UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

REINER HENRIQUE DOS SANTOS FILHO

Desenvolvimento de um Mecanismo de Provisionamento de Banda Dinâmico para um Centro de Dados Multi-Inquilino Utilizando Aprendizado por Reforço Difuso

NITERÓI

2021

UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

REINER HENRIQUE DOS SANTOS FILHO

Desenvolvimento de um Mecanismo de Provisionamento de Banda Dinâmico para um Centro de Dados Multi-Inquilino Utilizando Aprendizado por Reforço Difuso

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações.

Orientadores:

Profª. Dianne Scherly Varela de Medeiros, D.Sc.

Prof. Diogo Menezes Ferrazani Mattos, D.Sc.

NITERÓI

2021

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

S237d Santos filho, Reiner Henrique dos
Desenvolvimento de um Mecanismo de Provisionamento de Banda Dinâmico para um centro de Dados Multi-Inquilino Utilizando Aprendizado por Reforço Difuso / Reiner Henrique dos Santos filho, Dianne Scherly Varela de Medeiros, Diogo Menezes Ferrazani Mattos ; Dianne Scherly Varela de Medeiros, orientadora ; Diogo Menezes Ferrazani Mattos, coorientador. Niterói, 2021.
79 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2021.

DOI: http://dx.doi.org/10.22409/PPGEET.2021.m.14464048706

1. Aprendizado de máquina. 2. Lógica fuzzy. 3. Nuvem. 4. Otimização. 5. Produção intelectual. I. Medeiros, Dianne Scherly Varela de. II. Mattos, Diogo Menezes Ferrazani. III. Medeiros, Dianne Scherly Varela de, orientadora. IV. Mattos, Diogo Menezes Ferrazani, coorientador. V. Universidade Federal Fluminense. Escola de Engenharia. VI. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

REINER HENRIQUE DOS SANTOS FILHO

DESENVOLVIMENTO DE UM MECANISMO DE PROVISIONAMENTO DE BANDA DINÂMICO PARA UM CENTRO DE DADOS MULTI-INQUILINO UTILIZANDO APRENDIZADO POR REFORÇO DIFUSO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a Obtenção do Grau de Mestre em Engenharia Elétrica e de Telecomunicações.
Área de concentração: Sistemas de Telecomunicações.

Aprovado em 08 de julho de 2021.

BANCA EXAMINADORA

Prof. Dra. Dianne Scherly Varela de Medeiros – Orientadora
Universidade Federal Fluminense - UFF

Prof. Dr. Diogo Menezes Ferrazani Mattos - Orientador
Universidade Federal Fluminense - UFF

Prof. Dr. Tadeu Nagashima Ferreira
Universidade Federal Fluminense – UFF

Prof. Dr. Rodrigo de Souza Couto
Universidade Federal do Rio de Janeiro – UFRJ

Niterói
2021

À minha noiva.

Agradecimentos

Agradeço à minha noiva, Lyvia Lannder Guimarães Oliveira, por ter me apoiado de todas as formas possíveis para que eu pudesse concluir essa jornada. Seja por um abraço quando parecia que tudo iria dar errado ou virar a noite comigo quando eu precisava cumprir um prazo de um artigo.

Agradeço à minha mãe, Marcia Cristina Marinho Pereira Marques, por ter me dado carona e apoio financeiro nos primeiros meses, um dos fatores que tornou possível fazer mestrado.

Agradeço aos meus orientadores, Diogo e Dianne, por me darem todo o suporte possível e sempre tentar extrair o meu melhor a cada passo dessa jornada, por nunca desistirem de mim mesmo eu querendo o fazer por diversas vezes.

Agradeço aos professores da banca, Tadeu Nagashima Ferreira e Rodrigo de Souza Couto, pela disponibilidade.

Por fim, agradeço aos órgãos de fomento CNPq, CAPES, RNP e FAPERJ pelo financiamento parcial deste trabalho.

Resumo

A preocupação em violar Acordos de Nível de Serviço (SLA)s leva Provedores de Serviços em Nuvem a confiarem no sobredimensionamento da infraestrutura de rede para garantir a largura de banda necessária durante picos de demanda. Nesse sentido, durante a operação típica da infraestrutura de rede, existem recursos ociosos que podem ser redistribuídos, permitindo aos inquilinos usarem uma largura de banda maior do que a contratada e, consequentemente, aumentar a receita do provedor. Este trabalho propõe um mecanismo automatizado para provisionamento de largura de banda, com o objetivo de otimizar o uso de recursos de rede em provedores de serviço em nuvem. O mecanismo proposto permite que os inquilinos excedam o consumo da largura de banda contratada temporariamente. Simultaneamente, o mecanismo prioriza inquilinos de forma a manter a conformidade com os SLAs de cada inquilino. Para tanto, este trabalho avalia o uso de três estratégias para implementação de agentes que atuam sobre a regulação da largura de banda de cada inquilino. A primeira usa (i) lógica difusa, a segunda, (ii) aprendizado por reforço, e a terceira usa (iii) aprendizado por reforço difuso. A proposta é validada inicialmente em um cenário de pequena escala com tráfego UDP com taxa constante, verificando a viabilidade das estratégias (i) e (ii). Em seguida, avalia-se o mecanismo proposto em um cenário de maior escala no qual o tráfego é gerado com base em um conjunto de dados real que contém informações sobre fluxos de acesso a servidores *web*. Independentemente da estratégia utilizada, os resultados mostram que o mecanismo proposto regula o consumo de largura de banda e prioriza inquilinos para que os SLAs sejam cumpridos. O melhor resultado é obtido utilizando a estratégia baseada em algoritmos de aprendizado por reforço difuso. Essa estratégia aumenta a utilização da infraestrutura de rede do provedor por cada inquilino em 39% quando comparado à estratégia de lógica difusa, mantendo a carga de tráfego da nuvem dentro de um limite definido pelo provedor. Além disso, o mecanismo pode ser usado tendo somente o processamento como entrada e é robusto a entrada e saída de inquilinos.

Palavras-chave: Aprendizado por reforço, provisionamento de largura de banda, centro de dados multi-inquilino, lógica difusa.

