

Uma arquitetura para provisão de fatias de rede fim-a-fim em ambientes multi-inquilinos

Elton V. Dias¹, Lafaeit C. Silva¹, Murillo S. Nunes¹, Francielly S. Almeida¹,
Leandro A. Freitas², Sand L. Correa¹, Kleber V. Cardoso¹

¹Instituto de Informática – Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia
74.690-900 – Goiânia – GO – Brasil

²Instituto Federal de Goiás
Núcleo de Estudos aplicados a Redes de computadores e
Sistemas distribuídos (NumbERS)
Avenida Universitária, s/nº, Vale das Goiabeiras
75.400-000 – Inhumas – GO – Brasil

{eltondias, murillonunes, franciellyalmeida, sand, kleber}@inf.ufg.br

lafaietsilva@gmail.com, leandro.freitas@ifg.edu.br

Abstract. *In recent years, the world of telecommunications has experienced progress in the development of mobile communications technologies. Network slicing has been identified as the backbone of next-generation mobile networks, thanks to its rapid evolution. However, there are still a few deployed architectures that allow you to apply the concept of network slicing. The NECOS project defines and implements a Slice as a Service architecture for multi-domain scenarios where the main idea is to exercise the concept of network slicing as a complete solution for provisioning end-to-end services among federated providers. Therefore, the aims of this paper is to present the NECOS architecture for providing end-to-end slices, focusing on two components, DC and WAN Slice Controller; responsible for allocating computational resources, and network on infrastructure providers. Moreover, through the results obtained, the paper shows the viability of the architecture for resource slicing.*

Resumo. *Nos últimos anos, o mundo das telecomunicações experimentou progressos no desenvolvimento de tecnologias de comunicações móveis. O fatiamento de recursos em rede foi identificado como a espinha dorsal das redes móveis de próxima geração, graças à sua rápida evolução. No entanto, ainda existem algumas arquiteturas que permitem aplicar o conceito de fatiamento de recursos na nuvem. O projeto NECOS define e implementa uma arquitetura Slice as a Service para cenários de vários domínios, onde a ideia principal é exercitar o conceito de fatia de rede na nuvem como uma solução completa para provisionar serviços de ponta a ponta entre provedores federados. Portanto, o objetivo deste artigo é apresentar a arquitetura NECOS para provimento de fatias de recursos fim a fim, concentrando-se em dois componentes, o DC e o WAN Slice Controller, responsáveis pela alocação de recursos computacionais e de rede nos provedores de infraestrutura. Através dos resultados obtidos, o artigo mostra a viabilidade da arquitetura para fatiamento de recursos.*

1. Introdução

Com a proliferação das tecnologias de Computação em Nuvem, Virtualização e Centros de Dados (*Data Center* - DC), uma grande variedade de serviços em nuvem estão sendo fornecidos em diversas modalidades, incluindo software, plataforma ou infraestrutura [Afolabi et al. 2018]. Essa grande variedade de serviços cria novos desafios aos provedores de infraestrutura, os quais têm usado diferentes meios para gerenciar os recursos de computação, armazenamento e conectividade.

Atualmente, há um grande esforço em todo mundo dedicado à pesquisas de redes 5G [Zhang et al. 2014]. De maneira geral, essas pesquisas têm por objetivo permitir a conexão de maneira ubíqua e transparente de dispositivos conectados à Internet [Al-Fuqaha et al. 2015a], contando com avanços de áreas como Internet das Coisas (*Internet of Things* - IoT) [Al-Fuqaha et al. 2015b]. Aplicações avançadas como realidade virtual e aumentada, tele-medicina, cidades inteligentes e carros autônomos, são exemplos de serviços cuja concretização depende da próxima geração de redes móveis (redes 5G) [Zhang et al. 2014]. Um dos desafios para as redes 5G é garantir o suporte adequado a essa grande variedade de casos de uso, cada qual com seus próprios requisitos (e.g. taxas de transmissão de dados e atraso) [Ordonez-Lucena et al. 2017].

Para provisionar diferentes casos de uso sobre uma mesma infraestrutura de informação e comunicação (*Information and Communication Technologies* - ICT), provedores de infraestrutura terão de buscar novos modelos de negócio baseados em federação de recursos computacionais [Iwamura 2015]. Esses modelos, permitirão que provedores de serviço instanciem aplicações de forma distribuída geograficamente e sobre diferentes domínios tecnológicos, seja em um único domínio administrativo ou envolvendo outros.

O fatiamento de rede (*network slicing*) é uma das tecnologias que vem recebendo grande destaque da indústria e da academia [Foukas et al. 2017]. Esse conceito foi introduzido pelo *Third Generation Partnership Project* (3GPP) [thi 2016] e consiste em um mecanismo que permite instanciar fatias de recursos de rede sob demanda, criadas para atender a especificação de uma aplicação ou serviço em particular. Cada fatia de rede (*network slice*) é composta por um conjunto de funções de rede físicas e/ou virtuais, instanciadas para atender os requisitos de um caso de uso específico [thi 2017]. Além disso, essas fatias de rede possuem seus próprios planos de gerenciamento de dados e de controle, e orquestração, embora estejam instanciadas sobre a mesma infraestrutura.

Apesar de todos os benefícios apresentados, criar uma fatia de rede fim a fim sobre uma infraestrutura que envolve múltiplos domínios administrativos e tecnológicos pode trazer uma série de obstáculos [Foukas et al. 2017]. Como exemplo, podemos citar a heterogeneidade dos recursos presentes nessas infraestruturas (i.e. processamento, armazenamento, memória e rede) que são disponibilizados por cada provedor de recursos participante. Isso gera a necessidade do desenvolvimento de soluções que sejam agnósticas ao tipo de recurso que será provido [Kukliński et al. 2018]. Sendo assim, o objetivo deste trabalho é apresentar a arquitetura NECOS para provimento de fatias de rede fim a fim em ambientes multi-inquilinos. Em particular, nosso trabalho concentra-se em dois componentes desta arquitetura, o DC e o *WAN Slice Controller*, responsáveis pela alocação de recursos computacionais e de rede nos provedores de infraestrutura.

