Agentes Inteligentes baseados em Aprendizado por Reforço para Alocação Dinâmica de Tráfego em Nuvens*

Reiner H. Santos Filho[1], **Diogo M. F. Mattos**[1], **Dianne S. V. Medeiros**[1]

[1] MídiaCom – TET/PPGEET/UFF
Universidade Federal Fluminense (UFF)
Niterói, RJ – Brasil

Abstract. *The efficient allocation of cloud traffic is challenging due to resource sharing among clients. This can lead to idle resources if customers are limited to use the hired bandwidth only. Cloud usage can be optimized by providing resources to customers dynamically as demand exists. Reinforcement learning agents promote adaptive responses to time-varying environments. This paper proposes a multi-agent Q-learning based mechanism for managing resource access by each cloud client. We assess our proposal in an emulated environment where a controller is responsible for allocating traffic to clients. The results show that the mechanism reduces cloud idleness, allowing low-priority customers to use unused bandwidth while ensuring the hired bandwidth of higher priority clients. Our mechanism requires low total processing commitment, even varying the number of states and action space, while the memory cost per agent increases, peaking at 300 kB for 200 states and actions.*

Resumo. *A alocação eficiente do tráfego em nuvens é desafiadora devido ao compartilhamento de recursos entre os clientes. Isso pode implicar recursos ociosos caso os clientes estejam limitados a utilizar somente a banda contratada. O uso da nuvem pode ser otimizado provendo recursos aos clientes dinamicamente de acordo com a demanda. Agentes de aprendizado por reforço promovem respostas adaptáveis a ambientes variantes no tempo. Este artigo propõe um mecanismo baseado em Q-learning com múltiplos agentes para gerenciar o acesso aos recursos por cada cliente da nuvem. A proposta é analisada em um ambiente emulado, no qual um controlador é responsável pela alocação de tráfego aos clientes. Os resultados mostram que o mecanismo reduz a ociosidade da nuvem, permitindo que clientes com baixa priorização utilizem a banda disponível, ao mesmo tempo que garante a banda contratada a clientes prioritários. O mecanismo exige baixo comprometimento de processamento total, mesmo variando o número de estados e espaço de ações, ao passo que o custo em memória por agente aumenta, alcançando um máximo de 300 kB para 200 estados e ações.*

1. Introdução

A computação em nuvem vem sendo cada vez mais utilizada para diferentes finalidades. Nesse cenário, coexistem diversas infraestruturas físicas e virtuais, cada uma com

*Os autores agradecem ao CNPq, CAPES, RNP, FAPERJ e FAPESP pelo financiamento que viabilizou essa pesquisa.

seus requisitos de processamento, memória e armazenamento. Além disso, há grande diversidade nas demandas dos usuários, cada uma com exigências de capacidade diferentes durante períodos distintos [Chen et al., 2019, Medeiros et al., 2019]. A natureza dinâmica do sistema e a existência de diferentes requisitos dificultam a alocação eficiente de recursos, que continua sendo um desafio em aberto. Atender às demandas de processamento, armazenamento, memória e rede, fornecendo o desempenho requerido pelos usuários e maximizando o lucro dos provedores é um problema NP-difícil [Guo et al., 2017].

Os provedores de serviços em nuvens devem ser capazes de prover o desempenho mínimo requerido pelos clientes, mesmo quando a nuvem está próxima de sua carga máxima suportada [Fox et al., 2009, Carvalho et al., 2011]. Paralelamente, recursos ociosos, provenientes de capacidade disponível e não alocada, representam perdas de receita caso clientes com requisitos já atendidos solicitem mais recursos em um determinado período. Nesse contexto, surge o modelo de venda sob demanda, no qual clientes que contratam uma capacidade menor podem excedê-la, caso haja disponibilidade por parte da infraestrutura da nuvem, desde que o cliente pague uma tarifa adicional (*pay-as-you-go*). Assim, são necessários mecanismos que permitam distribuir o tráfego do conjunto de todos os usuários requisitando recursos da nuvem, de forma a atender o desempenho mínimo de cada usuário e, simultaneamente, redistribuir a capacidade ociosa. A estratégia de alocação dinâmica de tráfego baseada em aprendizado por reforço, utilizada em outros cenários [Yau et al., 2005], é uma tecnologia viabilizadora da distribuição dinâmica de demanda dos clientes da nuvem, priorizando aqueles que contratam maior vazão.

Este artigo propõe um mecanismo para promover a alocação ágil e dinâmica do tráfego de uma nuvem. O mecanismo proposto é baseado em agentes que executam a estratégia *Q-learning*. A proposta utiliza múltiplos agentes capazes de limitar a taxa de transmissão de cada cliente, mantendo, no entanto, a adequação ao desempenho mínimo de cada um. Considera-se que os agentes são independentes, uma vez que a decisão de outro agente não influencia nas ações executadas pelos demais. Todos os agentes compartilham as mesmas informações e estados, sendo que cada agente tem controle sobre a banda disponível para um único cliente. Além disso, existe uma restrição comum a todos os agentes de que não é permitido que o tráfego da nuvem aumente além de um limite estabelecido. A decisão de cada agente sobre a limitação do tráfego é tomada de acordo com os interesses do cliente ao qual está atribuído.

Trabalhos anteriores utilizam aprendizado por reforço para escalonamento de máquinas virtuais em uma nuvem, aumentando ou diminuindo o número de máquinas virtuais ativas [Dutreilh et al., 2011, Barrett et al., 2013, Chen et al., 2019]. Contudo, esses trabalhos não focam na alocação dinâmica de banda. A proposta deste artigo visa garantir de forma ágil que os clientes com maior prioridade sejam sempre atendidos, ao passo que, em caso de recursos ociosos, clientes de menor prioridade usam a banda disponível. Um protótipo do mecanismo proposto é implementado para controlar a banda dos clientes de uma nuvem. O desempenho do mecanismo proposto é avaliado em um ambiente emulado, no qual a nuvem possui um controlador de rede definida por software (*Software-Defined Netoworking* - SDN) responsável por controlar a banda dos clientes com base nas ações recebidas dos agentes. Resultados mostram que o mecanismo proposto previne a sobrecarga da nuvem, ao mesmo tempo que reduz os recursos ociosos e realiza a priorização dos clientes de forma dinâmica e ágil. Com o aumento do número de

ações e estados, o consumo de processamento se mantém constante, enquanto o consumo de memória aumenta exponencialmente, alcançando um máximo de 300 kB de memória para 200 estados e ações.

