Um componente de network slicing para o suporte de multi-inquilinos nas RANs do LTE.

Pedro H. A. Rezende[1] **e Edmundo R. M. Madeira**[1]

[1]Instituto da Computação
Universidade Estadual de Campinas (Unicamp)
Campinas, São Paulo, Brasil

pedrohenrique@lrc.ic.unicamp.br, edmundo@ic.unicamp.br

Abstract. *5G networks intend to integrate network slicing into their architecture aiming to satisfy the different service levels of an abundant amount of devices. Network Slicing relies on softwarization technologies, such as SDN and NFV, to instantiate slices (virtual networks) on top of the same physical substrate. This work introduces the "Otimizador de Slices", a component developed as an extension of LTE's evolved NodeB, responsible to perform network slicing for LTE downlink transmission. This component receives slice's information from multiple Service Providers and, based on the analysis of these information and on the network state, the proposed component selects the best slice to be scheduled at the moment. Simulations were performed to validate our proposal and expose the benefits that can be obtained by it, such as an enhancement of end user's QoS experience.*

Resumo. *As redes 5G pretendem integrar "network slicing" em sua arquitetura visando satisfazer os diferentes níveis de serviços de uma quantidade massiva de dispositivos. Network Slicing apoia-se em tecnologias de softwarização, como SDN e NFV, para instanciar slices (redes virtuais) sobre um mesmo substrato físico. Esses slices são mutualmente independentes e customizados de acordo com as necessidades dos usuários. Este trabalho introduz o Otimizador de Slices, um componente desenvolvido como uma extensão do evolved NodeB das redes LTE, responsável por concretizar o conceito de network slicing na transmissão de tráfego downlink das redes LTE. Este componente recebe informações sobre slices oriundas de múltiplos Provedores de Serviços e, a partir da análise dessas informações e do estado da rede, o componente proposto seleciona o melhor slice a ser escalonado no momento. Simulações foram realizadas para avaliar nossa proposta e mostrar os benefícios dela, como a melhora da QoS ofertada aos usuários finais.*

1. Introdução

As futuras redes 5G pretendem satisfazer os diferentes requisitos de serviço de uma quantidade abundante de aplicações. Para isso, os sistemas 5G planejam aumentar os recursos oferecidos aos usuários por meio de novas tecnologias de acesso de rádio e bandas no espectro, tais como MIMO (múltiplas entradas, múltiplas saídas) massivo e ondas milimétricas. Ademais, as redes 5G não empregarão a arquitetura monolítica presente nas redes atuais, uma vez que tal arquitetura é incapaz de atender os diferentes requisitos de serviço dos mais diversos casos de uso. Tais casos de uso incluem, por exemplo, veículos autônomos, hospitais inteligentes e tecnologias vestíveis.

Para suportar os mais diversos casos de uso, as redes 5G visam integrar o conceito de *Network Slicing* [Foukas et al. 2017] em sua arquitetura. Embora *Network Slicing* não seja uma tecnologia nova, apenas recentemente ela foi introduzida nas redes sem fio. *Network Slicing* permite que operadores de rede criem múltiplas redes virtuais (ou *slices*) sobre uma mesma infraestrutura física. Esses *slices* são mutualmente independentes e customizados de acordo com os requisitos definidos entre os operadores de rede e os usuários finais. A virtualização da infraestrutura de rede pode ser realizada, por exemplo, a partir de um *hypervisor* [Blenk et al. 2016].

O conceito de *Network Slicing* em redes 5G é realizado graças à consolidação de tecnologias de *softwarização* de redes, como Redes Definidas por Software (*Software Defined Networks* - SDN) [Kreutz et al. 2015] e Virtualização das Funções de Rede (*Network Functions Virtualization* - NFV) [Mijumbi et al. 2016]. Essas tecnologias de *softwarização* visam trazer os benefícios do *software* às redes. A flexibilidade, programabilidade e modularidade são exemplos de tais benefícios.

Network Slicing pode ser realizado tanto nas redes de acesso (*Radio Access Networks* - RAN) quanto no núcleo dos sistemas 5G. Recursos de computação, armazenamento, espectro, bem como as funções de rede podem ser virtualizadas e distribuídas aos diferentes *slices* pertencentes à rede. Neste trabalho, o foco é a realização do conceito de *Network Slicing* nas redes de acesso dos sistemas 5G, mais especificamente na alocação de espectro do canal *downlink* da RAN. Para isso, propomos um componente, chamado de Otimizador de *Slices* (OS), desenvolvido como uma extensão do evolved NodeB (eNB) das redes *Long Term Evolution* (LTE). As redes LTE são usadas como base para as futuras redes 5G.

O OS permite que múltiplos Provedores de Serviço (*Service Providers* - SPs) criem *slices*. O OS recebe mensagens sobre *slices* por intermédio de um *hypervisor*. Os múltiplos SPs oferecem seus serviços aos usuários finais por meio de controladores SDN, os quais operam em cima do *hypervisor*. O OS, além de concretizar o conceito de *network slicing*, é responsável por adaptar os *slices* ao estado da rede. Nossa proposta foi implementada e avaliada no NS-3 [NS-3 2017]. Os resultados obtidos mostram a eficiência de nossa proposta, como a melhora da Qualidade de Serviço (*Quality of Service* - QoS) percebida pelos usuários finais.

As demais seções deste trabalho estão organizadas da seguinte forma. A Seção 2 apresenta alguns conceitos importantes para o restante do trabalho, enquanto a Seção 3 introduz alguns trabalhos sobre *network slicing* presentes na literatura. A Seção 4 apresenta a nossa arquitetura do sistema e o componente proposto, bem como suas implementações. A Seção 5 expõe a avaliação de desempenho de nossa proposta. A Seção 6 conclui o artigo e apresenta trabalhos futuros.

2. Fundamentação Teórica

Nós apresentaremos nesta seção uma visão geral sobre Redes Sem Fio Definidas por *Software*, transmissão de tráfego *downlink* em redes LTE e rede multi-inquilinos.

2.1. Redes Sem Fio Definidas por Software

SDN separa o plano de controle do plano de dados dos dispositivos de encaminhamento de pacotes. O plano de controle, que torna-se uma entidade externa, é consolidado em componentes de *software*, chamado de controlador, o que facilita a gerência de redes.

Esse controlador é responsável pela programação de dispositivos por meio da *Application Programming Interface* (API) *Southbound* e pelo provisionamento de serviços para aplicações a partir da API *Northbound*. Os dispositivos de rede pertencentes ao plano de dados tornam-se simplificados, realizando um pequeno número de tarefas, dentre elas, o encaminhamento de pacotes.