O restante do trabalho está organizado da seguinte forma: a Seção 2 apresenta

organizações de definição de padrões que descrevem o conceito de fatiamento de rede, juntamente com projetos de pesquisa que propõem diferentes mecanismos para tratar o assunto. A Seção 3 aborda os principais conceitos, componentes e abstrações necessárias para dar suporte a *Slice as a Service* bem como a arquitetura do projeto NECOS. Os componentes DC e WAN *Slice Controller*, pertencentes a arquitetura NECOS, são apresentados na Seção 4 e na Seção 5, respectivamente. A Seção 6 apresenta a implementação e experimentos realizados para a criação das fatias de rede. Por fim, na Seção 7 apresentamos a conclusão de nosso trabalho, destacando resultados e contribuições.

2. Trabalhos Relacionados

Existem diversos projetos que lidam com o fatiamento de recursos nos mais diversos níveis, sejam recursos de computação como processamento, armazenamento e memória, ou recursos de rede. Esta seção está organizada da seguinte forma: na Seção 2.1, apresentamos conceitos de *network slicing* propostos por diferentes *Standards Definition Organizations* (SDOs). A Seção 2.2 discute sobre projetos de pesquisa que propõem mecanismos para tratar o assunto de *network slicing*.

2.1. Fatiamento de rede

Na literatura, diferentes SDOs como ITU, NGMN Alliance, IETF e 3GPP apresentam definições acerca de *network slicing*. De acordo com a ITU [ITU-T Y.3011 2012], o fatiamento segue o conceito de *Logically Isolated Network Partition* (LINP), em que uma fatia é uma unidade de recursos programáveis, como computação, armazenamento e rede.

A NGMN Alliance [Alliance 2016] determina que o conceito de *network slicing* seja composto por três camadas principais: camada de serviço, camada de *network slice* e camada de recursos. O padrão utiliza modelos pré-definidos para gerar instâncias em tempo de execução, que representam cada uma dessas camadas. Nesta arquitetura, é possível criar instâncias de sub-rede que podem ser utilizadas por mais de uma *network slice*. Uma instância de uma sub-rede é definida como um conjunto de funções de rede.

Para a *Internet Engineering Task Force* (IETF) [Galis et al. 2017], o *network slicing* é um conjunto de recursos de rede, físicos e/ou virtuais, particionados e que podem atuar como instâncias independentes. O *draft* apresenta as etapas do ciclo de vida em que o *network slicing* deve ter: criação, ativação/desativação, elasticidade, entre outros.

Por fim, o *Third Generation Partnership Project* (3GPP) [thi 2017] propõe uma arquitetura de rede para viabilizar o *network slicing*. Nesse caso, uma *network slice* é um conjunto de recursos e funcionalidades de rede que, juntos, compõem um ou mais serviços para o inquilino contratante da infraestrutura. Assim como em [Alliance 2016], a *network slice* é construída com base em modelos pré definidos que especificam um conjunto de funções necessárias para uma determinada finalidade.

2.2. Redes 5G

O projeto 5GEx [Bernardos et al. 2015] provê uma "Network Factory" na qual os recursos e funções de rede são negociados e provisionados para diferentes aplicações. No 5GEx, uma função de coordenação do *Slicer* é concebida com o objetivo de melhorar a aplicação do *Resource Orchestrator* a partir do fracionamento de recursos de multi-localização, além da funcionalidade de gerenciamento de ciclo de vida das *Virtualized Net-*

work Functions (VNFs). Além disso, o *Slicer* gerencia e integra funções de ciclo de vida específico para VNFs de vários fornecedores.

O projeto 5G-SONATA, apresentado em [Dräxler et al. 2017], trata uma *network slice* como um conjunto de recursos que pode ser agregado em uma infraestrutura distribuída, de maneira a dar suporte ao fornecimento de serviços e/ou funções de rede fim a fim. Nessa proposta, um domínio pode abrigar múltiplas *network slices* simultaneamente, em que todas funcionam de forma isolada.

O projeto 5G-NORMA [5G-PPP 2017] permite provisionar (alocar e desalocar) um conjunto de funções de rede, otimizar a *Radio Access Network* (RAN) e a *cloud*, além de controlar *Software Defined Mobile Network* (SDMN). Para isso, o 5G-NORMA implementa funções capazes de lidar com procedimentos voltados para serviços e recursos para *network slicing* intra e inter domínio.

3. Fatias de Rede como Serviço

Clayman *et al.* [Clayman 2017] apresenta funcionalidades, componentes e abstrações da arquitetura necessárias para viabilizar o modelo de *Slice as a Service* (SlaaS). Neste modelo, um *Virtualized Infrastructure Manager* (VIM) pode ser instanciado dinamicamente sobre um conjunto de recursos (tais como processamento, armazenamento e rede) que encontram-se geograficamente distribuídos. Para isso, o modelo possibilita que vários *Resource Providers* que operam suas próprias infraestruturas físicas, se coloquem à disposição no *Marketplace* para fornecer recursos computacionais [Maciel et al. 2019].

Inquilinos, denominados (*tenants*), contratam SlaaS instanciados sobre essas infraestruturas para executar seus serviços. Considerando que cada inquilino tem um conjunto de requisitos para seus serviços, ele deverá estar apto a receber uma fração isolada de recursos computacionais que atendam a sua especificação.

A Figura 1 ilustra a visão geral para provisionamento de *Slice as a Service* do projeto NECOS. A arquitetura é baseada no conceito de recursos federados providos por múltiplos *data centers* de computação em nuvem e provedores de rede. Três camadas compõem a arquitetura:

- *Orchestration Layer*: essa camada é responsável por gerenciar o ciclo de vida da *network slice* (criação, manutenção e exclusão), além de coordenar as múltiplas fatias de DC e de rede (*slice parts*) que devem ser abstraídas como uma única *network slice* de um inquilino. Isso significa que todos esses recursos devem se comportar como uma única unidade gerenciável;
- *Slice Control Layer*: na arquitetura NECOS, o componente DC *Slice Controller* deve ser instanciado para cada *data center* que seja um *Resource Provider*, bem como o componente WAN *Slice Controller* deve ser instanciado para cada provedor de conectividade. Eles são responsáveis pela criação de fatias no domínio administrativo, alocando os recursos necessários para atenderem a especificação do inquilino, além de instanciar um VIM que seja capaz de gerenciar esses recursos. A camada *Slice Control Layer* é um ponto de controle que lida com o gerenciamento das *slice parts* de DC e de rede;
- *Infrastructure Layer*: é a camada responsável por representar todos os recursos físicos que serão utilizados para o provimento de serviços de redes 5G.

