados da prova de conceito

Os testes desenvolvidos buscam comprovar a viabilidade da proposta, demonstrando que é possível um serviço em execução no núcleo da rede ser levado e executado na borda visando alcançar comunidades que não possuem conexão à Internet. Para que isso ocorra é necessário que este serviço seja transferido do núcleo até a borda sem perdas de conteúdo.

O processo de criação do serviço é iniciado com o provedor de serviços solicitando ao *Slice Provider* a criação de um *slice* com um contêiner no nucleo da rede, visto na Figura 6(a). Após o processamento da requisição exibido na Figura 6(b), o serviço é acessado através da sua URL que direcionará ao IP mostrado na Figura 6(e). Os usuários que já estiverem conectados ao IDC começam a ter acesso ao conteúdo, de acordo com a Figura 6(f).

Após a inicialização do serviço, este já está pronto para ser transferido à borda e levado até as comunidades através do IDC. O provedor de serviços então envia uma solicitação ao *Slice Provider* que irá replicar o contêiner do núcleo ao IDC, permitindo então que o serviço seja executado também na borda, as Figuras 7(a), 7(b) e 7(c) exibem o processo.

A partir deste momento, o serviço que era acessível somente aos usuários com conexão à Internet, é acessível para usuários em locais remotos uma vez que o serviço está hospedado no IDC e todas as requisições serão encaminhadas para ele através do

(a) Provedor de serviço solicitando a criação da fatia com o serviço

(b) Processamento da solicitação pelo *Slice Provider*

(c) Contêiner criado no núcleo da rede

(d) Contêiner ainda não transferido ao IDC

(e) Requisições sendo encaminhadas ao núcleo

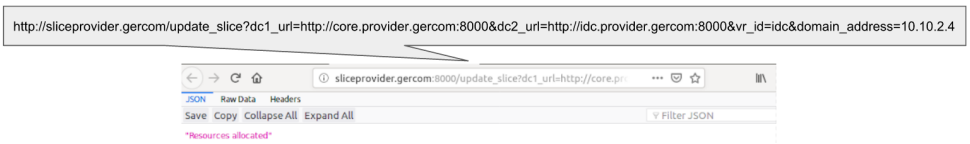
(f) Usuário com acesso ao serviço criado

**Figura 6. Criação do Slice**

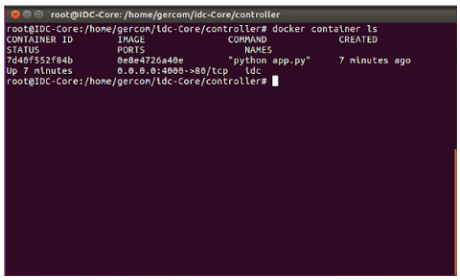
*Net-Slice-Part*, visto na Figura 8(b) e não mais à nuvem como na Figura 6(e). Mesmo com a perda de conexão entre o núcleo e o IDC visto na Figura 8(a), o serviço continua em execução proporcionando assim que os usuários consumam o conteúdo disponível visível na Figura 8(c).

Ao cumprir com a sua finalidade, o IDC retorna e estabelece uma conexão estável permitindo assim que o provedor de serviço solicite ao *Slice Provider* transferir o serviço que estava sendo executado no IDC de volta ao núcleo, as Figuras 9(a), 9(b) e 9(c) demonstram o processo. Com o retorno, requisições que *Net-Slice-Part* direcionava ao IDC será então direcionado ao nucleo novamente, sem interrupção e perdas de conteúdo aos usuários, como mostra a Figura 9(d).

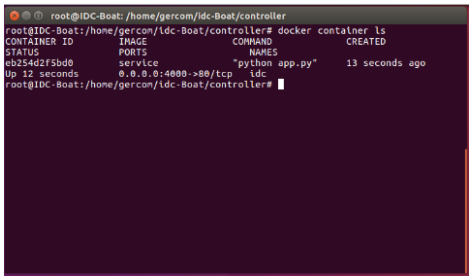
Portanto, do mesmo modo que os usuários que possuem acesso a internet interagem e consomem o conteúdo dos serviços disponíveis, usuários que estão distantes podem usufruir dos mesmos serviços. O NECOS permite o uso simples do data center itinerante fornecendo e unificando métodos de gerência de redes e nuvens. Com essa




(a) Comando de Replicação do serviço ao IDC

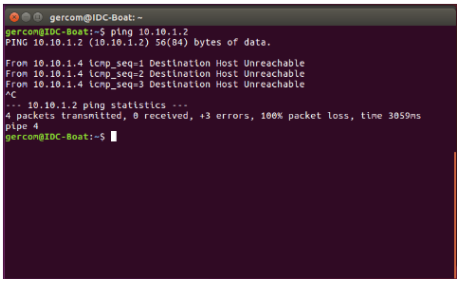



(b) Contêiner no núcleo

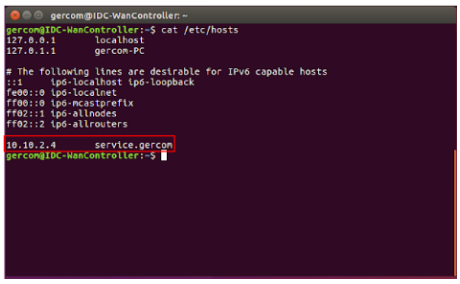



(c) Contêiner replicado no IDC

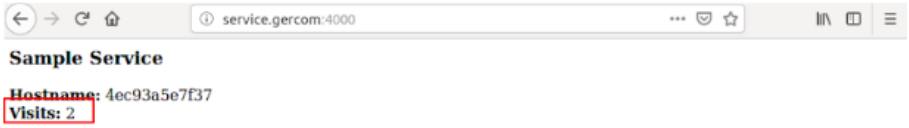
**Figura 7. Replicação do Serviço**




(a) IDC sem conexão com o núcleo da rede




(b) Requisições sendo encaminhadas ao IDC




(c) Usuário sem conexão com a internet com acesso ao serviço disponível

**Figura 8. IDC perde conexão com o nucleo da rede**

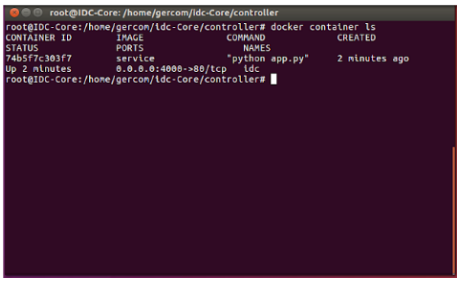
solução é possível prover serviços de nuvem para comunidades isoladas contribuindo para a redução dos custos desse tipo de serviços para estas comunidades.

## 6. Considerações Finais

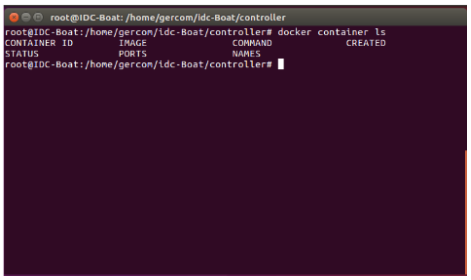
O isolamento tecnológico é uma realidade que afeta as comunidades ribeirinhas assim como aquelas que se encontram distantes dos grandes centros. Isso ocorre, não apenas pelas restrições tecnológicas, mas também por questões de retorno financeiro, visto que atender pequenas comunidades isoladas não cobre o custo financeiro de implantação das operadoras de telecomunicações em sua execução, independente do impacto social que a ação causaria.

Este artigo utilizou o conceito de LSDC da plataforma NECOS, para fornecer solução de computação em nuvem sustentável, de baixo custo e com pouco consumo de energia, para desenvolver um Centro de Dados Itinerante *(Itinerant Data Center – IDC)*,

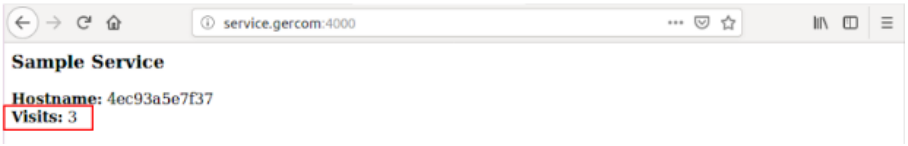
(a) Requisição para retomada do contêiner de serviço para nuvem




(b) Contêiner retornado ao nucleo




(c) Contêiner atualizado no nucleo e removido do IDC




(d) Serviço continua em execução

**Figura 9. IDC retorna e se conecta a rede novamente**

com o objetivo de melhorar a realidade de comunidades excluídas digitalmente, levando serviços essenciais para regiões sem ou com pouca infraestrutura.

Além disso, o desafio de conseguir implantar um ambiente que dê suporte para a criação de serviços de redes "in loco", ou seja, levar o serviço até o cliente e não apenas conectá-lo, e que seja de fácil implementação e ainda adaptado às condições climáticas e ambientais da região Amazônica, com a utilização de tecnologias inovadoras, como virtualização e o fatiamento de recursos para ambiente de nuvens, torna a proposta estimulante não apenas pelo ponto de vista de validar a plataforma NECOS, mas também pelo importante papel de inclusão social.

## Agradecimentos

Esta trabalho é financiado com recursos da 4º chamada colaborativa BR-EU no contexto do H2020, registrados no acordo 777067 (NECOS - Novel Enablers for Cloud Slicing), que é fomentado pelo Ministério da Ciência e Tecnologia no lado Brasileiro e pela Comissão Europeia de Tecnologia no lado Europeu.

## Referências

Castilho, G. U. and Kamienski, C. A. (2018). Aplicação de computação em névoa na internet das coisas para cidades inteligentes: da teoria à prática. In *Anais do XVI Workshop em Clouds e Aplicações (WCGA - SBRC 2018)*, volume 16, Porto Alegre, RS, Brasil. SBC.

Clayman, S., Tusa, F., and Galis, A. (2018). Extending Slices into Data Centers: the VIM on-demand model. In *2018 9th International Conference on the Network of the Future (NOF)*, pages 31–38.

Conectada, P. A. (2015). Disponível em http://www.amazoniaconectada.eb.mil.br. *Acessado: 29 de Março de 2019*.

Coutinho, M. M. (2009). Uso de redes codpon em aplicações de governo eletrônico. In *WCGE–I Workshop de Computação Aplicada em Governo Eletrônico*, pages 1451–1464.

de Castro, L. B., da Silva Ferreira, S. K., and Boas, B. V. (2018). Projeto celcom: Uma solução de baixo custo para a inclusão digital e social em comunidades isoladas no brasil. In *SENID–5º Seminário Nacional de Inclusão Digital*.

Harnal, S. and Kaur, J. (2016). Daknet (the village area network). *International Journal of Advanced Research in Computer Science and Software Engineering*, 06(10):302–306.

Kniess, J. and Ruck, D. (2016). Reduzindo a latência de aplicações web com fog computing. In *Conferência: ERAD-SP - VII Escola Regional de Alto Desempenho - SP*.

Marques, J. A. and Obelheiro, R. R. (2017). Um modelo de estimação para provisionamento de rede de máquinas virtuais em nuvens iaas. In *Anais do XV Workshop em Clouds e Aplicações (WCGA - SBRC 2017)*, volume 15, Porto Alegre, RS, Brasil. SBC.

McDonald, P., Geraghty, D., Humphreys, I., Farrell, S., and Cahill, V. (2007). Sensor network with delay tolerance (sendt). In *2007 16th International Conference on Computer Communications and Networks*, pages 1333–1338.

Silva, F. S. D., Lemos, M. O. O., Medeiros, A., Neto, A. V., Pasquini, R., Moura, D., Rothenberg, C., Mamatas, L., Correa, S. L., Cardoso, K. V., Marcondes, C., ABelem, A., Nascimento, M., Galis, A., Contreras, L., Serrat, J., and Papadimitriou, P. (2018). Necos project: Towards lightweight slicing of cloud federated infrastructures. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 406–414.

Vasconcelos, Y. (2017). Balões para acessar a internet. disponível em http://revistapesquisa.fapesp.br/2017/07/18/baloes-para-acessar-a-internet/. *Acessado: 21 de março de 2019*.

# I Workshop de Teoria, Tecnologias e Aplicações de Slicing para Infraestruturas Softwarizadas

# SBRC 2019
# Sessão Técnica 2

# Arcabouço para Orquestração Osmótica de *Cloud Slices*

**Aryadne G. P. Rezende[1], Gustavo S. Marques[1], Raquel F. Q. Lafetá[1], Rafael Pasquini[1]**

[1]Programa de Pós-Graduação em Computação UFU
Universidade Federal de Uberlândia (UFU) – Minas Gerais – MG – Brasil

{aryadne.guardieiro, guto_silveira_}@ufu.br

{raquel.lafeta, rafael.pasquini}@ufu.br

***Abstract.*** *This research builds upon the Slice as a Service paradigm proposed in the Novel Enablers for Cloud Slicing (NECOS) project. Assuming the provision of end-to-end slices which are composed by resources coming from multiple infrastructure providers, this work explores an osmotic orchestration approach in a cloud slice context, aiming to preserve SLAs during run time. The objective is to develop a mechanism to orchestrate the rearrangement of spare resources placed in a lightly loaded slice part towards a heavily loaded one. This paper brings an innitial experiment to validate the operation of the built testbed.*

***Resumo.*** *Essa pesquisa baseia-se no paradigma Slice as a Service proposto no projeto Novel Enablers for Cloud Slicing (NECOS). Assumindo o fornecimento de slices fim-a-fim, as quais são compostas por recursos vindos de multiplos provedores de infraestrutura, esse trabalho explora uma abordagem de orquestração osmótica, em um contexto de cloud slice, visando preservar o SLA durante o tempo de execução. O objetivo é desenvolver um mecanismo para orquestrar o rearranjo de recursos excedentes contidos em uma parte da slice pouco utilizada para uma parte que esteja mais sobrecarregada. Este artigo traz um experimento inicial para validar a operação do ambiente de testes construído.*

## 1. Introdução

A Computação em Nuvem (*Cloud Computing*) é o paradigma que emergiu da oferta de recursos computacionais na forma de um modelo *pay-as-you-go*, no qual o cálculo do pagamento pelo serviço é feito conforme seu uso em um determinado período. Um dos diferenciais da Computação em Nuvem é a elasticidade, a qual permite aos usuários adquirir ou liberar a quantidade necessária de recursos computacionais, de acordo com sua necessidade [Netto et al. 2014].

O modelo de negócio da Computação em Nuvem oferece serviços em três níveis: infraestrutura (*IaaS - Infrastructure as a Service*), plataforma (*PaaS - Platform as a Service*) e software (*SaaS - Software as a Service*), sendo que existem diversos provedores disponibilizando seus recursos em cada camada. Ortogonal a todas essas camadas está o conceito de *Slice* ou *Cloud Slice*. Uma *slice*, ou fatia, pode ser vista como um conjunto de recursos físicos ou virtuais (rede, processamento e armazenamento) que podem acomodar componentes de um serviço, de maneira integrada e independente de outras *slices* [Freitas et al. 2018]. O enorme número de combinações possíveis e muitas vezes a complexidade de integração desses recursos, tornam difícil o trabalho de organizações que

queiram oferecer seus serviços utilizando diferentes provedores de computação, armazenamento e conectividade. Toda essa complexidade torna o lançamento de novos serviços uma tarefa demorada e cara [Silva et al. 2018].

Com a finalidade diminuir essa complexidade, surgiu o projeto NECOS (*Novel Enablers for Cloud Slicing*), no qual se insere este trabalho [NECOS 2018b]. O objetivo geral do projeto é investigar um novo modelo de negócio chamado de *Slice as a Service*. Neste contexto, será testada a criação de *cloud slices* sob demanda, em um ambiente multi-domínio. Além da criação, será investigada a reconfiguração automática de recursos da *slice*, a fim de garantir a qualidade de serviço dos clientes do NECOS para com seus usuários finais.

Primeiro, o cliente fornecerá uma descrição da *slice* que irá precisar para o sistema NECOS. Depois de alocado esse conjunto de recursos, vindos de um ou mais provedores, o cliente poderá prover seus serviços ao seus usuários finais. Devido a natureza dinâmica da maioria das aplicações hospedadas na nuvem, a demanda do cliente poderá sofrer variações para mais ou para menos. Por essa razão, serão feitas constantes verificações do *Service Level Agreement* (SLA) que está sendo provido aos usuários finais do cliente (mais detalhes dessa estimativa em [Pasquini and Stadler 2017]). Buscando evitar quebras nesse acordo de serviço, o sistema NECOS ajustará as configurações da *slice*, objetivando manter a qualidade do serviço oferecido. Para manter o SLA contratado e os custos do cliente proporcionais a demanda de maneira dinâmica, é necessário um orquestrador inteligente que gerencie a alocação e desalocação de recursos.

Nesse cenário, o foco deste trabalho é o redimensionamento inteligente e automático de *slices*. A abordagem escolhida para a tratativa desse problema é a Computação Osmótica [Villari et al. 2016]. Como na biologia, em que a osmose se refere a passagem de água de um meio a outro para igualar concentrações de solutos, recursos também podem fluir para onde a demanda se concentra, evitando desperdícios e provendo melhor qualidade ao cliente final. Assim, o objetivo é avaliar o uso da Computação Osmótica no contexto da orquestração de *cloud slices*. Adiante, é proposto um arcabouço para orquestração osmótica de *cloud slices*, que adapta os recursos das partes da *slice* de maneira osmótica, com a finalidade de manter o SLA acordado aproximando o custo operacional ao mais baixo possível. Neste artigo, apresentamos também resultados de um experimento inicial, no qual o ambiente de testes desenvolvido coleta métricas de uma aplicação de vídeo sob demanda sendo ofertada em um *slice*. A partir destas métricas o orquestrador irá estimar a situação do SLA, ou seja, o primeiro passo na construção deste trabalho.

Na próxima Seção, são apresentados trabalhos que tratam da orquestração de recursos, no contexto de nuvem, IoT e serviços, intrinsecamente relacionados com este trabalho. Após, a Seção 3 expõe um arcabouço para orquestração de *cloud slices* de maneira osmótica, no contexto do projeto NECOS. Na sequência, a Seção 4 mostra a plataforma de experimentação elaborada para a investigação desse orquestrador. Em seguida, a Seção 5 mostra os resultados iniciais nessa plataforma. Por fim, a Seção 6 reúne as considerações finais juntamente com os próximos passos desta pesquisa.

## 2. Trabalhos Relacionados e Fundamentação Teórica

O problema da orquestração de recursos pode ser observado de diferentes ângulos, dependendo dos recursos que se planeja gerenciar. Os trabalhos citados em [Qu et al. 2018], por exemplo, lidam com a orquestração de máquinas virtuais. Em [Casalicchio and Perciballi 2017] e [Casalicchio 2019] analisa-se a orquestração a nível de contêineres. Já em [Carnevale et al. 2018] é descrito um orquestrador osmótico que gerencia dispositivos no contexto IoT (*Internet of Things*). Por outro lado, o trabalho [Sciancalepore et al. 2017] ilustra um orquestrador de *slices* em um contexto 5G.

Além de abordagens de orquestração, também estão relacionados a este artigo os temas: Computação Osmótica [Villari et al. 2016] e estimação de QoS (*Quality of Service*) a partir de métricas do provedor de serviço [Pasquini and Stadler 2017]. Adiante, nos aprofundaremos em conceitos importantes extraídos dessas publicações e em como elas se relacionam a este trabalho.

Em [Qu et al. 2018] é introduzido um tema pertinente na orquestração em nuvem: o *auto-scaling* (dimensionamento automático) de recursos. Dada a possibilidade de redimensionar facilmente os recursos virtuais na nuvem, originou-se o desafio de prover apenas a quantidade necessária de recursos para determinada aplicação em um dado momento. Na *survey*, é feita uma compilação das mais recentes pesquisas sobre o tema em ambientes de nuvem, focando no dimensionamento automático de *Virtual Machines* (VMs). Os trabalhos são classificados quanto a como são abordados os desafios do ciclo MAPE (*Monitoring, Analysis, Planning, and Execution*) [Kephart and Chess 2003]. As etapas desse ciclo, ilustradas na Figura 1, podem ser descritas como:

**Monitoramento:** nesta etapa são colhidos indicadores de performance, que serão usados para determinar estratégias de dimensionamento. O primeiro desafio dessa etapa é selecionar quais são os indicadores de performance mais significativos, o que aumenta a precisão da estimação dos recursos necessários e diminui os custos de tráfego de informação na rede. Em segundo lugar, é importante definir um intervalo de monitoria, o que tem impacto no fluxo de informação na rede e também influencia no quão sensível às mudanças será o orquestrador.

**Análise:** nesta fase, o orquestrador determina, por meio dos dados coletados, se é necessário executar alguma alteração nos recursos já alocados. Primeiramente, é preciso definir o momento de dimensionamento, que pode ser proativo ou reativo a mudanças. Se o método for proativo, é preciso escolher estratégias de estimação de carga. Também é preciso avaliar se o sistema suportará adaptação à mudanças, ou será composto de regras fixas. Outro ponto relevante refere-se a mitigar a oscilação na distribuição dos recursos, pois redimensionamentos sucessivos em um curto espaço de tempo ocasionam desperdícios e perdas de SLA.

**Planejamento:** neste ponto, é feita a estimativa de recursos que devem ser providos ou desprovidos na próxima ação de redimensionamento. É importante que seja selecionada a quantidade certa para reduzir custos financeiros. As dificuldades desse estágio incluem estimar quais recursos e em quais quantidades serão necessários para a situação atual ou futura dos serviços e como combiná-los. Os desafios dessa etapa a tornam um problema de otimização com um grande espaço de possibilidades, sendo um problema NP-hard a geração de um plano de provisão perfeito.

**Execução:** no fim do ciclo MAPE, a execução será a aplicação do plano de provisão produzido na etapa anterior, executando-se ações de redimensionamento por meio de

API's (*Application Programming Interface*) disponibilizadas pelos provedores de serviço. Quando se trata de aplicações que podem ser implantadas em diferentes *data centers*, ou caso ações em camadas distintas precisem ser executadas, é necessário lidar com diferentes API's dependendo dos provedores de serviço envolvidos, o que aumenta o grau de dificuldade de se construir um orquestrador genérico.

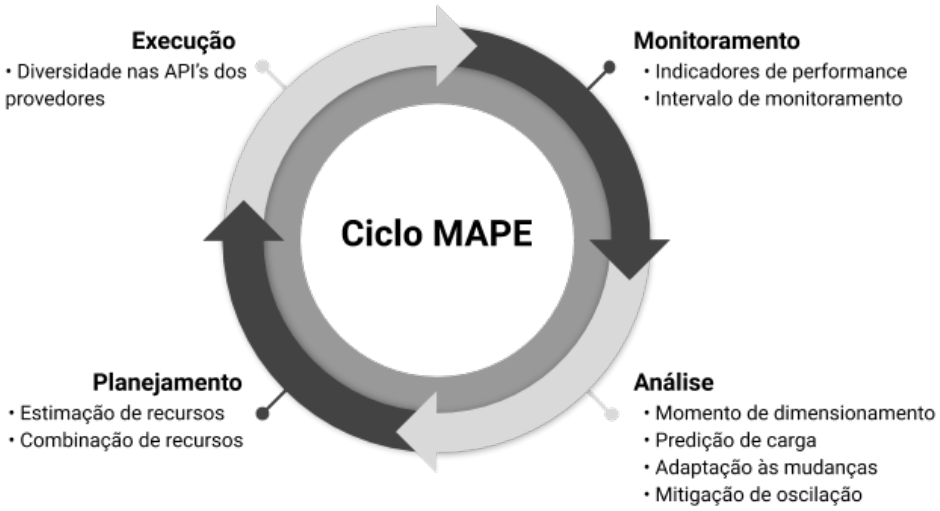



**Figura 1. Ciclo MAPE e os principais desafios de cada etapa**

Além do ciclo de orquestração, constam em [Qu et al. 2018] quatro operações diferentes de dimensionamento. No que se refere ao aumento de recursos, a operação *scaling up* aumenta os recursos internos de uma máquina virtual (número de CPU's, capacidade de memória, etc), enquanto a operação *scaling out* cria mais máquinas virtuais. Já quando é necessário diminuir recursos, pode ser feita a ação de *scaling down*, que reduz os recursos internos de uma VM, ou pode-se executar a operação *scaling in*, que reduz o número de VMs atribuídas a uma aplicação. As operações do tipo *scaling out* e *scaling in* são consideradas de redimensionamento horizontal e as ações restantes *scaling up* e *scaling down* como redimensionamento vertical.