Abstract

Concern about violating Service Level Agreements (SLAs) leads Cloud Service Providers to rely on over-scaling their network infrastructure to ensure the necessary bandwidth during peak demand. In this sense, during the typical operation of the network infrastructure, there are idle resources that can be redistributed, allowing tenants to use a larger bandwidth than hired and, consequently, increase the provider's revenue. This paper proposes an automated mechanism for provisioning bandwidth to optimize the use of network resources for cloud service providers. The proposed mechanism allows tenants to exceed the consumption of the temporarily contracted bandwidth. At the same time, the mechanism prioritizes tenants in order to maintain compliance with each tenant's SLAs. To this end, this work evaluates the use of three strategies for implementing agents that act on regulating the bandwidth of each tenant. The first uses (i) fuzzy logic, the second, (ii) reinforcement learning, and the third uses (iii) diffuse reinforcement learning. The proposal is initially evaluated in a small-scale scenario with UDP traffic at a constant rate, verifying the viability of strategies (i) and (ii). Then, the proposed mechanism is validated in a larger scale scenario in which traffic is generated based on a real data set that contains information about access flows to servers web. Regardless of the strategy used, the results show that the proposed mechanism regulates the consumption of bandwidth and prioritizes tenants so that the SLAs are fulfilled. The best result is obtained using a strategy based on fuzzy reinforcement learning algorithms. This strategy increases the use of the provider's network infrastructure by each tenant by 39% when compared to the fuzzy logic strategy, keeping the traffic load of the cloud within a limit defined by the provider. In addition, the mechanism can be used with only processing as input, and it is robust for tenants to enter and exit.

Keywords: Reinforcement Learning, Bandwidth Provisioning, Multitenant Datacenter, Fuzzy.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

SDN	*Sofware Defined Networking*
CPU	Unidade de Processamento Central
NP	Não Polinomial
NIST	*National Institute of Standards and Technology*
MLFQ	*Multilevel Feedback Queue*
HTB	*Hierarchy Token Bucket*
IIoT	*Internet das Coisas Industrial*
HTTP	*Hypertext protocol*
MDP	Processo de Decisão Markoviano
QoS	Qualidade de Serviço
CENIC	*Corporation For Education Network Initiatives In California*
NYU	Universidade Politécnica de Nova Iorque
EWMA	*Média Móvel Exponencial Ponderada*
HOE-DCN	*Hybrid optical-electrical switching based data center network*
SLAs	*Service Level Agreements*
AuTO	*Automatic Traffic Optimization*
ABC	*Adaptive Bandwidth Control*
DART	*Dynamic Bandwidth Distribution*
DRL	Aprendizado por Reforço Profundo
ABC	Controle Adaptativo de Largura de Banda
CBR	Taxa de Bit Constante
SLA	Acordos de Nível de Serviço
IaaS	Infraestrutura como um Serviço
PaaS	Plataforma como um Serviço

SaaS	*Software* como um Serviço
SARSA	*State-Action-Reward-State-Action*
RL	Reinforcement Learning
FRL	Fuzzy Reinforcement Learning
FSL	SARSA difuso
FIS	Sistema de inferência difuso
FQL	*Q-learning* difuso
PIAS	*Practical Information-Agnostic Flow Scheduling*
LAS	*Least Attained Service*
DIAS	*Distributed Information-agnostic Flow Scheduling*
UCP	*Urgency-based Congestion Control*
MAPE	*Monitor, Analysis, Plan, and Execute*
RLPAS	*Reinforcement Learning-based Proactive Auto-Scaler*
ARLCA	*Agente de Consolidação de Aprendizado por Reforço*
PBRS	*Modelagem Baseada na Recompensa Potencial*
REST	Transferência de Estado Representacional
SBRC	Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
IWSSIP	*International Conference on Systems, Signals and Image Processing*
NOF	*Network of Future*
MPLS	*Multiprotocol Label Switching*

Sumário

Capítulo 1

Introdução

A computação em nuvem é definida pelo *National Institute of Standards and Technology* (NIST) [1] como um modelo computacional que permite o acesso conveniente, ubíquo e sob demanda a um conjunto de diferentes recursos computacionais configuráveis, como processamento, largura de banda, aplicações e serviços. Elasticidade é a capacidade de um sistema de se adaptar a uma carga de trabalho por meio de provisionamento automático de recursos computacionais [2]. A elasticidade permite uma configuração de recursos de acordo com as necessidades do inquilino, ajustando-se de acordo com as mudanças na demanda. Migrar do gerenciamento próprio para a computação em nuvem gera cerca de 37% de economia para os inquilinos de um provedor [3]. O custo-benefício e a elasticidade atraem atenção de empresas e as leva a migrar suas aplicações e serviços atuais para arquiteturas baseadas em nuvem. Empresas como Facebook e Dropbox, por exemplo, migraram seu armazenamento de vídeo para a nuvem sem construir os seus próprios centros de dados [4]. Um provedor de recursos em nuvem é capaz de multiplexar disco, memória, processamento e recursos de redes entre inquilinos, possibilitando acomodar diferentes aplicações com diferentes requerimentos, incorrendo em diversidade nos tipos de serviços que acomoda.

Os serviços de computação em nuvem variam e podem ser classificados de acordo com o modelo. Existem três modelos de serviços atualmente, de acordo com o NIST: Infraestrutura como um Serviço (IaaS), Plataforma como um Serviço (PaaS) e *Software* como um Serviço (SaaS). O IaaS permite ao inquilino interagir diretamente com os recursos de *hardware* do centro de dados. O inquilino utiliza recursos computacionais e a infraestrutura de rede do provedor para executar as aplicações desenvolvidas pelo inquilino [5]. No modelo PaaS, os inquilinos usam infraestrutura, bibliotecas de linguagens de programação, ferramentas, e serviços fornecidos pelo provedor possuindo a vantagem

do inquilino ter a capacidade de desenvolver e executar suas próprias aplicações sem se responsabilizar pela infraestrutura prévia, fornecida pelo provedor. Entretanto, nesse modelo de serviço o inquilino não tem controle sobre o *hardware* utilizado, a adaptação à aplicação do inquilino é centrada no provedor do serviço [6]. Por fim, no modelo SaaS a aplicação é desenvolvida pelo provedor e é executada no centro de dados desse provedor. O inquilino utiliza diretamente a aplicação, que é completamente gerenciada pelo provedor. Esse modelo tem como vantagem não ser necessário instalar a aplicação localmente para poder utilizá-la [7].