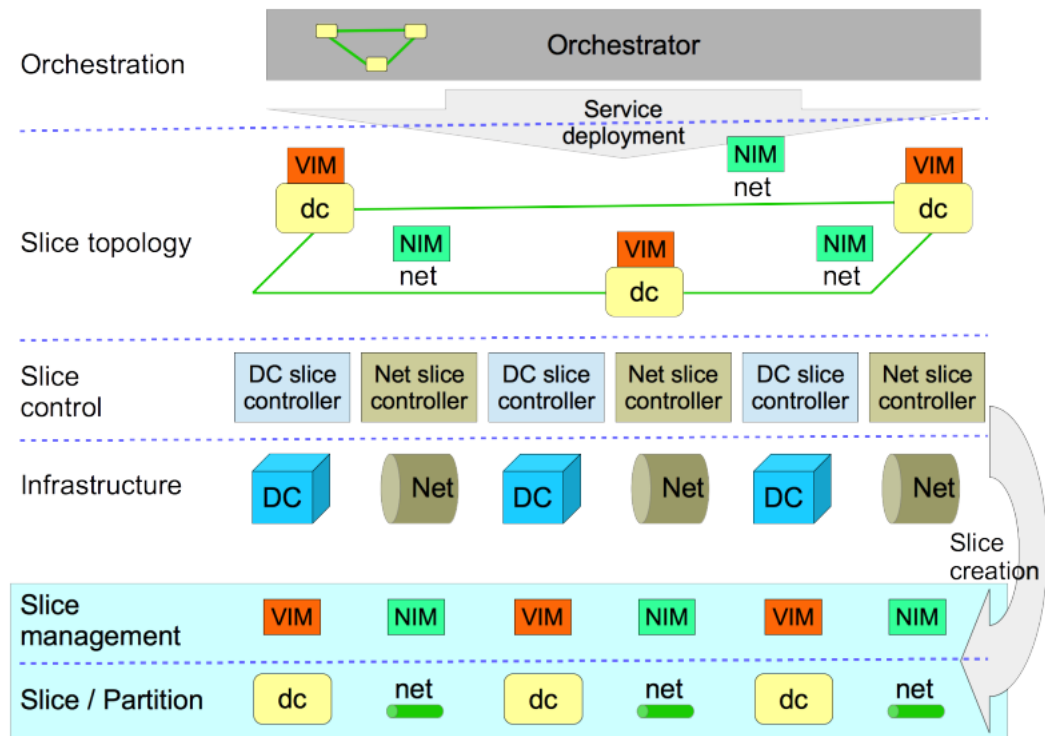


Figure 1. Visão geral para provisionamento de *Slice as a Service*.

O NECOS é um projeto de pesquisa que envolve instituições do Brasil e da União Européia, cujo principal objetivo é fornecer um mecanismo no qual *network slices* são instanciadas como um serviço. Para isso, o projeto define um modelo de *network slicing* em que a infraestrutura subjacente (de computação em nuvem e rede) é particionada em *slice parts*, as quais são combinadas para formar uma *network slice* fim a fim [NECOS 2017].

No NECOS, um *tenant* é uma organização que adquire *network slices* para executar seus serviços. Uma *Content Delivery Network* (CDN) [Pallis and Vakali 2006] é um exemplo de inquilino, que fornece conteúdo em uma rede distribuída geograficamente para diminuir a latência de acesso a seus usuários. É importante ressaltar que inquilinos no NECOS são organizações que proveem serviços para outras organizações ou para usuários finais e, portanto, são diferentes de clientes tradicionais de serviços *online*.

A Figura 2 ilustra a arquitetura NECOS. A requisição do *tenant* para criação da *slice*, denominada *Slice Description*, é iniciada a partir do componente *Slice Activator* que interage com o *Slice Specification Processor*. Essa descrição inclui as *slice parts* (DC e NET) que formam a *slice* fim a fim, bem como os recursos computacionais (processamento, armazenamento e rede), e informações de alto nível acerca do serviço a ser instanciado. Em contrapartida, o inquilino recebe um ponto de entrada (*entry-point*) para a fatia criada, através do qual será capaz de gerenciar os recursos alocados para a fatia, bem como o serviço implantado.

Em seguida, o *Slice Specification Processor* extrai as especificações de infraestrutura e serviço requisitadas, e as envia para o componente *Slice Builder*. O componente *Slice Builder* é responsável pela construção de uma *network slice* multi-domínio fim a

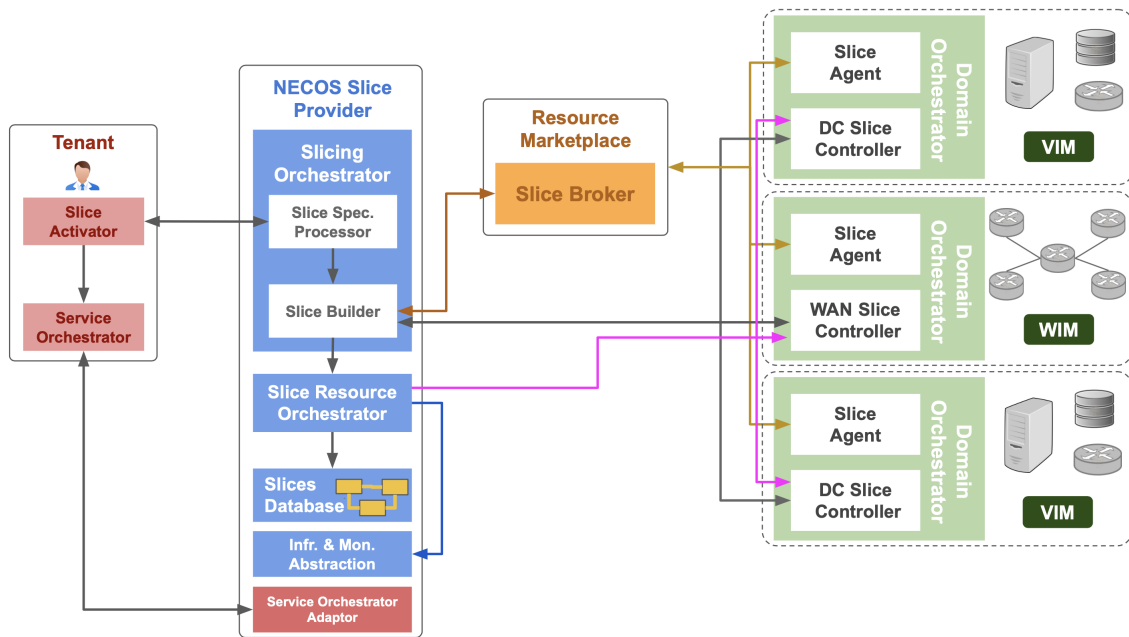


Figure 2. Arquitetura NECOS.

fim através das suas *slice parts* constituintes.