O restante do artigo está organizado da seguinte forma. A Seção 2 expõe o problema da alocação de banda em um serviço de nuvem. A Seção 3 introduz o aprendizado por reforço. O mecanismo proposto é apresentado na Seção 4. A avaliação da proposta e os resultados são discutidos na Seção 5. A Seção 6 discute trabalhos relacionados. A Seção 7 conclui o artigo.

2. O problema da alocação de banda em uma nuvem

A alocação de recursos é fundamental para os provedores de infraestrutura como serviço (*Infrastructure as a Service* - IaaS), responsáveis pela gerência e manutenção dos centros de dados, assim como para os inquilinos (*tenants*), que alugam recursos de centros de dados para implantar seus serviços na nuvem. O uso eficiente dos recursos para o provedor IaaS incorre na multiplexação dos diversos inquilinos sobre os recursos físicos e é significativo para sua receita. Isso porque a maior utilização de recursos indica que um número maior de requisições dos inquilinos estão sendo atendidas e, portanto, maior lucro é gerado. Para os inquilinos coexistentes, os recursos alocados determinam o desempenho dos serviços implantados na nuvem e incorrem em custos [Chen et al., 2014].

Diferente de recursos de processamento e memória que são compartilhados localmente em cada servidor, o recurso de rede é compartilhado globalmente entre os inquilinos que compartilham enlaces na rede. Assim, o compartilhamento da banda alocada para cada cliente depende das tarefas que estão alocadas no mesmo servidor, dos clientes que estão se comunicando com ele e dos clientes que passam pelo mesmo enlace de gargalo. A alocação de banda, então, diferencia-se da alocação dos recursos de processamento e memória, incorrendo em maiores dificuldades e acarretando desafios. Para cada inquilino, variações no desempenho da rede dificultam a previsão do locatário e limitam seu custo máximo. Sem isolamento de desempenho de rede, um inquilino malicioso pode aumentar seu uso de rede em detrimento de outros. Esse problema reside no compartilhamento da rede baseado no serviço de melhor esforço do IP.

Provedores de infraestrutura oferecem a seus inquilinos recursos computacionais sob-demanda para alocação de máquinas virtuais, entretanto não há garantias de desempenho para recursos de rede. Isso acarreta em imprevisibilidade do desempenho das aplicações, que pode gerar insatisfação por parte dos clientes e causar perda de receita [Bari et al., 2012]. É possível fazer o provisionamento de recursos em situações previsíveis com planejamento.A automatização da alocação dos recursos em situações não previsíveis [Dutreilh et al., 2011] requer que o provisionamento automático se adapte às variações da rede, levando em consideração a heterogeneidade dos requisitos.

Chen *et al.* definem propriedades da alocação de banda em provedores IaaS em nuvens: garantias determinísticas, justiça, utilização e praticabilidade [Chen et al., 2014]. Garantias determinísticas de largura de banda asseguram um desempenho previsível para cada inquilino, independentemente de qualquer comportamento dos inquilinos concorrentes. As garantias mínimas de largura de banda permitem aos inquilinos obter mais largura de banda além de suas garantias quando há largura de banda disponível. Isso permite que os inquilinos obtenham desempenho limitado em pior caso e, portanto, custos limitados.

A justiça é capaz de fornecer proteção relativa aos inquilinos bem-comportados com a presença de mau comportamento. Assim, garante a alocação justa de banda mesmo em cenários de disputa. A utilização relaciona-se ao uso da banda disponível no centro de dados. Alta utilização indica que a largura de banda não deve ficar ociosa enquanto houver demandas insatisfeitas. Um serviço com alta demanda de rede pode usar toda a largura de banda disponível quando outros serviços estão inativos. Garantir a utilização implica melhor desempenho de inquilinos cujos serviços têm demanda de tráfego atendidas. Provedores IaaS também se beneficiam de alta utilização da largura de banda, com a capacidade de acomodar mais solicitações de inquilinos e, assim, gerar mais renda. Por fim, a praticabilidade visa garantir a aplicação simplificada dos procedimentos de alocação de banda. Em especial a praticabilidade relaciona-se com a simplicidade e a escalabilidade da alocação de banda na nuvem.

A proposta deste artigo visa assegurar garantias determinísticas de que o tráfego mínimo contratado por um inquilino será sempre atendido, mesmo na presença de outros inquilinos que utilizem mais do que o contratado. O mecanismo proposto garante a justiça ao distribuir a banda ociosa proporcionalmente à prioridade dos clientes que disputam o uso da banda. A utilização é otimizada na proposta, pois um dos objetivos é garantir a alta utilização da banda da nuvem para atender o máximo de solicitações por banda possível. Por fim, a praticabilidade da proposta é ressaltada pela sua simplicidade e pelo baixo consumo de memória e processamento, implicando escalabilidade.

3. O aprendizado por reforço

O aprendizado por reforço é uma técnica de aprendizado de máquina utilizada em ambientes dinâmicos. A técnica consiste em utilizar um agente, definido como um módulo computacional capaz de agir de maneira autônoma para alcançar os objetivos para os quais foi desenvolvido [Isbell et al., 2001]. O agente analisa o estado atual do ambiente, responde com uma ação e recebe em retorno uma recompensa ou uma punição pela ação tomada [Xia et al., 2019]. Com o retorno, o agente altera seus parâmetros e se adapta ao ambiente em uma velocidade proporcional à modelagem das recompensas adquiridas e punições sofridas. Assim, o agente é capaz de ajustar suas ações de maneira otimizada para o ambiente em um dado momento t. Em $t+1$, no entanto, as ações podem não ser mais tão efetivas e, assim, o agente se adapta às novas condições.