Subindo o nível de abstração, ao invés de realizar o dimensionamento de máquinas virtuais, podemos orquestrar contêineres, que executam dentro dessas máquinas ou diretamente no metal. Um contêiner pode ser definido como um ambiente isolado e portável no qual é possível instalar uma aplicação, adicionar bibliotecas, binários e ainda uma configuração básica de como a aplicação deve ser executada [Casalicchio and Perciballi 2017]. Quando executamos uma imagem, que é a descrição de um contêiner, criamos uma instância do mesmo. Podemos criar ou excluir cópias dessas instâncias dependendo da necessidade dos usuários da aplicação em questão. Essas cópias são definidas como réplicas.

Instâncias de um contêiner têm seu tempo de vida administrado por um gerenciador de contêineres [Casalicchio 2019], por exemplo: Docker [Merkel 2014], Apache Mesos [Apache Software Foundation 2018] e Amazon ECS [Amazon Web Services 2018].

Já os orquestradores de contêineres permitem a seleção, implantação, monitoria e controle dinâmico de contêineres em um ambiente de *cloud* [Casalicchio 2019]. Esse tipo de orquestrador deve se preocupar com o controle de recursos em diversos nós, escalonamento de instâncias dentro do *cluster*, balanceamento de carga, checagem de saúde, tolerância a falhas e dimensionamento automático das instâncias ativas. Dessa maneira, os gerenciadores de contêineres atuam em um nível local, enquanto os orquestradores atuam a nível de *cluster*, orquestrando as instâncias entre diversos nós de uma rede. São exemplos desse tipo de orquestrador: Kubernetes [Kubernetes 2018c], Docker Swarm [Docker 2018] e Mesosphere Marathon [Mesosphere 2018]. É importante ressaltar que os orquestradores citados, até o momento da escrita deste artigo, não lidam com orquestração em diferentes domínios administrativos, um dos diferenciais deste trabalho.

Para ilustrar os conceitos apresentados, pode-se pensar em um dos casos de uso deste trabalho, no qual temos uma imagem de um contêiner contendo uma aplicação de vídeo sob demanda, que é instanciada em cada nó do nosso *cluster* pelo Docker e tem suas réplicas gerenciadas globalmente pelo Kubernetes. O Kubernetes possui um nível de organização básico de contêiner chamado *pod*, que representa um processo rodando dentro do *cluster* [Kubernetes 2018b]. Um *pod* encapsula um ou mais contêineres, que vão compartilhar recursos de armazenamento, um IP e outras configurações definidas na sua descrição.

O Kubernetes já conta com um dimensionador automático de *pods* nativo chamado: *Horizontal Pod Autoscaler* [Kubernetes 2018a]. Contudo, problemas em relação às métricas utilizadas por tal dimensionador foram apontados em [Casalicchio and Perciballi 2017]. [Casalicchio and Perciballi 2017] avalia quais são os melhores tipos de métricas para predição da quantidade necessária de *pods* de acordo com a demanda corrente. São definidos dois tipos de métricas: i) métricas relativas, que medem a porção de recursos que cada contêiner utiliza e ii) métricas absolutas, que representam a utilização real de recursos do sistema físico ou VM. Eles propõem um algoritmo que utiliza apenas métricas absolutas. É demonstrado que, para os *benchmarks* testados, as métricas absolutas são melhores na estimativa do número de réplicas necessárias do que as métricas relativas, utilizadas nativamente pelo Kubernetes. Porém, eles lidam apenas com casos de *scale out*, ou seja, aumentando o número de *pods*, não abordando mecanismos para executar *scale in*, que será abordado no decorrer deste trabalho.

Tratando-se da orquestração de *slices*, o trabalho [Sciancalepore et al. 2017] particiona os recursos em duas frentes de gestão: rede e dispositivos IoT. Nesse sentido, cria funções de maximização de uso da rede e dos dispositivos IoT, baseando-se em prioridade. Por se basear em regras estáticas, a abordagem tende a não ser adaptável às variações de demanda, o que não seria aplicável no contexto deste trabalho.

Devido a natureza multidomínio dos recursos que compõem as *slices* no projeto NECOS, nossa proposta é aplicar conceitos osmóticos com a finalidade de manter o equilíbrio entre os recursos das partes da *slice*, migrando recursos excedentes de uma parte pouco utilizada para uma parte que esteja mais sobrecarregada. É importante ressaltar que estamos tratando a migração de recursos como sendo uma deleção do recurso $X$ de uma parte $A$ da *slice* e a criação desse mesmo recurso $X$ em uma parte $B$. Isso se dá porque se trata de recursos físicos ou virtualizados, que dificilmente poderiam ser migrados literalmente pela *slice*. Pesquisas que já exploraram a Computação Osmótica

são mostradas adiante na Seção 2.1.

### 2.1. Computação Osmótica

Um novo paradigma que emerge da relação entre as novas estruturas de computação: *IoT*, *Edge Computing* (pequenos *data centers* na borda da rede) e *Cloud Computing* é a computação osmótica, ou do inglês *Osmotic Computing* [Villari et al. 2016]. O objetivo desse modelo de computação é permitir a implantação automática de micro serviços em ambientes altamente distribuídos e federados interconectados entre estruturas de *edge* e *cloud*. Pegando o termo emprestado da biologia, a computação osmótica visa migrar os micro serviços, geralmente na forma de contêineres, entre nuvem e borda de maneira fluída e automática. Essa migração de serviços entre partes da estrutura de um mesmo provedor, ou provedores diferentes, é o que inspira o uso desse paradigma na orquestração de recursos do NECOS, já que o projeto abrange a gerência da *cloud* até a *edge*.

O trabalho [Carnevale et al. 2018] propõe a arquitetura, ainda não implementada, de um orquestrador capaz de implantar micro serviços em um ambiente osmótico, no qual as aplicações podem migrar entre as camadas: *cloud*, *edge* e *IoT*. O trabalho foca na abstração de serviços, que são transformados em "MicroELementoS" (MELS) conteinerizados e na proposta abstrata de um orquestrador. A orquestração é tratada como um problema de otimização multiobjetivo, levando em conta o consumo de energia, o custo e também métricas de disponibilidade. Em tal proposta, os dispositivos IoT e os MELS são cadastrados em um *dashboard* para depois serem orquestrados pelo *Smart Orchestrator*.

Na proposta anterior, o *Smart Orchestrator* possuiria os módulos: gerenciamento de contêineres, gerenciamento de fluxo de dados, módulo de treinamento e módulo de predição. O módulo de fluxo de dados, pré-processaria as métricas enviadas pelos dispositivos, a fim de facilitar as tarefas de treinamento e predição. Já os módulos de treinamento e de predição seriam os responsáveis pelo aprendizado do orquestrador osmótico. O módulo de treinamento utilizaria *Deep Learning* [LeCun et al. 2015] para aprender padrões à partir das métricas enviadas pelos Agentes Osmóticos, a fim de gerar manifestos dinâmicos para a implantação mais assertiva dos MELS. Os modelos seriam salvos em um banco de dados. Já o módulo de predição utilizaria os modelos anteriores de uma implantação para a geração de novos manifestos que poderiam ser mais úteis na implantação dos MELS em diferentes níveis (IoT, *edge*, *fog* e/ou *cloud*).

No nosso trabalho, utilizaremos uma arquitetura similar, porém, tendo um módulo de predição de QoS, para saber a qualidade do serviço que está chegando nos usuários finais, assunto tratado em [Pasquini and Stadler 2017]. Na nossa abordagem, além das métricas da infraestrutura, o QoS recebido pelos usuários finais também será usado no treinamento do modelo de aprendizagem, o que deixará nosso orquestrador ciente da qualidade que chega aos mesmos. Vamos investigar quais algoritmos utilizar no módulo de treinamento para encontrar as causas raiz da violação de QoS e também para detectar o super provisionamento de recursos, assuntos não tratados em [Carnevale et al. 2018].

Em [Sharma et al. 2017] fica claro que o uso da *Fog Computing* pode acarretar em altos custos e redundância de recursos. Essa situação se torna ainda mais complexa quando se trata da administração de serviços heterogêneos, os quais são mais eficientemente executados em dispositivos específicos. Para contornar esse problema, [Sharma et al. 2017] propõe o uso da computação osmótica como forma de balancear os

serviços onde seriam mais eficientemente computados. Para tal, o artigo define uma taxonomia que liga os termos da computação osmótica ao problema a ser resolvido:

**Soluto:** parte solúvel a qual não é permitida atravessar a membrana semipermeável. No contexto do artigo [Sharma et al. 2017]: poder computacional, energia, tempo de processamento e carga atual.

**Solvente:** componente que absorve o soluto, podendo se mover pela membrana semipermeável. Neste caso, solvente são serviços que se deslocam entre os servidores.

**Solução:** composta pela infraestrutura, usuários, servidores, serviços e recursos disponíveis.

**Membrana semipermeável:** é o elemento responsável por manter a concentração apropriada da solução. No artigo, a camada de servidores na *fog* é responsável por esse papel. Para manter esse equilíbrio é preciso considerar: quando o serviço deve ser movido, para onde deve ser movido (um *data center* ou um dispositivo na *edge*?) e, por fim, como mover esse serviço (usando contêineres ou uma agregação de contêineres?).

**Concentração:** razão de soluto ou solvente, em computação osmótica pode ser interpretada como a razão do número de serviços a serem oferecidos pelos recursos computacionais disponíveis.

[Sharma et al. 2017] divide os serviços heterogêneos nas categorias micro e macro, com base nos requisitos de recursos computacionais, energéticos e de processamento. Além da taxonomia, o artigo propõe um algoritmo iterativo que visa balancear esses serviços entre as camadas *fog* e *cloud*, até que haja uma distribuição com um certo grau de tolerância $\epsilon$ de diferença no balanceamento. Enquanto tal estabilidade não é alcançada, o algoritmo tenta mover os serviços que couberem para a camada osmótica (formando um grupo de micro serviços), e os demais serviços que não couberem nessa camada (grupo de macro serviços) são migrados para a nuvem.

Olhando o contexto da computação osmótica como um meio distribuído de se montar modelos ainda mais inteligentes, [Morshed et al. 2017] exibe os desafios de se aplicar *Deep Learning* para análise de dados clínicos. Sendo o *Deep Learning* uma técnica de aprendizado de máquina que constrói conhecimento a partir de uma hierarquia de conceitos, dos mais concretos aos mais abstratos, ele promove grande flexibilidade e capacidade de aprendizado. No entanto, tal técnica precisa de um grande volume de dados, o que torna seu uso custoso, em termos de transporte e processamento, num contexto de computação distribuída.

Outro desafio do uso desse tipo de aprendizado de máquina quando aliado a computação osmótica, é a heterogeneidade dos dados vindos de diferentes fontes, que em diferentes formatos acrescenta mais complexidade no treinamento de modelos de aprendizagem. Mais um obstáculo seria garantir a privacidade e segurança dos dados, contudo, uma abordagem interessante proposta para evitar esse problema seria compartilhar apenas o resultado do modelo ao invés dos dados "crus" pela rede. Por fim, o trabalho apresenta uma proposta hierárquica de modelagem, na qual os dispositivos na *edge* seriam responsáveis por criar partes de um modelo que é unificado nas camadas superiores da aplicação, sendo que o modelo holístico central ficaria armazenado na nuvem. Tal abordagem de aprendizado pode ser considerada em futuras implementações deste trabalho, considerando as partes da *slice* como a *edge* que computa partes do modelo.

## 3. Arcabouço para Orquestração Osmótica de *Cloud Slices*

Esta seção utiliza os conceitos apresentados até então para elaborar uma proposta de orquestrador osmótico de *cloud slices* no contexto do projeto NECOS. Apesar do projeto contar com um módulo de orquestração a nível de serviço, o foco deste trabalho se limita na orquestração de recursos da *slice* no modo 0 (infraestrutura). Por esse motivo, neste trabalho, teremos a seguinte analogia osmótica:

**Soluto:** conjunto de partes de uma *slice*, distribuídas em um ou mais provedores de infraestrutura.

**Solvente:** conjunto de recursos que compõem uma parte de uma *slice*, que podem ser migrados a fim de balancear a saturação de uma parte da *slice*.

**Solução:** é composta por todos os elementos de infraestrutura e partes da *slice*.

**Membrana semipermeável:** representada pelo módulo *Slice Provider* do NECOS, descrito a seguir, será responsável por decidir quais recursos podem ser migrados entre partes da *slice*.

**Concentração:** razão de carga compartilhada entre todas as partes da *slice*.

A Figura 2 mostra a abstração da orquestração osmótica na conjuntura do projeto. Na base da arquitetura estão os provedores de infraestrutura. Eles podem oferecer recursos de *Data Center* (DC), conexão de *Data Centers* via *Wide Area Network* (WAN), e também outro tipos de recursos (*edge, fog*, etc.) aos *tenants*. Dentro de cada provedor, está um módulo do NECOS chamado *DC Controller*, ou *WAN Controller*, responsável por atuar na infraestrutura do provedor em nome do NECOS.

Um nível acima dos provedores de infraestrutura está o *Slice Provider*, o centro de operações do NECOS, responsável por administrar todas as *slices* solicitadas pelos *tenants*. Dentro dele, existe um módulo de monitoria (*Monitoring Module*), que é encarregado de receber dados de uso e saúde da infraestrutura alocada para as *slices*. Além de receber, ele também pré-processa esses dados, e os redireciona ao *Slice Resource Orchestrator*, que, por sua vez, tomará as decisões de elasticidade necessárias para manter o SLA contratado pelo *tenant*. Mais detalhes sobre a arquitetura do projeto podem ser encontrados em [NECOS 2018a].

No topo da arquitetura, se encontra o domínio dos clientes (*Tenants' Domain*), onde são utilizadas as *slices*. Um cliente nesse caso, são os usuários de *slice* de rede, nuvem ou *data center* no qual serviços customizados são hospedados. As *slices* são compostas por partes. Cada parte é um conjunto de recursos obtidos de apenas um provedor. No projeto NECOS, são definidos dois tipos de elasticidade: horizontal e vertical. Na elasticidade horizontal, são adicionadas ou removidas partes da *slice*, enquanto na vertical, existe o aumento ou a redução de recursos dentro de uma parte da *slice*. O objetivo de se utilizar um orquestrador osmótico é evitar a elasticidade horizontal, usando ao máximo os recursos já alocados, evitando o desperdício de recursos e a complexidade de se contratar mais provedores, possibilitando o aumento de barganha de preços dentro do fornecedor de infraestrutura daquela parte da *slice*.

Dentro do *Slice Resource Orchestrator* haverá um módulo chamado *Osmotic Slice Controller*, responsável por fazer a elasticidade automática das *slices*. Existirá uma instância desse módulo para cada *slice* administrada pelo *Slice Provider*. Dentro do *Osmotic Slice Controller* haverá um laço MAPE de controle customizado, que será implementado, seguindo o fluxo mostrado na Figura 2:

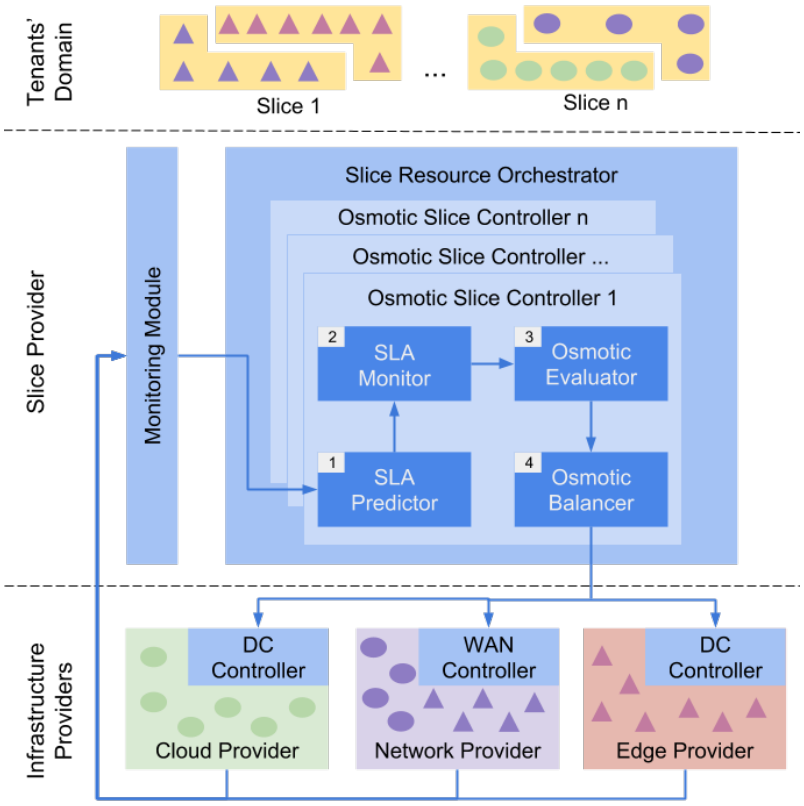



**Figura 2. Orquestração osmótica no contexto NECOS. Provedores de infraestrutura (base) fornecem os recursos a serem usados pelos *tenants* (topo), sendo que a gerência é de responsabilidade do *Slice Provider* (meio).**

1. O módulo *SLA Predictor* estimará o SLA do usuário final utilizando os dados pré-processados pelo *Monitoring Module*, ou seja, apenas as métricas mais relevantes na estimação do SLA serão levadas em consideração. A predição desse SLA será feita usando algoritmos de aprendizado de máquina do tipo regressão.
2. O fluxo seguirá para o *SLA Monitor*, que será responsável por analisar os SLAs estimados com a finalidade de detectar uma situação ou tendência de degradação da qualidade do serviço, usando algoritmos de reconhecimento de padrão, por exemplo.
3. O *Osmotic Evaluator* analisará a configuração atual da *slice*, para rebalancear os recursos entre as partes da mesma, migrando recursos excedentes de uma parte pouco utilizada para partes que necessitem mais daquele recurso. Nesse ponto, poderão ser usados algoritmos genéticos, para gerar arquivos de implantação que melhor se ajustem à fase corrente do serviço.
4. Por último, o *Osmotic Balancer* executará o plano criado pelo *Osmotic Evaluator*, disparando as ações sugeridas no passo 3 para os receptores do NECOS que residirão dentro dos provedores (*DC/WAN Controller*), usando API's REST.

O módulo *Osmotic Slice Controller* ainda está em fase de concepção. Atualmente, estão sendo testadas algumas hipóteses que validarão a implementação de seus submódulos, como por exemplo:

I - É possível estimar o SLA que o usuário final do serviço está recebendo num

contexto de *cloud slicing*?

II - É possível detectar uma tendência de degradação desse SLA de forma a agir preventivamente na elasticidade dos recursos da *slice*?

III - É possível gerar um arquivo descritivo de partes da *slice* de forma inteligente utilizando-se arquivos descritivos criados anteriormente, juntamente com seus históricos de qualidade de serviço?

Para testar essas hipóteses e o próprio orquestrador depois de implementado, foi implantada a plataforma de experimentação descrita na Seção 4.

## 4. Plataforma de Experimentação

Para estudo de caso, foi escolhida uma aplicação de vídeo sob demanda que utiliza o recente padrão de *streaming* de vídeos MPEG-DASH [Sodagar 2011]. De acordo com o relatório [Sandvine 2018], quase 58% do tráfego *downstream* de toda internet no ano de 2018 representava *streaming* de vídeos, tornando a avaliação desse tipo de aplicação de extrema importância. O MPEG-DASH é um padrão de *streamming* de vídeo adaptativo, no qual o dispositivo do usuário irá decidir qual segmento, entre os diversos segmentos de vídeo codificado com diferentes qualidades, melhor se adapta a sua largura de banda no momento, visando entregar o vídeo de modo ininterrupto ao usuário.

A plataforma de testes construída para esses experimentos é mostrada na Figura 3. Nela, é possível ver três componentes, seguindo os mesmos padrões de cores da Figura 2: (i) em amarelo, os componentes da *slice*, representando um provedor do tipo *edge*; em azul, representando o módulo de monitoria do NECOS o componente Prometheus; e (iii) um novo componente, até então não apresentado, os usuários da *slice*, que irão consumir o serviço de vídeo disponibilizado na *slice* de um *tenant*. O fluxo dos experimentos ocorre de acordo com a numeração da Figura 3, seguindo os passos:

1. O componente *Kubernetes Master* faz o *deploy* de réplicas do serviço de vídeo dentro do *cluster* de *raspberry pies* e também implanta uma instância do Balanceador de carga dentro de uma máquina virtual (VM), disponibilizada pelo OpenStack [Murphy 2018]. Os contêineres são administrados dentro de cada nó (*raspberry pies* e VM) pelo gerenciador de contêineres Docker [Merkel 2014].
2. A partir do momento que o serviço está em execução, métricas do monitoramento do serviço (contêineres) e das máquinas (virtuais e físicas) são coletadas e mantidas pelo Prometheus [Prometheus 2019]. Esse coletor de métricas executa em uma VM, também disponibilizada pelo OpenStack.
3. Com o serviço pronto para ser consumido, o Gerador de carga é responsável por gerar várias aplicações VLC [VLC 2019], com a finalidade de perturbar a qualidade do serviço oferecido, simulando uma demanda real. Já o cliente modificado é uma versão customizada do VLC, alterada para coletar métricas da qualidade do serviço que está sendo recebido pelo usuário final.

A plataforma foi elaborada de tal maneira que métricas tanto do lado do serviço como do lado dos usuários finais sejam coletadas com a finalidade de se estimar o SLA recebido (Hipótese I), usando algoritmos de aprendizado de máquina supervisionados. A arquitetura também facilita a elasticidade, uma vez que se recursos computacionais precisarem ser adicionados, removidos ou terem suas configurações alteradas, isso pode ser feito virtualmente por meio de chamadas às API's de controle do OpenStack ou do Kubernetes.

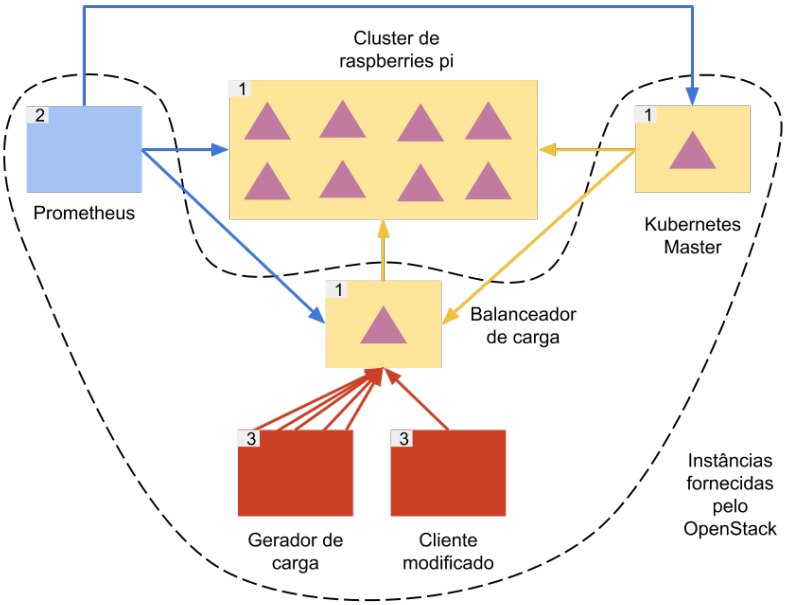



**Figura 3. Plataforma de experimentação preliminar**

## 5. Resultados Preliminares

No momento da escrita deste artigo, a plataforma de experimentos encontra-se em fase de calibragem de parâmetros. As variáveis: (i) número de clientes gerados pelo gerador de carga, (ii) número de réplicas do serviço e (iii) largura de banda entre clientes e serviço, precisam estar em uma área na qual seus valores provoquem oscilações realistas na plataforma de experimentos, levando em considerações as limitações da plataforma.

Por esse motivo, experimentos preliminares estão sendo feitos para avaliar, por exemplo, qual a carga máxima de clientes pode ser produzida pelo gerador de carga, qual a largura de banda mínima para os clientes receberem o serviço do servidor, porém, permitindo perturbações na qualidade de serviço. A Figura 4 ilustra um desses experimentos, com as seguintes configurações:

- Limitação de 35 MBits/s na máquina virtual que disponibiliza o serviço de vídeo;
- Carga de clientes com distribuição sinusoidal variando de 1 a 10 clientes, num período de 3600 segundos;
- Lógica de chaveamento de qualidade do tipo *rate* do visualizador de vídeo VLC, o qual utiliza uma estimação da largura de banda da máquina cliente solicitar os próximos trechos de vídeo.