Diversos trabalhos abordam o provisionamento de recursos em nuvens, que continua um desafio sem uma solução ótima. Dessa forma, esta dissertação foca em uma abordagem de provisionamento de recursos de rede em nuvens multi-inquilino utilizando um algoritmo de aprendizado por reforço difuso para cumprir os *Service Level Agreements* (SLAs) dos inquilinos ao mesmo tempo em que se busca maximizar o lucro para o provedor da infraestrutura.

1.1 Motivação

Os provedores de serviços em nuvem atendem a uma grande diversidade de demandas de aplicativos de vários inquilinos. Suprir os recursos de armazenamento, memória e rede exigidos por cada aplicação é um problema de otimização Não Polinomial (NP) difícil para o provedor de infraestrutura que visa maximizar a receita obtida [8]. Além disso, a existência de recursos ociosos pode ser interpretada como perda de receita, visto que os recursos ociosos poderiam ser alocados para suportar novos inquilinos ou aumentar a quantidade de recursos contratados por um inquilino ativo. Recursos de rede estão entre os recursos críticos necessários para garantir a conformidade com o acordo de nível de serviço em nuvens [9].

A alta utilização de recursos em uma nuvem é alcançada reduzindo a ociosidade de recursos, enquanto o uso proporcional de recursos implica em seu compartilhamento entre inquilinos de acordo com seus SLAs, incorrendo em priorização. Para tirar proveito da alta utilização, surge o modelo de negócios *pay-as-you-go*, em que os inquilinos podem exceder sua capacidade contratada na condição de haver recursos disponíveis na infraestrutura em nuvem, o que significa taxas adicionais para os inquilinos [10]. A redistribuição de recursos de rede ociosos requer o uso de mecanismos que garantam a conformidade com os acordos de nível de serviço e o aproveitamento ótimo dos recursos da infraestrutura. Os

provedores IaaS tendem a não empregar métodos para garantir a largura de banda mínima para um único inquilino, pois há uma relação de compromisso entre garantia mínima para inquilinos e alta utilização da nuvem, impor limites de utilização é contrário a utilizar o máximo de capacidade. Este compromisso dificulta o cumprimento dos limites de largura de banda contratada, ao mesmo tempo em que se visa alta utilização e proporcionalidade da rede.

Trabalhos anteriores alocam máquinas virtuais em uma nuvem, modificando o número de máquinas virtuais ativas [11, 12, 13]. Esses trabalhos se concentram no compartilhamento de memória e processamento localmente por inquilinos usando o mesmo servidor. No entanto, os recursos da rede são compartilhados entre todos os inquilinos, ao contrário de outros recursos computacionais [4], o que implica a possibilidade de os inquilinos compartilharem enlaces de gargalo, sem garantias de banda. Entretanto, a dependência de tráfego de dados para a carga de trabalho do centro de dados implica que a variabilidade da banda reflete em imprevisibilidade nas aplicações [14]. Existem aplicações que utilizam largura de banda de forma intensiva [15] ou que requerem um desempenho de rede confiável, como aplicativos médicos e de gerenciamento de desastres [16].

1.2 Objetivos

Esta dissertação tem como objetivo geral reduzir a ociosidade de recursos de rede em centros de dados com múltiplos inquilinos, mantendo a conformidade com os acordos de nível de serviço. Dessa forma, propõe-se um mecanismo dinâmico de provisionamento de recursos de rede, visando maximizar a receita do provedor de nuvem, considerando o modelo de negócios *pay-as-you-go*. O mecanismo proposto tem como base um modelo de aprendizado por reforço difuso [17] e se apoia no uso de um controlador de redes definidas por software, *Sofware Defined Networking* (SDN), para realizar o controle da largura de banda atribuída para cada inquilino. A fim de alcançar o objetivo desta dissertação, os seguintes objetivos previstos são (i) desenvolver modelos de aprendizado por reforço, utilizando os algoritmos de *Q-Learning*, *State-Action-Reward-State-Action* (SARSA) e lógica difusa para o provisionamento de recursos de rede em nuvens multi-inquilino maximizando o lucro para o provedor e respeitando as SLAs dos inquilinos e (ii) desenvolver um protótipo funcional de provisionamento de recursos em nuvens através de emulações.

1.3 Contribuições

Nesta dissertação, propõe-se um mecanismo para provisionamento de largura de banda utilizando um modelo de aprendizado por reforço difuso. As principais contribuições da dissertação são destacadas a seguir:

- elaboração de um mecanismo de provisionamento de recursos de rede usando aprendizado por reforço com a banda de cada inquilino como entrada;
- elaboração de um mecanismo de provisionamento de recursos de rede usando aprendizado por reforço com processamento gasto por cada inquilino como entrada;
- elaboração de um mecanismo de provisionamento de recursos de rede usando lógica difusa com a banda de cada inquilino como entrada.

Essas contribuições estão reportadas nos seguintes artigos, em ordem de publicação:

- Agentes Inteligentes baseados em Aprendizado por Reforço para Alocação Dinâmica de Tráfego em Nuvens, Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 2020;
- A Lightweight Reinforcement-Learning-Based Mechanism for Bandwidth Provisioning on Multitenant Data Center, *International Conference on Systems, Signals and Image Processing* (IWSSIP), 2020;
- A Rapid Fuzzy Controller for Decentralized Bandwidth Provisioning on a Multitenant Data Center, *Network of Future* (NOF), 2020.

1.4 Organização da dissertação

Esta dissertação está organizada em 6 capítulos, da seguinte forma. O Capítulo 2 apresenta a descrição do problema de provisionamento de recursos de rede em nuvens, juntamente com a fundamentação teórica de aprendizado por reforço, lógica difusa e os trabalhos relacionados. O Capítulo 3 apresenta a proposta desta dissertação, detalhando o desenvolvimento e a implementação do mecanismo para provisionamento de largura de banda em centros de dados com múltiplos inquilinos. No Capítulo 4, discute-se o conjunto de dados utilizado para validar o funcionamento do mecanismo proposto e um

detalhamento dos cenários utilizados para validação e avaliação do mecanismo proposto. O Capítulo 5 apresenta e discute os resultados obtidos. Finalmente, o Capítulo 6 apresenta as conclusões desta dissertação e sugestões para trabalhos futuros.

Capítulo 2

Computação em nuvem e aprendizado por reforço difuso

A proposta presente nesta dissertação para a resolução do problema de provisionamento de banda em nuvens multi-inquilino utiliza conceitos de aprendizado por reforço e lógica difusa. Este capítulo visa introduzir o provisionamento de banda em centros de dados multi-inquilino, aprendizado por reforço e lógica difusa, de modo a guiar a uma melhor compreensão da proposta e escolhas feitas para construí-la. Devido à grande quantidade de variáveis usada neste capítulo, sumariza-se a notação na Tabela 2.1 para facilitar referências futuras.