Existem três parâmetros principais que moldam o funcionamento de um sistema de aprendizado por reforço: a taxa de exploração ϵ, o fator de desconto γ e a taxa de aprendizado α. A taxa de exploração ϵ dita o quanto o agente toma decisões aleatórias em vez de usar o que já aprendeu. Um agente toma uma decisão com base nas experiências passadas com probabilidade igual a $1-\epsilon$. O fator de desconto γ traduz a importância das recompensas futuras. Quanto mais próximo a 1 estiver γ, maior será o peso das recompensas recebidas. Analogamente, quanto mais próximo a 0, menor é o peso da recompensa recebida. A taxa de aprendizado α representa a "resistência" do agente a mudanças, *i.e.*, a proporção entre novos valores obtidos pelas recompensas e os valores anteriores armazenados na Tabela Q. Quanto maior α, menor é o peso dado a aprendizados anteriores.

Problemas que envolvem aprendizado por reforço normalmente são modelados como um processo de decisão de Markov (*Markovian Decision Process* - MDP). O MDP é tradicionalmente representado através de uma tupla de quatro elementos,

$\langle \mathcal{S}, \mathcal{A}, p, q \rangle$ [Barrett et al., 2013]. Os elementos são o espaço de estados do ambiente ($\mathcal{S}$), o espaço de ações disponíveis para serem tomadas ($\mathcal{A}$), as probabilidades de transição entre os estados (p) e as probabilidades de recebimento de recompensas (q). Assim, partindo-se de um estado s_i é possível chegar a um estado s_j com probabilidade p. Quando todos os elementos da 4-tupla são conhecidos, existe um modelo completo do ambiente e não é necessário utilizar aprendizado por reforço. Quando não há um modelo completo, realiza-se uma tentativa de aproximá-lo através de algoritmos baseados em modelo, ou estima-se diretamente políticas ou funções para lidar com o problema usando algoritmos livres de modelo [Medeiros et al., 2019]. Um exemplo de algoritmo livre de modelo é o *Q-learning* [Barrett et al., 2013].

O algoritmo *Q-learning* é capaz de realizar predições levando em consideração estimativas passadas. O *Q-learning* possui uma tabela, conhecida como Tabela Q, na qual os elementos são os valores resultantes de uma função $Q(s, a)$, em que s é um estado, tal que $s \in \mathcal{S}$, e a é uma ação, tal que $a \in \mathcal{A}$. A função $Q(s, a)$ é definida pela Equação 1, com $s = s_t$ e $a = a_t$. O número de linhas na Tabela Q é igual à quantidade de estados e o número de colunas é igual à quantidade de ações. Um agente utiliza a Tabela Q para tomar suas decisões e a cada iteração a Tabela Q é atualizada conforme a equação

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot Q(s_{t+1}, a_{t+1})]. \quad (1)$$

Sempre que invocado, um agente que conhece o estado s_t busca na sua tabela a linha correspondente a esse estado e toma a decisão baseado na ação a_t que retorne o maior valor $Q(s_t, a_t)$ ou toma uma ação aleatória. A taxa de exploração ϵ determina se o agente realiza uma consulta à Tabela Q ou toma uma decisão aleatória. Paralelamente, o agente também recebe uma recompensa r, que utiliza para atualizar sua Tabela Q.

4. O mecanismo proposto de alocação dinâmica de banda

Este artigo propõe um mecanismo que visa otimizar o uso do recurso de largura de banda instalado e disponível na nuvem. A ideia é que clientes, inquilinos, que contrataram uma capacidade de tráfego R_1 possam excedê-la caso a nuvem tenha recursos ociosos. Simultaneamente, deve-se garantir que clientes que contrataram uma capacidade de tráfego $R_2 > R_1$ não sejam prejudicados pelo uso excedente da nuvem pelos outros clientes, garantindo o isolamento entre tráfego de diferentes clientes. Como não há um modelo completo para esse ambiente, o cenário é propício para o uso de técnicas de aprendizado por reforço livres de modelo, a fim de gerar políticas otimizadas. Esse modelo é vantajoso para clientes que se beneficiem de um aumento de banda sempre que possível, que reduz a banda ociosa e gera um aumento de receita para o provedor da infraestrutura. Assim, propõe-se utilizar o algoritmo de aprendizado por reforço *Q-learning* para estabelecer políticas de alocação dinâmica de banda usando agentes para otimizar o uso da capacidade de banda em uma nuvem, regulando o tráfego dos clientes. O *Q-learning* é usado com suporte a múltiplos agentes, uma vez que apresenta bom desempenho nesse tipo de cenário [Barrett et al., 2013, Dutreilh et al., 2011].

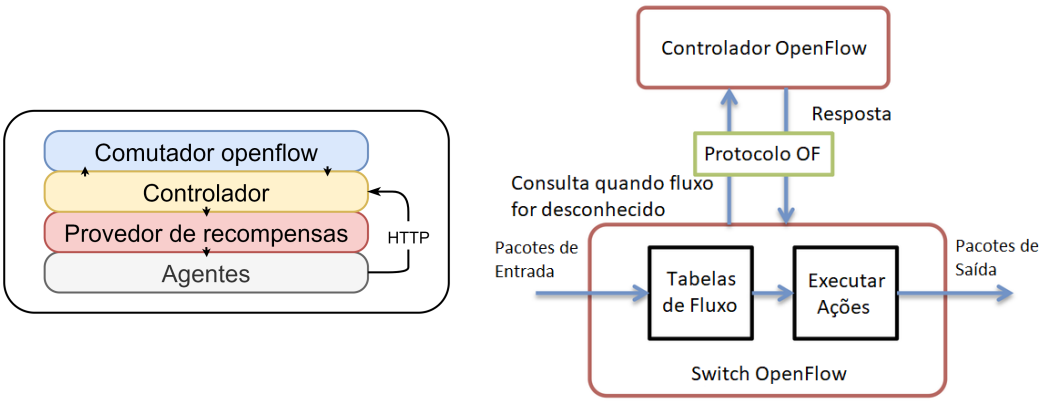
No mecanismo proposto, cada agente tem controle sobre a banda disponível para um único cliente e toma decisões sobre a limitação de tráfego com base nos interesses do cliente ao qual está atribuído. Utiliza-se um agente por cliente devido à simplicidade

