Controle de Admissão para *Network Slicing* Ciente de Recursos de Rede e de Processamento

Henrique Valle de Lima, Gustavo Zanatta Bruno, Felipe H. Grings, Cristiano Bonato Both, Antonio M. Alberti, Kleber Vieira Cardoso, Sand Luz Correa

***Resumo*— Neste trabalho, propomos dois algoritmos de controle de admissão para *Network Slicing* cientes de recursos de rede e de processamento e que utilizam a técnica de *overbooking* para aumentar a utilização da infraestrutura das redes 5G. Avaliamos os algoritmos propostos e comparamos com uma solução ótima, bem como um algoritmo estado da arte ciente apenas dos recursos de rede. Resultados mostram que os algoritmos propostos alcançam recompensas próximas da solução ótima, respeitando a restrição de capacidade de rede e de processamento e incorrendo em violações de SLA em menos de 1% dos casos.**

***Palavras-Chave*— Controle de admissão, *Slicing*, MEC.**

***Abstract*— In this work, we propose two admission control algorithms for *Network Slicing* aware of network and processing resources and that use *overbooking* techniques to increase the utilization of the 5G network infrastructure. We evaluate the proposed algorithms and compare them with an optimal solution, as well as a state of the art algorithm aware only of network resources. Results show that the proposed algorithms achieve rewards close to the optimal solution, respecting the network and processing capacity constraint and incurring SLA violations in less than 1% of cases.**

***Keywords*— Admission control, Network Slicing, MEC.**

I. Introduction

Network Slice (NS) é um habilitador fundamental para que as redes 5G e pós-5G suportem um grande espectro de serviços com diferentes requisitos de desempenho, banda, confiabilidade e latência. Ao fazer uso de tecnologias como *Network Function Virtualisation* (NFV) e *Software Defined Networking* (SDN), NS permite que operadores de rede criem e operem múltiplas redes lógicas isoladas e independentes, denominadas *slices*, ou simplesmente fatias, sobre uma infraestrutura física única e compartilhada. Cada fatia consiste em um conjunto de *Virtualized Network Functions* (VNFs), instanciadas para atender requisitos e *Service Level Agreements* (SLAs) de um serviço específico [1]. De fato, ao abrir a infraestrutura para múltiplos provedores de serviços, denominados inquilinos, o paradigma de NS cria novas oportunidades econômicas para o operador de rede, que pode aumentar a utilização da sua infraestrutura admitindo o maior número possível de fatias, enquanto explora os ganhos da multiplexação estocástica. Neste contexto, um mecanismo de controle de admissão de fatias se torna necessário para garantir que os SLAs das fatias ativas serão satisfeitos, ao mesmo tempo que os recursos da infraestrutura serão utilizados da forma mais eficiente possível.

Henrique Valle de Lima, Sand Luz Correa e Kleber Vieira Cardoso, Instituto de Informática, Universidade Federal de Goiás, Goiânia-GO, e-mail: henrivalle@discente.ufg.br, sand,kleber@ufg.br. Gustavo Zanatta Bruno, Felipe H. Grings e Cristiano Bonato Both, Programa de Pós-Graduação em Computação Aplicada, Unisinos, São Leopoldo-RS, e-mail: zanattabruno, felipehg@edu.unisinos.br, cbboth@unisnos.br. A. M. Alberti, Information and Communications Technology Laboratory, Instituto Nacional de Telecomunicações, Santa Rita do Sapucaí - MG, e-mail: alberti@inatel.br

Nos últimos anos, alguns mecanismos de controle de admissão de fatias foram propostos na literatura visando maximizar o lucro do operador de rede. Em [2], o controle de admissão de fatias é introduzido como um problema NP-difícil. Os autores em [3] apresentam uma solução prática para o controle de admissão de fatias baseada em um modelo simples de precificação. O trabalho em [4] analisa matematicamente o problema de admissão de fatias. Os autores também apresentam uma solução baseada em aprendizado de máquina para o problema, o qual se apoia em uma abordagem *offline*. No entanto, ao decidir sobre a admissão de uma fatia na infraestrutura, o operador de rede deve levar em consideração não apenas fatores determinísticos, como os requisitos das fatias já admitidas e ativas e a capacidade total do sistema, mas também fatores estocásticos como os requisitos de requisições futuras e a utilização efetiva de recursos nas fatias ativas.