Grande parte dos esforços em SDN tem sido feito nas redes cabeadas; entretanto, uma arquitetura de redes sem fio baseada em SDN pode oferecer inúmeros benefícios tanto aos operadores de rede quanto aos usuários finais [Haque and Abu-Ghazaleh 2016]. Em redes celulares, uma arquitetura baseada em SDN pode, por exemplo, facilitar a alocação de recursos de rádio, auxiliar no processo de *handover*, distribuir informações relacionadas ao estado das células e ajudar na redução de interferência entre elas.

2.2. Transmissão de tráfego downlink em redes LTE

O escalonador *downlink* na camada MAC do eNB é responsável pelo escalonamento de pacotes *downlink* [Grøndalen et al. 2017]. O *Physical Downlink Shared Channel* (PDSCH) é empregado para a transmissão de dados *downlink* do eNB aos equipamentos dos usuários (*User Equipments* - UEs). Todas as transmissões estão organizadas em quadros de rádio de 10 ms cada. Em cada quadro há 10 subquadros de 1 ms cada, que é o Intervalo de Tempo de Transmissão (*Transmission Time Interval* - TTI).

Em cada TTI, o escalonador realiza a alocação de recursos de rádio. Essa alocação é feita a partir de algumas informações, tais como: o estado do *buffer* da entidade *Radio Link Control* (RLC); a QoS recebida de camadas superiores; e do Indicador de Qualidade de Canal (*Channel Quality Indicator* - CQI), que é reportado pelos UEs. As restrições de QoS são fornecidas pelos *bearers* do *Evolved Packet System* (EPS). Tais *bearers* são associados a um Identificador de Classe de QoS (*QoS Class Identifier* - QCI) que, por sua vez, define alguns parâmetros, como a prioridade de escalonamento e perda de pacotes aceitável. O esquema *Adaptive Modulation and Coding* (AMC) seleciona uma modulação e codificação adequada para a transmissão de pacotes com o intuito de aumentar a eficiência da rede. A camada MAC também possui a entidade *Hybrid Automatic Repeat Request* (HARQ) para a retransmissão de pacotes perdidos.

2.3. Rede multi-inquilinos

As operadoras móveis vêm tendo dificuldade de acomodar em sua infraestrutura de rede o tráfego de seus usuários, uma vez que houve um aumento expressivo desse tráfego nos últimos anos devido à popularização dos *smartphones* e *tablets*, bem como do vídeo sob demanda. Dessa forma, as operadoras móveis têm um alto gasto operacional e de infraestrutura para acomodar esse tráfego crescente. Com isso, muitas operadoras móveis estão compartilhando entre si suas infraestruturas de rede. O compartilhamento de uma infraestrutura de rede é comumente chamado de rede multi-inquilinos [Samdanis et al. 2016]. O principal benefício de redes multi-inquilinos é a redução dos custos de capital e operacional para todos os envolvidos nesse compartilhamento.

3. Trabalhos Relacionados

Os autores em [Caballero et al. 2017] propõem um *framework* baseado em um modelo de alocação proporcional de recursos com o objetivo de realizar o conceito de *network slicing* em redes multi-inquilinos. O *framework* proposto permite o compartilhamento dinâmico de recursos entre os *slices*, aumentando o desempenho geral dos inquilinos. Além disso,

tal *framework* é genérico e não tem como base uma tecnologia celular específica. Em [Kokku et al. 2013], os autores introduzem CellSlice, um sistema que realiza o conceito de *network slicing* em redes de acesso na tecnologia WiMAX. Diferente de nossa proposta, ambos os trabalhos não focam nas redes LTE e não usam uma arquitetura baseada em SDN.

Em [Chartsias et al. 2017] os autores propõem um solução baseada em SDN para a gerência e orquestração de *slices* de múltiplos inquilinos. Todavia, diferente de nosso trabalho, em [Chartsias et al. 2017] a rede de acesso é baseada na tecnologia WiFi e o foco de [Chartsias et al. 2017] é no provisionamento de *slices* no *backhaul* das redes sem fio. É proposto em [Choyi et al. 2016] um *framework* que, a partir dos descritores de serviços definidos, realiza a negociação, seleção e atribuição dos *slices* na rede 5G aos respectivos usuários. Entretanto, em [Choyi et al. 2016] não são realizados experimentos para validar o *framework* proposto.

Em [Hu et al. 2016], os autores apresentam um mecanismo de escalonamento e *slicing* dinâmico para as redes LTE. No mecanismo proposto, uma quantidade de sub-canais é alocada para cada *slice*. Tais sub-canais podem ser realocados entre os *slices* com o objetivo de garantir os diferentes acordos de nível de serviço. O trabalho proposto em [Kamel et al. 2014] expõe um sistema para fatiar uma rede LTE em múltiplas redes virtuais. Após o fatiamento, tais redes virtuais são atribuídas a diferentes Provedores de Serviços e, baseado em um contrato de serviço, uma quantidade mínima de blocos de recurso é alocada a cada Provedor de Serviço.

Os autores em [Parsaeefard et al. 2015] apresentam um mecanismo de provisionamento de recursos e controle de admissão com o objetivo de maximizar o desempenho dos *slices* a partir da informação do estado do canal dos dispositivos dos usuários. A principal diferença entre nosso trabalho e [Hu et al. 2016, Kamel et al. 2014, Parsaeefard et al. 2015] é que nossa proposta usa uma arquitetura baseada em SDN. Em um trabalho anterior [Rezende and Madeira 2018], nós realizamos o conceito de *network slicing* na transmissão de tráfego *downlink* das redes LTE. Neste trabalho nós estendemos [Rezende and Madeira 2018], apresentando e implementando uma arquitetura multi-inquilinos. Dessa forma, permitimos que múltiplos SPs atuem na rede.

4. Arquitetura do Sistema e Componente proposto

Nesta seção serão apresentados a nossa arquitetura do sistema e o Otimizador de Slices, desenvolvido como uma extensão do eNB das redes LTE.

4.1. Visão Geral

A Figura 1 ilustra nossa arquitetura do sistema, que é constituída pelo Provedor de Infraestrutura (PI), o qual é proprietário de toda a infraestrutura física de rede; pelos controladores vSDNs e pelo *hypervisor*; além de toda infraestrutura física LTE, tanto da RAN quanto do núcleo (*Evolved Packet Core* - EPC). O PI é responsável por alocar recursos de rede aos Provedores de Serviços e pela gerência de tais recursos. Para a alocação desses recursos, o PI baseia-se nos Acordos de Níveis de Serviços (*Service Level Agreements* - SLAs) estabelecidos com os SPs.