de gerenciamento de memória ao criar e destruir instâncias de agentes quando clientes entram e saem do sistema. Essa abordagem é possível porque um processo de decisão markoviano distribuído para vários agentes ou com um único agente continua sendo um processo de decisão markoviano. Além disso, a abordagem usada permite a ativação e desativação de forma simples e independente de políticas distintas para um ou mais clientes. Todos os agentes compartilham as mesmas informações e estados. Há a restrição entre os agentes de que não é permitido que o tráfego total na nuvem aumente além de um limite determinado. As informações disponibilizadas para os agentes são coletadas de um elemento de interconexão, como um comutador. A recuperação dessa informação é responsabilidade de uma entidade chamada de "Provedor de Recompensas". O Provedor de Recompensas também força a tomada de decisão a cada intervalo de tempo T. As informações utilizadas pelos agentes são a carga da nuvem no momento atual, o aumento da taxa de transmissão de seu cliente e a taxa de transmissão agregada de todos os clientes. Em modo de operação normal os agentes são configurados com uma taxa de aprendizado $\alpha > 0$ e com base nas informações obtidas do Provedor de Recompensas, cada agente determina individualmente se receberá uma punição ou recompensa. Vale ressaltar que os agentes executam no provedor de infraestrutura (IaaS) e, portanto, não sofrem influência de aplicações que executem nos clientes. Então, cada agente tem comportamento confiável e, portanto, sempre executa corretamente.

A priorização é realizada pela alternância de políticas, sendo garantida pelo modo de operação de auto-coibição, executado pelos agentes. Nesse modo, sempre que clientes de maior prioridade requisitarem mais banda, os agentes tomam decisões que levam a uma limitação de tráfego mais intensa para os clientes menos prioritários. O único critério levado em consideração é a banda contratada. Uma vez que a banda de um cliente é reduzida abaixo da banda contratada devido à utilização da capacidade da nuvem por outros clientes, a taxa de aprendizado α é configurada como zero. A ideia é evitar que a tendência de crescimento do tráfego de cada cliente individualmente seja afetada pela auto-coibição dos agentes executada para garantir a priorização de clientes. Caso clientes de maior prioridade não solicitem mais banda, os agentes operam em modo normal.

A Figura 1 mostra o esquema do mecanismo proposto, com as interações entre as entidades que o compõe e entre o controlador e comutador de rede definida por *software* (*Software-Defined Network* - SDN). Na Figura 1(a), observa-se a interação entre as entidades do mecanismo. Os agentes são as únicas entidades que agem sobre o controlador. Provedor de Recompensas apenas realiza consultas ao controlador. Por sua vez, o controlador é a única entidade que se comunica diretamente com o comutador. A comunicação com o comutador ocorre conforme ilustrado na Figura 2. Quando não há uma configuração para o fluxo instalada na tabela do comutador, é realizada uma consulta para o controlador e a configuração é instalada na tabela de fluxos. A partir de então, o comutador é capaz de realizar ações de encaminhamento sobre o fluxo.

A cada período T, o Provedor de Recompensas consulta as informações de todas as portas ativas do comutador, à qual cada cliente está conectado, e as envia para cada um dos agentes. Cada agente processa essas informações de modo a obter o estado atual e a recompensa pela ação anterior, atualiza a sua Tabela Q, utilizando a Equação 1, e executa uma nova ação, fazendo uma requisição `HTTP` para o controlador. O controlador atualiza a limitação de tráfego para cada cliente de acordo com as requisições de seus respecti-



(a) Interação entre as entidades. (b) Interação entre o controlador e o comutador.

Figura 1. Esquema do mecanismo proposto. (a) A cada intervalo de tempo T, o Provedor de Recompensas consulta as informações no controlador via requisições `HTTP`, envia essas informações aos agentes e os estimula a executar uma ação. Os agentes limitam o tráfego para cada porta fazendo requisições `HTTP` ao controlador. (b) O controlador atualiza a nova limitação de tráfego na porta do comutador através do Openflow 1.3.

vos agentes. Através de requisições `HTTP` enviadas pelo Provedor de Recompensas para o controlador, os agentes recuperam informações sobre a carga da nuvem no momento atual e a quantidade de recursos que os clientes estão consumindo. Cada agente armazena as informações necessárias para realizar alterações de banda do seu cliente, como o endereço IP do cliente, a porta e o identificador numérico da regra de limitação de banda no controlador.

O tráfego total que chega à nuvem não pode ultrapassar um limite M, menor do que a capacidade total da nuvem. Assim, os estados são tomados como faixas de tráfego dependentes de M. O limiar M é dividido em n_{f1} faixas iguais, de forma que o número de estados, Q_d, pode ser dado por $\lceil M/n_{f1} \rceil$. No entanto, deve existir um estado adicional para contemplar o caso em que a demanda de tráfego atual, L, é maior do que o limite máximo tolerado pela nuvem, M. Assim, o número total de estados é dado pela equação

$$Q_d = 1 + \frac{M}{n_{f1}}, \tag{2}$$

em que quanto maior n_{f1}, maior será o número de estados Q_d. Consequentemente, maior será o número de linhas Q_s da Tabela Q. Quanto maior a Tabela Q, mais tempo é gasto para que os agentes convirjam para tomar decisões que impliquem resultados mais adequados para determinado estado.

Uma vez determinados os estados, o espaço de ações possíveis deve ser definido. Os agentes variam a taxa máxima de transmissão de cada uma das portas do comutador. As ações são modeladas com a proporção de tráfego que deve ser limitada ou concedida para a porta à qual o agente está designado. Para que o método possa ser usado considerando um limite M qualquer para o tráfego na nuvem, a quantidade de tráfego variada depende do número n_{f2} de frações de M, assim como as ações. Considerando um mesmo número de ações de aumento e redução da banda, para toda ação de aumento há uma ação de diminuição, portanto, a quantidade de ações é tomada como $2M/n_{f2}$. Há o caso em que o agente não realiza qualquer intervenção em sua porta, somando mais uma ação.