Neste contexto, um processo de decisão on-line se torna a solução mais apropriada para tratar o problema de controle de admissão em NS. Dessa maneira, os autores em [5] propõem o algoritmo *Online Network Slice Broker* (ONETS), um mecanismo on-line de controle de admissão de fatias baseado no problema *Multi-Armed Bandit* (MaB). O ONETS maximiza os ganhos de multiplexação do operador usando uma estratégia controlada de *overbooking*. Apesar de ser o estado da arte em controle de admissão para NS, o ONETS possui uma grande lacuna. Ao decidir o conjunto de fatias que devem ser admitidas no sistema, o ONETS considera apenas recursos de rádio. No entanto, os autores em [6] evidenciam quem em 5G e pós-5G, os recursos de computação também são necessários para executar as diversas VNFs que compõem os serviços.

Para preencher essa lacuna é proposto neste trabalho uma extensão para o ONETS denominada *Online Network Slicing with MEC and Overbooking by Demand* (M-ONETS-OBD), um algoritmo que leva em consideração recursos de rádio e recursos de computação. Propõe-se também uma variação do M-ONETS-OBD, denominada *Online Network Slicing with MEC and Overbooking by Empirical Rule* (M-ONETS-OBS) onde a estratégia de *overbooking* é suavizada para reduzir as chances de violações de SLAs. Resultados mostram que os algoritmos propostos alcançam recompensas próximas da solução ótima, respeitando a restrição de capacidade de rede e de processamento e incorrendo em violações de SLA em menos de 1% dos casos.

Este trabalho está organizado da seguinte forma. Na Seção II, descrevemos e reformulamos nosso modelo de controle de admissão. Na Seção III, apresentamos os algoritmos on-lines M-ONETS-OBD e M-ONETS-OBS propostos como solução para o problema apresentado. O modelo de controle de admissão proposto é avaliado numericamente na Seção IV. Finalmente, na Seção V, apresentamos as considerações finais.

II. Formulação do Problema

Neste trabalho, considera-se que um provedor de serviços de telecomunicações (operador de rede) deseja tornar sua infraestrutura disponível para clientes externos (inquilinos). É considerado apenas um segmento de *Radio Access Network* (RAN), que contém recursos de rádio e de processamento (e.g., servidores *Multi-access Edge Computing* (MEC)). Assume-se que a *Base Station* (BS) oferece suporte para NS, com uma capacidade total C, dada em número de *Physical Resource Block* (PRBs), e denota-se por M a capacidade total de processamento no servidor MEC, expressa em *CPU ticks*.

Considera-se $\mathcal{I} = \{1, 2, \cdots, |\mathcal{I}|\}$ um conjunto de inquilinos conhecidos previamente que submetem requisições para criação e utilização de NS. Seja $\mathcal{S} = \{1, 2, \cdots, |\mathcal{S}|\}$ o conjunto de *templates* (modelos) de fatias disponibilizados pelo operador. Cada inquilino $i \in \mathcal{I}$ pode requisitar uma fatia $s \in \mathcal{S}$ que melhor atenda aos requisitos do serviço por ele prestado. Cada *template* de uma fatia $s = \langle R^{(s)}, Q^{(s)}, L^{(s)} \rangle$ especifica uma quantidade $R^{(s)}$ de recursos de rádio, dada em número de PRBs, uma quantidade $Q^{(s)}$ de recursos computacionais, em *CPU ticks*, e um tempo $L^{(s)}$ de duração da fatia, em segundos.

Seja $\mathcal{T} = (1, 2, \cdots, |\mathcal{T}|)$ a sequência finita dos instantes de tempo em que um mecanismo de controle de admissão precisa decidir sobre quais das requisições em curso aceitar. Cada inquilino $i \in \mathcal{I}$ pode solicitar uma fatia $s \in \mathcal{S}$ em um instante $t \in \mathcal{T}$. Denota-se tal requisição por $r_{i,t}^{(s)} = \langle R_{i,t}^{(s)}, Q_{i,t}^{(s)}, L_{i,t}^{(s)} \rangle$. Neste estudo, a distribuição é modelada ao longo do tempo, das requisições de um inquilino $i \in \mathcal{I}$ como uma variável aleatória, de forma que o tempo Δ_t decorrido entre chegadas consecutivas seja distribuído exponencialmente com taxa ϕ_i. O valor de ϕ_i é obtido a partir de uma distribuição de Pareto com média ρ e desvio padrão ζ.