2.1 O paradigma da computação em nuvem

A computação em nuvem translada a localidade da infraestrutura computacional para a rede com o objetivo de reduzir custos associados à manutenção de recursos de *hardware* e *software* [18]. Tais recursos podem ser dinamicamente realocados com o intuito de se aproximar de um uso ótimo de recursos. Um uso ótimo significa consumir todos os recursos maximizando a vazão, diminuindo o tempo de resposta e evitando sobrecarga.

No contexto de computação em nuvem interagem provedores de serviço e seus inquilinos. Aos provedores de serviço, cabe tornar seus serviços acessíveis aos inquilinos através de interfaces de internet, geralmente por modelos de tarifação *pay-as-you-go*, cumprindo SLAs dos inquilinos [19]. Nesse contexto há diversos tipos de serviço em computação em nuvem, a Figura 2.1 ilustra os tipos de serviço existentes:

Tabela 2.1: Lista de variáveis

$\mathcal{S}$	:	Espaço de estados
$\mathcal{A}$	:	Espaço de ações
p	:	Conjunto de probabilidades de mudanças entre estados
q	:	Conjunto de probabilidades de receber recompensas
s_i	:	Um estado pertencente à $\mathcal{S}$
s_t	:	Estado no instante t
a_t	:	Ação no instante t
$Q(s_t, a_t)$	:	Função Q no instante t
ϵ	:	Probabilidade do agente agir de maneira randômica
γ	:	Fator de Desconto
α	:	Taxa de Aprendizado
$\pi(s, a)$	:	Probabilidade de escolha da ação a no estado S
T	:	Temperatura da estratégia softmax
m	:	Número de estados
r	:	Recompensa
r_n	:	Recompensa para o inquilino n
x	:	Um vetor de variáveis difusas
x_i	:	Uma variável difusa
$a(x)$	:	Uma ação escolhida, dependente de x
$\theta(x)$	:	Ativação da regra i, dependendo de x
N	:	Número de regras difusas
M	:	Limite de banda da nuvem
n_{fs}	:	Número de frações de M usado para determinar o número de estados
n_{fa}	:	Número de frações de M usado para determinar o número de ações
L	:	Carga na agregada medida na nuvem
L_n	:	Atual uso do inquilino n
g_n	:	Ganho de tráfego para o inquilino n
g'_n	:	Ganho de tráfego para o inquilino n na iteração anterior
F_n	:	Valor total pago pelo inquilino n
I_n	:	Banda contratada pelo inquilino n
r_n	:	Banda ultrapassada pelo inquilino n
R	:	Preço por cada unidade de largura de banda

1. **IaaS:** Recursos como processamento, memória, armazenamento e largura de banda são alocados em máquinas virtuais. Através da tecnologia de virtualização, os provedores de serviço podem clonar máquinas, realocar recursos ou construir sistemas com redes *ad-hoc* sob demanda dos inquilinos. É um modelo que dá liberdade ao inquilino para construir seus próprios *softwares* e próprias simulações, porém tem a desvantagem de normalmente não possuir facilitadores dados pelo provedor além da infraestrutura.

2. **PaaS:** Este modelo provê mais um nível de abstração. Ao invés de apenas dar ao

inquilino uma estrutura virtualizada, os provedores de serviço podem dar plataformas em que os sistemas dos inquilinos serão hospedados. Bibliotecas de linguagens de programação, ferramentas e serviços fornecidos pelo provedor são utilizados pelos inquilinos, agindo como facilitadores para acelerar o processo de desenvolvimento ao mesmo tempo em que infraestrutura é dada.

3. **SaaS:** É uma alternativa a executar aplicações localmente. Há uma aplicação desenvolvida pelo provedor, executada na infraestrutura do provedor e é utilizada pelo inquilino. Corretores de texto online são os exemplos mais conhecidos desse modelo de serviço em nuvem.

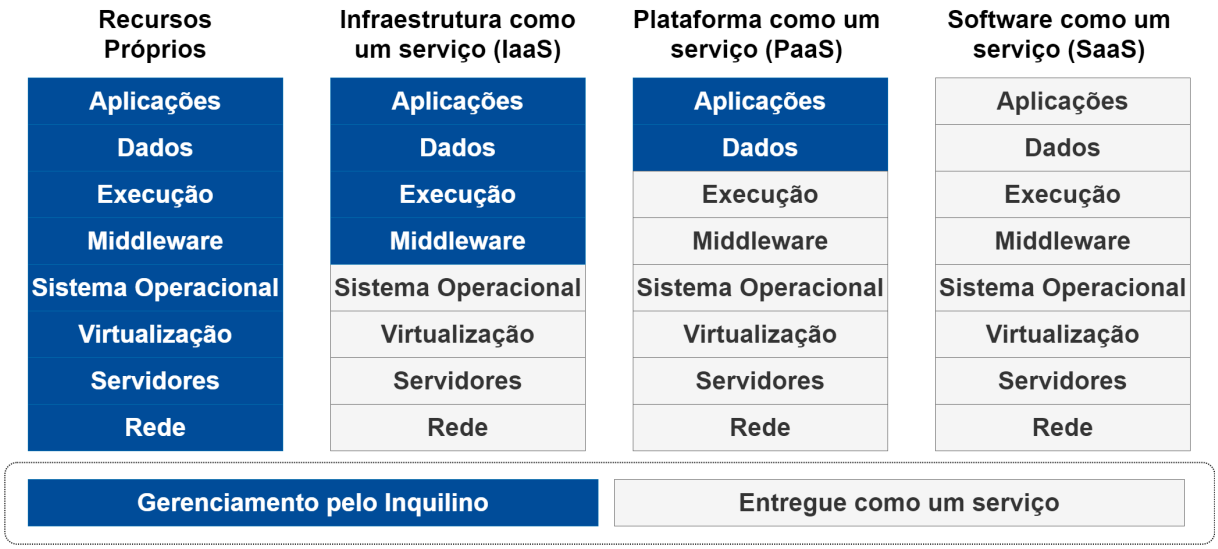


Figura 2.1: Tipos de serviços de computação em nuvem.