Assim, a quantidade de ações Q_a é dada por

$$Q_a = 1 + \frac{2M}{n_{f2}}. \tag{3}$$

O tamanho Q_t da Tabela Q é encontrado substituindo as Equações 2 e 3 em $Q_t = Q_a Q_e$. Assim, neste artigo, o tamanho da Tabela Q só depende da carga máxima suportada pela nuvem e da quantidade de frações escolhidas para os estados e as ações, dado por

$$Q_t = 1 + \frac{2M^2 + (2n_{f1} + n_{f2}) \cdot M}{n_{f1} \cdot n_{f2}}. \tag{4}$$

As informações do ambiente são compartilhadas por todos os agentes a cada intervalo de tempo T. Entre eles é necessário que haja a restrição de que o tráfego direcionado à nuvem não ultrapasse o valor M e que haja priorização entre os clientes. A função recompensa utilizada respeita

$$r_n(L, L_{c1}, L_{c2}, ..., L_{cn}, x) = \begin{cases} -\frac{L}{M} & , \text{ se } L > M \\ x & , \text{ se } x > 1 \\ -x & , \text{ c.c.} \end{cases}, \tag{5}$$

em que L é o tráfego atual total que chega na nuvem, L_{cn} é o tráfego atual do cliente n, r_n é a função recompensa dada ao agente do cliente n e x é a razão entre o tráfego atual e o tráfego anterior do cliente n. Caso um cliente i, que não esteja utilizando integralmente sua banda e tenha maior prioridade, solicite mais banda, todos os outros agentes atribuídos a clientes de menor prioridade e com capacidades contratadas menores passam a executar ações que restrinjam o uso de recursos para o seu cliente para garantir o atendimento de todos os acordos de nível de serviço (*Service Level Agreement* - SLA).

Tráfegos em rajada acontecem naturalmente na Internet e, dependendo da duração da rajada, as punições dadas aos agentes levam a uma queda no tráfego total do sistema, reduzindo o tráfego total abaixo de M. Além disso, é possível que o tráfego total demore para convergir de volta para as valores próximos a M. Para mitigar esse problema, a punição só ocorre quando detecta-se $L > M$ pelo menos 3 vezes seguidas, de maneira análago a implementações do TCP. Ademais, os parâmetros γ e α devem ser definidos de modo que fatos recentes só tenham grande relevância caso sejam persistentes, para evitar que o sistema convirja para um estado em que $L \ll M$ muito bruscamente.

5. A avaliação da proposta através de emulação

A avaliação do mecanismo proposto ocorre em ambiente emulado, utilizando a ferramenta Mininet [Lantz et al., 2010] e o controlador Ryu[1]. Para tanto, a emulação ocorre em um computador pessoal equipado com processador AMD FX(tm)-8350 de oito núcleos e 8 GB de memória RAM com sistema operacional Windows 10 executando o controlador e uma máquina virtual Ubuntu 18.04, sobre o VirtualBox, com instâncias do Mininet, e um Notebook equipado com processador Intel Core i5-8265U, 8 GB de RAM e sistema operacional Ubuntu 18.04 executando instâncias do Mininet.

[1]Disponível em https://osrg.github.io/ryu/.

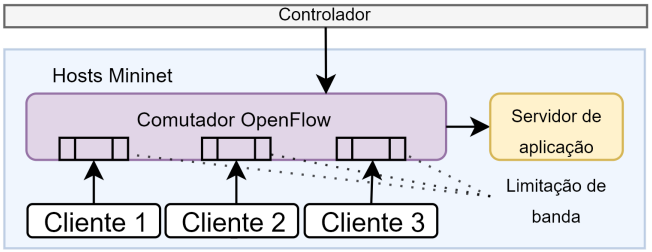
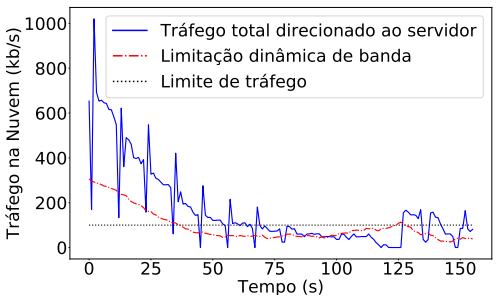


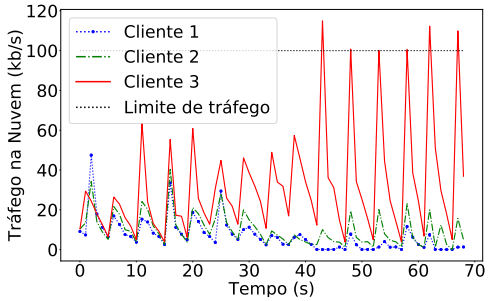
Figura 2. Topologia utilizada para avaliar a proposta. Três clientes estão conectados em portas diferentes do comutador *OpenFlow* enviando tráfego para um servidor de aplicação na nuvem.

O Mininet versão 2.3.0d6 é utilizado neste artigo para criação das instâncias que geram e recebem tráfego. O protocolo utilizado para comunicação entre nós é o *OpenFlow* versão 1.3. A Figura 2 ilustra a topologia utilizada no cenário de avaliação. Uma instância emula um servidor na nuvem e outras três emulam clientes da nuvem. Cada cliente utiliza uma porta do comutador *OpenFlow*. A taxa máxima de transmissão permitida em cada porta é determinada pelo controlador de acordo com as ações executadas pelos agentes. Os agentes obtêm as informações sobre uso de banda a partir do Provedor de Recompensas, que a cada instante de tempo coleta informações no controlador sobre cada cliente. O Provedor de Recompensas é implementado em *Python 3.6*. Com base nas informações obtidas, os agentes tomam a decisão quanto à limitação de banda. A decisão é enviada para o controlador que executa a modificação no comutador. O controlador *Ryu* é usado, pois interfaces de programação de aplicação (*Application Programming Interfaces* - APIs) que facilitam o desenvolvimento de diferentes formas de gerenciamento e controle de aplicações sustentadas pelo arcabouço SDN. O controlador é manipulado através da linguagem *Python* e a configuração dinâmica é feita via requisições REST, usando o protocolo `HTTP` com respostas em `JSON` [Ahalawat et al., 2019].