Considera-se ainda que cada inquilino $i \in \mathcal{I}$ ofereca um único tipo de serviço, como jogos on-line, vídeo por *streaming*, geolocalização e mapas. Existem diferentes *templates* de fatias para o mesmo serviço, que diferem entre si pela quantidade de recursos alocados. Um inquilino $i \in \mathcal{I}$ sempre requisita um NS para o mesmo tipo de serviço, contudo, a quantidade de recursos solicitados pode variar a cada requisição. Assume-se ainda que cada inquilino pode requisitar somente uma fatia por vez. Ou seja, novas solicitações para um inquilino i não podem ser aceitas enquanto i possui uma fatia ativa no sistema.

Denota-se o problema do controle de admissão como: a cada instante t, o operador deve decidir se aceita ou se rejeita, de forma on-line, um dado conjunto de requisições para criação de novos NS. As decisões devem ser tomadas de forma a maximizar a multiplexação estatística das fatias, buscando não violar as garantias acordadas (SLAs) para as fatias ativas no sistema. O inquilino selecionado recebe recursos de rede e computação, de acordo com o tipo de fatia acordado.

Denotando por $\lambda_{i,t}$ e $\xi_{i,t}$, o número de PRBs e o número de *CPU ticks* utilizados em t por i, é possível caracterizar as condições necessárias para viabilidade da alocação da fatia:

$$\lambda_{i,t} \leq R_{i,t}^{(s)} \leq C, \text{ enquanto que, } \xi_{i,t} \leq Q_{i,t}^{(s)} \leq M \quad (1)$$

onde C é a capacidade de transmissão de dados na rede e M é a capacidade de processamento de dados nos servidores MEC.

Ao aceitar uma requisição de um inquilino i no instante de tempo t, o operador recebe uma recompensa pela alocação dos recursos de rede, $\eta_{i,t}$, bem como pela alocação dos recursos de computação, $\kappa_{i,t}$. A recompensa $\eta_{i,t}$ leva em consideração a quantidade de recursos solicitada e o ganho de multiplexação, ou seja, a proporção entre o que foi realmente usado e o que está sendo solicitado, como mostrado na Equação (2). De um lado, inquilinos cujos recursos atribuídos passam a maior parte do tempo subutilizados são preferíveis àqueles que utilizam completamente seus recursos. Isso ocorre porque o operador pode realocar os recursos não utilizados naquele momento pelo inquilino para atender mais requisições no sistema, empreendendo uma estratégia de *overbooking*. Por outro lado, inquilinos que requisitam (e pagam) uma quantidade maior de recursos também são de interesse do operador. O peso $\alpha \in [0, 1]$ na Equação (2) expressa esse compromisso. A Equação (3) segue a mesma lógica, porém para recursos de computação.

$$\eta_{i,t} = \alpha \frac{R_{i,t}^{(s)}}{C} + (1 - \alpha) \frac{R_{i,t}^{(s)} - \lambda_{i,t}}{R_{i,t}^{(s)}} \quad (2)$$

$$\kappa_{i,t} = \sigma \frac{Q_{i,t}^{(s)}}{M} + (1 - \sigma) \frac{Q_{i,t}^{(s)} - \xi_{i,t}}{Q_{i,t}^{(s)}} \quad (3)$$

Para todo inquilino $i \in \mathcal{I}$ e para todo instante de tempo $t \in \mathcal{T}$, a variável de decisão $x_{i,t} \in \{0, 1\}$, assume valor 1, quando a requisição de criação da fatia do inquilino i é aceita em t, e o valor 0, caso contrário. Dessa maneira, é possível formular a admissão de fatias como o seguinte problema:

$$\textit{maximizar} \quad \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} (\eta_{i,t} + \kappa_{i,t}) x_{i,t} \quad (4)$$

$$\textit{sujeito a:} \quad \sum_{i \in \mathcal{I}} \lambda_{i,t} x_{i,t} \leq C, \qquad \forall t \in \mathcal{T} \quad (5)$$

$$\sum_{i \in \mathcal{I}} \xi_{i,t} x_{i,t} \leq M, \qquad \forall t \in \mathcal{T} \quad (6)$$

$$x_{i,t} \geq x_{i,t-1} \mathbb{F}(t - t_i^{start} \leq L_i^{(s)}) \quad (7)$$

$$x_{i,t} \in \{0, 1\}, \qquad \forall t \in \mathcal{T}, \ \forall i \in \mathcal{I} \quad (8)$$