É possível identificar na área entre 500 e 1500 segundos a degradação do vídeo (em azul) com a carga de clientes em alta (em vermelho), enquanto entre 2000 e 3000 segundos, o serviço apresenta estabilidade no recebimento de quadros por segundo, no momento em que a geração de carga está em baixa.

Neste trabalho, o primeiro passo para disparar a orquestração osmótica é a detecção de violações de SLA (Hipótese I). O protótipo construído é capaz de conduzir a carga suportada pelo *slice*, afetando, conforme resultado da Figura 4 o desempenho

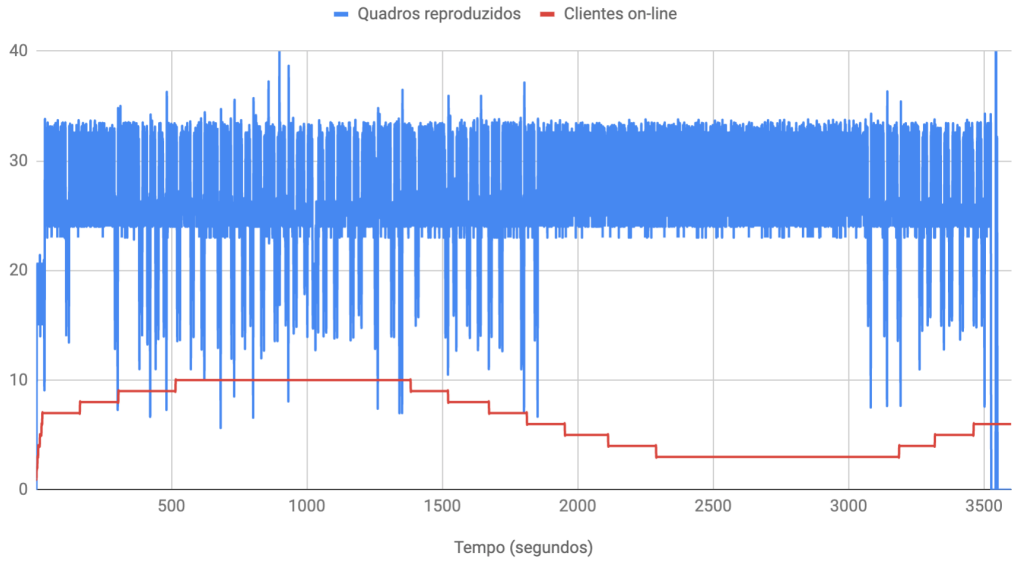



**Figura 4. Experimento de uma hora com geração de carga no serviço de *stream* de vídeo.**

do serviço em virtude de carga extra sob a infraestrutura. Esta funcionalidade torna automático o disparo da proposta de orquestração osmótica.

## 6. Conclusão e Trabalhos Futuros

Os estudos teóricos feitos até o momento apontam a computação osmótica como um paradigma promissor na gerência de recursos distribuídos nas três camadas de atuação de aplicações: *cloud, fog* e *edge*, que são o contexto de atuação do projeto NECOS. Além de promissor, o uso da computação osmótica apresenta características favoráveis ao combate ao desperdício dos recursos alocados por diminuir as chances de elasticidade horizontal, o que será investigado adiante nessa pesquisa.

Espera-se como desfecho deste trabalho, a validação de um protótipo de orquestrador de recursos de *slice*, que utilize principalmente o paradigma osmótico para realizar essa tarefa. Dessa maneira, as duas contribuições deste trabalho serão: i) propor um orquestrador inteligente de *slices* sob demanda e ii) expandir os conhecimentos atuais do uso do paradigma osmótico no contexto de gerência de *slices*.

É de nosso conhecimento que o uso da abordagem osmótica acarretará mudanças na camada de serviço, sendo necessária uma sincronização entre as ações do *Slice Resource Orchestrator* e o *Service Orchestrator*, incluindo a migração, criação e deleção de réplicas do serviço nas partes alteradas das *slices*. Essa situação será futuramente analisada, durante a implementação do orquestrador osmótico, avaliando-se o melhor tratamento de cada caso.

Outro trabalho futuro seria analisar casos em que o orquestrador osmótico se mostra mais eficiente do que outras tratativas de redimensionamento de recursos (baseado em regras, regressão, modelos analíticos, etc.). Podendo-se criar então um orquestrador híbrido, que adapta sua estratégia de acordo com exigências do cliente ou conforme a tendência de consumo de recursos apresentada pela *slice* gerenciada.

## Agradecimentos

Esta trabalho é financiado com recursos da 4º chamada colaborativa BR-EU no contexto do H2020, registrados no acordo 777067 (*NECOS - Novel Enablers for Cloud Slicing*), que é fomentado pelo Ministério da Ciência e Tecnologia no lado Brasileiro e pela Comissão Europeia de Tecnologia no lado Europeu.

## Referências

Amazon Web Services (2018). Amazon elastic container service. `https://aws.amazon.com/pt/ecs/`. Online; acessado em 12-12-2018.

Apache Software Foundation (2018). Apache mesos. `http://mesos.apache.org/`. Online; acessado em 12-12-2018.

Carnevale, L., Celesti, A., Galletta, A., Dustdar, S., and Villari, M. (2018). From the cloud to edge and iot: a smart orchestration architecture for enabling osmotic computing. In *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 419–424. IEEE.

Casalicchio, E. (2019). Container orchestration: A survey. In *Systems Modeling: Methodologies and Tools*, pages 221–235. Springer.

Casalicchio, E. and Perciballi, V. (2017). Auto-scaling of containers: The impact of relative and absolute metrics. In *Foundations and Applications of Self* Systems (FAS* W), 2017 IEEE 2nd International Workshops on*, pages 207–214. IEEE.

Docker (2018). Swarm mode overview. `https://docs.docker.com/engine/swarm/`. Online; acessado em 12-12-2018.

Freitas, L. A., Braga, V. G., Corrêa, S. L., Mamatas, L., Rothenberg, C. E., Clayman, S., and Cardoso, K. V. (2018). Slicing and allocation of transformable resources for the deployment of multiple virtualized infrastructure managers (vims). In *4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 424–432. IEEE.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kubernetes (2018a). Horizontal pod autoscaler. `https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/`. Online; acessado em 21-11-2018.

Kubernetes (2018b). Pod overview. `https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/`. Online; acessado em 21-12-2018.

Kubernetes (2018c). Production-grade container orchestration. `https://kubernetes.io/`. Online; acessado em 12-12-2018.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2.

Mesosphere (2018). Marathon: A container orchestration platform for mesos and dc/os. `https://mesosphere.github.io/marathon/`. Online; acessado em 12-12-2018.

Morshed, A., Jayaraman, P. P., Sellis, T., Georgakopoulos, D., Villari, M., and Ranjan, R. (2017). Deep osmosis: Holistic distributed deep learning in osmotic computing. *IEEE Cloud Computing*, 4(6):22–32.

Murphy, E. (2018). Welcome to openstack documentation. `https://docs.openstack.org`. Online; acessado em 16-03-2019.

NECOS (2018a). D3.1: Necos system architecture and platform specification. v1. `http://www.maps.upc.edu/public/NECOS\%20D3.1\%20final.pdf`. Online; acessado em 12-03-2019.

NECOS (2018b). Novel enablers for cloud slicing. `http://www.h2020-necos.eu/`. Online; acessado em 12-12-2018.

Netto, M. A., Cardonha, C., Cunha, R. L., and Assunçao, M. D. (2014). Evaluating auto-scaling strategies for cloud computing environments. In *Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on*, pages 187–196. IEEE.

Pasquini, R. and Stadler, R. (2017). Learning end-to-end application qos from openflow switch statistics. In *IEEE Network Softwarization (NetSoft)*, pages 1–9. IEEE.

Prometheus (2019). Overview prometheus. `https://prometheus.io/docs/introduction/overview/`. Online; acessado em 16-03-2019.

Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):73.

Sandvine (2018). The global internet phenomena report. `https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf`. Online; acessado em 18-03-2019.

Sciancalepore, V., Cirillo, F., and Costa-Perez, X. (2017). Slice as a service (slaas) optimal iot slice resources orchestration. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE.

Sharma, V., Srinivasan, K., Jayakody, D. N. K., Rana, O., and Kumar, R. (2017). Managing service-heterogeneity using osmotic computing. *arXiv preprint arXiv:1704.04213*.

Silva, F. S. D., Lemos, M. O., Medeiros, A., Neto, A. V., Pasquini, R., Moura, D., Rothenberg, C., Mamatas, L., Correa, S. L., Cardoso, K. V., et al. (2018). Necos project: Towards lightweight slicing of cloud federated infrastructures. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 406–414. IEEE.

Sodagar, I. (2011). The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, 18(4):62–67.

Villari, M., Fazio, M., Dustdar, S., Rana, O., and Ranjan, R. (2016). Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*, 3(6):76–83.

VLC (2019). Vlc media player. `https://www.videolan.org/vlc/`. Online; acessado em 16-03-2019.

# Uma Proposta de Arquitetura para o Monitoramento Multidomínio de Cloud Network Slices

**André L. Beltrami Rocha[1], Matheus N. Garcia[1], Vinicius F. Carvalho[1], Paulo Ditarso Maciel Jr.[1,2], Fábio Luciano Verdi[1]**

[1]Departamento de Computação (DComp)
Universidade Federal de São Carlos (UFSCar)
Sorocaba - SP, Brasil

[2]Unidade Acadêmica de Informação e Comunicação
Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB)
João Pessoa - PB, Brasil

{beltrami, verdi}@ufscar.br

{vinicius.furukawa,matheus.nadaleti}@dcomp.sor.ufscar.br

paulo.maciel@ifpb.edu.br

***Abstract.*** *The main feature of the cloud network slicing concept is the provision of a physical and virtual end-to-end infrastructure, capable of supporting a variety of vertical industries. Such a pool is instantiated over a large number of administrative and technological domains, which brings a greater challenge for the management and monitoring of allocated resources. Resource monitoring in this new entity so called slice is extremelly important to address MANO operations. Therefore, in this work we propose an architecture for the monitoring of physical and virtual resources in cloud network slices, considering multiples administrative and technological domains. Moreover, we present an information model for the monitoring of slices and a proof of concept implementation, in which we observe the slice monitoring resources metrics over time.*

***Resumo.*** *O conceito de cloud network slicing oferece como sua principal característica o provisionamento de uma infraestrutura física e virtual fim-a-fim capaz de dar suporte a uma variedade de indústrias verticais. Tal infraestrutura é instanciada ao longo de múltiplos domínios administrativos e tecnológicos, o que torna um desafio ainda maior gerenciar e monitorar os recursos alocados. O monitoramento dos recursos desta nova entidade chamada de slice é de suma importância para que as operações de gerência e orquestração sejam possíveis. Portanto, este trabalho propõe uma arquitetura para o monitoramento de recursos físicos e virtuais em cloud network slices, considerando multidomínios administrativos e tecnológicos. Além disso, este trabalho apresenta um modelo de informação preliminar para o monitoramento de slices e implementa uma prova de conceito, na qual foi possível observar algumas métricas das slices sendo monitoradas ao longo do tempo.*

## 1. Introdução

Por meio de tecnologias emergentes como Redes Definidas por Software (SDN, do inglês *Software-Defined Networking*) e Virtualização de Funções de Rede (NFV, do inglês

*Network Functions Virtualization*), além de tecnologias consolidadas de computação na nuvem, a "softwarização" de rede pode fornecer a flexibilidade e modularidade necessárias para criar várias redes (virtuais) lógicas, cada uma adaptada para um determinado caso de uso, em cima de uma infraestrutura de rede comum.

A partir destas tecnologias habilitadoras e da necessidade de prover uma infraestrutura fim-a-fim, surge o conceito de *cloud network slicing*. Este novo conceito encapsula em uma única unidade, a *slice*, categorias de elementos oriundos da computação em nuvem, redes e armazenamento. Portanto, o principal motivador deste novo conceito é prover a instanciação dos recursos de computação em nuvem, rede e armazenamento ao longo de múltiplos domínios administrativos e/ou tecnológicos de maneira integrada. Uma vez instanciados, o locatário que solicitou a criação da *slice* poderá acessar, configurar e gerenciar os recursos como preferir, de modo completamente isolado, até mesmo sublocando tais recursos para outros locatários.

Diversas entidades e projetos vêm estudando este tema para definir algumas opções de arquitetura para a criação e gerência de uma *slice*. Entretanto, operações importantes como o monitoramento e a orquestração dos recursos estão ainda incipientes na literatura e possuem muitos desafios a serem explorados. Em um ambiente onde os elementos envolvidos são tão diversos, um serviço de orquestração é extremamente necessário para coordenar os recursos alocados, realizar operações de elasticidade, entre outras. De acordo com a *Open Network Foundation* (ONF) [TR-521 2016], a orquestração é definida como um processo contínuo de seleção de recursos para atender às demandas de atendimento ao cliente de maneira ideal.

No cenário de *cloud network slicing* contextualizado acima, a orquestração dos recursos físicos ou virtuais não pode ser realizada eficientemente sem uma entidade capaz de monitorar diferentes recursos alocados em múltiplos domínios. Com o surgimento deste novo conceito *slice*, uma nova entidade fim-a-fim necessita ser monitorada, composta conceitualmente pelo monitoramento de recursos relacionados à computação, rede e armazenamento.

Este artigo busca equacionar alguns destes problemas através da proposta de uma arquitetura para o monitoramento de *cloud network slices*, considerando o desafio de monitorar múltiplos domínios administrativos e tecnológicos. Como contribuições deste trabalho, pode-se destacar a proposta de uma arquitetura e um modelo de informação para o monitoramento de *slices*. Foi implementada, uma prova de conceito que visa demonstrar de forma minimalista um cenário composto por duas *slices* sendo monitoradas ao longo do tempo. Com tais informações, diferentes ferramentas de visualização podem ser desenvolvidas de forma a facilitar a navegabilidade e operações de orquestração para esta nova entidade *slice* e seus respectivos recursos e/ou serviços.

Este trabalho está organizado como descrito a seguir. A Seção 2 apresenta alguns conceitos básicos para um melhor entendimento sobre *cloud network slicing*. A Seção 3 descreve alguns trabalhos relacionados à gerência e ao monitoramento de *slices*. As propostas de uma arquitetura multidomínio e de um modelo de informação para o monitoramento de *slices* estão presentes na Seção 4. A Seção 5 aborda questões relacionadas à implementação da proposta, como por exemplo as tecnologias habilitadoras. A Seção 6 apresenta a prova de conceito e os resultados preliminares obtidos. Por fim, a Seção 7

finaliza este artigo expondo as conclusões encontradas e perspectivas de trabalhos futuros.

## 2. Conceitos Básicos

Baseado na definição em [Clayman 2018], *cloud network slice*[1] é uma infraestrutura fim-a-fim que envolve recursos relacionados à computação em nuvem, rede e armazenamento. Esta variedade de recursos deve ser instanciada de forma transparente para o locatário, para que ele possa realizar operações de gerência e orquestração. Por ser um conceito fim-a-fim, *slices* incluem diversos componentes que pertencem a domínios administrativos diferentes e, portanto, exigem um nível de abstração mais alto para a gerência e monitoramento dos mesmos. Além disso, permite novos modelos de negócios, onde alguns atores desempenham funções comerciais distintas, dos quais valem a pena serem destacados (atores e terminologias):

- *Provedor de infraestrutura*: possui e gerencia uma determinada infraestrutura. Esses recursos, físicos ou virtualizados, são oferecidos para compor uma *slice* requisitada por um único ou vários locatários;
- *Provedor de slice*: possui ou não infraestrutura física própria, mas é responsável por oferecer "serviços" de *slices* fim-a-fim entre domínios administrativos distintos, como por exemplo criação, remoção e gerência de *slices*;
- *Slice Parte*: uma *slice* pode estar instanciada ao longo de muitos provedores de infraestrutura onde, para cada domínio administrativo a porção dos recursos que correspondem à determinada *slice* é denominada de *slice* parte;
- *Locatário*: contrata recursos de um ou mais provedores de *slices*, onde o locatário pode realizar, gerenciar e fornecer serviços para seus usuários. Um serviço é uma composição de elementos de rede, computação e/ou armazenamento, e é definido em termos de como tais elementos são acessados e conectados;
- *Domínio administrativo*: conjunto de recursos e infraestruturas gerido por uma única entidade, organização ou empresa; que mantém e governa tais recursos sob uma administração comum, podendo ou não utilizar diferentes domínios tecnológicos (tecnologias distintas de gerenciamento dos recursos);
- *Domínio entrante*: provedor de *slice* onde o locatário solicita a formação de uma *cloud network slice* (multidomínios administrativos), geralmente escolhido por critérios de localidade e/ou afinidade comercial.

A atuação dos membros acima é naturalmente recursiva, onde, por exemplo, provedores e locatários podem atuar em um padrão vertical multicamadas, em que um locatário em uma camada atua como provedor de infraestrutura ou *slice* na camada imediatamente acima. A recursão mencionada aqui implica que um locatário pode fornecer serviços de rede para um usuário final, mas também para outro locatário. Na Figura 1, observa-se as interações de cada uma das terminologias descritas, assim como um componente responsável por monitorar as *slices* instanciadas, que será descrito posteriormente.

Apesar da natureza efêmera dos recursos virtuais instanciados em serviços de nuvem, é provável que os dados gerados por tais recursos sejam importantes de serem coletados e observados. O monitoramento destes dados oferece não apenas uma medida de desempenho dos serviços, mas também uma valiosa fonte de informação do comportamento do sistema como um todo [Tusa et al. 2018].

[1]Para facilitar à leitura, abreviaremos *cloud network slice* para *slice* no decorrer de todo o texto.

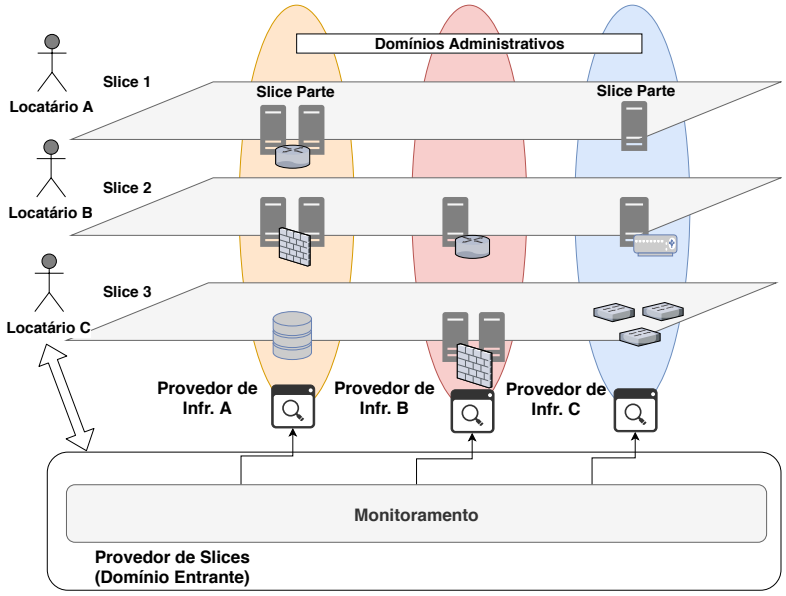



**Figura 1. Ilustração da terminologia utilizada.**

Dentre as características importantes desejadas em uma *cloud network slice* podem ser citadas: gerenciamento automático do ciclo de vida, operação autônoma, provisionamento e gerência de recursos simplificados, re-provisionamento dinâmico, alta escalabilidade e confiabilidade, eficiência econômica e rápida implantação da rede ou do serviço, possibilidade de novos modelos de negócios, dentre outras.

Há diversos esforços de padronização de *slices* por diferentes entidades, como por exemplo 3GPP [de Foy and Rahman 2017], ETSI [ETSI ISG NFV 2018] e NGMN [NGMN Alliance 2018]. Definir um modelo de informação faz parte destes esforços [Qiang et al. 2018], com o intuito de padronizar a comunicação entre os requisitos de um *slice*, que podem ser agnósticos as tecnologias, e o que realmente é implementado/provisionado nos provedores de *slices*, principalmente quando considerado o multidomínio tecnológico.

Contudo, apesar de todo o empenho em se criar padrões pelas entidades descritas acima, não há uma proposta definida especificamente para o monitoramento de *slices*, muito menos uma proposta de modelo de informação específico para o monitoramento de recursos de uma *slice*. Esta lacuna precisa ser preenchida e o objetivo do trabalho proposto neste artigo é dar um primeiro passo nesta direção.

## 3. Trabalhos Relacionados

O monitoramento de *slices* é de extrema importância para que os provedores e locatários possam gerenciar e orquestrar os recursos instanciados. Na literatura há um número considerável de trabalhos que abordam o monitoramento de infraestruturas físicas, de rede e de recursos virtuais. Entretanto, trabalhos que discorrem sobre o monitoramento de *cloud network slice* no mesmo contexto que o apresentado neste artigo não foram encontrados. Ou seja, nenhum trabalho propõe uma arquitetura capaz de monitorar tanto os recursos físicos quanto virtuais de *slices* com múltiplos domínios administrativos e tecnológicos. A seguir, são descritos alguns dos trabalhos estudados para fomentar esta proposta.

No artigo [Ordonez-Lucena et al. 2017] os autores descrevem os principais aspectos de *network slicing*, com atenção especial em sua aplicação às redes 5G. O artigo ressalta como *network slicing* oferta uma solução para estabelecer simultaneamente um amplo escopo de serviços em uma infraestrutura comum de rede 5G, cobrindo a demanda de casos de uso específicos das indústrias verticais. Os autores apresentam uma breve visão geral da arquitetura SDN proposta pela ONF, que indica ferramentas para o suporte a *slices*. De acordo com os autores, apesar da falta de recursos essenciais que podem ser fornecidos por NFV, a arquitetura fornece meios para implementações de *network slicing*.

No artigo [Fatema et al. 2014], é apresentada uma visão geral do termo monitoramento com ênfase em ambientes na nuvem e *datacenters*. O objetivo do trabalho é comparar qualitativamente algumas das ferramentas de monitoramento mais utilizadas, considerando alguns pilares do monitoramento, como por exemplo: a escalabilidade, a portabilidade, o ambiente com múltiplos locatários e a habilidade de adaptação de acordo com o locatário. Estas características e outras citadas no artigo devem ser consideradas para melhor satisfazerem as necessidades dos locatários.

O artigo [Kuklinski and Tomaszewski 2018] apresenta uma extensão da arquitetura MANO para suportar *network slicing* no contexto da gerência e orquestração de funções de rede virtualizadas. O trabalho traz uma discussão sobre possíveis atualizações da arquitetura MANO para abranger o conceito de *slice*. No artigo os autores descrevem detalhadamente os componentes propostos e suas responsabilidades. Além disso, discorrem qualitativamente os pontos fortes da proposta apresentada. Porém, não é apresentada uma análise quantitativa ou implementação de uma prova de conceito da arquitetura.

Em [Tusa et al. 2018] os autores apresentam uma proposta de monitoramento inteligente utilizando a ferramenta Lattice[2]. O escopo do monitoramento é voltado para sistemas de computação em nuvem, embora os autores mencionem que o Lattice pode ser estendido para suportar o conceito de *slice*. Em suma, eles monitoram um ambiente em computação em nuvem coletando algumas métricas e, ao detectar um aumento ou diminuição do tráfego, algumas taxas são ajustadas dinamicamente, como por exemplo o intervalo em que as métricas são coletadas e enviadas para o orquestrador. Entretanto, como o próprio trabalho destaca, os autores ainda não estenderam o trabalho para suportar o conceito de *slice*.

É importante fazer uma comparação entre os trabalhos descritos acima e o proposto neste artigo. O foco deste trabalho é apresentar uma arquitetura genérica que consiga fornecer as métricas de monitoramento referentes as *slices*, compostas por múltiplos domínios administrativos e/ou tecnológicos. A arquitetura irá coletar as métricas dos recursos individuais e agregar as métricas relacionadas a uma determinada *slice* como um todo. Além disso, será apresentado um modelo de informação preliminar para o monitoramento dos componentes que podem compor uma *cloud network slice*. O monitoramento poderá ser utilizado como uma visualização adaptada para cada locatário e/ou provedor de *slices*, bem como para realizar operações de gerência e orquestração dos recursos.