Escalabilidade é a habilidade de um sistema de sustentar cargas de trabalho crescentes com desempenho adequado [2]. O conceito de ambientes de nuvens multi-inquilino significa prover escalabilidade e benefícios econômicos para os inquilinos que compartilham a mesma plataforma de nuvem e a mesma plataforma de nuvem e sua infraestrutura [20]. Esta dissertação é direcionada para o tipo de serviço IaaS para uma nuvem multi-inquilino por ser o tipo de serviço mais amplamente oferecido [21]. Viabilidade, justiça, utilização, praticabilidade e garantias determinísticas são características necessárias para uma nuvem definidas por *Chen et al.* [22] como:

1. **viabilidade:** garantir a aplicação simplificada de procedimentos de alocação de largura de banda;

2. **justiça:** capacidade de proteger os inquilinos na presença de mau comportamento, garantindo a alocação justa de largura de banda mesmo em cenários de competição entre inquilinos;

3. **utilização:** capacidade de fazer com que recursos de rede não fiquem ociosos enquanto houver demandas não atendidas. 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 para os inquilinos cujos serviços têm a demanda de tráfego atendida. Provedores IaaS também se beneficiam do elevado uso de largura de banda, pois acomodam mais solicitações de inquilinos e, assim, gera-se mais receita.

4. **praticidade:** garantir a simplicidade e escalabilidade da alocação de largura de banda na nuvem, de forma que o mecanismo de alocação possa ser adaptado para um maior número de inquilinos com diferentes aplicações de maneira simples;

5. **garantia determinística:** garantir um desempenho previsível seguro para cada inquilino, independentemente de qualquer comportamento dos inquilinos concorrentes; Garantia de largura de banda mínima permite que os inquilinos ultrapassem sua largura de banda quando estiver disponível. Esse cenário permite que os inquilinos alcancem desempenho limitado no pior caso, quando não há largura de banda disponível, e custos limitados.

Todas estas características são levadas em consideração na construção do mecanismo usado neste trabalho. Provendo viabilidade por alocação por uma única requisição de um agente, justiça e utilização pela modelagem das regras difusas e parâmetros do aprendizado por reforço, praticidade por ser tratar de um sistema multiagente e garantia determinística por toda a sua estrutura.

2.2 O provisionamento de banda em nuvens multi-inquilino

Os provedores de serviços em nuvem precisam multiplexar em um conjunto limitado de recursos físicos as diversas solicitações advindas dos múltiplos inquilinos. Quando a disponibilidade de recursos é limitada, a alocação desses recursos torna-se um problema de otimização crítico e complexo. A eficiência da alocação dos recursos define o desempenho dos serviços implantados na nuvem [22]. A estratégia de alocação deve ter como um de

seus objetivos a minimização dos recursos ociosos, evitando desperdício. A minimização dos recursos ociosos pode implicar na necessidade de atribuir parte desses recursos a inquilinos cujas cotas já foram alcançadas, de forma a aumentar a receita do provedor.

Memória, processamento e armazenamento são recursos compartilhados por inquilinos de uma nuvem dentro do mesmo servidor. Diferentemente, os recursos de rede são compartilhados entre inquilinos que fazem solicitações ao mesmo centro de dados. Os inquilinos que consomem tarefas no mesmo centro de dados podem usar os mesmos enlaces para alcançar os servidores, levando ao compartilhamento de largura de banda em enlaces de gargalo. O provisionamento da largura de banda garante aos inquilinos desempenho previsível de suas aplicações. Previsibilidade, por sua vez, é um dos requerimentos para migrar aplicações para a nuvem. O provisionamento de largura de banda, no entanto, é um grande desafio devido à rápida variabilidade no tempo, que torna difícil a previsão de uso em cada enlace. Além da dificuldade de previsão devido à variabilidade, existe o problema do isolamento da rede [22] em nuvens decorrente do serviço IP de melhor esforço (*best-effort*) [23], comumente aplicado. O serviço de melhor esforço abre a possibilidade de um inquilino malicioso utilizar sua autonomia para aumentar o uso da largura de banda em detrimento dos demais, ou até mesmo inquilinos legítimos que eventualmente executam aplicações que requerem uso massivo de largura de banda. O isolamento de rede entre os inquilinos é uma ferramenta para minimizar as disrupções causadas por uso massivo de largura de banda por um pequeno grupo de inquilinos [24].

A alocação de recursos em lote, através da alocação de novas máquinas virtuais para o inquilino que necessita de mais recursos é o procedimento mais comum para resolver o problema provisionamento de recursos computacionais em provedores IaaS. No entanto, a alocação de novas máquinas virtuais não garante o desempenho para recursos de rede. Há uma série de dificuldades conhecidas para otimizar alocações de recursos computacionais em máquinas virtuais [25], da estratégia usada para mapear de maneira ótima um hospedeiro físico em máquinas virtuais até quantas e quais maquinas virtuais alocar para os inquilinos. Apesar dessas dificuldades, alocar máquinas virtuais é prático para provedores de nuvem. No entanto, essa ação por si só incorre em desempenho imprevisível de aplicações, o que resulta na insatisfação do inquilino e consequente perda de receita [26]. Em situações previsíveis, é eficaz fornecer recursos estaticamente. Entretanto, o uso de recursos de rede está aberto a situações inesperadas [11] que requerem um sistema de provisionamento dinâmico capaz de se adaptar à demanda de tráfego. Esse sistema leva em consideração as medidas atuais sobre o tráfego na rede e a heterogeneidade de requisitos demandados pelas aplicações dos inquilinos. Há uma relação de compromisso

entre a proporcionalidade do uso da rede e a garantia mínima. Garantir recursos mínimos a serem consumidos por inquilinos implica recursos ociosos, enquanto proporcionalidade implica uma utilização com o intuito de eliminar os mesmos recursos ociosos. Ao mesmo tempo, existem aplicações que exigem um limite de largura de banda confiável, como aplicações médicas [16]. Se uma nuvem possui M de capacidade de largura de banda para fornecer a todos os inquilinos, então, para cada inquilino, o problema de alocação de largura de banda pode ser modelado da seguinte forma. Para cada inquilino há uma tupla (c_n, L_n, F_n), onde c_n é a quantidade de contratos que um inquilino tem, L_n é a largura de banda total usada pelo inquilino e F_n é a taxa total paga pelo inquilino. O objetivo da alocação de banda para os provedores de nuvem é maximizar a receita, portanto, maximizar a expressão $\sum_{i=n} c_n F_n$, o que significa que a soma das taxas pagas por cada contrato é a maior possível. No entanto, esta alocação é submetida à restrição $\sum_{i=n} c_n L_n < M$ devido à quantidade limitada de recursos de rede. Esta definição é exatamente o problema da mochila (*knapsack problem*) unidimensional, considerando os contratos os objetos, a largura de banda consumida, o peso e as taxas o valor. O problema da mochila é um problema NP-Difícil, portanto, a alocação de recursos também é considerada NP-Difícil.