O *Resource Marketplace* é um sistema totalmente distribuído que localiza provedores de recursos que possam atender as especificações de uma *slice part*. O componente *Slice Broker* recebe requisições do *Slice Builder* e descobre esses provedores interagindo com um conjunto de *Slice Agents* hospedados nos domínios de recursos envolvidos.

A partir das informações retornadas pelo *Slice Broker*, o *Slice Builder* escolhe os provedores de recursos que irão fornecer cada *slice part* e entra em contato com o DC e o WAN *Slice Controller* desses provedores a fim de criar as *slice parts*.

O DC *Slice Controller* é responsável por provisionar uma *slice part* de recursos de computação em um centro de dados particular (DC *slice part*). Portanto, cada centro de dados pertencente à federação, deve executar um componente DC *Slice Controller*. O componente WAN *Slice Controller* é responsável por provisionar uma *slice part* com recursos de rede (NET *slice part*). Um NET *slice part* conecta dois DCs *slice parts*. Portanto, cada provedor de rede que pertence à federação deve executar um componente WAN *Slice Controller*.

Cada DC e WAN *Slice Controller* retorna para o *Slice Builder* *entry-points* para as *slice parts* alocadas. Posteriormente, essas informações são repassadas para o *Slice Resource Orchestrator*, que tem por objetivo orquestrar os recursos instanciados de modo a otimizar sua utilização na *network slice* criada. Na arquitetura NECOS, o DC e o WAN *Slice Controller* também são responsáveis por instanciar VIMs e WIMs, respectivamente. A instanciação destes elementos permite o *deployment* de serviços e/ou recursos virtuais, como por exemplo máquinas virtuais e *containers*.

O *Slice Resource Orchestrator* possui uma visão global do *network slice* e interage com o componente de monitoramento de dados denominado *Infrastructure and Monitoring Abstraction*, assim como a base de dados (*Slices Database*) que irá armazenar as

informações da *network slice* e os *Key Performance Indicators* (KPIs) monitorados. O componente *Service Orchestrator Adapter* interage com o *Service Orchestrator* do inquilino, que fornece recursos necessários para adaptar as chamadas do orquestrador de serviços em chamadas internas ao *Slice Resource Orchestrator*.

Na Seção 4 e na Seção 5 explicaremos com mais detalhes como os componentes DC e WAN *Slice Controller* foram projetados e implementados, respectivamente.

4. DC *Slice Controller*

Na arquitetura NECOS, cada *data center* executa uma instância do DC *Slice Controller* que é responsável por criar DC *slice parts* naquele domínio administrativo, alocando os recursos necessários descritos pelo *tenant* para uma *slice part* específica. Feito isso, o DC *Slice Controller* deve responder à chamada com um *entry-point* do VIM que foi instanciado na *slice part* criada. A Figura 3 ilustra a arquitetura do DC *Slice Controller*.

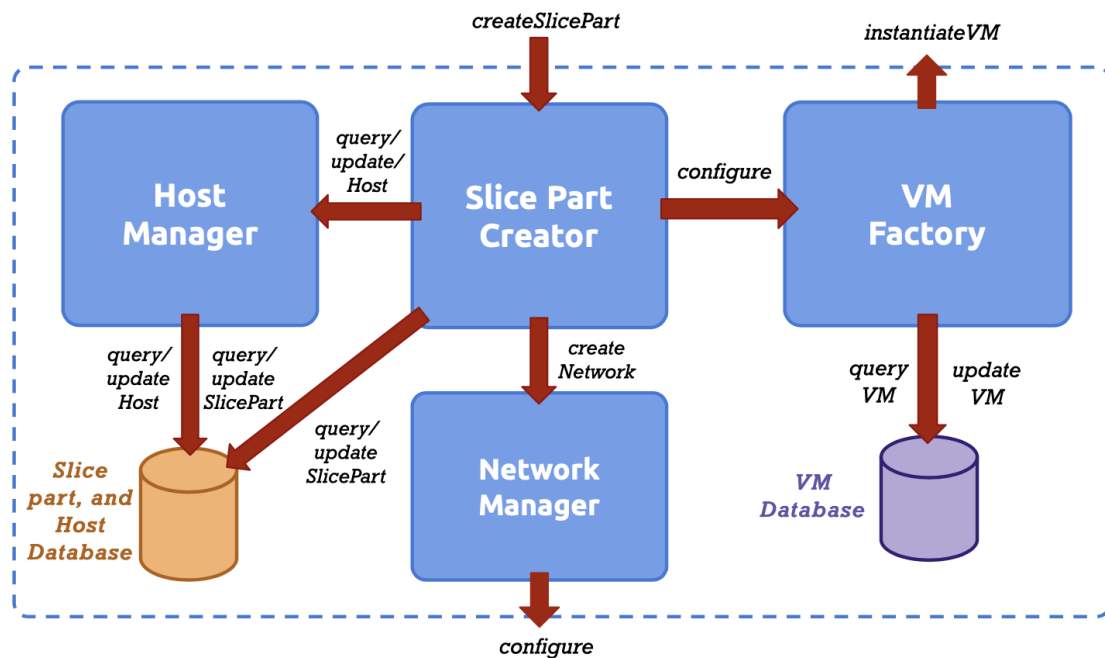


Figure 3. Componentes internos da arquitetura do DC *Slice Controller*.

O DC *Slice Controller* permite que provedores de *data center* e inquilinos interajam com o sistema NECOS. Através de uma API REST, o administrador pode gerenciar os recursos do *data center*, enquanto os *tenants* podem realizar as requisições de *network slice*. O componente *Slice Part Creator* é responsável por criar *slice parts* e instanciar VIMs sob demanda, baseado na requisição feita pelo *tenant*. As chamadas realizadas ao *Slice Part Creator* são divididas, basicamente, em duas: *request_slice_part* e *activate_slice_part*. A chamada *request_slice_part* é responsável por registrar uma nova *slice part* no banco de dados, enquanto a chamada *activate_slice_part* instancia um *slice part* previamente registrado.