## 4. Proposta

Esta seção apresenta uma arquitetura e um modelo de informação para o monitoramento de *slices*, que levam em consideração multidomínios administrativos e tecnológicos. No

[2]http://clayfour.ee.ucl.ac.uk/lattice/.

contexto desse novo conceito de *slicing*, surge a necessidade de monitorar recursos de diferentes tipos em tais infraestruturas, como por exemplo, computacionais, de armazenamento e de rede. Portanto, esta proposta visa ampliar a discussão sobre este tema apresentando uma possível arquitetura e um modelo de informação minimalista para esta nova unidade de monitoramento, denominada *slice*.

Na Seção 2, a Figura 1 ilustra uma visão mais ampla de como seria o monitoramento de diferentes *slices* que estão alocadas em domínios administrativos distintos. Assume-se que o componente responsável pelo monitoramento faz parte do provedor de *slice*, que representa o domínio entrante para os três locatários ilustrados na figura. Os provedores de infraestrutura A, B e C possuem recursos instanciados que pertencem a *slices* de diferentes locatários. É responsabilidade do provedor de *slice* disponibilizar uma visualização de cada uma das *slices* para os seus respectivos locatários. Para isso, o componente de monitoramento deverá ser capaz de coletar e apresentar as informações de cada *slice* parte.

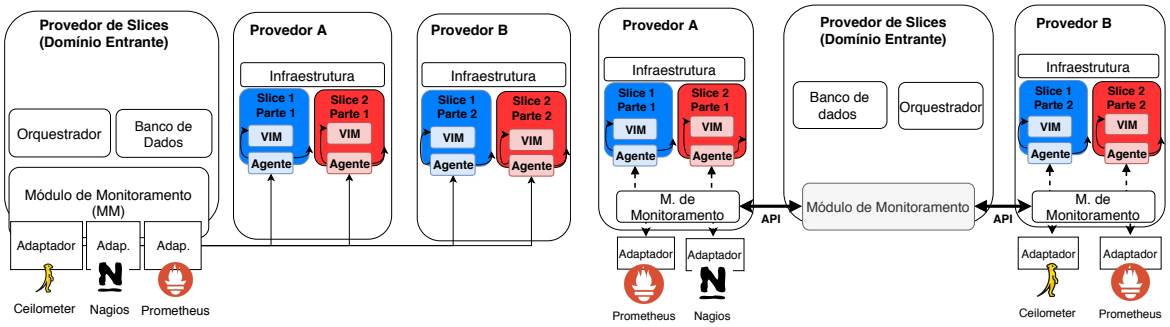
A tarefa de monitorar os recursos de uma *slice* pode ser considerada por diferentes perspectivas. A primeira é a perspectiva do locatário que, ao contratar uma *slice*, requer monitorar os recursos e/ou serviços que ele instanciou. Além do locatário, uma outra perspectiva em relação ao monitoramento é a do provedor de *slice*, que também precisa coletar métricas dos recursos alocados. Esta informação é utilizada para validar os contratos de utilização dos recursos previamente acordados, assim como, prover a orquestração inteligente dos recursos utilizando as informações providenciadas pelo componente de monitoramento. Considerando estas diferentes perspectivas baseadas nos interesses dos locatários e provedores, dois perfis podem ser elencados:

- **Perfil de monitoramento do locatário:** poderá requisitar informações relacionadas a recursos físicos, rede, armazenamento e serviços; baseadas no contrato definido no momento da instanciação de uma *slice*;
- **Perfil de monitoramento do provedor de *slice*:** deverá fornecer informações relacionadas aos recursos físicos, rede, armazenamento e serviços, instanciados por *slice*, que serão consumidas pelo locatário para as tarefas de orquestração.

### 4.1. Arquitetura

Este artigo busca apresentar duas possíveis arquiteturas que podem ser utilizadas com o objetivo de coletar métricas de diferentes tipos de recursos que estarão alocados em múltiplos domínios administrativos e tecnológicos. Como pode ser observado nas Figuras 2a e 2b, tal arquitetura possui como principais elementos: o Módulo de Monitoramento (MM), o Banco de Dados, o Orquestrador, os Adaptadores, os Agentes de Monitoramento (ou Ferramentas de Monitoramento) e os *Virtual Infrastructure Managers* (VIMs).

O MM é o principal componente da arquitetura, responsável pelo gerenciamento e monitoramento dos recursos alocados para uma determinada *slice*, de modo a disponibilizar informações sobre o uso dos recursos para o locatário e para o próprio provedor de *slices*. Para que o MM possa coletar as informações dos recursos e/ou serviços instanciados, ele deverá ser capaz de se comunicar com os diferentes provedores de infraestrutura que compõem a *slice*. Ao menos duas abordagens distintas podem ser implementadas para a comunicação do MM com os componentes de monitoramento. Estas abordagens representam a principal diferença entre as Figuras 2a e 2b e serão descritas a seguir.




**(a) Opção 1: Comunicação MM domínio entrante e Agentes.**

**(b) Opção 2: Comunicação MM domínio entrante e MM locais.**

**Figura 2. Arquiteturas para o monitoramento de *slices* multidomínios.**

Na Figura 2a observa-se apenas um componente MM instanciado no domínio do provedor de *slices*. Este MM será capaz de coletar e agregar as informações das diferentes *slice* partes de forma direta. Ou seja, o MM do domínio entrante deverá se comunicar com os agentes de monitoramento instanciados, através da implementação de diferentes adaptadores. Na Figura 2b, além do MM no domínio entrante, será necessário instanciar um MM em cada provedor de infraestrutura. Desta forma, o MM no domínio entrante, ao invés de se comunicar diretamente com os agentes de monitoramento, utiliza-se das interfaces com os MMs instanciados nos diferentes provedores de infraestrutura que compõem a *slice*. Em outras palavras, o papel que era desempenhado pelo MM no domínio entrante de coletar as métricas diretamente com os agentes de monitoramento, passa a ser responsabilidade dos MMs locais em cada um dos provedores de infraestrutura.

A partir das duas abordagens de implementação mencionadas acima, pode-se ressaltar as principais diferenças entre as mesmas:

- Opção 1: Comunicação do MM domínio entrante com os Agentes (Figura 2a).
  - O número de requisições é menor;
  - A informação estará concentrada no domínio entrante;
  - A implementação dos adaptadores será realizada no domínio entrante, retirando tal responsabilidade dos provedores de infraestrutura.

- Opção 2: Comunicação do MM domínio entrante com outros MMs (Figura 2b).
  - O número de requisições e o isolamento das informações é maior;
  - O provedor de infraestrutura tem a liberdade de configurar a visualização, podendo adaptar as métricas coletadas em um nível de granularidade maior;
  - A implementação dos adaptadores não precisa ser genérica, ou seja, cada provedor de infraestrutura possui diferentes implementações;
  - A implementação de uma API para a comunicação entre os MMs é necessária.

É válido ressaltar que as duas abordagens não são necessariamente excludentes e a arquitetura é genérica o suficiente para suportar as duas abordagens de modo a se adequar melhor ao funcionamento de cada provedor. Por exemplo, é provável que um provedor de *slices* que não possui infraestrutura física própria opte por implementar uma comunicação entre MMs, uma vez que não possui adaptadores para VIMs.

O adaptador é o componente responsável por interagir com diferentes agentes de monitoramento e/ou VIMs. Os adaptadores se comunicam com o Módulo de Monitoramento para obter as métricas de uma determinada *slice* parte. Como exemplo, pode-se destacar a implementação de adaptadores para as seguintes tecnologias: OpenStack, Docker, Prometheus, OpenStack-Telemetry, VLSP, Lattice, Gnocchi, SNMP, Nagios, entre outros.

O agente de monitoramento é o componente responsável por coletar as métricas, tanto da infraestrutura física quanto dos serviços. Cada provedor de infraestrutura pode disponibilizar para o locatário diferentes agentes de monitoramento. Alguns exemplos de agentes de monitoramento para recursos físicos são: SNMP, sFlow, IPMI, Prometheus, Nagios, entre outros. Alguns exemplos para recursos virtuais são: OpenStack-Telemetry, Prometheus, entre outros.

Os VIMs são responsáveis por instanciar os recursos virtuais relacionados à computação, ao armazenamento ou à rede. Alguns exemplos de VIMs são: OpenStack, VLSP, Kubernetes/Docker, entre outros. O Orquestrador utiliza as métricas monitoradas pelo MM com o intuito de otimizar a operação dos recursos alocados, além de armazenar constantemente no banco de dados as métricas monitoradas por *slice*. As outras funções que este componente desempenha estão relacionadas ao gerenciamento do *slice* e não fazem parte do escopo deste trabalho. O banco de dados armazena as métricas dos recursos que compõem um determinado *slice*, para que o Orquestrador ou o locatário possam acessar as informações de monitoramento.

### 4.2. Modelo de Informação para o Monitoramento em *Slices*

Esta subseção apresenta um modelo de informação com ênfase no monitoramento das *slices*. A Figura 3 ilustra um diagrama UML que detalha tal modelo, que pode ser estendido de acordo com a necessidade dos provedores de *slices*. Primeiramente, uma *slice* é composta basicamente por três diferentes tipos de infraestruturas principais: computação, armazenamento e rede. Sendo assim, uma solução de monitoramento para *slices* deve contemplar estas três categorias. A proposta a seguir foi baseada no draft do IETF[3] que aborda um modelo de informação para o conceito de *slice*, entretanto não possui como foco o monitoramento dos recursos desta nova entidade.

Discorrendo os relacionamentos presentes nesta entidade tem-se que, uma *slice* é composta por um ou mais domínios administrativos, onde cada domínio provê elementos de computação, armazenamento e/ou rede. Um domínio administrativo pode alocar um ou mais hosts, assim como elementos de rede. Os servidores físicos podem ser formados por nenhum ou vários elementos virtuais, que estão relacionados diretamente aos serviços, sejam eles instanciados em máquinas virtuais e/ou contêineres. Os elementos de rede, por sua vez, podem ser switches e/ou roteadores. Tanto os elementos de rede quanto as suas portas podem ser monitoradas pelo provedor de *slice* e/ou locatário. Um domínio pode conter um ou mais links nos quais estão associadas um conjunto de portas.

Os componentes descritos acima devem ser monitorados pelos provedores de modo a checar se os *KPIs* ou *SLAs* (do inglês, *Key Performance Indicators* e *Service Level Agreements*) acordados com o locatário serão cumpridos. Estes KPIs e SLAs, podem ser

[3]https://tools.ietf.org/html/draft-qiang-coms-netslicing-information-model-02

definidos para cada *slice* e estendidos para cada recurso que a compõem. Os exemplos de métricas a serem monitoradas por *slice* em cada componente podem ser visualizados na Figura 3. É importante salientar também que, caso seja necessário, tal modelo pode ser estendido de acordo com as requisições de cada locatário. Tanto a arquitetura de monitoramento quanto o modelo de informação descritos nesta seção, são propostas que visam ser genéricas o suficiente para que qualquer tipo de componente ou infraestrutura possam ser monitorados(as), considerando o conceito de *slices* multidomínios.

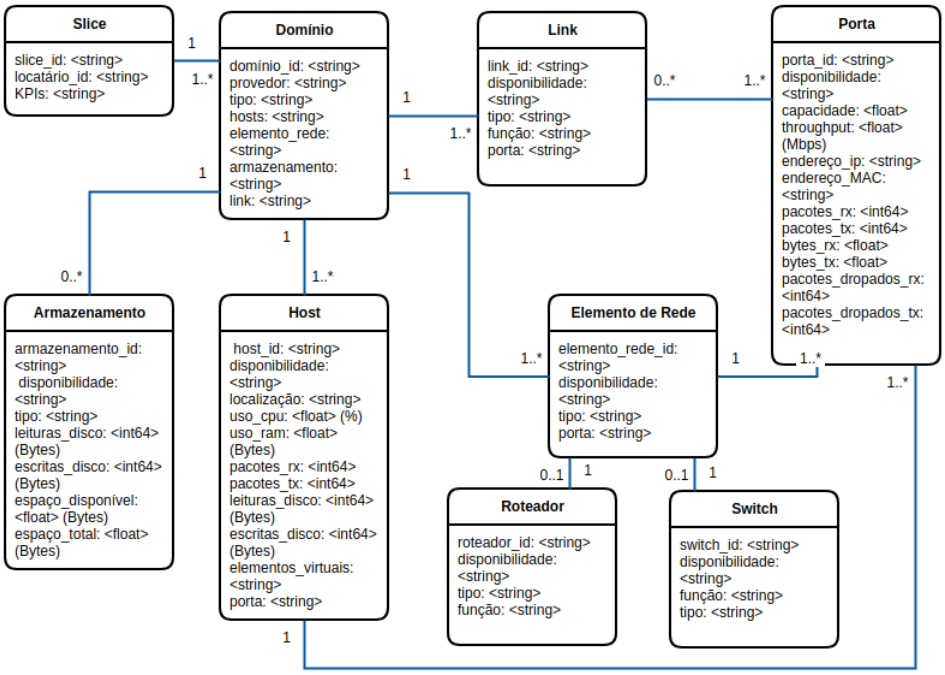



**Figura 3. Modelo de informação para o monitoramento de *slices*.**

## 5. Tecnologias Habilitadoras e Implementação

Esta seção expõe a implementação do MM e uma prova de conceito que busca exercitar tanto a arquitetura proposta quanto o modelo de informação. Para este trabalho, considera-se apenas o monitoramento dos elementos virtuais (máquinas virtuais e/ou contêineres) relacionados à infraestrutura de computação das *slices*. Inicialmente, optou-se pela arquitetura de comunicação representada na Figura 2a, assim como escolheu-se o perfil de monitoramento sob o ponto de vista do locatário (explicado na Seção 4). Em trabalhos futuros pretende-se implementar ambas as arquiteturas e analisar as diferenças das abordagens quantitativamente, o que neste trabalho foi exercitado qualitativamente.

### 5.1. Tecnologias Habilitadoras

Como mencionado anteriormente, uma *slice* pode ser formada por diferentes domínios, tanto administrativos como tecnológicos. Logo, para implementação de uma prova de conceito que represente tanto o monitoramento deste ambiente quanto o modelo de informação, decidiu-se por utilizar as seguintes tecnologias habilitadoras:

- O VIM utilizado será o Docker, responsável por instanciar contêineres;

- O Banco de dados responsável por armazenar as métricas coletadas foi construído em MySQL de acordo com o modelo de informação descrito na Subseção 4.2;
- O Agente de Monitoramento utilizado foi o Prometheus [4], que ficou responsável por coletar as métricas de monitoramento;
- O MM foi implementado na linguagem de programação Python e comunica-se com o Agente de monitoramento (Prometheus) via uma API REST.

O Prometheus é um framework capaz de monitorar diferentes tipos de infraestruturas e até mesmo aplicações por meio de elementos denominados *exporters*. Recentemente foi reconhecido pela Cloud Native Computing Foundation[5] e vêm sendo amplamente estudado e aprimorado. De certa forma, o Prometheus é capaz de disponibilizar as informações de diferentes *exporters*, como por exemplo, um *exporter* para cada tipo de infraestrutura como por exemplo, elementos físicos, recursos virtuais e até mesmo métricas de aplicações. Tal funcionalidade é muito interessante quando relacionada ao conceito de *slices*, pois facilita a forma em que diferentes métricas podem ser agregadas por componentes, como por exemplo os Adaptadores que compõem a arquitetura proposta neste artigo.

### 5.2. Implementação

Esta subseção detalha a implementação do componente MM, bem como discorre o esquema utilizado no banco de dados. O esquema baseia-se no modelo de informação apresentado na Figura 3 e abrange apenas o monitoramento das métricas dos elementos virtuais que compõem uma *slice*.

O *Algoritmo* 1 representa o pseudocódigo da implementação do componente MM. Na linha 1, observa-se os pré-requisitos para o início do algoritmo, que contempla um vetor contendo as *slices* que o MM irá monitorar e os agentes que o MM deverá se comunicar para obter as métricas. Então, da linha 4 a 6, uma nova *thread* é criada por *slice*, passando os agentes de monitoramento para o método *monitoraSlice*. Este método é observado entre as linhas 9 e 13, que tem como função se comunicar com cada um dos agentes informados para obter as métricas de monitoramento através do método $coletaMetricas()$ presente na linha 11. Este método recebe como parâmetro informações relacionadas ao agente de monitoramento. Após as métricas serem obtidas para cada agente, as mesmas são adicionadas no banco de dados.

Pelo fato deste trabalho considerar apenas a infraestrutura virtual relacionada ao elemento de computação de uma *slice*, o esquema do banco de dados criado engloba apenas os seguintes objetos apresentados na Figura 3: *slice*, domínio, servidores físicos e elementos virtuais. Portanto, no escopo atual do trabalho, as métricas relacionadas aos servidores físicos não são coletadas. Mais especificamente, as métricas coletadas dos elementos virtuais são: percentual de uso da CPU, quantidade de bytes utilizados de memória RAM, quantidade de bytes transmitidos e enviados pela rede, além da quantidade de bytes escritos no disco. As visualizações destas métricas são apresentadas na Seção 6.

## 6. Prova de Conceito e Resultados

Esta seção discorre sobre o ambiente de testes utilizado para validar a arquitetura proposta e sobre o modelo de informação. Esta prova de conceito está relacionada ao monitora-

[4]https://prometheus.io/
[5]https://www.cncf.io/

**Algoritmo 1** Pseudocódigo - MM

```
1: Slices = {Slice, [Agentes]}
2: procedure MÓDULO DE MONITORAMENTO
3:     enquanto true faça
4:         para cada slice em Slices faça
5:             novaThread = monitoraSlice(Agentes)
6:         fim para
7:     fim enquanto
8: fim procedure
9: procedure MONITORASLICE(Agentes de monitoramento)
10:     para cada agente em Agentes faça
11:         metricas = coletaMetricas(agente)
12:         adicionaBanco(metricas)
13:     fim para
14: fim procedure
```

mento de *slices* previamente instanciadas, através de multidomínios administrativos.

### 6.1. Prova de Conceito

Na Figura 4, observa-se a infraestrutura utilizada que consiste em duas *slices* instanciadas ao longo de dois provedores de infraestrutura, denominados *Provedor A* e *Provedor B*. Cada *slice* parte pertencente aos provedores de infraestrutura possui um servidor físico dedicado ilustrado por $SF\#$ ($SF1 - SF4$). Em cada servidor $SF\#$ há um contêiner representado por $C\#$ e um agente de monitoramento representado por $A\#$. O agente de monitoramento para todos os quatro $C\#$ é o Prometheus. O provedor de *slice* está representado mais à direita na Figura 4. A figura ilustra também os componentes: Banco de Dados, MM e Orquestrador. Entretanto, para os testes realizados, apenas os dois primeiros foram implementados e fazem parte do escopo deste trabalho.

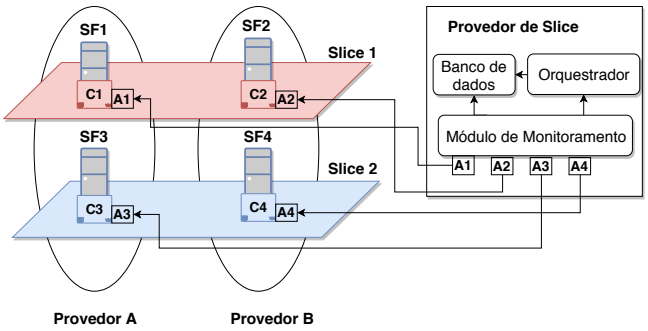



**Figura 4. Ambiente de Testes.**

De acordo com a arquitetura proposta, para cada agente ($A\#$) contido nos provedores de infraestrutura existe um adaptador implementado no MM. Contudo, como o framework de monitoramento é o mesmo (Prometheus) para todos os agentes na implementação da prova de conceito, foi implementado um adaptador no MM capaz de coletar as métricas dos diferentes agentes Prometheus. Em suma, o ambiente de testes consiste de duas *slices* instanciadas ao longo de dois provedores de infraestrutura (*Provedor A* e *Provedor B*), contendo os seguintes componentes por *slice*: dois servidores físicos ($SF\#$), dois contêineres ($C\#$) e dois agentes de monitoramento ($A\#$). Externamente aos provedores de infraestrutura, o *Provedor de Slice* executa o MM, que é responsável por monitorar e armazenar informações de cada uma das *slices*.

Para o cálculo das métricas relacionadas às *slices*, obteve-se a média aritmética dos valores individuais[6]. Mais especificamente, a cada intervalo de tempo $\Delta t = 10s$, foram coletadas as métricas dos contêineres pertencentes a uma *slice* e calculada a média das métricas em cada intervalo de tempo. Para cada um dos contêineres, simulou-se uma carga de trabalho adicional com o intuito de aumentar o uso dos recursos (CPU, memória, rede e disco). Para aumentar a carga de trabalho na CPU, memória e disco, utilizou-se o programa *stress-ng* disponível no Linux. Com relação à rede, gerou-se um tráfego de comunicação entre os contêineres e um servidor a partir do programa *iperf*[7]. Além disso, três perfis distintos de consumo dos recursos foram considerados: baixo, médio e alto. Nos testes realizados, estes perfis foram simulados a partir de uma variação sequencial a cada 10 minutos, resultando em testes de 30 minutos. Então, para cada uma das duas *slices* foram iniciados dois contêineres, onde um seguiu uma ordem crescente de perfil de consumo (baixo, médio e alto) e o outro a ordem inversa (alto, médio e baixo).

### 6.2. Resultados

Os resultados expostos a seguir têm o intuito de ilustrar algumas das métricas descritas acima, monitoradas ao longo do tempo. Vale ressaltar que os gráficos e visões apresentados nesta seção representam parte de um subconjunto do que poderia ser explorado com relação às métricas monitoradas atualmente. Por exemplo, optou-se por exibir as métricas por *slices* e também por recursos virtuais relacionados à uma determinada *slice*. Entretanto, poderia haver outras formas de exibir as métricas coletadas, uma vez que a arquitetura em conjunto com o modelo de informação proposto proporcionam o isolamento das métricas monitoradas por *slice*.

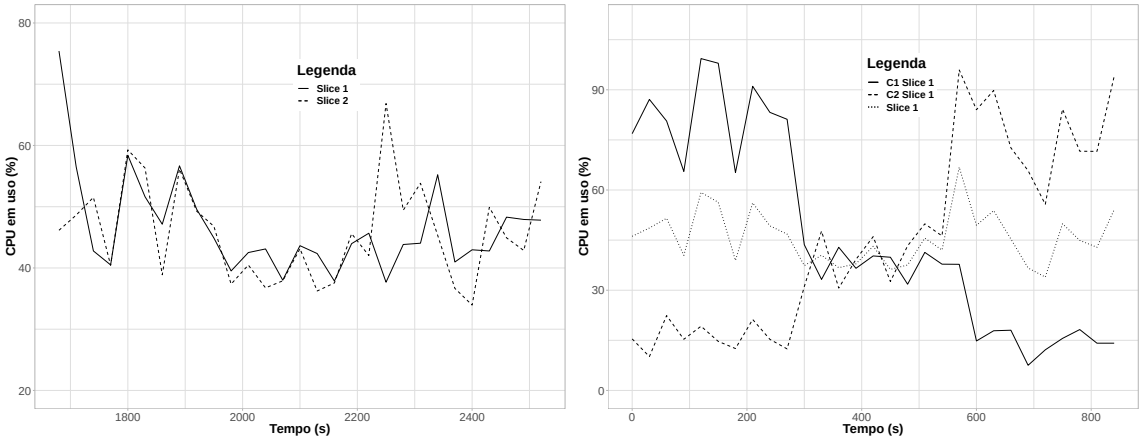
Na Figura 5, dois gráficos ilustram as métricas coletadas no ambiente de testes descrito na Subseção 6.1. Ambos os gráficos apresentam as métricas monitoradas (eixo y) ao longo do tempo (eixo x). A Figura 5a mostra o uso médio de CPU dos recursos virtuais para as *Slices* 1 e 2. Pela Figura 5b, pode-se observar um detalhamento nos valores de monitoramento de cada um dos recursos virtuais específicos para a *Slice* 1. Como dito anteriormente, nos experimentos executados, o $C1$ iniciou com um perfil de carga de trabalho definido como baixo, enquanto o $C2$ iniciou com o perfil alto.