O mecanismo proposto é avaliado para verificar a capacidade dos agentes de pre-



(a) Convergência da carga de tráfego recebida pela nuvem para o limiar M previamente estabelecido.



(b) Priorização de clientes quando qualquer cliente é impedido de usar a banda contratada.

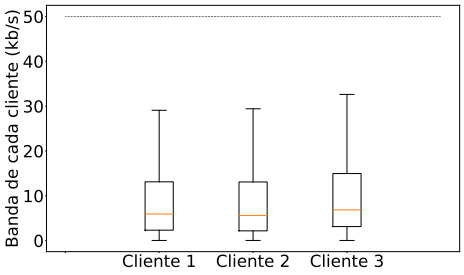
Figura 3. O mecanismo proposto a) leva à convergência da carga de tráfego recebida. Para o caso $L > M$, a convergência é rápida devido à abordagem de estado único utilizada. A limitação dinâmica de banda é a média das limitações feitas pelos agentes na banda de seus clientes em cada instante. b) Quando os clientes demandam o uso da banda contratada, os agentes entram em modo de auto-coibição, coordenando a limitação de banda dos clientes, a fim de garantir a priorização dos clientes.

venir a sobrecarga da nuvem e garantir a priorização entre os clientes através do modo de auto-coibição. Além de verificar o consumo de memória e processamento com o aumento da Tabela Q. No primeiro cenário de avaliação, a capacidade do servidor é elevada até um valor maior que um limite máximo de tráfego destinado à nuvem M. Sempre que o limite M é ultrapassado, o mecanismo deve ser capaz de convergir para próximo de M novamente. A Figura 3(a) mostra a ação do mecanismo proposto ao longo do tempo. Os resultados são obtidos utilizando $n_{f2} = 10$, $n_{f1} = 200$ e $T = 0{,}7$ s. Os valores para n_{f2} e n_{f1} são definidos empiricamente, de forma que os agentes não provoquem variações bruscas nas taxas de transmissão dos clientes, incorrendo em queda na utilização da nuvem em função do estouro de temporizadores TCP. O intervalo de $0{,}7$ s é suficiente para que o controlador *Ryu* atualize suas métricas.

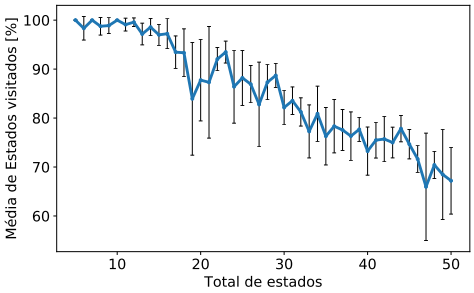
A Figura 3(a) mostra que há convergência do tráfego na nuvem para o limite de tráfego M representado pela linha pontilhada. Ademais, para um tráfego total direcionado à nuvem muito maior que o limite de tráfego imposto, *i. e.*, $L \gg M$, os agentes diminuem a taxa máxima permitida para cada cliente até que o tráfego se mantenha próximo a M, convergindo rapidamente para o limite. Esse caso é mapeado em um único estado, uma única linha da Tabela Q. A convergência rápida para valores ótimos está de acordo com a modelagem proposta. A ideia chave é que o mecanismo trate com urgência o cenário no qual a nuvem recebe mais tráfego do que o limite estabelecido, em contraste ao cenário em que o tráfego está abaixo de M.Observa-se na Figura 3(a) que, quando o tráfego está abaixo de M, a taxa máxima permitida para os clientes aumenta com o objetivo de evitar que haja uma grande fração de banda ociosa, caso haja demanda pelos clientes. No cenário avaliado, a utilização média dos recursos da nuvem, alcança 82% durante um período de 7 horas.

Além de reduzir os recursos ociosos na rede, as prioridades definidas nos acordos de nível de serviço dos clientes devem ser atendidas. Assim, é importante verificar a capacidade de priorização dos agentes, garantindo que aqueles com maior banda contratada sejam priorizados. Nesse cenário, os clientes partem do mesmo valor de taxa de transmissão e os agentes são encarregados de distribuir o tráfego de acordo com a priorização. A Figura 3(b) mostra como o mecanismo proposto prioriza os clientes. Quanto maior o número de identificação do cliente maior a prioridade, o Cliente 3 é definido como mais prioritário do que o Cliente 2, que por sua vez é mais prioritário do que o Cliente 1. Observa-se na Figura 3(b) que a auto-coibição dos agentes provoca um efeito de dente de serra no tráfego dos clientes. Isso ocorre porque no modo de auto-coibição, os agentes são limitados a executar somente ações que diminuam o tráfego permitido quando qualquer outro cliente é impedido de usar a banda contratada.

A Figura 4(a) mostra o consumo de banda de cada cliente, considerando o cenário no qual um cliente com maior prioridade consome recursos em um momento em que os recurso da nuvem estão completamente comprometidos. Os resultados mostrados são as médias de 20 repetições de um cenário em que os agentes possuem 15 estados e 4 ações possíveis, dependentes do limite máximo admitido pela nuvem, durante um período para a tomada de 200 decisões. O mecanismo proposto deve ser capaz de limitar a banda máxima dos Clientes 1 e 2, menos prioritários, para servir ao Cliente 3 mais prioritário, ao mesmo tempo em que garante a banda mínima contratada. O resultado mostra que o modo de auto-coibição é efetivo na priorização entre os clientes. Quando o Cliente 3 começa a



(a) Consumo de banda dos clientes, considerando um cenário com 15 estados e 4 ações possíveis, em um período de 200 decisões.



(b) Percentual de estados visitados pelo agente, considerando um cenário com 9 ações possíveis em um período de 100 decisões.