O *Host Manager* é o componente que gerencia todos os recursos no *data center* e armazena suas informações. Essas informações são necessárias por validar os componentes de um *slice part*. A etapa de validação inclui verificar se há alguma inconsistência na requisição de *slice part* (se o tipo de VIM existe, por exemplo) e, verificar,

se há disponibilidade de recursos para sua criação. O componente *Network Manager* é responsável por gerenciar toda a parte de rede do *slice part*. Ele é capaz de criar e remover *bridges Open vSwitch* (OVS) sob demanda. Essas *bridges* são utilizadas para prover comunicação entre as VMs pertencentes a um *slice part*, ao mesmo tempo que as isola das VMs pertencentes a outros *slice parts*.

O componente *VM Factory* é responsável por implantar um determinado tipo de VIM e configurá-lo para usar os recursos que compõe a *slice part*. Além disso, ele gerencia todas as máquinas virtuais que compõem as *slice parts*. Para implantar as VMs, o *VM Factory* clona e reconfigura o(s) *template(s)* do VIM sob demanda. Um *template* de um VIM (e.g. Kubernetes e OpenStack), é um modelo vazio ainda não configurado.

Para cadastrar ou ativar uma *slice part*, o *Slice Part Creator* recebe as requisições da API REST e interage com o *Host Manager*. Quando a requisição é *request_slice_part*, o *Slice Part Creator* se comunica com o *Host Manager* para verificar se há recursos disponíveis para criar uma nova *slice part*. Se for possível, o *Host Manager* registra o *slice part* no banco de dados e retorna as informações para o *Slice Part Creator*. Quando a requisição é *activate_slice_part*, o *Slice Part Creator* se comunica com o *Host Manager* para verificar se a *slice part* existe e se está disponível para ser instanciado. Caso isso ocorra, o *Slice Part Creator* se comunica com o *VM Factory* solicitando a implantação do *slice part*.

Para desenvolvimento do DC *Slice Controller*, utilizamos um conjunto diverso de tecnologias. O componente *Slice Part Creator* foi implementado utilizando uma API REST que permite ao administrador e ao usuário controlar todas funcionalidades do DC *Slice Controller*.

A implementação do *VM Factory* foi realizada utilizando o Libvirt [Libvirt 2019], juntamente com XEN [XEN 2019] e volumes lógicos do tipo LVM. Através do Libvirt, é possível controlar todo o ciclo de vida das VMs, assim como os recursos alocados para elas. Dessa forma, o *VM Factory* é capaz de implantar *templates* pré-configurados e modificá-los. O componente *Slice Part Creator* do DC *Slice Controller* recebe requisições REST de modo a realizar as operações responsáveis pela criação da DC *slice part*.

5. WAN *Slice Controller*

Para cada *Network Provider*, há um WAN *Slice Controller* que cria NET *slice parts* dinamicamente. Uma NET *slice parts* é conjunto de *links* virtuais que servem para conectar dois DC *slice parts* e garantir a comunicação entre elas a fim de viabilizar uma *network slice* fim a fim. Para isso, o WAN *Slice Controller* gerencia todos os recursos de rede no domínio de um *Network Provider* e mantém registros dos recursos que estarão disponíveis, bem como aqueles já alocados para *network slices* existentes.

O WAN *Slice Controller* recebe requisições de criação de NET *slice parts* e verifica se possui recursos (e.g. largura de banda) suficientes para atender a solicitação. Caso seja possível, o WAN *Slice Controller* instancia o conjunto de *links* virtuais necessários para conectar as DC *slice parts*. A Figura 4 ilustra os componentes internos do WAN *Slice Controller*:

- **WAN Master:** esse componente é executado no ambiente do *Network Provider* e é a principal função do WAN *Slice Controller*. É responsável por iniciar todo o

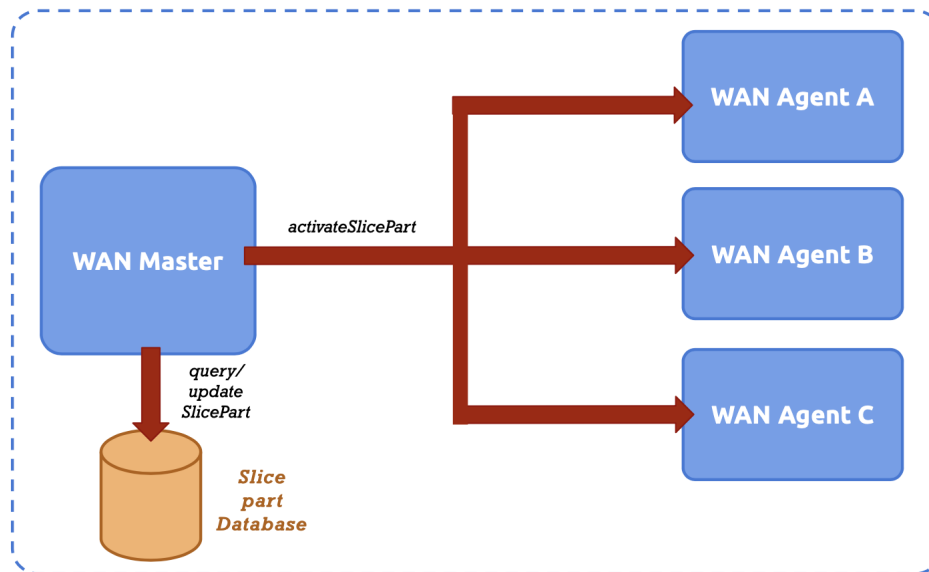


Figure 4. Componentes internos arquitetura do WAN Slice Controller.

- procedimento de criação da NET *slice part* de WAN a partir da requisição recebida do *tenant*. A partir da requisição recebida do *Slice Builder*, o WAN Master cria um registro para a NET *slice part* no seu banco de dados local (*Slice Part Database*). Essa atividade gera uma identificação única que é retornada ao componente solicitante. Posteriormente, para ativar uma NET *slice part* específica, o WAN Master interage com os WAN Agents localizados nos *data centers* que hospedam as duas DC *slice parts* a serem conectadas. De forma geral, o WAN Master se comunica com dois WAN Agents e os gerencia para configurar um túnel VXLAN, conectando *bridges Open vSwitch* (OVS) de ambas as partes da *slice*. Essa conexão é realizada através da implementação de uma sobreposição da rede L2 na camada de rede L3 existente, que na maioria dos casos é a Internet;
- WAN Agent: esse componente é executado em cada *data center* onde o WAN Slice Controller é capaz de prover conectividade. A sua função é receber instruções do WAN Master para a configuração de túneis VXLAN.