A análise dos gráficos na Figura 6 é análoga a descrita no parágrafo anterior, onde o gráfico mais à esquerda (Figura 6a) ilustra a quantidade de bytes transmitidos pelos contêineres que compõem a *Slice* 2. O gráfico na Figura 6b detalha o consumo de memória RAM dos contêineres que fazem parte da *Slice* 1. Como este ainda é um trabalho preliminar, o foco dos resultados apresentados é apenas ilustrar alguns componentes de uma *slice* sendo monitorados ao longo do tempo.

A partir das Figuras 5 e 6, é possível observar a variação dos perfis de carga de trabalho no comportamento de consumo dos contêineres instanciados. Nota-se, por exemplo, que no primeiro terço do tempo de coleta um dos contêineres apresenta um alto perfil de carga de trabalho, enquanto o outro simula uma carga baixa de trabalho. Como esperado, este comportamento se inverte no terceiro terço do tempo de coleta, enquanto no segundo terço o comportamento dos dois é similar com uma carga média de trabalho.

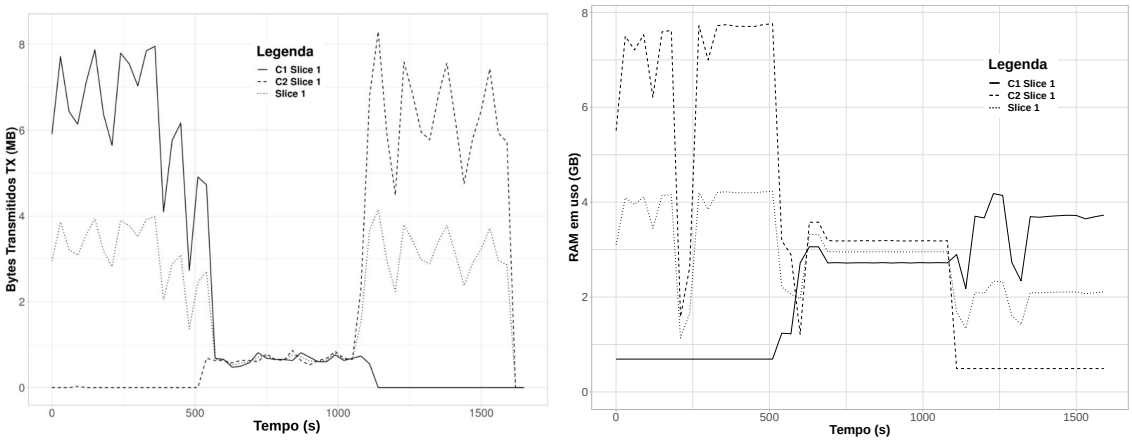
[6]Tais métricas têm o intuito apenas de ilustrar o monitoramento dos recursos, pois a informação da média de uso de um recurso de uma slice pode não agregar tanta informação para possíveis ações de orquestração.

[7]`https://iperf.fr/`




**(a) CPU por *Slice*.** **(b) CPU para a *Slice* 1.**

**Figura 5. Métricas relacionadas à CPU (%).**




**(a) Bytes transmitidos para a *Slice* 2.** **(b) RAM para a *Slice* 1.**

**Figura 6. Métricas relacionadas a Rede 6a e a Memória 6b.**

## 7. Conclusões e Trabalhos Futuros

O conceito de *cloud network slicing* traz novos desafios que precisam ser investigados em relação ao monitoramento dessa nova entidade. Coletar informações dos recursos fim-a-fim através de múltiplos domínios administrativos e tecnológicos é de suma importância para aprimorar operações de gerência e orquestração.

Neste contexto, este artigo propõe uma arquitetura para o monitoramento de *slices* fim-a-fim, considerando o cenário de multidomínios administrativos e/ou tecnológicos. Outra contribuição importante é a proposta de um modelo de informação para o monitoramento de *slices*, que envolve diferentes infraestruturas. Por fim, pode-se destacar a implementação de uma prova de conceito baseada na arquitetura proposta. A partir desta prova de conceito foi possível visualizar métricas relacionadas aos recursos virtuais que haviam sido instanciados, bem como as métricas por *slice* de forma isolada.

A extensão imediata deste trabalho envolve avaliar quantitativamente a escalabilidade da arquitetura proposta, aumentando o número de métricas, *slices*, recursos, dentre outros. Pretende-se também adicionar adaptadores para outras infraestruturas e/ou agen-

tes de monitoramento, como por exemplo Kubernetes e OpenStack. Por fim, planeja-se aproveitar a arquitetura proposta para servir de base para a implementação do componente responsável pela gerência e orquestração dos recursos alocados, levando em consideração as métricas monitoradas.

## Agradecimentos

Este trabalho é financiado pela chamada conjunta Europa-Brasil H2020 número 777067, no contexto do Projeto NECOS (*Novel Enablers for Cloud Slicing*), co-financiado pela União Europeia e pelo Ministério de Ciência, Tecnologia, Inovação e Comunicação (MCTIC), através do Centro de Pesquisa e Desenvolvimento em Tecnologias Digitais para Informação e Comunicação (CTIC) da Rede Nacional de Pesquisa (RNP).

## Referências

Clayman, S. (2018). D3.1: NECOS System Architecture and Platform Specification. V1. Technical report.

de Foy, X. and Rahman, A. (2017). Network Slicing - 3GPP Use Case. Internet-Draft draft-defoy-netslices-3gpp-network-slicing-02, IETF Secretariat.

ETSI ISG NFV (2018). GR NFV-IFA 022 - V3.1.1 - Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on Management and Connectivity for Multi-Site Services. Technical report.

Fatema, K., Emeakaroha, V. C., Healy, P. D., Morrison, J. P., and Lynn, T. (2014). A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, 74(10):2918 – 2933.

Kuklinski, S. and Tomaszewski, L. (2018). Dasmo: A scalable approach to network slices management and orchestration. *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6.

NGMN Alliance (2018). 5G End-to-End Architecture Framework v2.0. Technical report.

Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J., and Folgueira, J. (2017). Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Communications Magazine*, 55(5):80–87.

Qiang, L., Galis, A., Geng, L., Makhijani, K., Martinez-Julia, P., Flinck, H., and de Foy, X. (2018). Technology Independent Information Model for Network Slicing. Internet-Draft draft-qiang-coms-netslicing-information-model-02, Internet Engineering Task Force. Work in Progress.

TR-521, O. (2016). SDN Architecture. Technical report, Open Networking Foudation.

Tusa, F., Clayman, S., and Galis, A. (2018). RealTime Management and Control of Monitoring Elements In Dynamic Cloud Network Systems. In *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, pages 1–7. IEEE.

# Arcabouço de um Sistema Inteligente de Monitoramento para *Cloud Slices*

**Gustavo S. Marques[1], Aryadne G. P. Rezende[1], Ian R. da Cunha[1], Raquel F. Q. Lafetá [1], Rafael Pasquini[1]**

[1]Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Caixa Postal 593 – 38.400-902 – Uberlândia – MG – Brazil

{guto_silveira_, aryadne.guardieiro, ianresende}@ufu.br

{raquel.lafeta, rafael.pasquini}@ufu.br

***Abstract.*** *This work builds upon the slice-as-a-service paradigm proposed in the Novel Enablers for Cloud Slicing (NECOS) project. Assuming the provision of end-to-end slices which are composed by resources coming from multiple infrastructure providers, this work investigates an intelligent monitoring system, aiming to dynamically select a set of features that best fits the real life time management needs of the slices, keeping the accuracy of that management. We want to avoid the movement of unnecessary information from the infrastructure providers, delivering essential data to management functions. An innitial experiment in our testbed is presented and some benefits of feature selection techniques can already be seen.*

***Resumo.*** *Esse trabalho se baseia no paradigma slice-as-a-service proposto no projeto Novel Enablers for Cloud Slicing (NECOS). Assumindo fatias (slices) fim-a-fim, compostas por recursos de múltiplos provedores de infraestrutura, esse trabalho estuda um sistema inteligente de monitoramento capaz de selecionar dinamicamente métricas que melhor atendam às necessidades de gerenciamento das slices, mantendo a sua precisão. Basicamente, deseja-se evitar o tráfego de dados desnecessários extraídos dos diferentes provedores de infraestrutura, entregando um conjunto essencial de dados para as funções de gerenciamento. Um primeiro experimento em nosso protótipo é apresentado e alguns dos benefícios da seleção de características já podem ser observados.*

## 1. Introdução

Nos últimos anos, computação em nuvem se tornou um assunto bastante relevante no meio tecnológico. Entre as suas contribuições pode-se destacar algumas, tais como: permitir processamentos pesados, maior capacidade de armazenamento de informação, acesso a partir de qualquer localidade e etc.

Empresas como Google, Microsoft, Amazon, IBM e outras, disponibilizam seus servidores (nuvem) oferecendo aos seus usuários uma variedade de serviços, criando conceitos como IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*) e SaaS (*Software as a Service*) [Kächele et al. 2013]. Estes serviços demonstram a versatilidade da computação em nuvem, a qual será mais ou menos aproveitada dependendo da necessidade do usuário. Alguns exemplos destes serviços são: hospedagem de sites, serviços de e-mail, serviços de armazenamento de dados e etc.

Entretanto, embora a computação em nuvem traga muitos benefícios, ela traz igualmente alguns desafios. Nesse contexto, pode-se levantar alguns questionamentos: 1) um único provedor de servidores de nuvem é sempre capaz de suprir todas as demandas de seus clientes? 2) Não sendo capaz, o que pode ser feito? 3) Seria possível estabelecer comunicação entre diversos provedores, de maneira que, um provedor possa contratar serviços de terceiros para ofertar ao seu cliente uma solução completa, de maneira transparente, quando não for capaz de o atender integralmente?

Nesse contexto, o projeto NECOS (*Novel Enablers for Cloud Slicing*) [Necos 2017c] propõe um novo paradigma definido como *slice-as-a-service*. Neste cenário, fatias (*slices*) são provisionadas entre provedores federados que fornecem diferentes tipos de infraestrutura, isto é, entre provedores de nuvem, de rede, etc. Portanto, uma *slice* é um conjunto de recursos de infraestrutura ofertado por vários provedores e que atende a necessidade de um usuário do sistema (aqui chamado de *Tenant*). Um *Tenant* pode ser entendido como um usuário final de serviços de nuvem, mas também como os próprios provedores e empresas.

Na arquitetura do projeto NECOS[1], pode-se encontrar quatro subsistemas: *Tenant's Domain*; *Slice Provider*; *Marketplace*; e *Infrastructure Providers*. Todos suportam o paradigma *slice-as-a-service* proposto, oferecendo *slices* sob demanda aos *Tenants* [Necos 2017a] [Necos 2017b]. O fluxo de criação de uma *slice* no NECOS inicia-se no *Slice Provider* ao receber uma requisição para uma nova *slice* vinda de um *Tenant*. Isto é, informações como quantidade de CPU, tamanho do disco, quantidade de memória RAM, velocidade de *uplink* e *downlink* e descrição do serviço a ser hospedado na *slice*, são fornecidos pelo *Tenant* ao *Slice Provider*. Em interação com os demais subsistemas, o *Slice Provider* cria uma *slice* fim-a-fim que preenche por completo os requisitos do *Tenant*.

Finalizada a criação da *slice*, o *Slice Provider* passa a monitorá-la para fornecer ao *Tenant*, bem como aos outros módulos do sistema, as informações sobre o estado da infraestrutura na qual a *slice* está hospedada. Esse monitoramento é essencial para o gerenciamento da *slice*, porque além de ela poder sofrer alterações na sua infraestrutura, é necessário saber se a composição atual continua atendendo às necessidades dos serviços sendo ofertados nela.

Entretanto, dada a variedade de recursos que compõem as *slices*, o sistema de monitoramento pode sofrer problemas de escalabilidade, devido a grande quantidade de dados disponíveis na infraestrutura. Considerando a relevância da monitoria para o bom funcionamento da *slice*, se faz necessário pensar num sistema inteligente de monitoramento, capaz de não apenas coletar dados, mas também determinar um conjunto essencial de métricas, que realmente estão relacionadas ao bom funcionamento do serviço sendo ofertado. Idealmente, essa seleção precisa se dar de maneira automática, uma vez que esse conjunto pode ser diferente para cada serviço e para cada composição da *slice*.

Nesse contexto desafiador, o principal objetivo deste trabalho é investigar algoritmos de aprendizado de máquina, aplicados em conjunto com as técnicas de seleção de características, para a construção de um sistema inteligente de monitoramento. Feita essa seleção, há também o objetivo de evitar o tráfego de informações desnecessárias, o que significa que, uma vez selecionadas as métricas mais relevantes, é importante que as

[1]http://www.h2020-necos.eu/documents/deliverables/

demais deixem de ser trafegadas o mais próximo possível da sua fonte. Resultados iniciais coletados em nosso protótipo demonstram a eficácia da seleção automática de características, não apenas reduzindo o volume de métricas a serem coletadas, como também permitindo estimativas mais acuradas quanto ao serviço em execução.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta alguns trabalhos relacionados, nos quais também se fez necessária a seleção de características; a Seção 3 introduz a fundamentação teórica, resumindo conceitos de aprendizado de máquina e de seleção de características; a Seção 4 detalha o sistema inteligente de monitoramento em construção; a Seção 5 apresenta os resultados coletados em nosso primeiro experimento; e, por fim, a Seção 6 traz as conclusões e trabalhos futuros.

## 2. Revisão da Literatura Correlata

Esta Seção apresenta alguns trabalhos relacionados, apresentando algoritmos de seleção de características, além de um trabalho que buscou aplicar algumas técnicas de seleção de características em um cenário bastante similar ao trabalho apresentado neste artigo.

No trabalho [Yang et al. 2005] há um cenário em que as características são ondas cerebrais emitidas pelo usuário e, com base nelas, procura-se identificar quem são os usuários do sistema. Entretanto, várias são as ondas captadas e a aplicação de um algoritmo de seleção de característica se fez necessária. Para isso, foi proposto um algoritmo novo de seleção de características, nomeado *Corona*, baseado em *Support Vector Machine* (SVM). Em [Yang et al. 2005] as características são séries temporais multivariadas, isto é, para um mesmo instante de tempo numa mesma série temporal há mais de uma informação a ser analisada. Isso também ocorre neste artigo, uma vez que um mesmo serviço pode ter várias instâncias, o que significa que para um mesmo instante de uma mesma métrica poderá existir mais de um valor a ser analisado. Em um serviço de vídeo sob demanda, por exemplo, onde há mais de um servidor sendo utilizado para ofertá-lo, haverá uma série temporal multivariada, o que permite que o *Corona* seja testado, embora o cenário seja diferente.

No trabalho [Kira and A. Rendell 1992] foi desenvolvido um novo algoritmo para seleção de características, denominado *Relief*, executado com base em métodos estatísticos que, segundo os autores, não depende de heurísticas. Além disso, é dito que o algoritmo tem tempo linear com relação ao número de características dadas e na quantidade de instâncias de treinamento. Para testar o algoritmo, os autores forneceram um conjunto de características em que apenas duas delas eram consideradas relevantes. Em um dos piores resultados apresentados, o algoritmo *Relief* conseguiu encontrá-las em $70\%$ dos casos, com um tempo relativamente baixo de aprendizado. Portanto, o resultado obtido pelos autores pode servir de base de comparação para os que serão obtidos no trabalho aqui apresentado.

Em [John et al. 1994] é apresentado um algoritmo de aprendizado supervisionado que determina o grau de relevância de uma característica. No método proposto, os autores compararam as abordagens *backward stepwise elimination* e *forward stepwise elimination*. Essas abordagens são baseadas, respectivamente, nos algoritmos de *backward elimination* e *forward elimination*. O primeiro consiste em inicialmente se tomar o conjunto completo de características e, de forma gulosa, retirar aquela em que o conjunto resultante é melhor que o conjunto atual, ou que o conjunto restante seja ligeiramente

pior, mas que compense a diminuição de uma característica. O segundo funciona de forma análoga, porém iniciando com um conjunto vazio.

As abordagens comparadas adicionam uma melhoria nos algoritmos *backward elimination* e *forward elimination*. Esta melhoria consiste em se considerar a ação de adicionar ou remover características a cada passo. No primeiro algoritmo há apenas remoção, enquanto no segundo há apenas adição. Mas com a melhoria, tomando o *backward stepwise elimination* como exemplo, é possível, em cada passo, adicionar características que foram removidas, caso estas melhorem a performance do conjunto resultante. Desta forma, é válido tentar aplicar o método proposto em [John et al. 1994] para executar os algoritmos de seleção de características no sistema proposto neste trabalho.

Em [Pasquini and Stadler 2017] o cenário é bem relacionado com o deste trabalho. O objetivo era prever as métricas de qualidade de serviço de vídeo sob demanda e de leitura e escrita de dados em uma DHT (*Distributed hash table*). Para isso, os autores compararam dois algoritmos de aprendizado de máquina (*random forest* e *regression tree*). Ambos os algoritmos também foram utilizados na construção de conjuntos reduzidos de características correlacionadas as da qualidade do serviço. Os autores chegaram a conclusões relevantes, atestando que, embora as predições passassem a ser feitas com base no conjunto reduzido de métricas, o erro não subiu de maneira substancial, e em alguns casos o erro foi bastante próximo do obtido com o conjunto completo.

Futuramente, os dois serviços apresentados em [Pasquini and Stadler 2017] também serão avaliados no contexto deste trabalho, porém com duas importantes modificações: 1) no serviço de vídeo sob demanda, o protocolo DASH [Encoding.com 2016] será utilizado com intuito de analisar se os algoritmos de aprendizado continuam sendo capazes de construir bons conjuntos de métricas a partir das do serviço monitorado. Esse protocolo gera uma dificuldade a mais, pois altera dinamicamente a qualidade do vídeo de acordo com a carga aplicada ao servidor e a velocidade de banda do cliente. Portanto, suspeita-se que a predição de qualidade de serviço seja dificultada; e, 2) no serviço de DHT, será avaliado o serviço do Cassandra [Apache 2016] e não o Voldemort [Voldemort 2016].

## 3. Fundamentação Teórica

Esta Seção introduz algumas definições sobre os métodos de seleção de características, bem como conceitos sobre algoritmos de aprendizado de máquina.

Em [Chandrashekar and Sahin 2014] os autores apresentam uma revisão da literatura abordando os algoritmos reconhecidos como estado da arte de seleção de características. Além disso, é apresentada uma classificação das técnicas, dividindo-as segundo a estratégia de busca implementada para selecionar as características, que podem ser classificadas em: 1) *wrapper*, métodos que utilizam o mesmo algoritmo de aprendizado de máquina tanto no processamento/mineração dos dados quanto para seleção de características; 2) *filter*, métodos que primeiro constroem um *ranking* das características baseado em algum critério, para utilizá-lo posteriormente selecionando as características de acordo com a posição no *ranking*; 3) *embedded*, métodos que selecionam as melhores características a medida em que se cria o modelo de aprendizado.

Nesse sentido, todas as classes podem ser aplicadas. Conforme se vê em [Pasquini and Stadler 2017], tanto se pode utilizar os algoritmos de aprendizado de

máquina para a seleção de características, quanto se pode construir um modelo de seleção independente do algoritmo utilizado para o aprendizado. Sendo assim, qualquer uma das estratégias de busca podem ser exploradas. No contexto do trabalho aqui apresentado, explorou-se a estratégia *filter*, portanto, criou-se um *ranking* e em seguida selecionou-se as $K$ primeiras características deste *ranking*.

No que tange a algoritmos de aprendizado de máquina, pode-se estabelecer a seguinte divisão: algoritmos de aprendizado não-supervisionado, de aprendizado semi-supervisionado e de aprendizado supervisionado.

No primeiro caso, são algoritmos aplicados quando não há qualquer informação sobre os dados coletados e se deseja aprender algo sobre eles [Fulmari1 and Chandak 2013]. Diz-se que não se tem os respectivos rótulos dos dados coletados. Um exemplo, seria um cenário em que se desejasse estabelecer relação entre pessoas com base nas suas características. Neste caso, poderia se aplicar um algoritmo de aprendizado não-supervisionado para agrupar (*clustering*) e detectar, por exemplo, que pessoas com determinado estilo de vida também são mais propensas a desenvolver determinada doença, sem que essa condição seja conhecida anteriormente.

No segundo caso, são algoritmos em que os dados em sua maioria não possuem os respectivos rótulos. Segundo [Zhu 2005], algoritmos deste tipo buscam construir classificadores. Ainda segundo [Zhu 2005], o fato de esses algoritmos trabalharem com dados não rotulados faz com que esta abordagem seja interessante em muitos casos, pois definir tais rótulos é uma tarefa frequentemente árdua.

No terceiro caso, todos os dados são rotulados. Isso significa que para amostragem do conjunto de dados coletados, encontra-se disponível um ou mais rótulos que indicam o que a amostra representa [Fulmari1 and Chandak 2013]. Os rótulos podem ser discretos, utilizados em algoritmos de classificação, como também podem ser valores contínuos, utilizados em algoritmos de regressão.

Portanto, em aprendizado supervisionado há dois conjuntos de dados essenciais, comumente denominados $X$ e $Y$. O conjunto $X$ contém as amostras coletadas para as análises e o conjunto $Y$ contém os rótulos relativos a estas amostras coletadas. Por exemplo, num cenário onde se deseja prever se o dia será ensolarado, chuvoso ou nublado, podem ser analisadas amostras relativas a temperatura, umidade do ar, latitude e longitude. Nesse caso, se construiria um conjunto $X$ formado pelas informações de temperatura, umidade do ar, latitude e longitude, e um conjunto $Y$ com as respectivas condições do dia.

Desta forma, o objetivo de um algoritmo de aprendizado supervisionado é construir uma função $F$ que receba os dois conjuntos de dados, $X$ e $Y$, e busca estabelecer a relação entre ambos os conjuntos. Assim, quando $F$ for aplicada aos valores encontrados em $X$, deseja-se obter os respectivos rótulos em $Y$. Uma vez estabelecida tal função $F$, esta passa a ser aplicada em novas amostras de $X$ para que se possa estimar o rótulo que existiria em $Y$.

Há ainda uma divisão quanto ao objetivo de se utilizar um algoritmo de aprendizado de máquina. Nesse contexto, pode-se desejar fazer uma classificação ou uma regressão. Classificação significa que o objetivo é dividir os dados coletados em classes discretas ou categorias. O exemplo mencionado sobre prever a condição do dia é uma

forma de classificação. Na regressão deseja-se construir uma função $F$ com base nos valores encontrados em $X$ e em $Y$, de tal forma que se possa usar $F$ para estimar valores contínuos de $Y$, dados novos valores de $X$.

Para este trabalho serão utilizados algoritmos de aprendizado supervisionado, buscando executar uma regressão nos dados, para que se possa estimar as métricas do serviço em questão. Esses algoritmos também poderão, eventualmente, ser utilizados para fazer a seleção de características em nossos experimentos. A técnica supervisionada se adéqua melhor ao cenário deste trabalho, pois as métricas coletadas possuem seus rótulos e estes são conhecidos do sistema de monitoramento. O protótipo desenvolvido é capaz de coletar métricas relativas ao funcionamento dos serviços, de tal forma que as coletadas na infraestrutura são automaticamente rotuladas. Esta é uma característica essencial do protótipo, pois, conforme dito anteriormente, obter rótulos é uma tarefa geralmente custosa.

## 4. Monitoramento Inteligente

O NECOS auxilia os *Tenants* a obterem suas *slices* sob demanda. Este processo inclui desde uma simples combinação de recursos para construção da *slice* até um gerenciamento complexo e automático, que consiste em alterações nos recursos que compõem as *slices*, ou até mesmo do local físico em que ela está hospedada. Ao requisitar uma *slice*, o *Tenant* opta, entre outros pontos, pelo nível de gerenciamento que necessita. Esses níveis de gerenciamento podem ir desde o nível $0$, quando o *Tenant* gerencia a *slice* por conta própria, até o nível $3$, em que o *Tenant* deseja que o NECOS faça todo o controle da *slice*. O sistema também é responsável por realizar as alterações nas *slices*, visando manter os *Service Level Agreements* (SLAs) [Necos 2017a].

Desta forma, para possibilitar que tais funcionalidades existam, o NECOS possui um sistema de monitoramento contido no módulo IMA (*Infrastructure & Monitoring Abstraction*). Atualmente, o IMA é capaz de coletar métricas a partir dos diferentes provedores de infraestrutura presentes na federação, trabalhando com diferentes tecnologias e atuando em diferentes granularidades de coleta.