A entidade Funções de Gerência e Orquestração (*Management and Orchestration* - MANO) presente no PI é responsável por alocar os recursos de rede aos SPs a partir dos SLAs definidos entre o PI e os SPs. Para isso, a MANO alimenta periodicamente

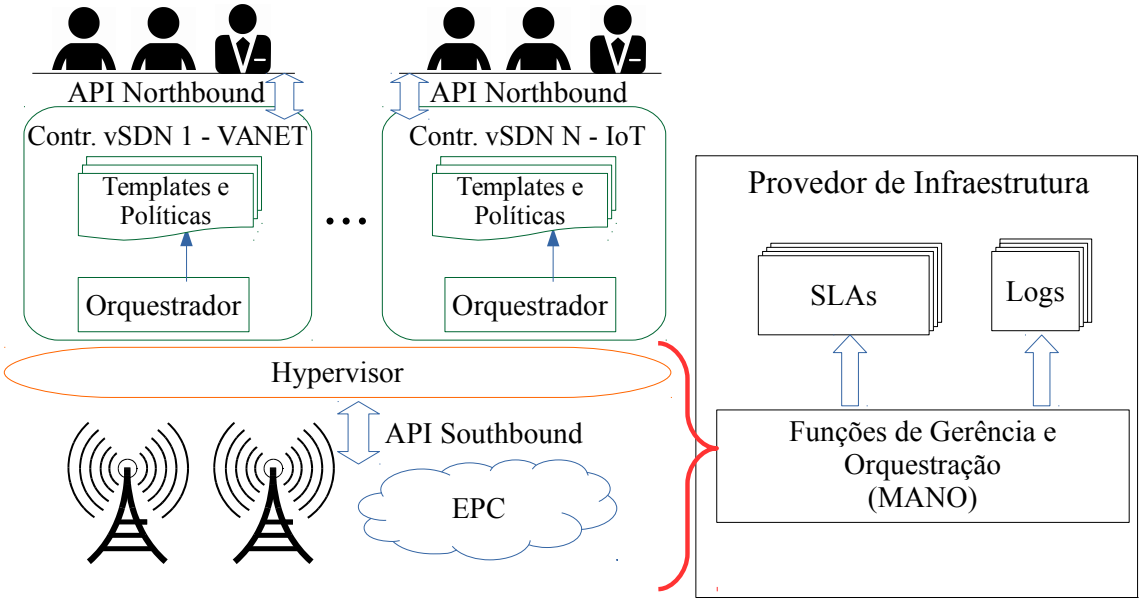


Figura 1. Visão Geral da Arquitetura do Sistema.

um *hypervisor* (Ex.: OpenVirteX [Al-Shabibi et al. 2014]) com informações relativas aos recursos contratados em cada SP. A quantidade de recursos de rádio ou o número de eNBs contratados são exemplos dessas informações. Uma vez que um quadro LTE contém 10 subquadros, os recursos de rádio são alocados pelo PI em um vetor de dez posições, chamado de Vetor de Mapeamento (VDM), onde cada posição representa a numeração do subquadro e o valor corresponde ao identificador do SP. É criado um VDM para cada eNB no sistema. O *hypervisor*, após receber o VDM, envia-o ao OS, que usará esse vetor para saber qual SP tem a prioridade de escalonamento neste subquadro. Além disso, o *hypervisor*, a partir dessas informações recebidas pelo PI, abstrai os recursos físicos de rede, alocando-os aos SPs como *virtual Software Defined Networks* (vSDNs). Cada SP, por sua vez, utiliza o seu próprio sistema operacional de rede (controlador) para configurar a sua vSDN.

Ao receber a vSDN, o controlador (e.g. POX[1], Floodlight[2]) é responsável por alocar os recursos virtualizados aos seus usuários. Para isso, cada SP define seus *templates* e políticas de orquestração de recursos em seu controlador. *Templates* podem ser definidos, por exemplo, para tráfego de vídeo e tráfego de voz, onde cada um desses tráfegos tem restrições distintas de QoS. Esses *templates* são acessados pelos usuários finais a partir da API *Northbound*, que é disponibilizada pelo controlador. Por meio dessa mesma API, os operadores de rede definem suas políticas de alocação de recursos. Em todos os controladores há um orquestrador que recebe como entrada: os recursos de rede, as políticas e os *templates*, e gera como saída dois parâmetros: os descritores de *slices* (DSs) e um vetor de alocação (VA). Os DSs contêm informações acerca dos *slices*, por exemplo, o conjunto de tuplas contendo os fluxos registrados no *slice*, assim como o identificador e a prioridade de cada *slice*. Nesse conjunto de tuplas, cada tupla é composta por dois itens, o *International Mobile Subscriber Identity* (IMSI) e o QCI. São necessários esses dois parâmetros uma vez que um mesmo UE pode gerar tráfegos pertencentes a diferentes

[1]https://github.com/noxrepo/pox
[2]http://www.projectfloodlight.org/floodlight/

slices. O VA define a distribuição dos recursos de rádio aos *slices*.

Os DSs e o VA gerados pelo orquestrador são enviados ao *hypervisor*. Ao receber ambos os parâmetros, o *hypervisor* verifica a integridade deles e, caso haja algum problema com eles, o *hypervisor* envia uma mensagem ao MANO que, por sua vez, escreve os erros em um arquivo de *log*. Se os DSs e o VA estiverem corretos, uma coleção de pares chave-valor é criada para mapear os valores do VA em subquadros de rádio. Tal coleção é chamada de mapa de alocação (MA). O MA e os DSs são enviados pelo *hypervisor* ao OS por meio da API *Southbound*. Exemplos de protocolos que podem ser usados na API *Southbound* incluem o OpenFlow [McKeown et al. 2008] e o NETCONF.

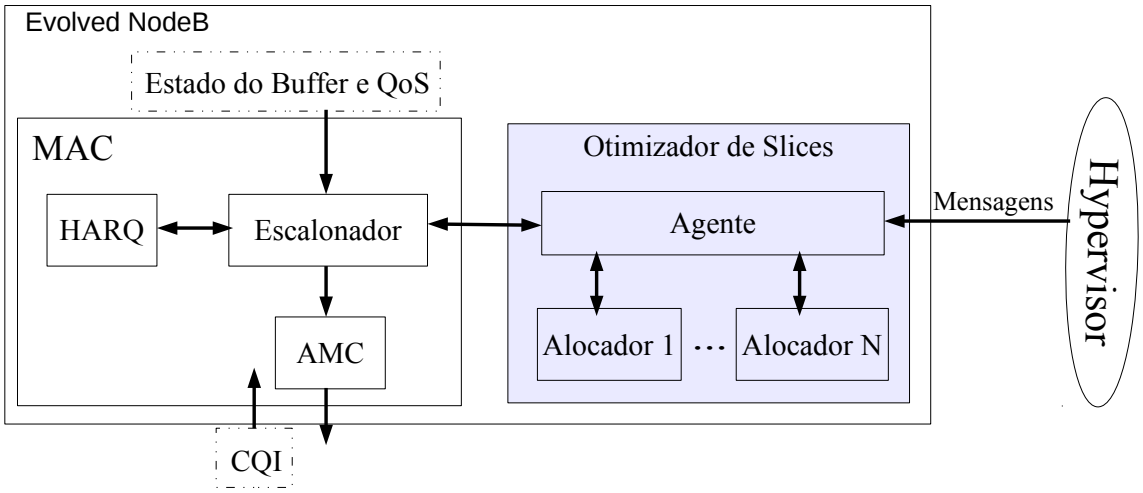


Figura 2. Otimizador de *Slices* e entidades relevantes para a transmissão de tráfego *downlink*.