Em sistemas estacionários, aplicar a teoria de filas é eficaz [26, 27] uma vez que o modelo criado apresenta comportamento previsível de requisições de inquilinos sendo processadas até o seu término. Em sistemas com natureza dinâmica e sujeitos a diferentes cargas, o modelo utilizado para o controle precisa ser corrigido periodicamente. Apesar de a teoria de controle visar lidar com abordagens reativas, é comum usar essa teoria para abordagens proativas [28]. A teoria de controle geralmente falha em cenários de tráfego intenso, com *flash crowds*, sendo necessário utilizar outra ferramenta. Nesse caso, o aprendizado por reforço desempenha papel importante, por ser capaz de garantir uma utilização estável do sistema, mesmo em condições de carga altamente dinâmica [29]. No entanto, o sistema deve apresentar um comportamento previsível, pois o aprendizado por reforço depende da modelagem do problema de provisionamento como um Processo de Decisão Markoviano (MDP). Uma vantagem dos sistemas baseados em aprendizado por reforço é que eles identificam o modelo de desempenho e as políticas para o cenário de implantação sem conhecimento prévio. Embora seja uma solução robusta e adaptativa para automatizar os mecanismos de provisionamento de recursos [30], o aprendizado por reforço depende de um espaço discreto de estados e ações, e implica que o tempo entre as mudanças na rede deve ser menor do que o tempo de convergência do mecanismo. Essa condição é um requisito para o MDP, uma vez que o processo controlado deve ser estacionário durante o período de controle. Além disso, as soluções de aprendizado por

reforço puro apresentam uma convergência lenta e problemas de escalabilidade ao tentar expandir o número de estados e ações [31].

A lógica difusa é uma ferramenta robusta que permite criar modelos traduzindo a noção humana em regras compreensíveis por máquinas, tornando o processo enxuto e eficaz. Um sistema de inferência difusa é tão bom quanto as regras inseridas nele. A lógica difusa possui a desvantagem de não ser adaptativa, ou seja, o modelo não consegue se alterar automaticamente de acordo com as mudanças no ambiente. Mudanças na saída das regras que poderiam resultar em melhora considerável no desempenho podem não ser notadas pelo olho humano, não sendo, portanto, adicionadas ao modelo do sistema. Essa limitação é superada pelo aprendizado por reforço difuso, que altera a saída das regras difusas de acordo com a observação do ambiente. Nesse sentido, o aprendizado por reforço difuso [17] visa somar a adaptabilidade do aprendizado por reforço com a robustez das regras difusas. Dessa forma, a limitação do aprendizado por reforço de ações discretas é mitigada, uma vez que o conhecimento prévio pode ser inserido no modelo através das regras difusas. Ao mesmo tempo a limitação de lógica difusa, que é o fato das regras serem sempre as mesmas, não acontece no aprendizado por reforço difuso já que as regras são alteradas a cada iteração. O aprendizado por reforço difuso é tão simples de ser modelado quanto lógica difusa e converge mais rápido no cenário de alocação de largura de banda comparado ao aprendizado por reforço puro. Isso ocorre porque mais de um elemento de uma Tabela Q é atualizado ao mesmo tempo. Esse mecanismo de aprendizado não possui a limitação de possuir um conjunto discreto de ações predefinidas, uma vez que as regras difusas possibilitam a escolha de uma ação que é combinação linear de outras ações, podendo assumir qualquer valor contínuo dentro de um intervalo. Uma vez que possui estas vantagens, o aprendizado por reforço difuso é um bom candidato a algoritmo para determinar políticas de provisionamento de largura de banda.

2.3 Aprendizado por reforço

O aprendizado por reforço é uma das técnicas de aprendizado de máquina usadas em ambientes dinâmicos. Essa técnica possui um mecanismo de aprendizagem orientado a objetivos. O mecanismo conta com um agente, que é um módulo computacional capaz de atuar autonomamente para atingir os objetivos para os quais foi desenvolvido [32]. O agente otimiza uma recompensa cumulativa interagindo com o ambiente [33], a fim de aprender boas políticas para resolver problemas sequenciais de decisão [34]. O ambiente, por sua vez, é o lugar onde ocorre a aprendizagem. Além do agente e do ambiente, o

mecanismo conta com o sensor, uma entidade que analisa todas as variáveis do ambiente e fornece realimentação ao agente. Dependendo da implementação, o sensor pode ou não ser parte do próprio agente, sendo essa decisão limitada pela disponibilidade do *hardware*.

O ciclo comum de um processo de aprendizado por reforço está representado na Figura 2.2 com a interação entre as entidades que compõem o processo. A cada iteração o sensor analisa o ambiente (1), provê o estado atual (2) e a recompensa (3) para o agente, que atualiza sua política (4) e a utiliza (5) para tomar uma ação (6) de acordo com a estratégia de exploração. Assim, o agente muda a política a cada iteração, com base na realimentação do sensor, buscando sempre a melhor política.

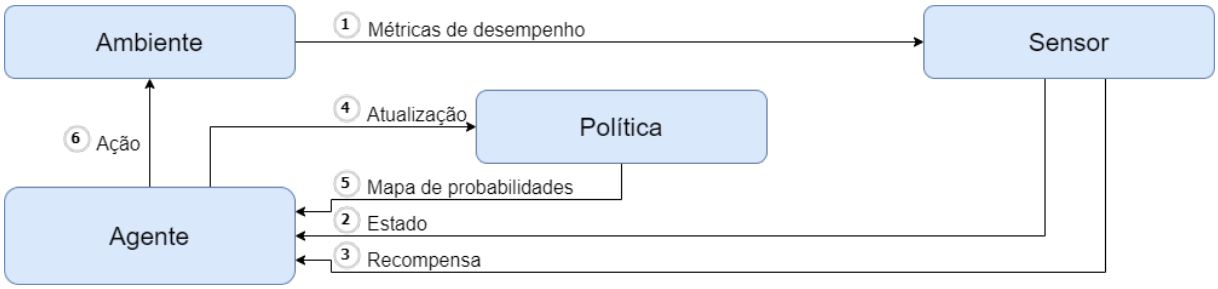


Figura 2.2: Esquema do funcionamento do processo de aprendizado por reforço.