Neste projeto, o IMA é complementado com a capacidade de coletar dados dos serviços sendo ofertados nas *slices*. Os dados do serviço, conforme mencionado anteriormente, constituem uma alternativa para composição do conjunto $Y$ de dados para o aprendizado supervisionado neste artigo. Além da coleta de $Y$ relativo ao serviço, outra contribuição deste trabalho está em melhorar o atual sistema de coleta de dados do NECOS, adicionando uma seleção inteligente de características. O cenário ideal consiste em reduzir significativamente o conjunto de métricas coletadas por *slice*, mantendo a precisão das tarefas de gerenciamento.

A Figura 1 ilustra o fluxo de monitoramento do NECOS, o qual envolve principalmente o IMA. O processo de monitoramento se inicia com a alocação de recursos nos provedores de infraestrutura para uma determinada *slice*. De acordo com a requisição do *Tenant*, uma grande quantidade de tecnologias pode ser instanciada nos provedores para compor a *slice*. Isso implica que o IMA precisa se comunicar com diferentes provedores, que estabelecem padrões de comunicação diferentes uns dos outros, diferença essa que pode se dar tanto na forma de requisitar os dados, quanto na formatação da resposta. Portanto, o IMA implementa *Adaptors* (AD) para que seja possível comunicar-se com os

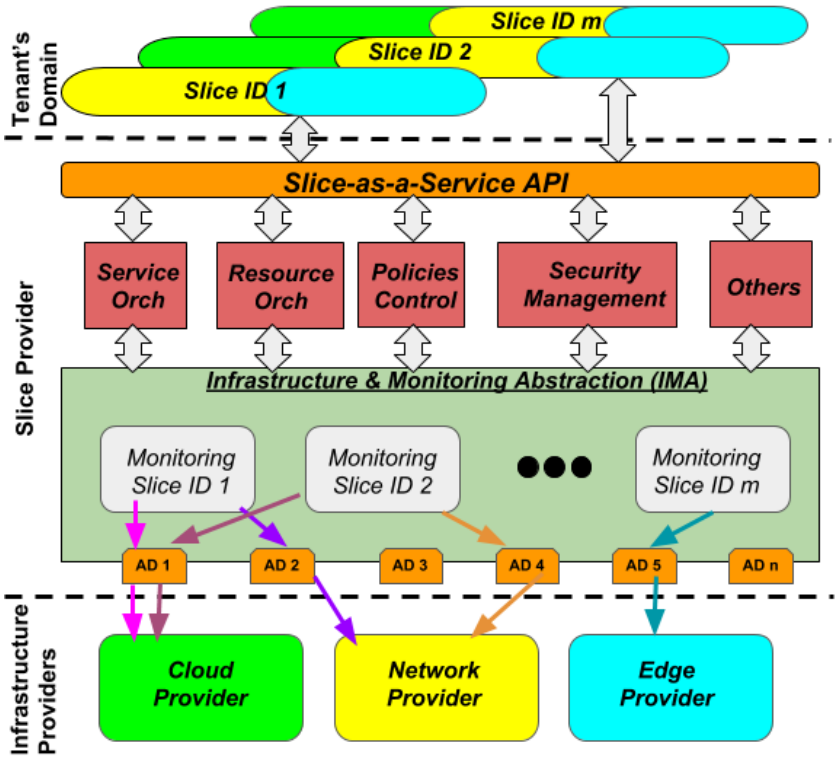



**Figura 1. Visão geral do monitoramento no NECOS. O módulo IMA usa adaptadores para coletar métricas de diferentes tecnologias que os *Tenants* requisitam para construção das suas *slices*, instanciadas na infraestrutura federada de provedores. As métricas coletadas são fornecidas para as diferentes tarefas de gerenciamento desempenhadas durante o ciclo de vida das *slices*.**

diversos provedores e, assim, coletar as métricas. Os *Adaptors* realizam a tradução da solicitação do IMA para uma sintaxe que o provedor aceite e vice-versa. Além disso, no IMA, instâncias dos módulos de monitoramento são geradas para cada *slice*.

Tão logo as métricas sejam coletadas da infraestrutura que compõe a *slice*, os módulos de gerenciamento, tais como o de Segurança, o de Controle de Políticas, o Orquestrador de Serviços e o de Recursos, etc, consomem essas informações durante o gerenciamento das *slices* para que decisões possam ser tomadas com base nelas. Por exemplo, após analisar os dados de monitoramento, o Orquestrador de Recursos pode concluir que determinada *slice* necessita de aumento no número de CPUs. Neste caso, o Orquestrador executa a ampliação do recurso da *slice*.

Finalizada a instanciação, o IMA recebe do módulo de criação da *slice* uma descrição completa desta. Tal descrição contém não somente a topologia da *slice*, com toda a sua composição tecnológica, mas também com seus pontos de acesso, isto é, IPs e portas, para os *Virtual/WAN Infrastruture Managers* (VIMs/WIMs) que fazem parte da *slice*. Usando tais pontos, o IMA é capaz de monitorar todas as métricas disponíveis.

Os módulos de gerenciamento indicam ao IMA qual $Y$ a ser utilizado na seleção de características. Este conjunto $Y$ pode ser um conjunto de informações do serviço ou, até mesmo, uma métrica coletada na própria infraestrutura, cujo módulo de gerenciamento

tem relação/interesse. Além de fornecer o conjunto $Y$, os módulos podem especificar um valor $K$, que representa o tamanho do conjunto de métricas que deve ser produzido pela seleção de características. Existem técnicas de seleção nas quais o valor $K$ não precisa ser informado.

A Figura 2 ilustra o Orquestrador de Recursos solicitando a seleção de características ao módulo de monitoramento (IMA). Neste exemplo, o Orquestrador de Recursos envia um conjunto $Y$ ao módulo de monitoramento da *slice* (passo 1). Por sua vez, o módulo de monitoramento recupera todas as métricas $X_{Completo}$ de infraestrutura relativas à *slice* (passo 2). Na sequência, executa-se o algoritmo de seleção de características para construção de um conjunto reduzido de métricas $X_{Selecionado}$ (passo 3), instanciando o monitoramento destas características (passo 4) a serem fornecidas ao Orquestrador de Recursos (passo 5).

O fluxo ilustrado ocorre sempre que o módulo de monitoramento ainda não realizou a seleção de características, ou quando a renovação desse conjunto for requisitada. Nos demais casos, o sistema de monitoramento simplesmente retorna os dados coletados, sem necessidade de refazer a seleção.

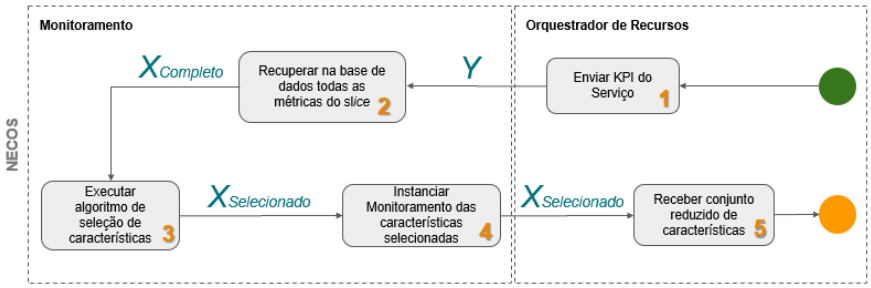



**Figura 2. Fluxo proposto do sistema inteligente de monitoramento considerando solicitações vindas do módulo Orquestrador de Recursos.**

Feita a seleção, o IMA passa a monitorar as métricas selecionadas mais frequentemente que as demais, o que significa que todas as métricas continuam a ser coletadas, porém em intervalos maiores. A frequência na coleta das métricas selecionadas $X_{Selecionado}$ respeita a solicitação do *Tenant*. A frequência de coleta para $X_{Completo}$ é um parâmetro a ser investigado neste trabalho. É preciso ter uma frequência de coleta que não prejudique a eficácia de seleção. Ao manter ambas as coletas, o presente trabalho permite que o conjunto $X_{Selecionado}$ possa ser renovado dinamicamente, o que pode ser necessário devido a mudanças as quais os serviços e *slices* estão sujeitos.

## 5. Avaliação

Os resultados apresentados nesta Seção foram coletados durante um experimento no qual um serviço de *Key-Value Store* foi monitorado. Para a construção do serviço, foram instanciadas 5 máquinas virtuais que hospedam um *cluster* do serviço Cassandra $v3.11.3$ [Apache 2016]. Além disso, outras duas máquinas virtuais complementam os experimentos: uma na qual um cliente do serviço Cassandra é executado, sendo esta utilizada para a coleta de métricas $Y$; e outra na qual um gerador de carga foi desenvolvido para simular uma comunidade dinâmica de clientes do Cassandra.

O conjunto de métricas $Y$ contém estatísticas relativas a ambas as operações de leitura e escrita no Cassandra que foram desempenhadas pelo cliente. Dentre as métricas temos, número de operações realizadas e tempos de resposta, incluindo, por exemplo, média e percentis dos tempos de resposta. Ao utilizarmos o gerador de carga, o objeto é conduzir o serviço do Cassandra através de diferentes níveis de carga. Como resultado, o protótipo desenvolvido consegue coletar as flutuações causadas nas métricas observadas no cliente que coleta $Y$, gerando um cenário ideal para rotular amostragens de $X$.

Para a coleta de dados da infraestrutura $X$ utilizou-se uma ferramenta denominada *Prometheus* em conjunto com um *exporter* de dados [Kochie 2018], que foi instalado em cada uma das $5$ máquinas que hospedam o serviço do Cassandra. Isso permitiu que métricas sobre os mais diversos aspectos fossem coletadas, incluindo métricas de uso de CPU, memória, tráfego em *bytes* na rede, etc. Para este experimento[2], o intervalo de coleta de tais métricas, tanto as do conjunto $X$ quanto as do conjunto $Y$, foi de um segundo. Relativamente ao conjunto $X$, $374$ métricas foram coletadas, desta forma, temos que o conjunto $X_{Completo}$ é formado por todas estas métricas coletadas a partir das $5$ máquinas onde o serviço do Cassandra é executado.

Toda a infraestrutura é virtualizada no OpenStack. Cada uma das $5$ máquinas virtuais com o Cassandra que estão envolvidas no teste possuem a seguinte configuração: 1 vCPU atribuído pelo OpenStack, 4GB de RAM e 50GB de disco. A máquina cliente para coleta do $Y$ possui: 1 vCPU atribuído pelo OpenStack, 2GB de RAM e 20GB de disco. A máquina do gerador de carga possui: 4 vCPUs atribuídos pelo OpenStack, 8GB de RAM e 20GB de disco. Todas estas máquinas virtuais rodam com sistema operacional Ubuntu Server 16.04. Os mecanismos de aprendizado de máquina para seleção de características foram executadas em um *notebook* com processador Intel i7 de sétima geração e 16GB de memória RAM, num ambiente com sistema operacional Ubuntu 18.04 LTS.

O experimento para a coleta dos conjuntos $X_{Completo}$ e $Y$ teve duração total de três horas. O gerador de carga, implementado utilizando a linguagem Python, segue uma distribuição de *Poisson* para a chegada de novos clientes, aplicando uma sinusoide para determinar $\lambda$. Iniciou-se o teste com $\lambda = 11$ clientes ativos ($10$ criados pelo gerador de carga, mais o cliente para coleta de $Y$). A sinusoide possui período de uma hora e amplitude de $5$ clientes, ou seja, o número de clientes ativos durante o período de uma hora oscila entre $16$ e $6$ clientes. Desta forma, o experimento coletou dados referentes a três sinusoides completas.

A Figura 3 apresenta o $95$ percentil do tempo de escrita na DHT observado no cliente onde o $Y$ é coletado, durante o experimento de três horas. É possível notar as oscilações causadas no tempo de resposta, reflexo do número de clientes ativos que foram controlados pelo gerador de carga.

O objetivo central deste artigo é avaliar a eficácia da seleção de características, aplicando diferentes valores de $K$ para a composição do conjunto $X_{Selecionado}$. Porém, para permitir uma melhor ilustração dos resultados da seleção de características, os experimentos aplicaram também um regressor para estimar o $95$ percentil dos tempos de resposta das operações de escrita efetuadas pelo cliente coletor de $Y$. Este seria, por exemplo, a tarefa de um módulo orquestrador que precisa avaliar se acordos de nível de

[2]Os dados do experimento podem ser acessados em: *https://github.com/gustavosm/traces-wslice-2019*

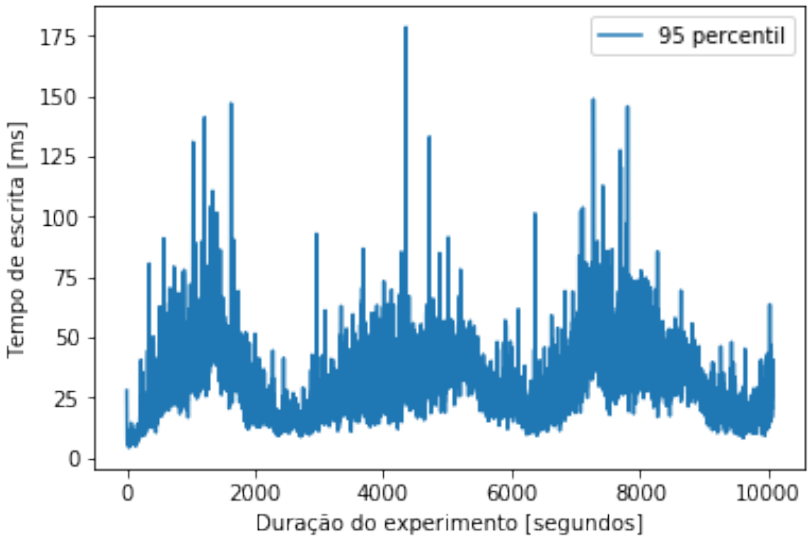



**Figura 3. 95 percentil do tempo de escrita na DHT ao longo de três horas de experimentação.**

serviço estão sendo honrados na *slice*.

O algoritmo de aprendizado de máquina utilizado para a regressão foi o *Random Forest*, utilizando-se a implementação oferecida no pacote de *machine learning* para Python denominado *scikit-learn* [INRIA 2018]. Os parâmetros de configuração foram: *max_depth* $= 10$, *random_state* $= 17$ e *n_estimators* $= 120$. Nesta avaliação, utilizou-se $70\%$ dos dados coletados para o treinamento do regressor e os outros $30\%$ para avaliar a acurácia deste. A acurácia é apresentada em termos do Erro Absoluto Médio Normalizado (*Normalized Mean Absolute Error - ($NMAE$)*), que é expressada por: $\frac{1}{\overline{y}}(\frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|)$, onde $\hat{y}_i$ é estimativa obtida pelo regressor, $y_i$ é o valor coletado no cliente, e $\overline{y}$ é a média dos valores $y_i$ do conjunto de teste, cujo tamanho é igual a $m$.

A Figura 4 apresenta duas séries de dados referentes ao valor amostrado de $Y$ (conjunto de teste com $30\%$ das amostras) pelo cliente e o valor estimado $\hat{Y}$ pelo regressor, a partir das métricas $X_{Completo}$ observadas na infraestrutura onde o serviço Cassandra é executado. A taxa de erro das estimativas feitas pelo regressor corresponde a $NMAE = 20,07\%$. Conforme ilustra a Figura 3, as estimativas conseguem seguir o comportamento dos valores amostrados, ou seja, a partir das métricas $X_{Completo}$, é possível estimar o comportamento do serviço Cassandra que está sendo entregue aos clientes.

Conforme mencionado anteriormente, o conjunto $X_{Completo}$ contém todas as métricas da infraestrutura, coletadas pelo *Prometheus*. O tamanho deste conjunto de dados varia conforme a composição da infraestrutura, podendo representar um volume pesado de dados a ser coletado. Além disso, o conjunto $X_{Completo}$, geralmente, possui métricas (ruídos) que podem degradar o comportamento, por exemplo, dos regressores. Outro fator importante é que, para cada serviço diferente ofertado em uma *slice*, a composição ideal de $X$ para representar o comportamento do serviço pode variar.

A Figura 5 apresenta o primeiro resultado do mecanismo inteligente de seleção de características desenvolvido neste trabalho. Os resultados apresentam a $NMAE$ obtida

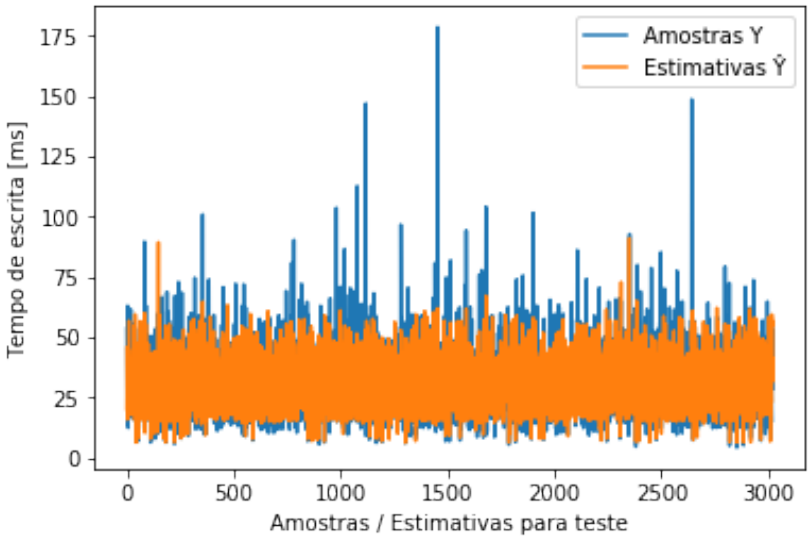



**Figura 4. Valores amostrados de $Y$ pelo cliente do Cassandra e estimativas $\hat{Y}$ obtidas pelo regressor *Random Forest*.**

para regressores treinados com diferentes conjuntos $X_{Selecionado}$, considerando os valores $K = \{50, 100, 150, 200\}$ para a seleção. A estratégia escolhida para seleção de características foi a *filter* e, portanto, as $K$ métricas representam as primeiras posições de um *ranking* que elege as métricas mais correlacionadas com o $Y$ considerado.

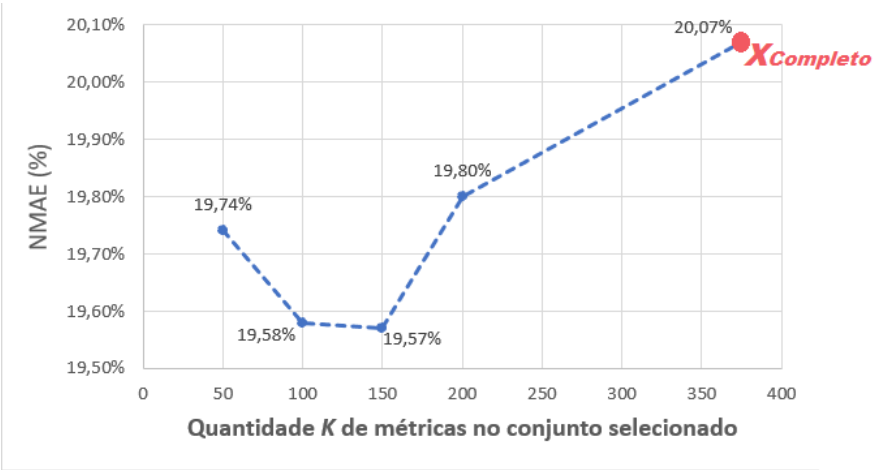



**Figura 5. $NMAE$ observada para os diferentes valores de $K$ utilizados para seleção de métricas que compõem o conjunto $X_{Selecionado}$.**

Como pode ser observado na Figura 5, os diferentes conjuntos $X_{Selecionado}$ são capazes de melhorar a acurácia do regressor, apresentando melhores valores de $NMAE$. O gráfico indica que, neste cenário de experimentação, valores para $K$ entre 100 e 150 são os mais indicados. Primeiramente, este resultado indica a eficácia da seleção de características, permitindo que um conjunto menor de métricas sejam coletadas a partir da infraestrutura, evitando a movimentação desnecessária de grandes volumes de dados.

Outro aspecto importante do resultado da Figura 5 refere-se a extração de ruído

presente no conjunto $X_{Completo}$. Vale destacar que o valor $K = 50$ apresentou $NMAE$ maior do que $K = 100$ e, conforme o *ranking*, as primeiras $50$ métricas são as mesmas em ambos os conjuntos $X_{Selecionado}$. Entretanto, além de eliminar ruído, o valor adequado para $K$ deve ser capaz de produzir um conjunto $X_{Selecionado}$ que represente adequadamente o comportamento de $Y$.

Finalmente, a redução no número de métricas permite o treinamento de regressores em um menor tempo. Por exemplo, considerando o conjunto $X_{Selecionado}$ para $K = 150$, o regressor do *random forest* foi treinado em $27.73$ segundos, enquanto que para o conjunto $X_{Selecionado}$ com $K = 100$ este tempo foi de $15.51$ segundos.

## 6. Conclusões e trabalhos futuros

O cenário de investigação do projeto NECOS é desafiador. Por se tratar de um sistema que deve ser altamente escalável, os tempos de resposta de cada módulo devem ser os menores possíveis. Desta forma, o módulo de monitoria possui um requisito de selecionar as características de maneira rápida, para que possibilite aos demais módulos maior velocidade na tomada de decisões. Como resultado da seleção de características, um volume menor de métricas passa a ser fornecido aos módulos, fator central para a escalabilidade de todo o sistema. Conforme os resultados apresentados, a seleção de características contribui, inclusive, para uma melhor estimativa das métricas $Y$. Entretanto, apesar de o cenário ser bastante desafiador, os primeiros testes são promissores e indicam a importância do sistema inteligente de monitoramento apresentado neste trabalho.

O conjunto de trabalhos futuros inclui: 1) experimentar outros serviços conforme indicado na Seção 2; 2) avaliar coletas de métricas $X_{Completo}$ em intervalos diferentes do atual (um segundo); 3) aplicar diferentes padrões de carga aos serviços, além do atual baseado na sinusoide; 4) utilizar outras ferramentas de coleta de dados, tais como *Ceilometer* [Ceilometer 2018] e *SAR* [SAR 2018]; e 5) aplicar outras metodologias de seleção de características, que não a *filter*, conforme discutido na Seção 3.

## Agradecimentos

Esta trabalho é financiado com recursos da 4º chamada colaborativa BR-EU no contexto do H2020, registrados no acordo 777067 (*NECOS - Novel Enablers for Cloud Slicing*), que é fomentado pelo Ministério da Ciência e Tecnologia no lado Brasileiro e pela Comissão Europeia de Tecnologia no lado Europeu.

## Referências

Apache (2016). Apache cassandra. `http://cassandra.apache.org/`. Acessado em: 17/03/2019.

Ceilometer (2018). Ceilometer. `https://docs.openstack.org/ceilometer/latest/`. Acessado em: 21/12/2018.

Chandrashekar, G. and Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28. 40th-year commemorative issue.

Encoding.com (2016). Mpeg-dash an overview. `https://www.encoding.com/mpeg-dash/`. Acessado em: 17/03/2019.

Fulmari1, A. and Chandak, M. B. (2013). A survey on supervised learning for word sense disambiguation. https://pdfs.semanticscholar.org/58bb/8f4b9a0e7257ca15555e505e9fd35992f66c.pdf. Acessado em: 17/03/2019.

INRIA (2018). Scikit learn. https://scikit-learn.org/stable/. Acessado em: 12/12/2018.

John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *ICML 1994*.

Kira, K. and A. Rendell, L. (1992). The feature selection problem: Traditional methods and a new algorithm. In *The Feature Selection Problem: Traditional Methods and a New Algorithm.*, pages 129–134.

Kochie, B. (2018). Prometheus node-exporter. https://github.com/prometheus/node_exporter/releases. Acessado em: 06/02/2019.

Kächele, S., Spann, C., Hauck, F. J., and Domaschka, J. (2013). Beyond iaas and paas: An extended cloud taxonomy for computation, storage and networking. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 75–82.

Necos (2017a). D3.1: Necos system architecture and platform specification. v1. http://www.maps.upc.edu/public/NECOS%20D3.1%20final.pdf. Acessado em: 12/03/2019.

Necos (2017b). D5.1: Architectural update, monitoring and control policies frameworks. http://www.maps.upc.edu/public/D5.1_final.pdf. Acessado em: 12/03/2019.

Necos (2017c). Motivation and vision. http://www.h2020-necos.eu/motivation-and-vision/. Acessado em: 04/12/2018.

Pasquini, R. and Stadler, R. (2017). Learning end-to-end application qos from openflow switch statistics. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–9.