A função objetivo (4) incentiva a aceitação das requisições dos inquilinos, considerando recursos de rede e processamento. As restrições em (5) e (6) garantem que a quantidade de PRBs e *CPU ticks* não excedam a capacidade disponível. Em (7) é imposta a seleção do inquilino i enquanto durar a reserva da fatia s previamente concedida. Denota-se por t_i^{start} a rodada em que a requisição s do inquilino i foi aceita, e $\mathbb{F}$ é uma função indicadora que fornece valor 1, se a condição entre parênteses for satisfeita, e o valor 0, caso contrário. Em (8) é estabelecida a variável binária de decisão.

III. Solução Proposta

O problema exploratório MaB serve de base para resolver o problema definido nas Equações (4)-(8) de forma on-line. No MaB, um jogador tem diversas máquinas caça-níquel à disposição e cada máquina retorna uma recompensa através de uma distribuição de probabilidade desconhecida. O jogador pode (1) explorar e observar a recompensa retornada por uma

máquina caça-níquel ainda não escolhida; ou (2) aproveitar mantendo-se com a máquina caça-níquel que retornou a maior recompensa em rodadas anteriores. O objetivo é maximizar a recompensa recebida após um certo número finito de rodadas. Propõe-se a resolução do problema a cada rodada t, pela seleção de K máquinas caça-níquel, cada uma representando um inquilino i, respeitando a restrição de orçamento C e M, de forma que a média da distribuição empírica seja maximizada. A solução é denominada M-ONETS-OBD, ilustrado no Algoritmo 1. No algoritmo, as linhas 1-4, correspondem a uma etapa de treinamento em que todos os inquilinos são selecionados. Para cada rodada, nas linhas 6-7, é coletada a recompensa v_i retornada para cada inquilino i e computada a recompensa média fornecida por i até aquele instante ($\bar{\theta}_i(t)$). O termo $\sqrt{\frac{2\ log\ t}{W_i}}$ é introduzido para dar peso maior para médias de recompensa obtidas em janelas mais longas. Nas linhas 8-10, inquilinos com fatias ativas são incluídos na próxima rodada. Se esses inquilinos não são suficientes para consumir o orçamento disponível, busca-se, entre os inquilinos restantes, aqueles que possuem as maiores recompensas médias ($\bar{\theta}_i(t)$), obedecendo a restrição de orçamento. As linhas 11-15 implementam a estratégia de *overbooking*, contabilizando os recursos alocados para i não pela quantidade prevista no seu *template* de fatias, i.e., $R_{i,t}^{(s)}$ e $Q_{i,t}^{(s)}$, mas pelo número médio de PRBs e de CPU *ticks* utilizados por i até t, i.e., $\hat{\lambda_i}$ e $\hat{\xi_i}$.

Algoritmo 1 M-ONETS-OBD - Algoritmo de seleção de inquilinos, com garantias de orçamento.

Entrada: K, $\mathcal{T}$, $\mathcal{I}$, C, M.
Inicialização: $B = 0$, $Z = 0$, $L \leftarrow \emptyset$, $W_i = 0$, $\bar{\theta}_i(0) = 0$, $n = 0$
1: **for all** $i \in \mathcal{I}$ **do**
2: $v_i \leftarrow \eta_{i,0} + \kappa_{i,0}$
3: ATUALIZE $\bar{\theta}_i(0)$
4: $W_i = W_i + 1$
5: **for all** $t \in \mathcal{T}$ **do**
6: **for all** $i \in \mathcal{I}$ **do**
7: $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2\ log\ t}{W_i}}$
8: **while** $n \leq K$ **do**
9: **if** $L(t) \neq \emptyset$ **then** $\hat{i} \leftarrow L(t)$
10: **else** $\hat{i}$: arg max $\hat{\theta}_i(t)$
11: **if** $B + \hat{\lambda_i} \leq C$ **and** $Z + \hat{\xi_i} \leq M$ **then**
12: $R \leftarrow R \cup \hat{i}$
13: $B = B + \hat{\lambda_i}$
14: $Z = Z + \hat{\xi_i}$;
15: $n = n + 1$
16: **for all** $i \in R$ **do**
17: $v_i \leftarrow \eta_{i,t} + \kappa_{i,t}$
18: ATUALIZE $\bar{\theta}_i(t)$
19: $W_i = W_i + 1$
20: ATUALIZE $L(t)$; $B = 0; Z = 0, n = 0$