A Figura 2 ilustra o Otimizador de *Slices*, representado pela cor azul, assim como as entidades da camada MAC relevantes para a transmissão de dados *downlink*. O OS precisa ser inserido em cada um dos eNBs na rede para que eles possam comunicar com o *hypervisor*. O OS possui um módulo Agente e vários Alocadores, onde a quantidade de Alocadores é a mesma do número de SPs atuando na rede. O *hypervisor*, além de entregar os DSs e o MA de cada SP ao Agente, também envia uma requisição para a criação de um novo Alocador sempre que um SP é adicionado à rede. Esse novo Alocador estará vinculado ao SP e, portanto, sempre que o Agente receber os DSs e o MA de um SP, ambos os parâmetros serão entregues ao Alocador associado ao respectivo SP.

O Agente recebe do Escalonador dois parâmetros: o subquadro de rádio atual e um conjunto contendo o estado do *buffer* de cada fluxo na rede. A partir desses dois parâmetros, o Agente define o SP responsável por transmitir nesse subquadro e, subsequentemente, envia uma permissão de transmissão e o subquadro atual de rádio ao Alocador que corresponde ao SP escolhido. O Alocador ao receber essa permissão e o subquadro escolhe o *slice* a ser escalonado no momento. Essa escolha é feita tendo como base os DSs e o MA do respectivo SP. Após a escolha do *slice*, o Alocador envia ao Agente um conjunto de tuplas contendo os fluxos que pertencem ao *slice* escolhido. O Agente, ao receber esse conjunto, envia-o ao Escalonador para que o mesmo aloque os blocos de recurso aos UEs do conjunto.

4.2. Detalhes de Implementação

A implementação da arquitetura e do OS foi realizada no NS-3. O NS-3 é um simulador de rede de eventos discretos desenvolvido em C++. O NS-3 implementa as redes LTE no módulo LENA [LENA 2017]. A nossa implementação é basicamente composta por seis classes C++. São elas: OS, Alocador, *hypervisor*, PI, controlador vSDN e orquestrador. O NS-3 não fornece um protocolo para a comunicação do eNB com uma entidade externa à rede LTE. Dessa forma, a comunicação do OS com o *hypervisor*, bem como entre o *hypervisor* e os controladores vSDNs, é feita por meio de chamadas de função C++. Similarmente, a API *Northbound* exposta pelos controladores vSDNs e a comunicação entre o *hypervisor* e o PI são chamadas de funções C++. Em nossa implementação, um *slice* é um conjunto de fluxos pertencendo ao mesmo tipo de aplicação. Um *slice* contém fluxos de apenas um único SP, não podendo ser compartilhado com outro SP. Foi implementada a comunicação entre o *hypervisor* e um único eNB. Além disso, não há comunicação entre o *hypervisor* e o EPC, visto que o foco deste trabalho é no provisionamento de *slices* na rede de acesso.

O PI é responsável por gerenciar a infraestrutura de rede. Atualmente, a única função implementada no PI é o provisionamento de recursos definidos nos SLAs. Apenas os recursos de rádio no canal *downlink* são ofertados aos SPs. Ou seja, no SLA de cada SP é definido o identificador e uma quantidade mínima de subquadros que podem ser disponibilizados aos SPs. Dessa forma, os 10 subquadros de rádio são divididos entre os SPs a partir da análise dos SLAs, e armazenados no VDM. Em nossa implementação, definimos dois SLAs. Um deles para um SP que proverá conexão de rede a veículos e um outro SLA para um SP oferecendo os recursos de rede às pessoas participando de um evento. Ambos os SPs operam na mesma região. No SLA do SP responsável pelos veículos foram garantidos no mínimo quatro subquadros, enquanto que no outro foram assegurados no mínimo seis subquadros. Com isso, o PI por meio de sua função de provisionamento de recursos (MANO) cria o VDM, que sempre terá dez posições, e distribui aleatoriamente os identificadores dos SPs nele, sempre respeitando o número de subquadros definidos nos SLAs. Uma vez que em nossa implementação o número de subquadros requisitados pelos SPs é o mesmo do total de subquadros de um quadro, ou seja, 10, os SPs receberão o mínimo de subquadros definidos nos SLAs.

Após a criação e o preenchimento do VDM, o PI envia-o ao *hypervisor*, que por sua vez salva uma cópia desse vetor, e o envia ao OS. Ademais, o *hypervisor* conta quantas vezes o identificador de cada SP apareceu no VDM, enviando ao controlador vSDN de cada SP sua respectiva soma. A partir dessa soma, o controlador vSDN saberá o número de subquadros disponibilizados a ele. O controlador vSDN não terá acesso aos índices dos subquadros alocados a ele pelo PI, sendo essa uma informação escondida pelo *hypervisor*. Ao receber o número de subquadros disponibilizados, o orquestrador, que é um objeto instanciado pelo controlador vSDN, cria e aloca os subquadros aos *slices* baseando-se nas políticas definidas pelos operadores de rede e na popularidade dos *templates*. O orquestrador gera como saída os DSs e o VA. O VA é um vetor de N posições, onde N é o número de subquadros disponibilizados ao controlador vSDN pelo PI. Nesse VA, os identificadores dos *slices* são distribuídos de acordo com a alocação computada pelo orquestrador. Dessa forma, se um *slice* receber três subquadros, então três das N posições disponíveis no VA serão alocadas para esse *slice*. Tanto os DSs quanto o VA são enviados ao *hypervisor* pelo controlador vSDN.