Problemas que são resolvidos com técnicas de aprendizado por reforço podem ser modelados como um MDP. Quando todos os elementos da tupla do modelo do MDP são conhecidos, existe um modelo completo do ambiente e não é necessário usar o aprendizado por reforço. Quando não há um modelo completo, uma tentativa aproxima-o por meio de algoritmos baseados em modelo. Outra abordagem é estimar diretamente políticas ou funções para lidar com o problema usando algoritmos livre de modelo [35], como os algoritmos *Q-learning* e SARSA [12].

Um problema de decisão markoviano MDP é um problema de tomada de decisão de tempo discreto em que o tomador de decisão tem o objetivo de encontrar as melhores ações nos estados visitados pelo sistema. Para cada ação somente a política usada e o estado atual são levados em consideração. Para um MDP, o termo "Markov", significa que ações futuras dependem apenas do estado atual e da ação tomada [36]. Uma ação é considerada a melhor se maximiza uma métrica de desempenho, como uma recompensa acumulada [37]. Modelar um problema como um MDP é normalmente o primeiro passo para aplicar aprendizado por reforço na sua resolução.

Os MDP podem ser representados por uma tupla de quatro elementos $\langle \mathcal{S}, \mathcal{A}, p, q \rangle$ [12], sendo $\mathcal{S}$ o espaço de estados, $\mathcal{A}$ espaço de ações, p a probabilidade de transição entre os estados, e q a probabilidade de receber recompensas. Assim, partindo de um estado s_i,

é possível chegar a um estado s_j com probabilidade p. A escolha das ações para um determinado estado, depende da política utilizada.

Política, em aprendizado por reforço, é um mapa que relaciona um estado a uma ação. Para cada ação, há uma probabilidade de execução para um determinado estado dentro de $\mathcal{S}$ [38]. A política é modificada até ser considerada ótima segundo as métricas de desempenho escolhidas ou segue sendo modificada indefinidamente para casos de ambientes com características variantes no tempo.

Existem dois tipos de política, a política de comportamento é aquela usada para gerar dados para aprendizagem, e a política de destino é usada para tomar decisões. Se a política de comportamento for igual à política de destino, o algoritmo é denominado dentro da política (*on-policy*), caso contrário, o algoritmo é denominado fora da política (*off-policy*). Os algoritmos fora da política fornecem maior adaptabilidade em ambientes dinâmicos por necessitarem de menos amostras para convergência das políticas, embora sejam considerados mais difíceis de modelar [39]. Os algoritmos dentro da política são considerados de fácil modelagem, mas possuem uma eficiência mais baixa no aprendizado se comparados com fora da política [40]. A forma com que a política é usada, assim como a atualização da política, são ditadas pela estratégia de exploração escolhida.

A exploração (*exploration*) é o ato de testar novas ações em diferentes cenários, sendo necessária para diminuir a probabilidade da política de destino se fixar em ótimos locais. Aproveitamento (*exploitation*) é o ato de utilizar diretamente a política de destino para tomar decisões. Explorar em excesso leva a uma baixa recompensa acumulada, enquanto aproveitar em excesso leva o algoritmo a se fixar em um ótimo local, não explorando outras possibilidades. Esse dilema é conhecido como dilema de exploração-aproveitamento (*exploration-exploitation dilemma*) [41]. Uma estratégia de exploração define a relação entre exploração e aproveitamento.

Uma das estratégias de exploração mais amplamente utilizadas é a ϵ-*greedy* por simplicidade e por ter seu desempenho dificilmente superado segundo experimentos [42, 43]. A estratégia ϵ-*greedy* funciona com um parâmetro ϵ, que representa a probabilidade de se realizar uma ação aleatória, ou seja, a probabilidade de explorar. Por outro lado, existe uma probabilidade de $1 - \epsilon$ de aproveitar a politica de destino existente.

Outra estratégia comumente utilizada é a exploração *softmax* (ou Boltzmann), que funciona classificando as ações, em vez de apenas selecioná-las aleatoriamente como a ϵ-*greedy*. Na estratégia *softmax*, cada ação possui uma probabilidade de ser selecionada, e esta probabilidade é atualizada a cada iteração de acordo com as métricas de desempenho.

A equação (2.1) descreve o comportamento da estratégia *softmax*

$$\pi(s_t, a) = \frac{e^{Q_t(s_t,a)/T}}{\sum_{i=1}^{m} e^{Q_t(s_t,a_i)/T}}, \tag{2.1}$$

em que T é um parâmetro de aleatoriedade que varia no intervalo $[0, +\infty)$ e π é a probabilidade da ação a ser selecionado no estado s_t. Quanto maior T, menor é a aleatoriedade da escolha de ações. É importante destacar que quando $T \to \infty$ ou $\epsilon = 1$, tanto a estratégia *softmax* quanto a *ϵ-greedy* são equivalentes a um algoritmo de passeio aleatório.

Q-learning e SARSA são algoritmos de aprendizado por reforço. A Figura 2.3 ilustra o algoritmo utilizado tanto pelo SARSA quanto *Q-learning*. A Tabela Q é inicializada uma vez, com valores aleatórios ou todos iguais, a recompensa pela ação anterior é processada, a Tabela Q é atualizada pela função de atualização, uma nova ação é escolhida e o processo se repete. A repetição deste ciclo se repete até que o agente atinja o desempenho desejado ou continua com o processo de aprendizado para constantemente se adaptar a um ambiente dinâmico.

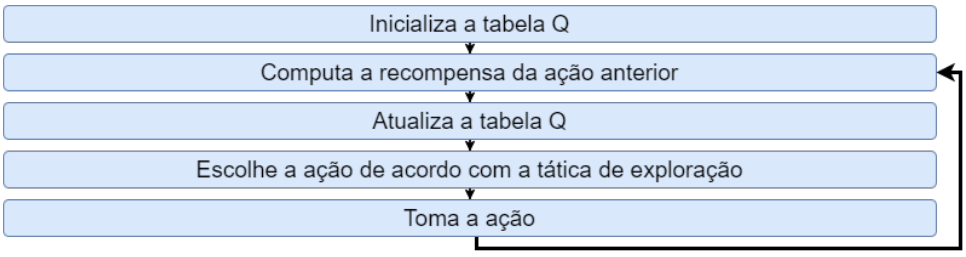


Figura 2.3: Esquema do funcionamento de um sistema de aprendizado por reforço.