6. Experimentos

Com o intuito de avaliar a viabilidade de uma *network slice* envolvendo múltiplos domínios, criamos uma slice fim a fim entre o Brasil e a Espanha, e implantamos a plataforma Dojot [CPqD 2017] sobre a *slice* criada. A Dojot é uma plataforma aberta de serviços para Internet das Coisas (*Internet of Things* - IoT) e seu principal objetivo é fornecer recursos que facilitem o desenvolvimento de soluções IoT. Esses recursos são, por exemplo, APIs para acesso à plataforma, armazenamento de grandes volumes de dados, conexão com dispositivos e sensores, processamento de eventos, entre outros.

Em nosso ambiente de testes, utilizamos três servidores em diferentes domínios administrativos: UFG e UNICAMP (Brasil) e, 5TONIC [IMDEA Networks Institute 2019] (Espanha). No servidor localizado na UFG, foram instalados os componentes da arquitetura NECOS do *tenant*, (*Slice Activator*, *Service Orchestrator*) e *Slice Provider* (*Slice Specification Processor*, *Slice Builder*, *Slice*

Resource Orchestrator e Infrastructure and Monitoring Abstraction). Já na UNICAMP e no 5TONIC, foram instaladas instâncias do DC Slice Controller e dos WAN Agents. O WAN Master foi implantado na UNICAMP.

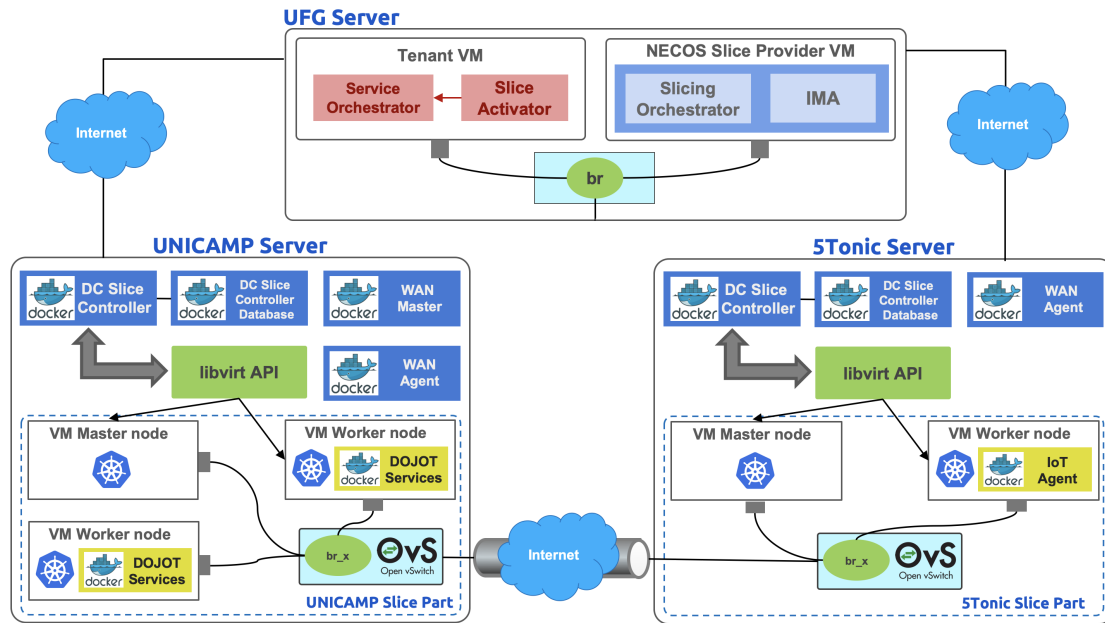


Figure 5. Cenário de experimentação.

Para ilustrar um cenário de redes 5G, definimos um ambiente em que a UNICAMP representa a *cloud computing* e o 5TONIC a *edge computing*. Em nosso experimento, criamos uma *network slice* fim a fim alocando uma *DC slice part* na UNICAMP e outra no 5Tonic. A *DC slice part* da UNICAMP é formada por 3 VMs (um *master* e dois *workers nodes*) gerenciadas pelo Kubernetes. O *DC slice part* no 5Tonic é um cluster Kubernetes formado por duas VMs (um *master* e um *workers nodes*).

No *DC slice part* da UNICAMP instanciamos a maioria dos microsserviços da Dojot. Já na *DC slice part* do *data center* do 5Tonic, apenas o microsserviço responsável pela comunicação com os dispositivos de IoT foi instanciado, neste caso, o *broker* MOSCA [Wirawan et al. 2018]. Para validar o ambiente instanciado, foram criados dispositivos IoT simulados que, por sua vez, eram programados para submeter dados para a Dojot através do componente supracitado. A Figura 5 ilustra os servidores utilizados no experimento, os componentes da arquitetura NECOS instalados em cada servidor e as *slice parts* (DC e NET) criadas no experimento.

A seguir, listamos os microsserviços essenciais para o funcionamento da Dojot, bem como seus papéis em nosso cenário de experimentação, em suas respectivas *slice parts*, UNICAMP e 5Tonic:

- UNICAMP: GUI Web App (React), API Gateway (Kong), Authorization (PostgreSQL e Redis), History (MongoDB), Flow Broker (Docker, RabbitMQ e MongoDB), Device Manager (PostgreSQL e Redis), Persister (MongoDB) e Data Broker (Kafka e Redis);
- 5Tonic: *broker* de mensagens MOSCA.

A Figura 6 apresenta o gráfico contendo os tempos contabilizados para criação das DC *slice parts* (5Tonic e UNICAMP) e da NET *slice part*. Podemos observar que o tempo para criar DC *slice parts* é significativamente maior que aquele utilizado para criar NET *slice parts*. Isso se deve ao fato da alocação de recursos nos *data centers* envolver passos que consomem mais tempo (por exemplo, clonagem de VMs, configuração da rede da *slice part* e configuração da ferramenta de monitoramento) que a conexão das *slice parts*. Para a criação da DC *slice part* no 5Tonic o tempo consumido foi de 861,14 segundos. Para a criação da DC *slice part* na UNICAMP o tempo foi de 1708,74 segundos. A NET *slice part* de rede contabilizou 1,56 segundos.