Em nossa implementação, o SP responsável pela rede de veículos instancia dois

slices, um para voz e um outro para vídeo. Cada *slice* recebe dois subquadros. Já o SP responsável pelas pessoas participando de um evento instancia dois *slices*, um para tráfego TCP e um outro para vídeo. O primeiro recebe dois subquadros, enquanto que o segundo recebe quatro subquadros. Cada *slice* possui um DS, que contém os seguintes campos: o identificador do *slice*, a prioridade do *slice* e um conjunto contendo os IMSIs de todos os UEs pertencentes ao *slice*. É possível que um UE esteja em mais de um *slice* ao mesmo tempo. Nesse caso, o QCI seria usado para distinguir os *slices*. Entretanto, o NS-3 não suporta completamente o escalonamento por fluxo. Dessa forma, mesmo que diferentes *bearers* fossem empregados a um mesmo UE, o que representaria *slices* distintos, apenas um *bearer* seria usado pelo Escalonador. Portanto, em nossa implementação assumimos que um UE pertence a apenas um *slice*.

Ao receber os DSs e o VA do controlador vSDN, o *hypervisor* constrói o conjunto de pares chave-valor, o MA, a partir do VA e o VDM. No MA, a chave é o subquadro e o valor é o identificador do *slice*. Cada SP possui seu próprio MA. Os DSs e o MA são encapsulados em uma mensagem e enviados ao OS, mais especificamente ao Agente. Nessa mensagem, também há o identificador do SP responsável por gerar os DSs. Após receber essa mensagem, o Agente envia os DSs e o MA ao Alocador vinculado ao controlador vSDN. Em cada TTI, o OS recebe do Escalonador dois parâmetros: o subquadro atual e um vetor contendo o estado do *buffer* de cada UE. Esse *buffer* é atualizado pela entidade RLC sempre que dados entram ou saem do *buffer* RLC.

O Algoritmo 1 mostra o processo de seleção realizado pelo OS para a escolha do *slice* a ser escalonado. Em cada TTI, o Escalonador *downlink* requisita ao Agente um *slice* a ser escalonado. O Agente, por sua vez, chama o procedimento *RetornarMelhorSlice*, passando como parâmetros o subquadro atual ($subQuadro$) e o vetor contendo o estado dos *buffers* de todos os UEs ($vetorEB$). Nas linhas 2-4, o algoritmo seleciona o SP definido nesse subquadro e verifica se algum UE pertencente a algum dos *slices* definidos por esse SP tem dados a serem transmitidos (linhas 14-19). O algoritmo vai para a linha 6 se algum UE tiver dados a serem transmitidos; caso contrário, vai para a linha 8. Na linha 6, o algoritmo seleciona o *slice* a ser escalonado nesse subquadro e retorna ao Escalonador (linha 7) os UEs que pertencem a esse *slice*.

O melhor *slice* é computado pelo procedimento *SelecionaSlice* (linhas 21-34). Neste procedimento, o *slice* escolhido é o definido na coleção de pares chave-valor da linha 22 ($mapaAlocacao$) para esse subquadro. Entretanto, caso nenhum dos UEs pertencentes a esse *slice* tenha dados a serem transmitidos, o procedimento seleciona o *slice* com maior prioridade que tenha dados a transferir.

No laço *for* das linhas 8-12 o algoritmo escolhe o *slice* do SP de maior prioridade que tenha dados a transmitir. Na linha 12, o algoritmo retorna o conjunto de UEs (IMSIs) pertencentes ao *slice* para o Escalonador. Após receber o conjunto de IMSIs, o Escalonador converte esse conjunto para uma coleção de *Radio Network Temporary Identifiers* (RNTIs), pois os escalonadores do NS-3 não trabalham com o IMSI. Posteriormente, o Escalonador aloca os recursos de rádio *downlink* aos UEs de acordo com sua política.

Conforme pode ser visto no Algoritmo 1, o SP definido pelo PI no subquadro atual sempre tem maior prioridade; entretanto, caso nenhum UE pertencente aos *slices* instanciados por esse SP tenha dados a serem transmitidos, um outro SP será escolhido para transmitir nesse subquadro. Dessa forma, os recursos de rádio são usados mais eficientemente, uma vez que o Escalonador vai alocar recursos para um *slice* que realmente

Algorithm 1 Seleciona o slice a ser escalonado nesse subquadro.

```
1: procedure RETORNARMELHORSLICE(subQuadro, vetorEB)
2:     spAlocadoID ← vetorMapeamento[subQuadro]
3:     spAlocado ← vetorSPs[spAlocadoID]
4:     resultado ← VERIFICASITUACAOBUFFER(spAlocado.imsis, vetorEB)
5:     if resultado = verdadeiro then
6:         slice ← SELECIONASLICE(subQuadro, vetorEB)
7:         return slice.vetorIMSIS
8:     for each sp in vetorSPsOrdenadoDecresPorPrioridade − {spAlocado} do
9:         resultado ← VERIFICASITUACAOBUFFER(sp.imsis, vetorEB)
10:        if resultado = verdadeiro then
11:            slice ← SELECIONASLICE(subQuadro, vetorEB)
12:            return slice.vetorIMSIS
13:
14: procedure VERIFICASITUACAOBUFFER(vetorIMSIS, vetorEB)
15:    for each imsi in vetorIMSIS do
16:        tamanhoBuffer ← vetorEB[imsi]
17:        if tamanhoBuffer > 0 then
18:            return verdadeiro
19:    return falso
20:
21: procedure SELECIONASLICE(subQuadro, vetorEB)
22:    sliceIDAtual ← mapaAlocacao[subQuadro]
23:    sliceAtual ← descritoresSlices[sliceIDAtual]
24:    if sliceAtual = Nil then
25:        sliceIDAtual ← vetorSlicesOrdenadoDecresPorPrioridade[0]
26:        sliceAtual ← descritoresSlices[sliceIDAtual]
27:    resultado ← VERIFICASITUACAOBUFFER(sliceAtual.imsis, vetorEB)
28:    if resultado = verdadeiro then
29:        return sliceAtual
30:    for each sliceID in vetorSlicesOrdenadoDecresPorPrioridade − {sliceIDAtual} do
31:        descSlice ← descritoresSlices[sliceID]
32:        resultado ← VERIFICASITUACAOBUFFER(descSlice.imsis, vetorEB)
33:        if resultado = verdadeiro then
34:            return descSlice
```

tenha dados a serem transmitidos.