Apesar de promissora, a estratégia de *overbooking* que aloca recursos no sistema considerando a utilização média ($\hat{\lambda_i}$ e $\hat{\xi_i}$) pode resultar em violações de SLAs. Para atenuar esse efeito, propõe-se o M-ONETS-OBS, que usa a regra empírica do desvio padrão para contabilizar os recursos alocados. Denota-se por φ_λ e φ_ξ o desvio padrão do número médio de PRBs e o desvio padrão do número médio de CPU *ticks* utilizados por i, respectivamente, o algoritmo M-ONETS-OBS é similar ao Algoritmo 1, exceto pelas linha 11, 13 e 14 onde $\hat{\lambda_i}$ é substituída por $\hat{\lambda_i}+(3\times\varphi_\lambda)$ e $\hat{\xi_i}$ é substituída por $\hat{\xi_i}+(3\times\varphi_\xi)$.

IV. Resultados

No modelo, PRBs indicam a quantidade de recursos de rede disponível em uma BS. A taxa efetiva de transferência, em PRBs, será de aproximadamente 648 Mbps [7]. Para quantificar os recursos de computação oferecidos por MEC, é utilizado o conceito de CPU *ticks* [8]. O modelo de carga de processamento segue Malandrino et al. [9], que utiliza os perfis de tráfego apresentados na Equação (9) para caracterizar os serviços providos por inquilinos, i.e., (1) vídeo por *streaming*, (2) jogos online e (3) geolocalização e mapas.

$$CPU_T = \begin{cases} 0,25 \ * \ tr_{Mbit} + 6,76 & videos \\ 161,38 \ * \ tr_{Mbit} + 1675,03 & jogos \\ 67,44 \ * \ tr_{Mbit} - 7,53 & mapas \end{cases} \quad (9)$$

De posse das relações da Equação (9), é possível calcular a quantidade de CPU *ticks* necessária para cada tipo de tráfego. Consideramos dois *templates* de fatias para os mesmos serviços: demanda máxima e demanda mínima. A Tabela I apresenta a quantidade de recurso de rede e de computação disponibilizada para cada *template* em cada tipo de serviço.

TABELA I: *Templates* disponibilizados à inquilinos.

Aplic.	D. Mínima		D. Máxima	
	Mbps	CPU_T	Mbps	CPU_T
vídeo	142	42	426	113
jogos	33	7.000	99	17.651
mapas	41	2.757	123	8.287

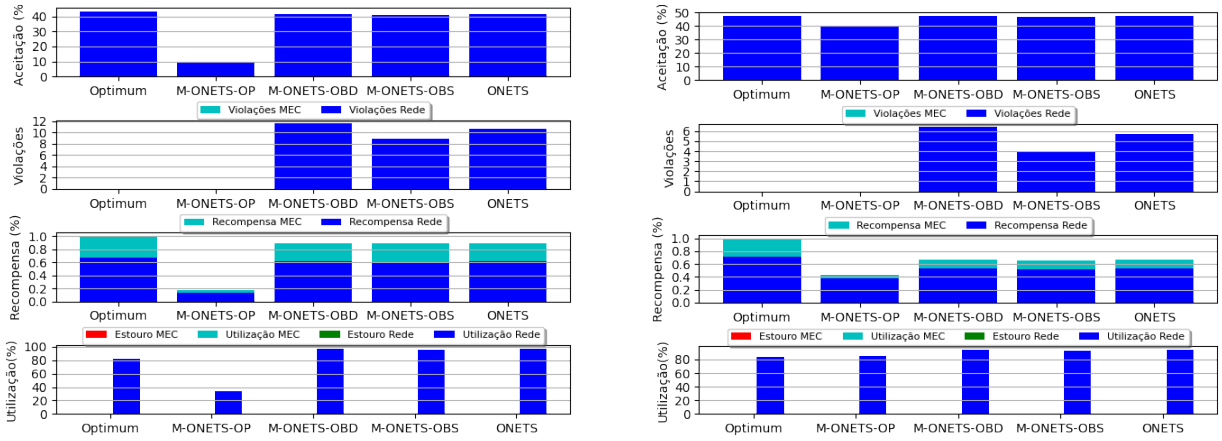
A. Cenários e Experimentos

Foram gerados seis cenários de experimentação para as estratégias de *overbooking* analisadas. Em todos os cenários, foram geradas as requisições para criação de fatias a partir de uma distribuição exponencial, com parâmetros ρ e ζ fixos, que determinam a frequência das ocorrências de requisições. Para gerar o parâmetro $\lambda_{i,t}$ é usada uma distribuição gaussiana com média μ_i e desvio padrão ω. Para criar diferentes cenários de tráfego efetivo, μ_i varia entre 10% a 90% do parâmetro $R_{i,t}^{(s)}$, definido no *template* de fatia associado ao inquilino i. A Tabela II resume os parâmetros utilizados na avaliação.