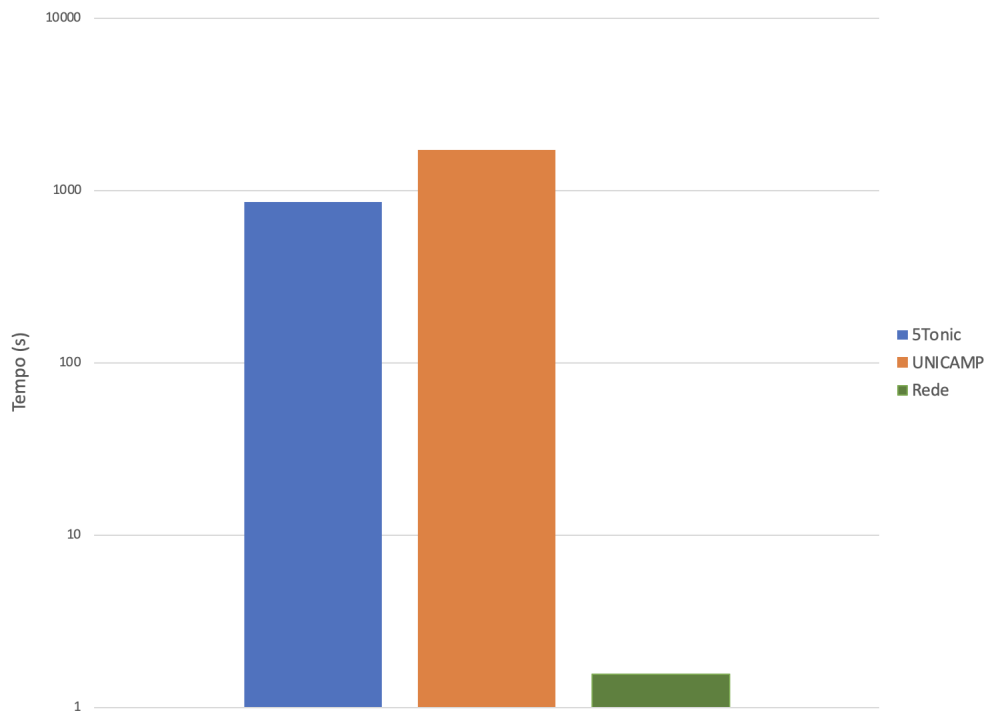


Figure 6. Tempos totais de criação

A Figura 7 detalha os tempos dos principais passos executados na criação das DC *slice parts*. Dividimos esses passos em seis partes:

1. Clonagem das VMs: os recursos computacionais que utilizamos em nossos experimentos foram criados na forma de VMs que, por sua vez, são utilizadas para compor *clusters* de VIMs. Neste caso, mantemos um repositório de *templates* de VMs pré configuradas com um sistema operacional e suas dependências já instaladas, o que permitiu reduzir o tempo total de criação. Quando uma DC *slice part* é solicitada, os *templates* de VMs são clonados de acordo com a especificação do *tenant*. Como mostrado na Figura 5, o DC *Slice Controller* do 5Tonic clonou duas VMs resultando no tempo de 460,45 segundos. Já o DC *Slice Controller* da UNICAMP, clonou três VMs, resultando em um tempo de 899,82 segundos. Em geral, quanto maior a quantidade de VMs a serem clonadas, maior será o tempo total de clonagem;
2. Configuração de rede da *slice part*: nesta operação, o DC *Slice Controller* realiza o isolamento entre todas as VMs pertencentes a uma mesma *slice part*, de maneira

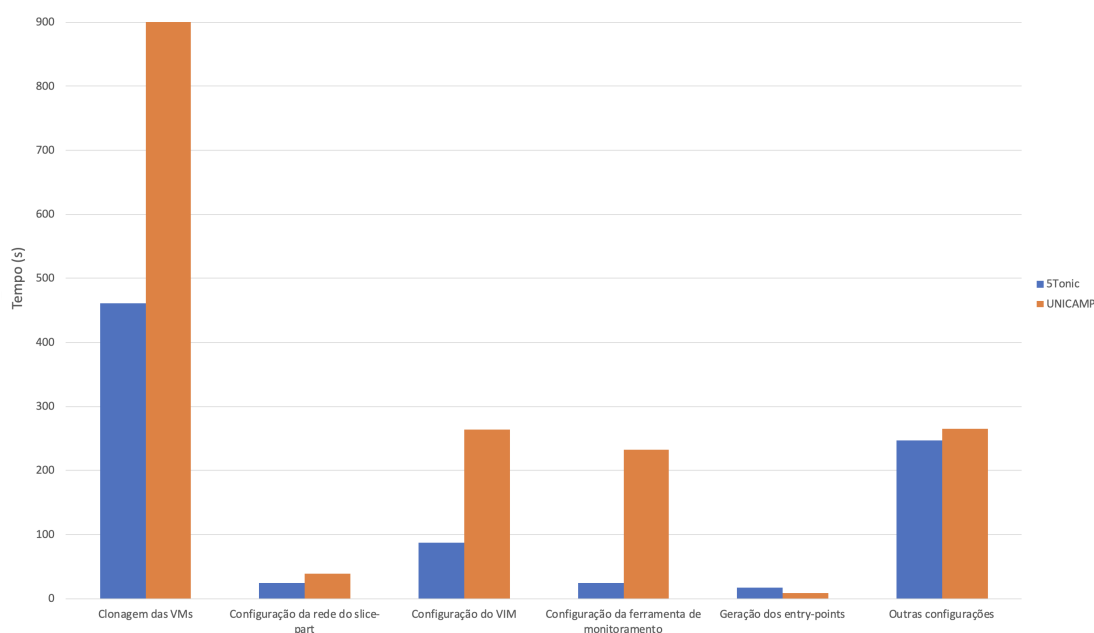


Figure 7. Tempos de criação na UNICAMP e 5Tonic.