Figura 4. Desempenho do mecanismo proposto em relação à banda fornecida aos clientes e ao percentual de estados visitados para um limite de 50 kb/s. a) Os agentes realizam de forma efetiva a priorização entre os clientes, reduzindo a banda permitida para os demais clientes e aumentando para o Cliente 3 mais prioritário usando o modo de auto-coibição, o que fica visível no boxplot considerando o us. b) O percentual de estados visitados diminui com o aumento no número de estados e implica o não preenchimento da Tabela Q, levando a decisões aleatórias em caso de visita a estados ainda não visitados.

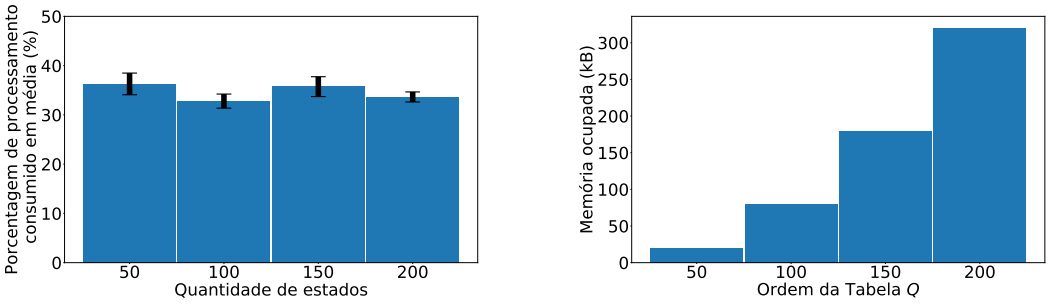
utilizar a nuvem, os agentes dos outros clientes limitam suas ações a somente diminuir a banda máxima do seu cliente, até que todos os clientes estejam com a banda mínima contratada para garantir a priorização e a justiça na alocação. Simultaneamente, reduz-se a taxa de aprendizado α para zero e os agentes entram em modo de auto-coibição. Ao retornar à operação normal, a tendência dos agentes de aumentar o tráfego não é modificada pela diminuição brusca provocada pela auto-coibição para garantir os acordos de nível de serviço.

A Figura 4(b) mostra a porcentagem média de estados visitados pelo mecanismo proposto em função do número total de estados considerados na modelagem. No cenário avaliado, os agentes podem tomar um máximo de 100 decisões e 9 ações possíveis. Observa-se uma redução na média de visitação de estados à medida em que o número total de estados considerados aumenta. A não visitação de um estado implica não haver atualização da Tabela Q para esse estado. Consequentemente, se o estado até então não visitado for visitado, o agente tomará uma ação aleatória. Dessa forma, não é recomendável um aumento exacerbado da quantidade de estados.

A métrica definida para verificar a adequação [Mattos e Duarte, 2012] calcula o percentual de acerto da alocação dos recursos baseado na prioridade de cada cliente:

$$adequacao = 1 - \left| 1 - \frac{\text{Banda Obtida}}{\text{Banda Esperada}} \right|. \tag{6}$$

Essa métrica determina o quão próximo da banda contratada a utilização do cliente está. A adequação no cenário avaliado é de 30%, em média. Esse valor se deve ao comportamento de dente de serra provocado pelo modo de auto-coibição dos agentes.



(a) Consumo de processamento.

(b) Consumo de memória.

Figura 5. Impacto do aumento de estados no processamento e na memória dos agentes. a) Relação entre a quantidade de estados e o processamento utilizado. Há pouca variação no processamento comprometido. b) Uso de memória considerando uma Tabela Q de n ações por n estados. A variação em n aumenta a memória consumida exponencialmente.

A Figura 5 mostra a variação do consumo de memória e de processamento em função do tamanho da Tabela Q. Considerando uma Tabela Q de ordem k, isto é, com k ações e k estados, o aumento de k implica um aumento exponencial no consumo de memória, conforme ilustrado na Figura 5(a). O processamento se mantém constante, em torno de 36% no cenário avaliado, independentemente do tamanho da Tabela Q. O fato do processamento se manter constante, independente do tamanho da Tabela Q utilizada, permite a variação do ponto de estabilidade do sistema, variando o número de estados e de ações sem restrições de desempenho. Quanto à memória, no caso do uso de grande quantidade de ações e estados, o mecanismo tem a praticabilidade prejudicada, perdendo a escalabilidade, dado que cada novo cliente necessita uma Tabela Q.

6. Os trabalhos relacionados

Russel e Zindars utilizam o método de decomposição Q, que visa não utilizar a estrutura comum de um agente monolítico, mas uma divisão de subagentes munidos da técnica de *Q-learning* que recebem as mesmas informações, cada um com a sua própria Tabela Q [Russell e Zimdars, 2003]. Em contraste, este artigo busca uma otimização para cada agente ao mesmo tempo em que implementa um modo de auto-coibição que funciona como uma restrição entre os agentes para não causar o colapso do sistema.

Mera-Gomez *et al.* propõem uma abordagem baseada em *Q-learning*, semelhante à de Russel e Zindars. Os estados são modelados de acordo com a proporção de clientes com uma ou mais requisições na fila e a priorização é realizada de acordo com a distância do período de tarifação [Mera-Gómez et al., 2017]. Diferentemente, neste artigo os estados são baseados inteiramente na banda contratada. Habib e Khan adotam uma proposta de ações similar a de Mera-Gomez *et al.*, com foco na alocação de recursos em nuvens, porém comparando com outro algoritmo de aprendizado por reforço, o SARSA (*State-Action-Reward-State-Action*) [Habib e Khan, 2016]. O trabalho conclui que o algoritmo *Q-learning* apresenta melhor desempenho em relação ao tempo de convergência.

Barret *et al.* reduzem o tempo de convergência do *Q-learning* utilizando paralelismo. Na proposta, compartilha-se as experiências entre os agentes a fim de benefi-

ciar o sistema como um todo [Barrett et al., 2013]. Dutreilh *et al.* utilizam uma abordagem adaptativa baseada em *Q-learning* com um limite variável de carga na nuvem modelado por uma função sinusoidal. Rao *et al.* desenvolvem uma abordagem baseada em aprendizado por reforço para realizar autoconfiguração de máquinas virtuais em nuvens. A abordagem aponta soluções próximas às ótimas para sistemas de baixa escala, fazendo as máquinas virtuais se auto-reconfigurarem de acordo com mudanças nas aplicações [Rao et al., 2009]. Diferentemente dos outros autores, Wang *et al.* utilizam uma técnica de *deep Q-learning* para o provisionamento de uma nuvem buscando balancear os custos e o desempenho da nuvem [Wang et al., 2017].