TABELA II: Parâmetros da simulação.

Parâmetros	Valores	Parâmetros	Valores
$\|\mathcal{I}\|$	10	α	0,5
$\|\mathcal{T}\|$	10000	σ	0,5
C	216 PRBs	ω	1
M	26.052 CPU *ticks*	K	10
μ_i	10% a 90% de $R_{i,t}^{(s)}$	L	100, 500
ρ	100		ou 1000
ζ	0,2		

A cada rodada t, a política de seleção implementada (M-ONETS-OBD e M-ONETS-OBS) analisa o melhor conjunto de inquilinos para serem aceitos. O algoritmo M-ONETS-OP, que provisiona recursos pela quantidade prevista no *template*, i.e., $R_{i,t}^{(s)}$ e $Q_{i,t}^{(s)}$ é implementado para efeitos de comparação. Utiliza-se também a formulação ótima do problema (*Optimum*), descrita na Seção II, resolvida com o uso da biblioteca *docplex* do Python. Por fim, considera-se o algoritmo ONETS proposto em [5]. Os resultados representam as médias das recompensas, do número de violações de SLA e dos percentuais de utilização e aceitação obtidas durante 30 execuções.



(a) Vídeo com demanda máxima.

(b) Vídeo com demanda mínima.

Fig. 1: Resultados para requisições de criação de fatias envolvendo apenas aplicações de video.

B. Avaliação das soluções M-ONETS-OBD e M-ONETS-OBS

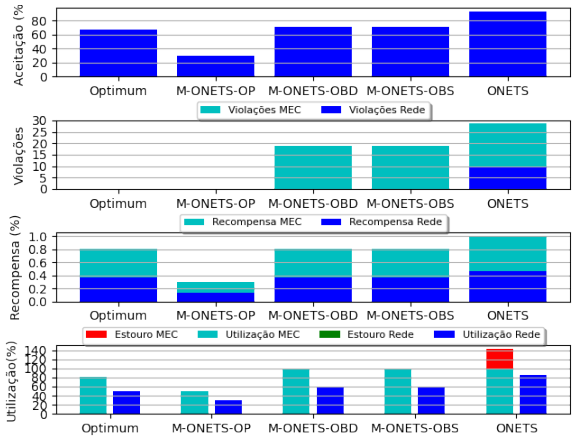
A Fig. 1 apresenta os resultados dos inquilinos que requisitam fatias com serviços de vídeo. A taxa de aceitação média é ilustrada no gráfico superior, seguida pelo número de violações, média da recompensa e média de utilização do sistema. As estratégias M-ONETS-OBD, M-ONETS-OBS e ONETS, que implementam *overbooking*, alcançam taxas de aceitação mais elevadas que M-ONETS-OP, que utiliza uma estratégia de *overprovisioning*. M-ONETS-OP retorna em torno de 24% da recompensa recebida por *Optimum* para cenários com demanda máxima. M-ONETS-OBD, M-ONETS-OBS e ONETS obtém recompensas próximas ao *Optimum*, em torno de 98%, 95% e 98%, respectivamente. Analisando os gráficos de violações de SLA, observa-se que *Optimum* e M-ONETS-OP não apresentam violações. M-ONETS-OBD, M-ONETS-OBS e ONETS incorrem em violações de SLA. Entretanto, mesmo no cenário com demanda máxima, o número de violações máximo é igual a 12 ocorrências, que correspondem à 0,12% dos casos. Observa-se que nos gráficos de utilização de recursos, devido às características implícitas à aplicação de vídeo, a utilização dos recursos de rede é alta. Entretanto, a utilização de recursos de processamento é baixa, próximo de 1%. Para os cenários de vídeo, a demanda máxima permite melhor utilização dos recursos e maiores recompensas.