a garantir que outras *slice parts* tenha acesso apenas a seus recursos alocados. Neste experimento, o tempo para configuração da rede da *slice part* do 5Tonic foi de 24,63 e da UNICAMP de 38,20 segundos. A configuração da rede da *slice part* da UNICAMP levou um tempo maior por conter mais VMs;

3. Configuração do VIM: o DC *Slice Controller* é responsável por configurar o VIM escolhido, neste caso, o Kubernetes. Para esse experimento, o tempo consumido para a configuração do VIM no 5Tonic foi de 87,62 segundos e 264,08 segundos na UNICAMP. Novamente, o tempo consumido na *slice part* da UNICAMP foi maior devido a maior quantidade de nós *workers*;
4. Configuração da ferramenta de monitoramento das *slice parts*: neste experimento utilizamos duas ferramentas de monitoramento, o Prometheus na DC *slice part* do 5Tonic e o NetData no DC *slice part* da UNICAMP. Usamos diferentes ferramentas de monitoramento em cada *slice part* para demonstrar que o DC *Slice Controller* é agnóstico da ferramenta de monitoramento disponível no provedor de recursos. O tempo consumido para configuração na *slice part* do 5Tonic foi de 24,46 segundos e, no da UNICAMP, foi de 264,08 segundos. O tempo maior no DC *slice part* da UNICAMP foi maior em virtude da ferramenta NetData envolver mais passos que aqueles presentes na ferramenta do Prometheus;
5. Geração dos *entry-points*: para disponibilização do *entry-point* do DC *slice part* do 5Tonic o tempo contabilizado foi de 16,66 segundos. Já na UNICAMP, o tempo foi de 8,77 segundos;
6. Outras Configurações: estas configurações são relacionadas a operações realizadas pelo DC *Slice Controller* que não são relevantes para o contexto deste trabalho. No DC *slice part* 5Tonic o tempo consumido por essas operações foi de 247,32 segundos e, na UNICAMP de 265,41 segundos.

De maneira geral, esses experimentos revelaram que o DC e o WAN *Slice Controller* são capazes de criar *network slices* fim a fim sob demanda de maneira automática

e eficiente, uma vez que os tempos aqui apresentados envolvem *slice parts* gerenciadas por um VIM complexo como o Kubernetes.

7. Conclusão

Neste trabalho nós apresentamos a arquitetura NECOS que tem por objetivo lidar com as limitações das infraestruturas de nuvem atuais. O NECOS propõe uma abordagem denominada *Lightweight Slice Defined Cloud* (LSDC) que atua como um *enabler* de *Cloud Slicing* através do conceito de *Slice as a Service*. A arquitetura proposta permite dar suporte ao gerenciamento de multi-domínios, além de prover orquestração inteligente de infraestruturas de nuvem federadas.

Este artigo concentrou-se nos componentes da arquitetura NECOS, DC e WAN *Slice Controller*, responsáveis pela alocação de recursos computacionais e de rede nos provedores de infraestrutura. Os experimentos realizados com esses componentes nos permitiram concluir que eles são capazes de criar *network slices* fim a fim sob demanda de forma automática e em tempo satisfatório, haja visto que as *slice parts* são gerenciadas pelo VIM Kubernetes, que possui certo grau de complexidade.

References

- (2016). *3GPP, Feasibility Study on New Services and Markets Technology Enablers: Stage 1, Technical Specification*. Third Generation Partnership Project (3GPP). Version 14.2.0.
- (2017). System architecture milestone of 5g phase 1 is achieved. Disponível em: <http://www.3gpp.org/news-events/3gpp-news/1930-sysarchitecture>, Acessado em 03/09/2019.
- 5G-PPP (2017). 5g-norma eu project. Disponível em: <http://www.it.uc3m.es/wnl/5gnorma/>, Acessado em 02/09/2019.
- Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., and Flinck, H. (2018). Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453.
- Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015a). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376.
- Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015b). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376.
- Alliance, N. (2016). Description of network slicing concept. *NGMN 5G P*, 1.
- Bernardos, C. J., Dugeon, O., Galis, A., Morris, D., Simon, C., and Szabó, R. (2015). 5g exchange (5gex)-multi-domain orchestration for software defined infrastructures. *focus*, 4(5):2.
- Clayman, S. (2017). Network slicing supported by dynamic vim instantiation. *IETF 100, Singapore*.
- CPqD (2017). Dojot. Disponível em: <http://www.dojot.com.br>, Acessado em 18/09/2019.

- 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.
- Foukas, X., Patounas, G., Elmokashfi, A., and Marina, M. K. (2017). Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100.
- Galis, A., Kuklinski, S., Dong, J., Geng, L., and Makhijani, K. (2017). Network slicing - revised problem statement. *Working Draft, IETF Secretariat, Internet-Draft draft-galis-netslices-revised-problem-statement-01*.
- IMDEA Networks Institute, T. (2019). 5tonic. Disponível em: <https://www.5tonic.org/>, Acessado em 18/09/2019.
- ITU-T Y.3011 (2012). Framework of Network Virtualization for Future Networks, Next Generation Network. Technical report.
- Iwamura, M. (2015). Ngmn view on 5g architecture. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE.
- Kukliński, S., Tomaszewski, L., Osiński, T., Ksentini, A., Frangoudis, P. A., Cau, E., and Corici, M. (2018). A reference architecture for network slicing. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 217–221. IEEE.
- Libvirt (2019). Libvirt documentation.
- Maciel, P. D., Verdi, F. L., Valsamas, P., Sakellariou, I., Mamatas, L., Petridou, S., Papadimitriou, P., Moura, D., Swapna, A. I., Pinheiro, B., et al. (2019). A marketplace-based approach to cloud network slice composition across multiple domains. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 480–488. IEEE.
- NECOS (2017). System architecture and platform specification v1. Disponível em: <http://www.h2020-necos.eu/documents/deliverables/>, Acessado em 20/07/2019.
- 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.
- Pallis, G. and Vakali, A. (2006). Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106.
- Wirawan, I. M., Wahyono, I. D., Idfi, G., and Kusumo, G. R. (2018). Iot communication system using publish-subscribe. In *2018 International Seminar on Application for Technology of Information and Communication*, pages 61–65. IEEE.
- XEN (2019). Xen documentation.
- Zhang, S., Xu, X., Wu, Y., and Lu, L. (2014). 5g: Towards energy-efficient, low-latency and high-reliable communications networks. In *2014 IEEE international conference on communication systems*, pages 197–201. IEEE.