SAR (2018). Sar. https://linux.die.net/man/1/sar. Acessado em: 21/12/2018.

Voldemort (2016). Voldemort Project. http://www.project-voldemort.com/voldemort/. Online; acessado em 20/03/2019.

Yang, K., Yoon, H., and Shahabi, C. (2005). A supervised feature subset selection technique for multivariate time series. In *In Proceedings of the Workshop on Feature Selection for Data Mining: Interfacing Machine Learning with Statistics, 92–101*.

Zhu, X. (2005). Semi-supervised learning literature survey. https://minds.wisconsin.edu/bitstream/handle/1793/60444/TR1530.pdf?sequence=1. Acessado em: 17/03/2019.

# Virtualização em Redes Terrestre-Satélite 5G

**Karine L. M. da Costa**[1], **Tibério Tavares Rezende**[1], **Antônio Marcos Alberti**[1]

[1]ICT Lab – Instituto Nacional de Telecomunicações (INATEL)
Caixa Postal 37.540-000 – Santa Rita do Sapucaí – MG – Brasil

{karine.costa;tiberio}@mtel.inatel.br, alberti@inatel.br

***Abstract.*** *In order to propose a hybrid terrestrial-satellite architecture for the 5G infrastructure in Brazil, it is sought to discover and understand the resources and tools used in the simulation or emulation of these network architectures for performance analysis purposes when evaluating various technological interoperability scenarios. Based on the main projects of the European Commission H2020 and standardization organizations of the 5G network, such as 3GPP, ETSI, ITU, among others, this article investigates how the standardization and implementation of 5G terrestrial network management and virtualization technologies are being applied. This article outlines usage scenarios and the tools used in their investigations.*

***Resumo.*** *No intuito de propor uma arquitetura híbrida terrestre-satélite para infraestrutura da rede 5G no Brasil, almeja-se descobrir e entender quais os recursos e ferramentas utilizados na simulação ou emulação dessas arquiteturas de rede para fins de análise de desempenho dos diversos cenários possíveis de interoperabilidade tecnológica. Fundamentado nos principais projetos da Comissão Europeia H2020 e órgãos padronizadores da rede 5G, tais como 3GPP (3rd Generation Partnership Project), ETSI (European Telecommunications Standards Institute), ITU (International Telecommunications Union), dentre outros, este artigo investiga como vêm sendo realizadas a padronização e implementação das tecnologias de virtualização e gerenciamento das redes 5G terrestre-satélite no mundo. Este artigo destaca cenários de uso e as ferramentas utilizadas em suas investigações.*

## 1. Introdução

O notável desenvolvimento do ambiente de redes e das tecnologias de comunicação levou a nossa sociedade a um passo mais próximo da comunicação onipresente, em que numerosos dispositivos, por exemplo, computadores pessoais, telefones celulares e equipamentos inteligentes, podem ser conectados à Internet a qualquer momento. Em um futuro próximo, espera-se que, além desses dispositivos, todas as outras coisas físicas do planeta também estejam ligadas à rede [T. Pasca 2017]. Um dos principais diferenciais do sistema de Internet móvel de quinta geração, conhecido como 5G, é sua capacidade de suportar as tecnologias necessárias para a implantação da Internet das Coisas (em inglês, *Internet of Things* - IoT) [Y. Kawamoto 2013]. Neste contexto, a tecnologia de comunicação via satélite deve ser exposta sempre como uma possibilidade chave, visando atender os mais diversos cenários de uso no 5G. Esse tipo de tecnologia desempenhará um papel importante ao lado das tecnologias terrestres existentes. O uso da comunicação satelital nas tecnologias geoestacionárias (GEO) e não geoestacionárias (NGEO) visando estabelecer

comunicação entre as redes de acesso, núcleo e de dados no 5G é uma oportunidade única da atual conjuntura de convergência tecnológica exponencial.

É amplamente reconhecido pelos pesquisadores dos sistemas 5G que nenhuma tecnologia única atenderá a todos os requisitos e que a maneira de cumprir os principais indicadores de desempenho 5G é considerando uma abordagem híbrida de rede. As comunicações via satélite evoluíram para fornecer conectividade contínua e alta taxa de dados com ampla cobertura e alta confiabilidade. Devido suas capacidades inerentes, as comunicações via satélite possuem um papel importante a desempenhar no 5G, o qual está sendo padronizado no 3GPP, com a intenção de ser implementado comercialmente até 2020. O objetivo do 5G é oferecer suporte a vários cenários de uso, como banda larga móvel aprimorada (*enhanced Mobile Broadband* - eMBB), comunicações ultra-confiáveis e de baixa latência (*Ultra Reliable Low Latency Communications* - URLLC) e comunicações massivas do tipo de máquina (*massive Machine Type Communications* - mMTC) [3GPP 2016b]. Nesses cenários, as comunicações via satélite serão usadas para ampliar a cobertura de redes terrestres e fornecer *backhaul* para redes 5G. Os satélites têm outras funções importantes no 5G, incluindo o descarregamento de redes terrestres e o fornecimento de comunicações de emergência confiáveis.

Historicamente, em gerações de redes móveis anteriores, a integração do *satcom* se baseava em soluções sob medida proprietárias, tanto no nível de rede móvel quanto na comunicação por satélite. Na maioria das vezes, os satélites de telecomunicações eram considerados independentemente das redes terrestres. Nos raros casos em que as soluções híbridas foram propostas, a rede de satélites foi usada principalmente para fornecer *backhaul* a algumas células individuais remotas e pouco acessíveis como uma rede de transporte simples, não flexível e muito cara. A visão do 5G em conjunto com satélite na oferta de serviços avançados, como virtualização, capacitação de cidades inteligentes, ambientes inteligentes e comunicação M2M (em inglês, *Machine-to-Machine*), podem suprir esta demanda [Naser Al-Falahy 2017]. As comunicações por satélite desempenham um papel significativo na implementação do 5G como uma solução complementar às redes terrestres devido às suas características de cobertura onipresente, *broadcast*, *multicast* e recuperação rápida em casos de emergência.

A adoção de tecnologias de virtualização para 5G, como SDN (do inglês, *Software Defined Network*) e NFV (do inglês, *Network Function Virtualization*), se mostram como um passo necessário na evolução dos sistemas de segmentos terrestres e de satélite, operando por exemplo, em *gateways* e terminais de satélite, dentre outros componentes de rede. Esta virtualização permitirá que os serviços de comunicações por satélite sejam entregues de uma forma mais flexível, ágil e eficaz em termos de custos do que é feito hoje, facilitando grandemente a integração e operação sem descontinuidades de redes híbridas. Estes fatores têm sido um dos principais motivadores deste trabalho. Assim sendo, este documento está organizado em 5 Seções. A Seção 2 apresenta as principais atividades de padronização e projetos referentes à comunicação via satélite no contexto 5G. Entendendo que a virtualização é um habilitador potencial para a implementação do 5G, a Seção 3 apresenta um *overview* das tecnologias de virtualização de rede, SDN e NFV, destacando o estado atual de suas padronizações e como vêm sendo empregadas no cenário terrestre-satélite 5G. Visando contribuir com a padronização 5G no Brasil, tem-se como objetivo principal, propor uma arquitetura híbrida terrestre-satélite para a rede

5G. Para isso, o desenvolvimento de um ambiente de testes para essas redes se faz necessário. Logo, a Seção 4 apresenta o ambiente de testes, cenários de usos e um *overview* das possíveis ferramentas que serão utilizadas no ambiente de testes SAT-5G em fase inicial de implementação no ICT Lab (*Information and Communications Technologies Laboratory*), laboratório do Instituto Nacional de Telecomunicações - Inatel. A Seção 5 apresenta as principais conclusões.

## 2. Comunicação via Satélite para Atendimento das Redes 5G

Pretende-se que a rede 5G seja uma rede heterogênea e que torne realidade o conceito de conectividade ubíqua. É possível presenciar um momento de transformação tecnológica no qual há muitos modelos de negócios disruptivos, onde empresas passam a ser plataformas aumentando significativamente a quantidade de dados disponíveis e acessados pelos usuários. O cenário IoT em que qualquer dispositivo pode ser conectado e controlado através da Internet, satélite e redes móveis, traz um aumento significativo na quantidade de dados e dispositivos interligados. A maior dificuldade deste cenário é alcançar a heterogeneidade das diversas redes que formam o universo 5G e futuramente IoT, pois estão surgindo diversas soluções para essa nova geração de conectividade, mas com diferentes protocolos. Sabe-se que a rede 5G não está sendo projetada para resolver um problema específico, mas sim para ter capacidade de lidar com uma ampla gama de cenários de uso [5G 2016]. Quatro cenários de aplicação têm sido alvo desta pesquisa:

- ***Wireless Regional Area Network* (WRAN):** ambiente que exige taxa e latência média da rede. Este cenário foi incluído visando atender países de grandes extensões geográficas como o Brasil.
- **Comunicação M2M:** aplicações que demandam taxas menores e mais tempo de vida da bateria como rede de sensores, IoT, comunicação veicular e *smart places*.
- **Comunicação com baixa latência:** cobre aplicações de Internet Tátil que exigem taxas razoáveis, baixíssima latência e alta confiabilidade [B. Zoran 2016]. Com o padrão de Internet Tátil, as aplicações são executadas em nuvem e o usuário utiliza equipamentos que não processam nenhum dado localmente. Trata-se de uma rede ultra-confiável com respostas ultra-rápidas, capaz de entregar experiências táteis remotamente. A abordagem com baixa latência é de grande interesse para a indústria, também por causa dos problemas de segurança de dados [Szymanski 2017].
- **Comunicação com alta vazão:** abrange cenários como multidões em eventos, ambientes fechados e áreas urbanas densas.

As redes 5G operando juntamente com satélites podem revolucionar a maneira como a Internet atual está construída. O padrão da Internet atual é formado por provedores de serviços de Internet (em inglês, *Internet Services Providers* - ISP) que estão interligados entre si, juntamente com grandes prestadores de serviços que criam redes menores para conectar os usuários finais como mostrado na Figura 1. Este padrão se mostra inviável para as realizações esperadas com a ubiquidade e com o advento da IoT [A. Siris 2016]. Existem desafios que precisarão ser vencidos, como a alta quantidade de informação que passará a trafegar na rede, uma vez que todo objeto físico terá seu correspondente virtual.

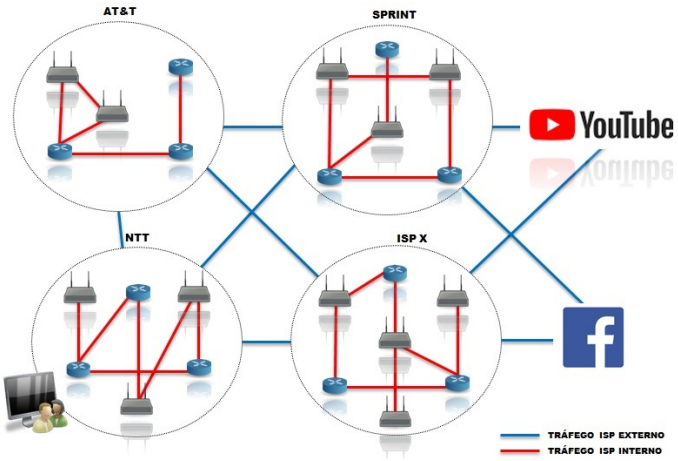



**Figura 1. Infraestutura de alto nível da Internet atual.**

Embora não tenha sido construído especificamente para IoT, o 5G aparece como um componente chave na camada de comunicação e conectividade da pilha de tecnologias IoT. Promete oferecer confiabilidade, latência, escalabilidade, segurança e mobilidade onipresente que serão exigidos em várias aplicações/serviços de missão crítica no ambiente IoT [Ejaz et al. 2016]. Sua característica de banda larga móvel aprimorada (em inglês, *enhanced Mobile Broadband* - eMBB) é vista como uma habilitadora da IoT massiva por oferecer capacidade e desempenho de rede em áreas carentes e não atendidas, especialmente em regiões com receita média por usuário baixa (em inglês, *Average Revenue per User* - ARPU) , nos mercados emergentes e em plataformas móveis de aplicações como embarcações e aeronaves [SaT5G 2018].

Os sistemas de comunicação por satélite são a única solução econômica para lidar com esses cenários. Para isso, é necessário que haja a integração perfeita da futura arquitetura 5G e a eficiência ideal seja obtida por meio de colaborações tecnológicas entre os sistemas 5G móveis e *satcom*. Atualmente, o sistema de comunicação via satélite está passando por uma nova fase de operação com várias renovações de constelação e configuração. O principal caso de uso não está mais construído apenas em torno da voz, como era com a utilização do 2G. A grande diferença com relação as implantações satelitais comerciais anteriores é que estas novas constelações suportam comunicação M2M com foco em IoT. Elas apostam em IoT como uma grande parte de sua estratégia de geração de receita. A tecnologia de satélite serve como um facilitador chave para esses novos serviços, em todas as indústrias, através de fronteiras geográficas para transformar a conectividade IoT como hoje é conhecida [M. De Sanctis 2016].

O principal projeto que trata de *satcom* para o 5G é o Sat5G [SaT5G 2018]. Este projeto define soluções de *backhaul* e de descarregamento de tráfego baseadas em satélite. O seu escopo cobre tecnologias 5G para obter o melhor valor dos recursos de satélites. Por exemplo, *multicast* para conteúdo e entrega de função de rede virtualizada e atenuação de latência. O projeto também cobre novos modelos de negócios e colaborações operacionais economicamente viáveis que integram as partes interessadas de satélite e terrestres.

Um dos principais objetivos da arquitetura 5G é um alto grau de flexibilidade para atender melhor às necessidades específicas de campos fora do setor de telecomunicações,

como por exemplo, indústria automotiva e operação de organizações de missão crítica. Para este tipo de operação a integração da rede 5G e satélite deverá ser clara e robusta, em particular em relação aos desempenhos de rede e segurança [G. Arfaoui 2018]. Requisitos como QoS de ponta a ponta para esse sistema integrado, segurança e privacidade, como também o uso de vários modelos de confiança, políticas e esquemas de autenticação devem ser explorados visando a rede híbrida. Uma área promissora é a dos sistemas de transporte inteligente (em inglês, *Intelligent Transport Systems* - ITS), em que a integração de rede 5G e satélite permitirá a implantação em larga escala de ITS, maximizando o desempenho da rede e proporcionando segurança e privacidade aos usuários apesar da grande quantidade de compartilhamento de dados pessoais neste sistema. O projeto 5G CHARISMA [M.C. Parker 2017] pode ser utilizado para aplicações ITS.

Os papéis e vantagens dos satélites em 5G foram estudados pelo 3GPP. A cobertura de satélite traz valores agregados em conjunto com as tecnologias de acesso para 5G, especialmente para aplicações onde a cobertura e a disponibilidade onipresentes são cruciais [S. Parkvall 2017]. O 3GPP agora está padronizando o uso de redes não-terrestres (em inglês, *Non-Terrestrial Networks* - NTN), isto é, satélites e outros, incluindo veículos aéreos não tripulados, de tal forma que redes de acesso múltiplo podem ser usadas para suportar redes 5G. O primeiro item do estudo NTN começou em fevereiro de 2018 e tem como foco a modelagem de canais para operação das tecnologias. Além da modelagem de canal, as discussões sobre o impacto de NTN do 5G *New Radio* (NR) [Jin Liu 2016] são realizadas para uma variedade de tópicos como: *Random Access Channel* (RACH), *Hybrid-ARQ* (HARQ), *Phase Tracking Reference Signal* (PTRS), *Demodulation Reference Signal* (DMRS), *Initial Access* e *Timing Advance*.

Com redes de experiência *gigabit* por segundo com latência controlada e os esperados bilhões de dispositivos conectados à Internet, é clara a necessidade de novas arquiteturas e padrões de sistemas de comunicação. O uso de cargas úteis comumente usadas em satélites pode ser reaproveitado em plataformas próximas à terra. Os serviços de *backhaul* 5G podem ser ativados usando essas plataformas próximas à terra que atendem aos requisitos de alta taxa de dados e latência exigidos em 5G. Essas plataformas apresentam vantagens exclusivas como implantação rápida de infraestrutura de banda larga de alta velocidade, cobertura digital em mercados carentes e não atendidos, densificação e implantação de tecnologias de ativação 4G, 5G e IoT, além da maximização da eficiência espectral que permite a oportunidade de melhorar o compartilhamento de espectro.

## 3. Virtualização da Rede Terrestre-Satélite 5G

A visão do 5G em conjunto com satélite na oferta de serviços avançados, como virtualização, capacitador de cidades inteligentes, ambientes inteligentes e M2M, podem suprir a elevada demanda no tráfego móvel [Naser Al-Falahy 2017]. A adoção de tecnologias de virtualização para 5G, como SDN e NFV, se mostram como passo necessário na evolução dos sistemas de segmentos terrestres e de satélite, operando por exemplo, em *gateways* e terminais de satélite. Esta virtualização permitirá que os serviços de comunicações por satélite sejam entregues de uma forma mais flexível, ágil e eficaz em termos de custos do que é feito hoje, facilitando a integração e operação sem descontinuidades de redes híbridas.

Em SDNs, o plano de controle é desacoplado do plano de dados e a comunicação

entre eles é realizada através de APIs (em inglês, *Application Programming Interface*) (por exemplo, OpenFlow). Em síntese, a arquitetura SDN é composta por três camadas: (1) plano de dados, (2) plano de controle / camada do controlador e (3) aplicação. O plano de dados, baseado em *hardware*, consiste em dispositivos de encaminhamento (por exemplo, roteadores e comutadores) sendo responsável somente pelo encaminhamento dos dados de acordo com as instruções do controlador. Atualmente, comutadores programáveis estão sendo desenvolvidos usando P4 (em inglês, *Programming Protocol-Independent Packet Processors*). Atuando como o cérebro, o controlador possui uma visão global e opera toda rede. As necessidades do cliente são abstraídas sobre a camada de aplicação que se comunicam com controlador através das APIs *Northbound* (por exemplo, API RESTful). Todos os aplicativos / programas são executados acima do controlador. Dentre os controladores mais usados estão: OpenDaylight, Floodlight, NoX / POX, etc. O controlador SDN define a regra para os fluxos de entrada do plano de dados. Camadas SDN se comunicam entre si por meio de APIs abertas chamadas API *Northbound Interface* (NI) e API *Southbound Interface* (SI).

Na tecnologia NFV, as funções de rede são implementadas em componentes de *software* conhecidas como funções de rede virtual (em inglês, *Virtual Network Functions* - VNFs). As VNFs são executadas em servidores comuns ou em infraestrutura de nuvem, ao invés de *hardwares* dedicados, reduzindo o CAPEX (*CAPital EXpenditure*) e otimizando o provisionamento de recursos. As VNFs podem não apenas ser vinculadas a um serviço e/ou aplicativo específico, mas também podem se relacionar aos requisitos de usuários específicos, políticas de rede e permitir diferenciação de serviços com base nos acordos de nível de serviço (em inglês, *Service Level Agreement* - SLAs) da assinatura de usuário [Abdelwahab et al. 2016]. A principal motivação é que, uma vez implementadas em *software*, essas VNFs podem ser executadas em *hardware* de uso geral, onde podem ser implantadas, configuradas, dimensionadas, migradas e atualizadas sob demanda. Espera-se que a operação das VNFs aconteça de acordo com a estrutura de gerenciamento e orquestração (em inglês, *Management and Orchestration* - MANO) da NFV definida pelo ETSI.

Analisando as especificações publicadas pelos órgãos padronizadores 3GPP, IETF e ETSI, nota-se que há uma cooperação mútua entre esses órgãos para delinear a utilização das técnicas de SDN e NFV nos sistemas 5G a fim de oferecer suporte à conectividade de dados e serviços. A especificação TR 32.842 [3GPP 2015] que define o gerenciamento das redes 3GPP, realizada na *Release* 13, é o primeiro trabalho em conjunto com o ETSI NFV. Nesta especificação, com base no *framework* arquitetural NFV-MANO do ETSI, estuda-se o gerenciamento do *core* de redes móveis 3GPP que incluem funções do *core* de rede virtualizadas. A Figura 2, apresenta o mapeamento do gerenciamento de rede entre 3GPP e ETSI NFV. Baseado nas definições do ETSI, o 3GPP define os elementos de rede (em inglês, *Network Element* - NE) que não são virtualizados (em inglês, *Physical Network Functions* - PNF) e aqueles que são virtualizados (VNF). O NFV MANO, conforme definido pelo ETSI NFV, gerencia a infraestrutura virtualizada, as funções de rede virtualizadas (VNF) e os serviços de rede (NS) que abrangem VNFs e PNFs. Na *Release* 14, o 3GPP expandiu essa abordagem com as especificações 3GPP TS 28.525, 526 e 527 referindo-se as interfaces Os-Ma-nfvo, Ve-Vnfm-em e Ve-Vnfm-vnf. Atualmente, o ETSI NFV está expandindo seu trabalho no ETSI NFV *Release* 3 para explorar e abordar alguns dos novos casos e requisitos do 5G.

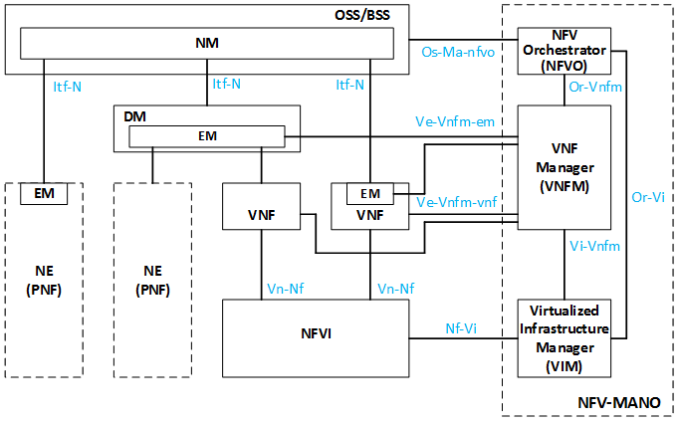



**Figura 2. Mapeamento da relação de gerenciamento de rede mista entre o *framework* arquitetural 3GPP e ETSI NFV-MANO, adaptado de 3GPP TR 32.842 [3GPP 2016a].**

Não há especificação ou padronização que engloba a integração SDN/NFV *satcom* e o 5G entre os órgãos padronizadores visitados. No entanto, Ferrús *et al.* [R. Ferrús 2015] descrevem uma arquitetura para sistemas de segmentos terrestres-satélite aptos à SDN/NFV para integração do componente de satélite em sistemas 5G. A Figura 3 refere-se ao modelo de referência de arquitetura para uma rede terrestre-satélite integrada com funções de federação fornecidas pelo 3GPP. Referindo-se ao domínio do operador de rede de satélites (em inglês, *Satellite Network Operator* - SNO), a parte inferior do modelo é responsável pela gerência dos recursos das infraestruturas de virtualização de função de rede (em inglês, *Network Function Virtualization Infrastructure* - NFVI) e da conectividade entre NFVIs.

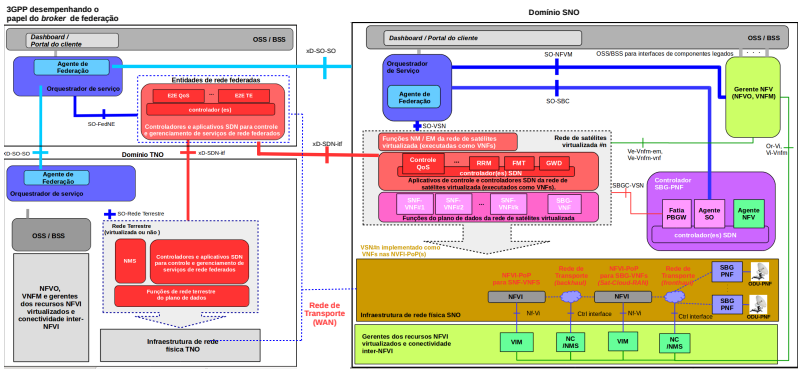



**Figura 3. Modelo de referência de arquitetura para uma rede terrestre-satélite integrada com funções de federação fornecidas pelo 3GPP, adaptado de [R. Ferrús 2015].**

A infraestrutura de rede física é composta pelos seguintes elementos:

- NFVI-PoPs (NFVI-*Points of Presence*) para SNF-VNFs (*Satellite Network Function*-VNFs): fornecendo principalmente recursos de rede, processamento e armazenamento, as NFVI-PoPs executam as SNF-VNFs. Essas infraestruturas não estarão localizadas necessariamente próximas ao *hub* do satélite.