A Fig. 2 apresenta os resultados, considerando que todos os inquilinos que requisitam criação de fatias oferecem serviço de mapas e geolocalização. Analisando M-ONETS-OBD e M-ONETS-OBS, oberva-se que as recompensas médias obtidas se aproximam do *Optimum*. Percebe-se que M-ONETS-OBD e M-ONETS-OBS incorrem em um número de violações de SLA reduzido, com 19 ocorrências para o cenário com demandas máximas, correspondendo a 0,19% dos casos. O algoritmo ONETS atinge um número de 30 violações de SLA. Analisando os gráficos de utilização de recursos, diferentemente do vídeo, os cenários de mapas necessitam de relevantes recursos computacionais. Entretanto, em M-ONETS-OBD e M-ONETS-OBS, os recursos computacionais não foram superestimados. A recompensa média obtida por ONETS é maior que as recompensas obtidas pelos algoritmos equivalentes propostos neste trabalho. Entretanto, analisando os gráficos de aceitação e utilização, observamos que esse ganho vem juntamente com uma superestimação dos recursos de computação. No cenário com demanda máxima, ONETS necessita de uma capacidade de processamento de dados 40% maior que a disponível, como ilustrado em vermelho nos gráficos de utilização média. Isso ocorre porque o algoritmo não tem ciência dos recursos de computação, apenas dos recursos de rede. Nas redes 5G, algumas aplicações são sensíveis a atraso, como por exemplo, carros autônomos que utilizam serviços de geolocalização. Uma alocação incorreta de recursos de rede e MEC, nessa aplicação, pode levar ao não cumprimento dos prazos necessários para garantir a correta execução do serviço.

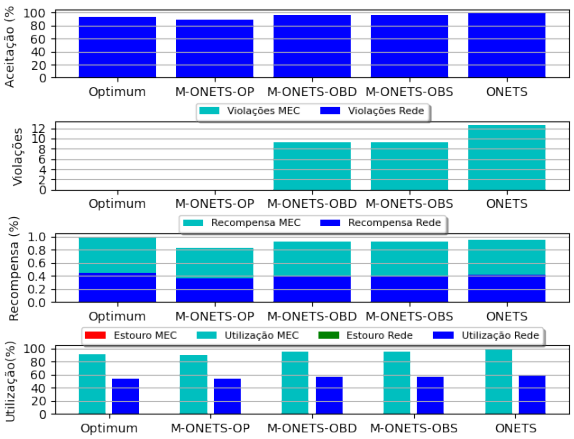
A Fig. 3 apresenta os resultados considerando que todos os inquilinos requisitam criação de fatias para serviços de jogos on-line. As recompensas médias obtidas com M-ONETS-OBD e M-ONETS-OBS se aproximam do *Optimum*, no cenário com demanda máxima, conseguindo 99% da recompensa recebida pelo *Optimum*. M-ONETS-OBD e M-ONETS-OBS apresentam uma taxa de violação de SLA de 0,19% dos casos analisados. Observa-se que os jogos necessitam de uma quantidade relevante de recursos de computação. No cenário com demandas máximas, M-ONETS-OBD e M-ONETS-OBS não superestimam os recursos computacionais. Novamente, o ONETS alcança as maiores recompensas e as maiores taxas de aceitação. No entanto, como mostrado nos gráficos de utilização, esses valores são alcançados porque o algoritmo considera apenas a capacidade dos recursos de rede, violando em ate 300% a capacidade de processamento. Apesar de não se tratarem de aplicações sensíveis, incorrer em atrasos no processamento de dados nessas aplicações poderia impactar os níveis de *Quality of Experience* (QoE) dos usuários finais, comprometendo o serviço ofertado.

V. Conclusões

As redes móveis de 5^a geração (5G) e NS trouxeram a capacidade de divisão e isolamento completo de recursos na forma de fatias. Aproveitando essa gama de oportunidades, este trabalho analisa, um conjunto de algoritmos para solução

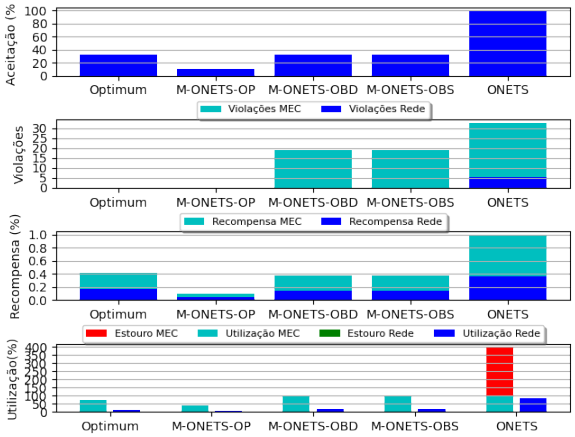


(a) Mapas com demanda máxima.