As abordagens citadas assumem um processo de decisão de Markov (*Markov Decision Process* - MDP) e aplicam algoritmos de aprendizado por reforço para resolver o problema. Este artigo também assume um MDP, com agentes independentes, já que a decisão dos outros agentes não tem influência na ação tomada, havendo somente a restrição de que o tráfego total direcionado à nuvem não pode ultrapassar o limite M. Além disso, este trabalho se diferencia ao utilizar políticas diferentes para garantir o cumprimento das SLAs, o modo de operação normal e o modo de auto-coibição.

7. Conclusão

Este artigo propôs a utilização de aprendizado por reforço para realizar o compartilhamento de recursos de rede em uma nuvem. Os agentes são modelados para impedir que a nuvem seja sobrecarregada e para garantir a priorização entre os clientes. A proposta utiliza a técnica *Q-learning* com múltiplos agentes e alternância de políticas para coordenar o tráfego na nuvem, respeitando um limite máximo de tráfego. Os resultados mostram que o mecanismo proposto de fato previne a sobrecarga da nuvem e reduz, simultaneamente, os recursos ociosos. Além disso, o mecanismo prioriza os clientes de forma dinâmica e ágil, garantindo os níveis de acordo de serviço. O mecanismo proposto é simples e apresenta baixo consumo de memória e de processamento, sendo, portanto, escalável.Vislumbra-se como trabalho futuro a utilização de outras técnicas de aprendizado por reforço, como *deep Q-learning* e SARSA (*State-Action-Reward-State-Action*). Além disso, pretende-se explorar as relações entre processamento e banda para reduzir a necessidade de monitoramento.

Referências

Ahalawat, A., Dash, S. S., Panda, A. e Babu, K. S. (2019). Entropy based DDoS detection and mitigation in OpenFlow enabled SDN. Em *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, p. 1–5. IEEE.

Bari, M. F., Boutaba, R., Esteves, R., Granville, L. Z., Podlesny, M., Rabbani, M. G., Zhang, Q. e Zhani, M. F. (2012). Data center network virtualization: A survey. *IEEE Communications Surveys & Tutorials*, 15(2):909–928.

Barrett, E., Howley, E. e Duggan, J. (2013). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674.

Carvalho, H. E. T., Fernandes, N. C. e Duarte, O. C. M. B. (2011). Um controlador robusto de acordos de nível de serviço para redes virtuais baseado em logica nebulosa. Em *Anais do XXIX Simpósio Brasileiro de Redes de Computadores-SBRC'11*, Campo Grande, MS.

Chen, L., Li, B. e Li, B. (2014). Allocating bandwidth in datacenter networks: A survey. *Journal of Computer Science and Technology*, 29(5):910–917.

Chen, Z., Hu, J. e Min, G. (2019). Learning-based resource allocation in cloud data center using advantage actor-critic. Em *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, p. 1–6. IEEE.

Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N. e Truck, I. (2011). Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. Em *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, p. 67–74.

Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. e Stoica, I. (2009). Above the clouds: A Berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28(13):2009.

Guo, J., Song, Z., Cui, Y., Liu, Z. e Ji, Y. (2017). Energy-efficient resource allocation for multi-user mobile edge computing. Em *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, p. 1–7.

Habib, A. e Khan, M. I. (2016). Reinforcement learning based autonomic virtual machine management in clouds. Em *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, p. 1083–1088. IEEE.

Isbell, C., Shelton, C. R., Kearns, M., Singh, S. e Stone, P. (2001). A social reinforcement learning agent. Em *Proceedings of the fifth international conference on Autonomous agents*, p. 377–384. ACM.

Lantz, B., Heller, B. e McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. Em *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, p. 19:1–19:6, New York, NY, USA.

Mattos, D. M. F. e Duarte, O. C. M. B. (2012). Qflow: Um sistema com garantia de isolamento e oferta de qualidade de serviço para redes virtualizadas. Em *Anais do XXX Simpósio Brasileiro de Redes de Computadores - SBRC 2012*, Ouro Preto, MG.

Medeiros, D. S. V., Cunha Neto, H. N., Andreoni Lopez, M., Magalhães, L. C. S., Silva, E. F., Vieira, A. B., Fernandes, N. C. e Mattos, D. M. F. (2019). Análise de dados em redes sem fio de grande porte: Processamento em fluxo em tempo real, tendências e desafios. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2019:142–195.

Mera-Gómez, C., Ramírez, F., Bahsoon, R. e Buyya, R. (2017). A debt-aware learning approach for resource adaptations in cloud elasticity management. Em *International Conference on Service-Oriented Computing*, p. 367–382. Springer.

Rao, J., Bu, X., Xu, C.-Z., Wang, L. e Yin, G. (2009). VCONF: a reinforcement learning approach to virtual machines auto-configuration. Em *Proceedings of the 6th international conference on Autonomic computing*, p. 137–146. ACM.

Russell, S. J. e Zimdars, A. (2003). Q-decomposition for reinforcement learning agents. Em *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, p. 656–663.

Wang, Z., Gwon, C., Oates, T. e Iezzi, A. (2017). Automated cloud provisioning on AWS using deep reinforcement learning. *arXiv preprint arXiv:1709.04305*.

Xia, S.-M., Guo, S.-Z., Bai, W., Qiu, J.-Y., Wei, H. e Pan, Z.-S. (2019). A new smart router-throttling method to mitigate DDoS attacks. *IEEE Access*, 7:107952–107963.

Yau, D. K., Lui, J. C., Liang, F. e Yam, Y. (2005). Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. *IEEE/ACM Tran-*

sactions on Networking, 13(1):29–42.