- NFVI-PoP (s) para SBG-VNFs (*Satellite Baseband Gateway*-VNFs): essa infraestrutura é responsável pela implementação das funções dos *gateways* de banda base dos satélites. Devido as limitações de distância que podem ser impostas pela rede de *fronthaul* em termos de latência e largura de banda entre SBG-VNFs e SBG-PNFs, provavelmente essas infraestruturas estarão localizadas nas instalações do *hub* de satélite ou próximas delas.
- SBG-PNFs: conectado diretamente a uma ODU (*Out-Door Unit*), cada SBG hospeda a parte não virtualizada do *gateway* de satélite.
- Redes de transporte: a rede *backhaul* localiza-se entre as NFVI-PoPs e a *fronthaul* entre as NFVI-PoPs, onde as SBG-VNFs são executadas, e o local que hospeda as SBG-PNFs.

Conforme ilustra a Figura 3, uma ou múltiplas redes de satélite virtualizadas (em inglês, *Virtualized Satellite Network* - VSN) podem ser estabelecidas. A operação de uma VSN pode ser executada pelo próprio operador de redes de satélite (em inglês, *Satellite Network Operator* - SNO) ou por outra empresa que exercerá, desse modo, o papel do operador de redes de satélite virtualizadas. Uma VSN é composta pelo plano de dados, pelos controladores e aplicativos de controle e pelos gerenciadores de rede e de elementos. O plano de dados pode ser constituído por uma ou várias SNF-VNFs e SBG-VNFs. Formado por controladores SDN e aplicativos de controle, o plano de controle interage com as VNFs do plano de dados mediante diferentes protocolos. São inúmeras as aplicações de controle que podem ser implementadas, das quais destacam-se: gerenciamento de recursos de rádio (em inglês, *Radio Resource Management* - RRM), técnicas para mitigação da atenuação (em inglês, *Fade Mitigation Techniques* - FMT), diversidade de *gateway* (em inglês, *Gateway Diversity* - GWD) e controle de QoS. Localizados acima do plano de controle, as funções de NM e EM, podem ser providas como VNFs ou máquinas virtuais (em inglês, *Virtual Machines* - VMs), fornecendo ao operador de uma determinada VSN um pacote de funções para o gerenciamento da rede de satélites (por exemplo, gerenciamento FCAPS (em inglês, *Fault*, *Configuration*, *Accounting*, *Performance and Security*)).

## 4. Rede Terrestre-Satélite 5G Virtualizada - *Hands on*

Pretende-se desenvolver um ambiente de testes ICT SAT-5G no ICT Lab. do Inatel, conforme demonstra a Figura 7, de modo que a equipe de pesquisa possa realizar testes de diferentes configurações da arquitetura terrestre-satélite 5G com a finalidade de encontrar e escolher as melhores ferramentas para modelar as redes satelitais nas concepções GEO e NGEO. Visando contribuir com a padronização 5G no Brasil, tem-se como objetivo principal de pesquisa, propor uma arquitetura híbrida terrestre-satélite para a rede 5G. Para isso, o desenvolvimento de um ambiente de testes para essas redes se faz necessário. Logo, baseado em resultados obtidos em trabalhos anteriores, esta Seção apresenta a configuração para a implementação inicial do ambiente de testes e um *overview* das possíveis ferramentas que serão utilizadas para investigar os seguintes cenários de uso:

- **Cenário de uso 1 - Entroncamento IoT:** constituído por uma aplicação IoT implementada em uma Escola Rural de Santa Rita do Sapucaí, este cenário refere-se a uma horta inteligente conectada à Internet, que realiza o controle e gerenciamento do processo de irrigação de seus canteiros. A ideia é demonstrar a vi-

abilidade da oferta de serviços IoT-5G em áreas rurais utilizando o satélite no entroncamento das rede, conforme ilustra a Figura 4.

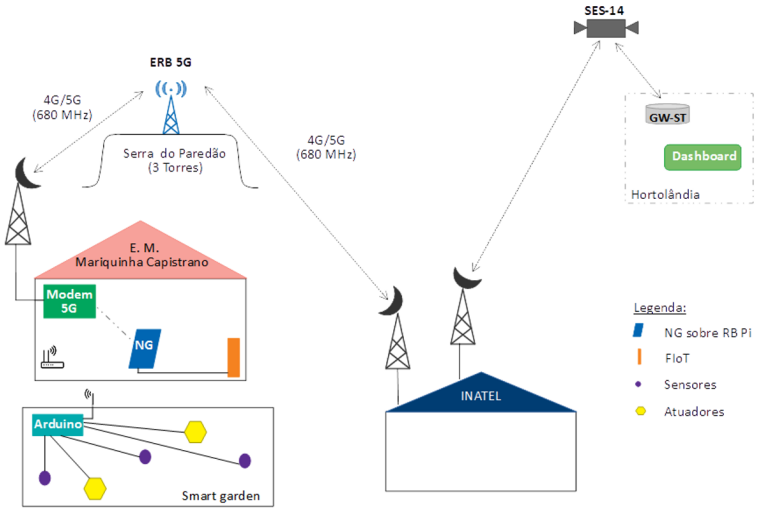



**Figura 4. Aplicação IoT-5G rural utilizando o satélite no entroncamento das redes.**

- **Cenário de uso 2 - Teste visando distribuição de conteúdo (*Content Delivery Network* - CDN e *Integrated Delivery Network* - IDN) considerando *Caching*:** utilizar *caches* e outras ferramentas *open source* para rede híbrida 5G. Avaliar o suporte ao gerenciamento de *cache* (e *streaming* adaptativo, se possível) para distribuição de conteúdo. Trabalhos futuros visam testes de *cache* com arquitetura de Internet do Futuro *NovaGenesis*, uma rede ICN (em inglês, *Information Centric Networking*), através do enlace de satélite até o ponto final da rede para comparações.

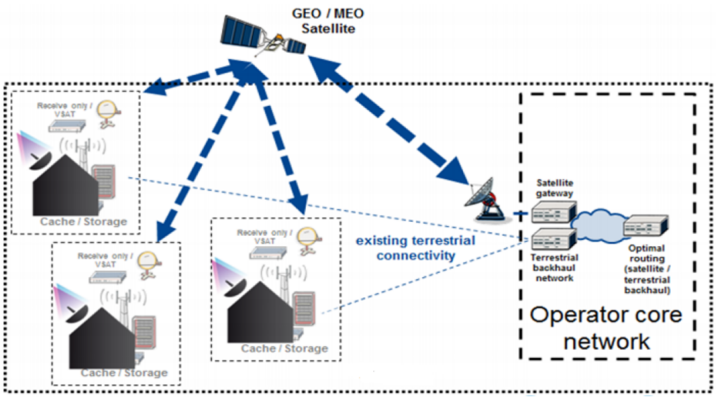



**Figura 5. Cenário para testes visando distribuição de conteúdo (CDN) considerando *caching*. Fonte: *Use cases and scenarios of 5G integrated satellite-terrestrial networks for enhanced mobile broadband* [Konstantinos Liolis 2018].**

- **Cenário de uso 3 - Virtualização de funções de rede e compatibilidade com redes definidas por *software* (SDN):** uma rede terrestre-satélite integrada definida

por *software* pode ser divida em três planos: dados, controle e gerenciamento, conforme ilustra a Figura 6. Responsável pelo encaminhamento dos dados, o plano de dados compõe a camada de infraestrutura e é constituído pelos dispositivos terrestres e satélites, e *switches* Openflow. Os controladores SDN fazem parte do plano de controle, responsáveis pelo gerenciamento e controle da rede SDN. O plano de gerenciamento é composto pelas funções de rede, sendo elas virtuais ou físicas. O objetivo é inserir o satélite no enlace entre controlador e controlado, avaliando o desempenho em termos de atraso, perda de pacotes, etc.

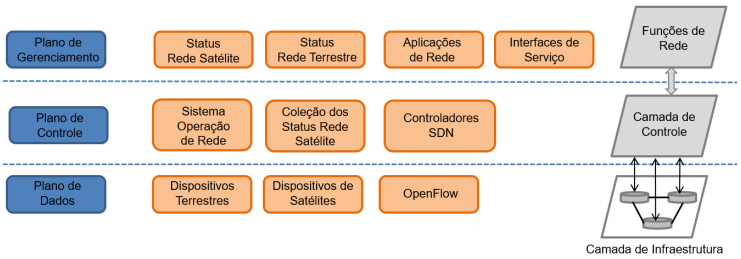



**Figura 6. Estrutura funcional de uma rede terrestre-satélite integrada definida por *software*.**

A Figura 7 apresenta a configuração inicialmente desenvolvida para a construção do ambiente de testes ICT SAT-5G, incluindo as plataformas e ferramentas mencionadas abaixo. As ferramentas serão utilizadas nos cenários apresentados para realização de testes, tais como: conectividade IoT e 5G, distribuição de conteúdo de maneira mais eficiente com o uso de satélites e criação de redes virtuais orquestradas por SDN minimizando a latência da rede. Este trabalho apresenta as principais plataformas e ferramentas selecionadas, que vêm sendo integradas na construção do cenário híbrido 5G terrestre-satélite utilizando as tecnologias SDN, NFV, CDN, IDN e ICN:

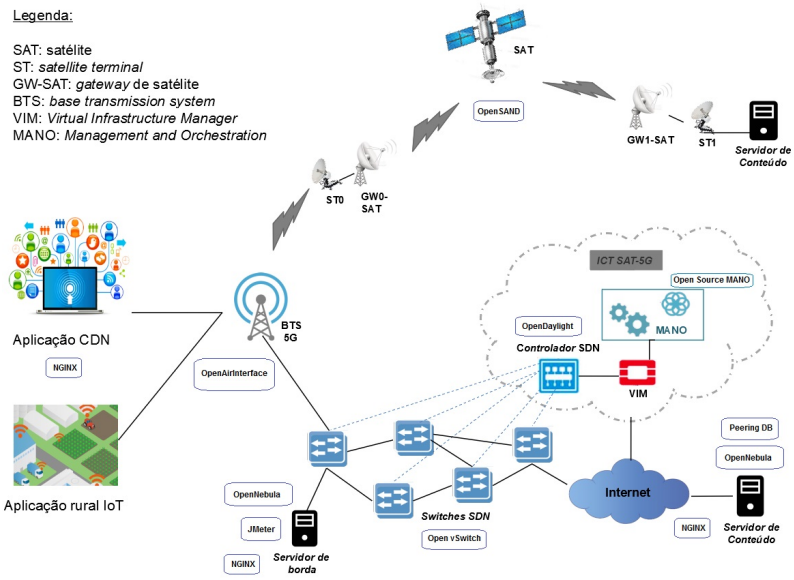



**Figura 7. Configuração do ambiente de testes ICT SAT-5G para análises e simulações de arquiteturas de redes terrestre-satélite 5G.**

- **OpenSAND:** ferramenta mais empregada na virtualização dos componentes de uma rede de satélite, o OpenSAND é uma plataforma *open source* inspirada nos padrões DVB-S2 / RCS2, capaz de emular as principais características de um sistema *satcom*. Suporta conectividade IPv4, IPv6 e Ethernet e pode ser interconectada com equipamentos reais e outras redes baseadas em IP (terrestre e/ou satélite), ou até mesmo com o *backbone* da Internet. Além de fornecer ferramentas de configuração e monitoramento (em tempo real e *off-line*), permitindo avaliar o desempenho dos cenários emulados.
- **OpenDaylight:** é uma plataforma SDN. Suporta os principais protocolos utilizados em SDN: OpenFlow, OVSDB, NETCONF, BGP e outros. Com uma estrutura modular, permite que a seleção dos recursos que serão utilizados seja feita pelo usuário, possibilitando a criação de um controlador personalizado.
- ***Open Source* MANO (OSM):** baseado no *framework* definido pelo ETSI para o gerenciamento e orquestração de todos os recursos no data center em nuvem, o OSM é uma comunidade *open source* que fornece uma pilha MANO para NFV. Esta ferramenta é independente de VIM (do inglês, *Virtualized Infrastructure Manager*) e adequada a todas as funções de rede virtualizadas.
- **Open vSwitch:** é uma ferramenta utilizada para a criação de *switches* virtuais [OvS 2019]. Através desta ferramenta é possível criar múltiplos *switches* para a automação da rede através de programação. Nos *switches* criados com o Open vSwitch é possível implementar diversos protocolos, entre eles o *OpenFlow*, e também definir um controlador como o POX (plataforma de desenvolvimento de *software* livre para um controlador SDN baseado em Python), o NOX (sistema operacional para a rede), ou o próprio controlador nativo do Open vSwitch [OvS 2019]. O Open vSwitch permite a criação de *switches* virtuais em máquinas físicas diferentes, similar ao *software* da VMware. Desta forma é possível criar diversas redes virtuais para testes.
- **OpenAirInterface:** A missão da OpenAirInterface *software* Alliance (OSA) é fornecer *software* e ferramentas para pesquisa e desenvolvimento de produtos sem fio 5G [OpenAirInterface 2019]. A atual geração de hardware e *software* para a rede de acesso por rádio (RAN) consiste em um grande número de elementos proprietários que sufocam a inovação e aumentam o custo para as operadoras implantarem novos serviços em redes celulares em constante mudança. *software* de código aberto executado em processadores de uso geral (como x86, ARM) pode simplificar muito o acesso à rede, reduzir custos, aumentar a flexibilidade, aumentar a velocidade de inovação e acelerar o lançamento de novos serviços. A implementação em código aberto de pilha totalmente em tempo real (eNB, UE e rede principal) em processadores de propósito geral quando combinados com SDN, NFV e OpenStack e trazem eficiência significativa no design de RAN tanto de inovação quanto de custo.
- **OpenNebula:** O OpenNebula é uma solução de código aberto simples, porém poderosa e flexível, para criar Nuvens Privadas e gerenciar a virtualização de Data Centers [OpenNebula 2019]. O OpenNebula fornece recursos nas duas principais camadas de virtualização de data center e infraestrutura de nuvem. O Gerenciamento de Virtualização de Data Center integra-se diretamente a hipervisores (como o KVM) e tem controle total sobre recursos físicos e virtuais, fornecendo recursos avançados para gerenciamento de capacidade, otimização de recursos,

alta disponibilidade e continuidade de negócios. O Gerenciamento de Nuvem para fornecer uma camada de provisionamento multilocatária semelhante a uma nuvem sobre uma solução de gerenciamento de infraestrutura existente (como o VMware vCenter).

- **JMeter:** JMeter é uma ferramenta *opensource* utilizada para testes de carga. Esta ferramenta é parte do projeto Jakarta da Apache *software* Foundation [JMeter 2019]. O JMeter foi desenvolvido para criar e executar testes de carga em serviços computacionais. Neste projeto de rede terrestre-satélite 5G virtualizada, pode ser usado para testar o desempenho em recursos estáticos e dinâmicos, e também aplicativos dinâmicos da *Web*. Pode ser usado para simular cargas em um servidor, grupo de servidores e rede ou para analisar o desempenho geral sob diferentes tipos de utilizações de carga [JMeter 2019].
- **Peering DB:** é um projeto que tem o objetivo de facilitar a troca de informações relacionadas a *peering* (acordos de troca de tráfego) e conta com o apoio dos principais ASNs e IXPs (*Internet Exchange Points*) do mundo. Trata-se de um registro de organizações e Sistemas Autonômos em banco de dados com informações dos locais em que fazem troca de tráfego, e qual a política de *peering* que possuem. Facilidades para o usuário: Gestão mais simples com uma única conta para *network*, *facility* e *exchange*; Um único usuário pode gerenciar múltiplas organizações; Possibilidade de gerenciamento de usuários e atribuição de níveis de acesso.
- **NGINX:** é uma ferramente utilizada como servidor HTTP, IMAP, POP3, Proxy e também para criação de redes CDN. Em termos práticos, pode ser descrito como um *software* que processa as solicitações dos usuários da rede garantindo que a troca de informações seja efetuada. Nginx funciona com o método EDA (*Event-driven Architecture*), que é preparado para lidar com uma maior demanda de conexões se comparado ao modo convencional, como é o caso do Apache, no qual o processamento é baseado em um único segmento para cada usuário. No Nginx, todo o caminho operado pela conexão de rede é *non-blocking*. O ponto a destacar é que este servidor se mostra eficiente por consumir poucos recursos da rede. Algumas interfaces de *socket* são bloqueadas em determinadas situações, enquanto outras retornam o resultado de imediato [NGINX 2019].

Diversas ferramentas e plataformas de virtualização e gerenciamento de redes existentes no mercado têm viabilizado o desenvolvimento de diferentes soluções no que refere-se a arquitetura híbrida terrestre-satélite. A ferramenta OpenSAND é utilizada para simulação da rede de satélite real, e as configurações SDN operam através da OpenDaylight. A orquestração dos recursos da rede é realizada pela OSM. Essa rede híbrida pode conter vários *switches* criados pela ferramenta Open vSwitch. Já, a OpenAirInterface desenvolve códigos abertos para controle de *Radio Access Network* (RAN). As ferramentas voltadas para distribuição de conteúdo com *cache* de rede são: NGINX para criação de uma rede CDN *cloud* e *edge* operando com o OpenNebula (gerenciador de nuvens privadas ou públicas), a JMeter e PeeringDB para testes de carga da rede e controle de informações de vários usuários diferentes.

## 5. Conclusão

Percebe-se que o 5G está sendo construído a partir da convergência terrestre/satélite. Isso significa que não existe mais 5G ou satélite, mas sim 5G operando através da rede via

satélite. A arquitetura híbrida é a única que faz sentido no contexto da evolução tecnológica. Para tanto, a padronização do 5G deve contemplar a arquitetura híbrida na sua totalidade. A interoperabilidade deve ser garantida via padrões amplos e que integrem o enlace satelital ao 5G, sem amarras ou remendos. Diversas ferramentas e plataformas de virtualização e gerenciamento de redes existentes no mercado têm viabilizado o desenvolvimento de diferentes soluções no que refere-se a arquitetura híbrida terrestre-satélite. Desse modo, este artigo destaca as principais ferramentas que podem ser integradas para realizar cenários de avaliação híbrido 5G terrestre-satélite. Atualmente o cenário de telecomunicações é de transição, com mais perguntas do que respostas. O 5G é uma oportunidade única para a convergência terrestre/satélite e para a evolução das comunicações globais. A virtualização das funções da rede para *broadcast* faz sentido pela otimização da rede, pela visibilidade da rede em relação ao gerenciamento e pela orquestração de forma eficiente. Sendo assim, como próximo trabalho, pretende-se explorar as ferramentas apresentadas neste artigo na implementação de uma rede terrestre-satélite 5G definida por *software* para distribuição de conteúdo.

## Agradecimentos

Este trabalho foi parcialmente financiado pelo Sindicato Nacional das Empresas de Telecomunicações por Satélite (SINDISAT). Também foi parcialmente financiado pela RNP, com recursos do MCTIC, processo No 01250.075413/2018-04, sob o projeto Centro de Referência em Radiocomunicações (CRR) do Instituto Nacional de Telecomunicações – Inatel, Brasil. Os autores agradecem também a FAPEMIG, CNPq e CAPES.

## Referências

3GPP (2015). Telecommunication management; study on network management of virtualized networks. In *Release 13*.

3GPP (2016a). Tr 32.842 study on network management of virtualized networks. In *3GPP*.

3GPP (2016b). Tr 38.913 study on scenarios and requirements for next generation access technologies. In *Release 14*. [Online: 12/12/2015] Disponível em: http://www.3gpp.org/DynaReport/38913.htm Acesso em: 19/07/2018 às 13:40.

5G, A. (2016). 5g americas white papers. *http://www.5gamericas.org/pt-br/resources/white-papers/*, Novembro.

A. Siris, Y. Thomas, G. P. (2016). Supporting the iot over integrated satellite-terrestrial networks using information-centric networking. In *8th IFIP Int. Conf. New Tech. Mobi. Secur. NTMS 2016*, volume 16, pages 03–07.

Abdelwahab, S., Hamdaoui, B., Guizani, M., and Znati, T. (2016). Network function virtualization in 5g. *IEEE Communications Magazine*, 54(4):84–91.

B. Zoran, B. Bakmaz, M. B. (2016). Vision and enabling technologies of tactile internet realization. In *Adv. Tech., Sys. and Ser. in Teleco. (TELSIKS)*, volume 16, pages 01–07.

Ejaz, W., Anpalagan, A., Imran, M. A., Jo, M., Naeem, M., Qaisar, S. B., and Wang, W. (2016). Internet of things (iot) in 5g wireless communications. *IEEE Access*, 4:10310–10314.

G. Arfaoui, P. Bisson, R. B. R. B. H. E. E. F. F. K. P. K. N. M. N. P. O. J. P. J. S. M. S. J.-P. W. e. A. Z. (2018). A security architecture for 5g networks. In *IEEE Access*, volume 06, pages 22466–22479.

Jin Liu, S. Sharma, K. S. (2016). Initial access, mobility, and user-centric multi-beam operation in 5g new radio. In *IEEE Communications Magazine*, volume 18, pages 68–84.

JMeter, A. (2019). In *JMeter*. [Online]: Disponível em: https://jmeter.apache.org. Acesso em: 26/03/2019 às 11:21.

Konstantinos Liolis, Alexander Geurtz, R. S. N. C. (2018). Use cases and scenarios of 5g integrated satellite-terrestrial networks for enhanced mobile broadband: The sat5g approach. *International Journal of Satellite Communications and Networking*, Maio.

M. De Sanctis, E. Cianca, G. A. I. B. R. P. (2016). Satellite communications supporting internet of remote things. In *IEEE Int. of Th. J.*, volume 03, pages 113–123.

M.C. Parker, G. Koczian, S. W. K. H. V. J. T. R. I. N. M. S. P. K. e. P. J. (2017). Ultra-low latency 5g charisma architecture for secure intelligent transportation verticals. In *2017 19th International Conference on Transparent Optical Networks (ICTON)*, volume 06.

Naser Al-Falahy, O. Y. A. (2017). Technologies for 5g networks: Challenges and opportunities. In *IEEE IT Professional*, volume 19.

NGINX (2019). Nginx. [Online]: Disponível em: https://www.nginx.com/resources/wiki/. Acesso em: 29/03/2019 às 14:20.

OpenAirInterface, S. A. (2019). In *Open Air Interface*. [Online]: Disponível em: https://www.openairinterface.org/. Acesso em: 28/03/2019 às 17:07.

OpenNebula, P. (2019). In *OpenNebula*. [Online]: Disponível em: https://opennebula.org/key-features/. Acesso em: 01/04/2019 às 13:45.

OvS, L. F. (2019). In *OpenvSwitch*. [Online]: 01/11/2017 Disponível em: https://www.openvswitch.org/. Acesso em: 25/03/2019 às 16:44.

R. Ferrús, H. Koumaras, O. S. e. a. (2015). Sdn/nfv-enabled satellite communications: Ground segment network architecture for 5g integration. *Physical Communication*.

S. Parkvall, E. Dahlman, A. F. M. F. (2017). Nr: The new 5g radio access technology. In *IEEE Communications Standards Magazine*, volume 1, pages 68–84.

SaT5G, H. (2018). Sat5g project h2020. *http://sat5g-project.eu*, Junho.

Szymanski, T. (2017). Strengthening security and privacy in an ultra-dense green 5g radio access network for the industrial and tactile internet of things. *Wire. Commun. and Mob. Comp. Conf. (IWCMC)*, page 01–08.

T. Pasca, S. Dama, V. S. K. K. (2017). A feasible cellular internet of things. In *IEEE Consum. Elect. Mag.*, volume 16, pages 66–72.

Y. Kawamoto, H. Nishiyama, Z. M. F. (2013). Effective data collection via satellite-routed sensor system (srss) to realize global-scaled internet of things. In *IEEE Sens. J.*, volume 13, page 3645–3654.

Realização

SBC Sociedade Brasileira de Computação

LARC

Organização

.inf INSTITUTO DE INFORMÁTICA UFRGS

Apoio Institucional

RNP 30 anos

Patrocínio

Diamante

CAPES

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico

cgi.br Comitê Gestor da Internet no Brasil

nic.br Núcleo de Informação e Coordenação do Ponto BR

BEDU TECH

Ouro

SAP

Prata

HUAWEI

DATACOM

acm sigcomm

@dyLNET TELECOM

BRDigital Telecomunicações

Bronze

FAPERGS

Google