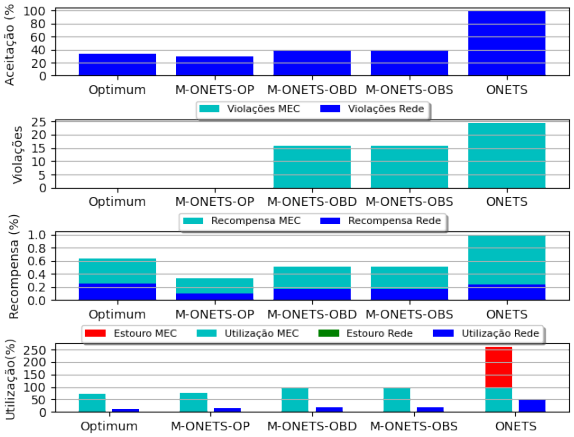


(b) Mapas com demanda mínima.

Fig. 2: Resultados para requisições de criação de fatias envolvendo apenas aplicações mapas.



(a) Jogos com demanda máxima.



(b) Jogos com demanda mínima.

Fig. 3: Resultados para requisições de criação de fatias envolvendo apenas aplicações jogos.

on-line de controle de admissão de inquilinos na rede da operadora, considerando recursos de comunicação e computação. A solução é baseada em problemas da classe MaB, com restrição de *lock-up*. Propõe-se uma solução de baixa complexidade, denominada *Online Network Slice Broker with MEC* (M-ONETS), que alcança uma acomodação eficiente de fatias na rede da operadora. Por fim, são apresentados os algoritmos M-ONETS-OBD e M-ONETS-OBS: soluções on-line de controle de admissão que adotam técnicas de *overbooking*. Os resultados demonstram que os recursos computacionais são importantes aos recursos de comunicação no processo de admissão de inquilinos na rede. Os recursos computacionais, escassos na borda, podem ser superestimados, levando a interpretações errôneas de admissão de inquilinos e alocação dos recursos.

AGRADECIMENTOS

Este trabalho foi parcial financiado pelo CNPq/CAPES. Além disso, a RNP apoiou parcialmente o trabalho, com recursos do MCTIC, nº 01245.010604/2020-14 no âmbito do projeto de Sistemas de Comunicações Móveis 6G do CRR da Inatel e pelo MCTIC/CGI.br/FAPESP por meio do Projeto SAMURAI - Smart 5G Core And MultiRAn Integration sob a Bolsa 2020/05127-2.

REFERÊNCIAS

[1] E. Husni and A. Bramantyo, "Design and implementation of mpls sdn controller application based on opendaylight," in *2018 International Symposium on Networks and Communications (ISNCC)*, pp. 1–5, 2018.

[2] S. E. Elayoubi *et al.*, "5G RAN slicing for verticals: Enablers and challenges," *IEEE Communications Magazine*, vol. 57, pp. 28–34, 2019.

[3] D. Bega *et al.*, "A machine learning approach to 5G infrastructure market optimization," *IEEE Transactions on Mobile Computing*, vol. 19, no. 3, pp. 498–512, 2019.

[4] R. Li *et al.*, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.

[5] V. Sciancalepore *et al.*, "ONETS: online network slice broker from theory to practice," *arXiv preprint arXiv:1801.03484*, 2018.

[6] L. Cominardi, T. Deiss, M. Filippou, V. Sciancalepore, F. Giust, and D. Sabella, "Mec support for network slicing: Status and limitations from a standardization viewpoint," *IEEE Communications Standards Magazine*, vol. 4, no. 2, pp. 22–30, 2020.

[7] S. Hwang and S. Park, "On the effects of resource usage ratio on data rate in LTE systems," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pp. 78–80, 2017.

[8] F. Weisbecker, "Status of Linux dynticks," in *9th annual workshop on Operating Systems Platforms for Embedded Real-Time applications*, 2013.

[9] F. Malandrino *et al.*, "From megabits to cpu ticks: Enriching a demand trace in the age of mec," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 43–50, 2020.