5. Avaliação de Desempenho

Nesta seção é avaliado o desempenho de nossa proposta usando dois esquemas de *network slicing*, estático e dinâmico. No esquema estático, o OS, após escolher o SP a ser servido neste subquadro, a partir da alocação dos subquadros feita pelo PI, seleciona um *slice* desse SP a ser escalonado nesse subquadro. Mesmo que nenhum dos *slices* desse SP tenha dados a serem transmitidos neste subquadro, ainda assim um de seus *slices* precisa ser fornecido ao escalonador. No esquema dinâmico é dada prioridade de escalonamento a um dos *slices* pertencentes ao SP definido pelo PI neste subquadro. Entretanto, se nenhum dos *slices* desse SP possuir dados a serem transmitidos neste subquadro, o OS escolhe um SP de maior prioridade em que pelo menos um dos *slices* desse SP tenha tráfego a ser transmitido. Após escolher o SP, o OS seleciona o *slice* e envia-o ao escalonador. O esquema dinâmico é o definido no Algoritmo 1.

5.1. Configuração do Cenário

O cenário de simulação contém um eNB, dois SPs (veículos e evento), um EPC e um hospedeiro responsável por gerar tráfego de *downlink* na rede. Há diversos UEs na rede,

variando de 20 a 50, com incrementos de 10. O *framework Simulation of Urban MObility* (SUMO) [3], versão 0.31.0, foi usado para gerenciar a mobilidade dos veículos. Foi empregado um trecho do cenário de Manhattan Grid, contendo 5 ruas de 2 pistas. Tais ruas são espaçadas por 100 metros, o que totaliza uma área de 160.000 m^2.

O eNB está localizado no centro do cenário. Os UEs são uniformemente distribuídos no cenário. Um UE recebe apenas um tipo de tráfego. Os tipos de tráfego são: VoIP, FTP e Vídeo. Há 4 *slices* na rede, dois para cada SP. Os *slices* 1 e 2 pertencem ao SP responsável pelos veículos (SP-VEIC), enquanto que os *slices* 3 e 4 são do SP responsável pelo evento (SP-EVEN). Quatro subquadros de rádio foram alocados ao SP-VEIC, e seis subquadros ao SP-EVEN. Todos os participantes do evento andam a pé. Todo o tráfego tem como origem o hospedeiro, que está conectado ao EPC, e como destino os UEs. A largura de banda e o atraso da conexão cabeada entre o hospedeiro e o EPC são de 100Gb/s e 1 ms, respectivamente. O tamanho do *buffer* RLC é de 150 KBytes, que é o recomendado para dispositivos da Categoria 1 [3GPP 2017]. Dispositivos da Categoria 1 suportam apenas SISO (*Single Input, Single Output*), que são aqueles usados na simulação. As Tabelas 1 e 2 sumarizam o modelo do tráfego e a descrição dos *slices* empregados na simulação.

Tabela 1. Modelo de Tráfego.

Aplicação	VoIP	FTP	Vídeo
Descrição	G.711[1]	TCP Bulk[2]	MPEG 4[3]
BitRate	64 Kbps	N/A	742 Kbps
QCI	1	9	4

[1] Modelo On/Off. Tempo On = 0.352; Tempo Off = 0.650
[2] Definido em: http://www.nsnam.org/doxygen/tcp-bulk-send_8cc_source.html. Foi empregado o TCP Vegas.
[3] Baseado em trace. Nós usamos o trace ARD NEWS, que está disponível em: http://www2.tkn.tu-berlin.de/research/trace/ltvt.html.

Tabela 2. Descrição dos Slices.

ID do Slice	1	2	3	4
Aplicação	VoIP	Vídeo	FTP	Vídeo
Dono do Slice	VEÍC.	VEÍC.	EVENTO	EVENTO
Subquadros	{3, 5}	{7, 10}	{1, 6}	{2, 4, 8, 9}
Porcentagem de UEs	75%	10%	5%	10%

Foram empregados o modelo de propagação Kun e um *trace* do modelo de *fading* para cenários suburbanos. O modelo de *fading* foi gerado pelo *script fading-trace-generator.m*, que é disponibilizado pelo NS-3. A entidade HARQ está ativada. A Tabela 3 expõe os principais parâmetros usados na simulação.

Tabela 3. Parâmetros da Simulação.

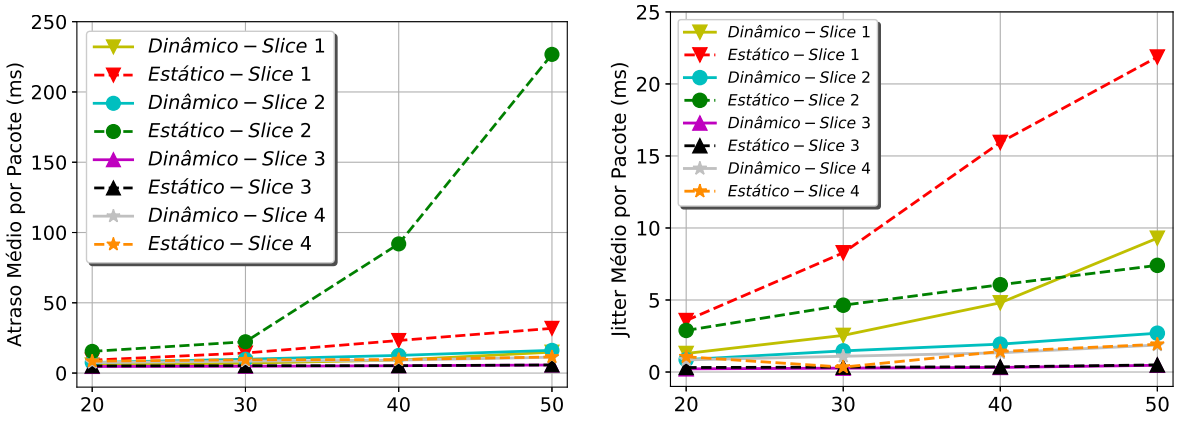
Parâmetro	**Valor**
Número de eNBs	1
Área do Cenário	160.000 m^2
Frequência do Sistema	2655 MHz
Modelo de Propagação	Kun 2600Mhz
Potência de TX eNB	34 dBm
Potência de TX UE	10 dBm
Largura de Banda do Sistema	10 MHz
Escalonador Downlink	Proportional Fair
Modo RLC	Não Reconhecido (Unacknowledged Mode - UM)
Tamanho do Buffer RLC	150 KBytes
Velocidade dos UEs	3 Km/h (A pé); 60 km/h (veículo)
Número de Replicações	34
Duração da Simulação	150 s

5.2. Resultados

As figuras presentes nesta seção ilustram os valores médios por UE para as seguintes métricas de QoS: atraso, *jitter*, perdas e vazão. Todas essas métricas são expostas como função do número de UEs na rede. Foram realizados 8 experimentos, um para cada combinação de esquema de *slicing* e número de UEs na rede. Cada experimento foi executado 34 vezes com sementes distintas. Dessa forma, foram executadas 272 simulações. Nas figuras, as linhas contínuas representam o *slicing* dinâmico, enquanto que as linhas

[3]http://sumo.dlr.de/index.html

tracejadas correspondem ao *slicing* estático. Os intervalos de confiança de 95% são mostrados nas figuras. Note que eles são bem pequenos e imperceptíveis nas figuras. Em todas as figuras, os *Slices* 1 e 2 pertencem ao SP-VEIC, enquanto que os *Slices* 3 e 4 são do SP-EVEN.



(a) Atraso Médio por Pacote.

(b) *Jitter* Médio por Pacote .

Figura 3. Atraso e Jitter médio de cada pacote por UE em cada *Slice* como função do número de UEs na rede.

A Figura 3(a) apresenta o atraso médio de cada pacote por UE. Como pode ser observado na figura, tanto o *slicing* estático quanto o *slicing* dinâmico possuem resultados parecidos nos *slices* pertencentes ao SP-EVEN. Isso ocorre, pois os 6 subquadros alocados são suficientes para o tráfego do SP-EVEN, mesmo no cenário com 50 UEs na rede. Entretanto, os dispositivos dos *slices* pertencentes ao SP-VEIC têm o atraso de seus pacotes aumentado consideravelmente no *slicing* estático, em especial na rede com 50 UEs.

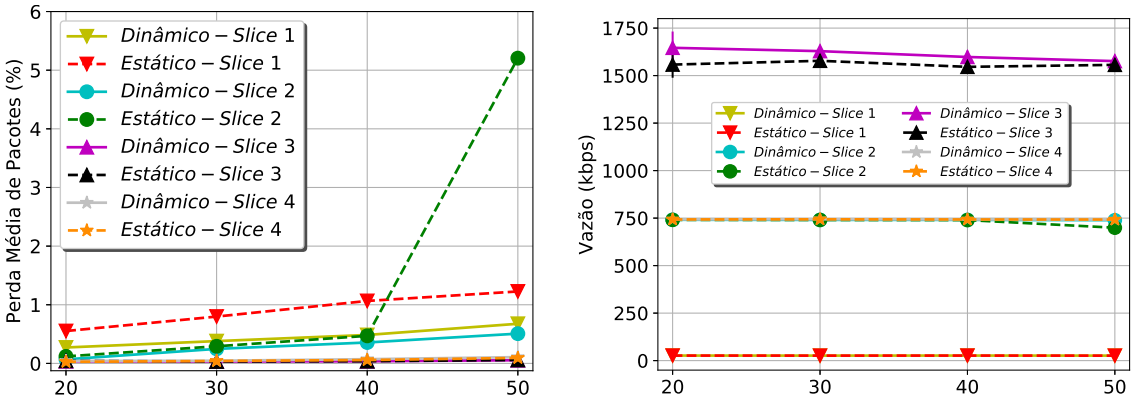
O *slicing* estático não permite que os subquadros alocados ao SP-EVEN possam ser usados pelos *slices* do SP-VEIC quando nenhum dos *slices* pertencentes ao SP-EVEN tenha dados a serem transmitidos. O *slicing* dinâmico, por outro lado, permite que o SP-VEIC use os recursos não usados pelo SP-EVEN, fazendo com que os pacotes do SP-VEIC sejam servidos pelo escalonador de modo mais rápido e, portanto, é reduzido consideravelmente o atraso percebido pelos UEs pertencentes ao SP-VEIC. Os UEs pertencentes ao *Slice* 2 são os que mais sofrem com o *slicing* estático, visto que seus pacotes carregam mais *bytes* em comparação com os pacotes do *Slice* 1, ocasionando um maior tempo de escalonamento. No cenário com 50 UEs, o atraso médio dos pacotes do *Slice* 2 é aproximadamente 16 vezes maior (226 ms contra 16 ms) em comparação com os pacotes do *Slice* 2 no *slicing* dinâmico.

A Figura 3(b) expõe o *jitter* médio de cada pacote por UE. De modo similar à Figura 3(a), o *slicing* estático afeta negativamente os fluxos pertencentes ao SP-VEIC. Entretanto, diferentemente da Figura 3(a), o *Slice* 1 é o mais afetado pelo *slicing* estático. O *Slice* 1 também tem o pior desempenho em relação ao *jitter* dentre os 4 *slices* no *slicing* dinâmico. Esse maior *jitter* no *Slice* 1 deve-se a um conjunto de fatores. Primeiro, a maior prioridade de escalonamento do Slice 2 em relação ao Slice 1, o que faz com que no *slicing* dinâmico, o SP-VEIC priorize a alocação de recursos de subquadros não utilizados pelo SP-EVEN ao *Slice* 2.

Além disso, 75% dos UEs transmitem tráfego VoIP, o que faz com que em alguns

subquadros de rádio muitos sejam os pacotes VoIP gerados ao mesmo tempo pelos fluxos VoIP, o que aumentaria o atraso de tais pacotes, visto o maior tempo para escaloná-los. Entretanto, em vários instantes poucos são os pacotes VoIP gerados ao mesmo tempo, acarretando um atraso menor a esses pacotes. Dessa forma, existem pacotes VoIP que são servidos rapidamente, enquanto alguns outros lentamente, o que, consequentemente, aumenta o *jitter*. No cenário com 50 UEs, o *jitter* médio no *Slice* 1 do *slicing* estático é cerca 2,3 vezes maior que o mesmo *slice* no *slicing* dinâmico (21 ms contra 9 ms).

A Figura 4(a) ilustra a perda média de pacotes por UE. De modo similar às Figuras 3(a) e 3(b), os *slices* pertencentes ao SP-VEIC são afetados negativamente pelo *slicing* estático. De modo especial, o *Slice* 2 tem uma perda de pacotes bastante significativa quando há 50 UEs na rede. Conforme visto na Figura 3(a), no *slicing* estático, o atraso médio dos pacotes do *Slice* 2 aumenta consideravelmente à medida que cresce o número de UEs na rede. Com o aumento do atraso, mais pacotes vão sendo enfileirados nos *buffers* RLC do eNB, os quais são responsáveis por armazenar temporariamente os pacotes dos UEs. Os pacotes ficam nesses *buffers* até serem servidos pelo escalonador; ou, no pior caso, até serem descartados.



(a) Perda Média de Pacotes.

(b) Vazão Média.

Figura 4. Perdas de Pacotes e Vazão por UE em cada *Slice* como função do número de UEs na rede.

Como os pacotes do *Slice 2*, no *slicing* estático, tem um alto atraso e, além disso, o tamanho dos pacotes de vídeos carregam muitos *bytes*, os *buffers* RLC ficam cheios. Quando esses *buffers* enchem, pacotes precisam ser descartados, conforme mostrado na Figura 4(a). Por outro lado, no *slicing* dinâmico, as perdas de pacotes são relativamente baixas. O *Slice* 2 no cenário com 50 UEs, por exemplo, tem no *slicing* estático uma perda de aproximadamente 5,2% de pacotes, enquanto que no *slicing* dinâmico apenas 0,5% dos pacotes são descartados, uma quantidade dez vezes menor.

A Figura 4(b) apresenta a vazão média dos UEs. De modo geral, a vazão média dos UEs em cada *slice*, independente do esquema de *slicing* usado, é basicamente a mesma. Embora aparentemente o *slicing* dinâmico ofereça uma vazão superior, de modo especial no *Slice* 3, essa vantagem do *slicing* dinâmico sobre o *slicing* estático não é estatisticamente significativa. Entretanto, quando há 50 UEs no cenário, a vazão média dos UEs no *Slice* 2 usando o *slicing* dinâmico é ligeiramente superior que no *slicing* estático. Isso ocorre, pois como visto na Figura 4(a), o *slicing* estático aumenta consideravelmente a quantidade de pacotes perdidos no *Slice* 2 em comparação com o *slicing* dinâmico. Dessa forma, a vazão dos UEs no *Slice* 2 precisa ser necessariamente menor no *slicing*

estático, o que realmente acontece, conforme exposto na Figura 4(b). Vale mencionar que no *Slice* 1, em ambos os esquema de *slicing*, a vazão é de no máximo 23 Kbps, uma vez que a aplicação VoIP só transmite dados quando no estado ON.

A partir dos resultados obtidos neste trabalho, pode-se concluir que o esquema de *slicing* dinâmico é superior ao *slicing* estático. Tendo em vista que o estado de uma rede de comunicação de dados altera-se com frequência, bem como nem sempre os operadores de rede aplicam uma política de alocação de recursos correta, o Otimizador de *Slices*, ao utilizar o *slicing* dinâmico, adapta os *slices* ao estado da rede, permitindo que recursos de rádio não usados por SPs sejam alocados a outros SPs que necessitam de tais recursos e, consequentemente, há uma melhora na QoS ofertada ao usuário final.

6. Conclusões e Trabalhos Futuros

Neste artigo foi apresentado o Otimizador de *Slices*, um componente que realiza o conceito de *network slicing* no canal *downlink* das redes LTE para múltiplos Provedores de Serviço. Os Provedores de Serviço constroem seus *slices* e definem o escalonamento deles por meio de um módulo de orquestração próprio, que reside em um controlador vSDN. A partir de informações recebidas sobre *slices* do controlador vSDN e do estado da rede, o Otimizador de *Slices* seleciona o melhor *slice* a ser escalonado no momento. Simulações foram realizadas para mostrar que o Otimizador de *Slices* usa eficientemente os recursos de rádio *downlink* e, consequentemente, melhora a QoS percebida pelos usuários finais.

Como trabalhos futuros, nós planejamos permitir ao Otimizador de *Slices* manipular *slices* de múltiplos SPs no canal de rádio *uplink*. Além do mais, esperamos integrar funções virtualizadas de rede nos *slices* e avaliar o *overhead* causado pela comunicação entre os componentes de nossa arquitetura.

Agradecimentos

Os autores gostariam de agradecer a CAPES pela bolsa de doutorado. Este trabalho é parte do INCT sobre Internet do Futuro para Cidades Inteligentes (CNPq 465446/2014-0, CAPES 88887.136422/2017-00 e FAPESP 2014/50937-1).

Referências

3GPP (2017). Disponível em: http://http://www.etsi.org/deliver/etsi_ts/136300_136399/136306/10.02.00_60/.

Al-Shabibi, A., De Leenheer, M., Gerola, M., Koshibe, A., Parulkar, G., Salvadori, E., and Snow, B. (2014). OpenVirteX: Make Your Virtual SDNs Programmable. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 25–30, New York, NY, USA. ACM.

Blenk, A., Basta, A., Reisslein, M., and Kellerer, W. (2016). Survey on Network Virtualization Hypervisors for Software Defined Networking. *IEEE Communications Surveys Tutorials*, 18(1):655–685.

Caballero, P., Banchs, A., de Veciana, G., and Costa-Pérez, X. (2017). Network slicing games: Enabling customization in multi-tenant networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9.

Chartsias, P. K., Amiras, A., Plevrakis, I., Samaras, I., Katsaros, K., Kritharidis, D., Trouva, E., Angelopoulos, I., Kourtis, A., Siddiqui, M. S., Viñes, A., and Escalona,

E. (2017). SDN/NFV-based end to end network slicing for 5G multi-tenant networks. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–5.

Choyi, V. K., Abdel-Hamid, A., Shah, Y., Ferdi, S., and Brusilovsky, A. (2016). Network slice selection, assignment and routing within 5G Networks. In *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–7.

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.

Grøndalen, O., Zanella, A., Mahmood, K., Carpin, M., Rasool, J., and Østerbø, O. N. (2017). Scheduling Policies in Time and Frequency Domains for LTE Downlink Channel: A Performance Comparison. *IEEE Transactions on Vehicular Technology*, 66(4):3345–3360.

Haque, I. T. and Abu-Ghazaleh, N. (2016). Wireless Software Defined Networking: A Survey and Taxonomy. *IEEE Communications Surveys Tutorials*, 18(4):2713–2737.

Hu, M., Chang, Y., Sun, Y., and Li, H. (2016). Dynamic slicing and scheduling for wireless network virtualization in downlink LTE system. In *2016 19th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 153–158.

Kamel, M. I., Le, L. B., and Girard, A. (2014). LTE Wireless Network Virtualization: Dynamic Slicing via Flexible Scheduling. In *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pages 1–5.

Kokku, R., Mahindra, R., Zhang, H., and Rangarajan, S. (2013). CellSlice: Cellular wireless resource slicing for active RAN sharing. In *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–10.

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76.

LENA (2017). Disponível em: http://networks.cttc.es/mobile-networks/software-tools/lena/.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Mijumbi, R., Serrat, J., Gorricho, J. L., Bouten, N., Turck, F. D., and Boutaba, R. (2016). Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262.

NS-3 (2017). Disponível em: https://www.nsnam.org.

Parsaeefard, S., Jumba, V., Derakhshani, M., and Le-Ngoc, T. (2015). Joint resource provisioning and admission control in wireless virtualized networks. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2020–2025.

Rezende, P. and Madeira, E. (2018). An adaptive network slicing for LTE Radio Access Networks. In *2018 Wireless Days*.

Samdanis, K., Costa-Perez, X., and Sciancalepore, V. (2016). From network sharing to multi-tenancy: The 5G network slice broker. *IEEE Communications Magazine*, 54(7):32–39.