2.3.1 *Q-learning*

O *Q-learning* é um algoritmo fora da política, que faz previsões com base em estimativas anteriores. O algoritmo armazena o conhecimento em uma tabela, conhecida como Tabela Q, atualizada pela função $Q(s, a)$. Nessa função, s é um estado, tal que $s \in \mathcal{S}$, e a é uma ação, tal que $a \in \mathcal{A}$. A função é definida pela Equação (2.2), com $s = s_t$ e $a = a_t$. O número de colunas na Tabela Q é igual ao número de ações, enquanto o número de linhas é igual ao número de estados. Para tomar decisões, um agente usa a Tabela Q, que é atualizada a cada iteração de acordo com a Equação 2.2.

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot \max_a Q(s_{t+1}, a)], \tag{2.2}$$

em que r é a recompensa. α é a taxa de aprendizado, e γ é o fator de desconto. Assim, ao conhecer o estado atual s_t, o agente pesquisa em sua tabela a linha correspondente a esse estado e toma uma decisão com base em um algoritmo de exploração.

2.3.2 SARSA

O SARSA (*State-Action-Reward-State-Action*) é um algoritmo de diferença temporal livre de modelo, que apresenta muitas semelhanças com o *Q-learning* exceto pela ausência da operação não linear *max* em sua função de atualização. A ausência desse fator o força a utilizar sempre a mesma ação tomada na iteração para atualizar a política de destino, tornando o SARSA um algoritmo dentro da política.

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot [r + \gamma \cdot Q(s_{t+1}, a_{t+1})]. \tag{2.3}$$

2.4 Lógica Difusa

A lógica difusa permite modelar o conhecimento humano, convertendo-o e armazenando-o na forma de regras. A consulta a essas regras, por sua vez, permite concluir qual ação é a mais indicada em uma determinada situação [44]. Para que essa interpretação seja possível, é necessário traduzir um processo complexo em variáveis linguísticas. Variáveis linguísticas são variáveis cujos valores são palavras ou frases em uma linguagem natural. Por exemplo transformar o valor claro 80% na variável linguística "Alta utilização". Enquanto na lógica booleana os elementos pertencem diretamente a um grupo, na lógica difusa os elementos difusos têm um grau de pertinência associado ao conjunto. Dessa forma, um elemento x_i está associado a um conjunto da seguinte forma

$$\{x_i \in U \mid \mu(x_i) > 0\},$$

em que U é o universo dos elementos e $\mu(x_i)$ é a pertinência do i-ésimo elemento ao conjunto, variando no intervalo $[0, 1]$. As funções de pertinência definem o quanto um elemento pertence a um conjunto difuso. O Sistema de inferência difuso (FIS) é um sistema de controle que utiliza a lógica difusa.

As entradas de um FIS são chamadas de antecedentes e as saídas são chamadas de consequentes. As regras utilizadas para tomar decisões são representadas como: *se* (x_1 *é* D_1) *e/ou* (x_2 *é* D_2) ... *e/ou* (x_m *é* D_m) *então* (y_1 *é* E_n). Nesse caso, x é o vetor

de variáveis antecedentes, y é o vetor de variáveis consequentes e D e E são vetores de variáveis linguísticas. A Figura 2.4 ilustra as etapas existentes em um FIS. O processo de fuzzificação consiste na transição do valor numérico de entrada, também conhecido como valor claro (*crisp*), para uma variável difusa antecedente utilizando funções de pertinência. Uma vez formatados como variáveis linguísticas, os antecedentes são submetidos a uma avaliação segundo regras difusas. Do resultado das regras obtém-se os elementos dos conjuntos difusos. Esses elementos são transformados em valores claros em um processo conhecido como defuzzificação. Existem muitas estratégias conhecidas para o processo de defuzzificação, como escolha de um máximo aleatório, último dos máximos, primeiro dos máximos, centroide, dentre outros.

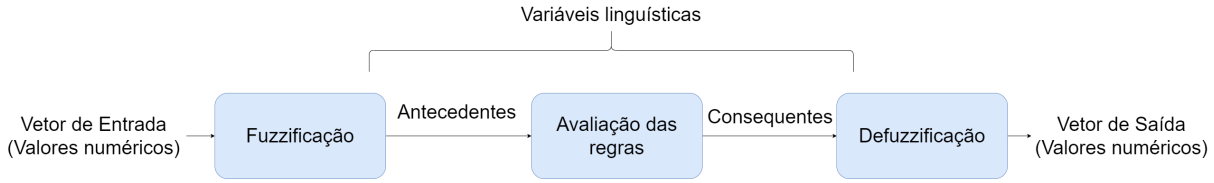


Figura 2.4: Esquema do funcionamento de um sistema de inferência difuso.

2.5 Aprendizado por reforço difuso

Ambos o Q-learning e o SARSA, algoritmos de aprendizado por reforço, podem levar a políticas ótimas. No entanto, esses algoritmos não apresentam escalabilidade quando as quantidades de estados e ações crescem. Além disso, os dois algoritmos não podem ser usados diretamente em ambientes contínuos [45], é necessário dividir o ambiente em um número finito de estados e ações. Para lidar com essas desvantagens, utiliza-se frequentemente o aprendizado por reforço difuso em diversas áreas de pesquisa [46, 47, 48, 49, 50]. Mesmo que com um conjunto de estados finito, as ações possíveis não são limitadas em aprendizado por reforço difuso por se tratarem de uma combinação linear das ações escolhidas. A principal motivação da existência do aprendizado por reforço difuso é que os sistemas de inferência difusos são aproximadores universais [51] e o conhecimento prévio pode ser embutido nas regras difusas, reduzindo o treinamento. O *Q-learning* difuso (FQL) [17] e o SARSA difuso (FSL) são exemplos de algoritmos de aprendizado por reforço difuso. Enquanto o FQL combina um sistema de inferência difuso (FIS) com *Q-learning*, o FSL combina FIS com SARSA. A Figura 2.5 resume o funcionamento do geral dos algoritmos de aprendizado por reforço difuso. Assim como em aprendizado por reforço, inicializa-se a Tabela Q, é computada a recompensa pela ação anterior e atualiza-se a

Tabela Q. Os estados, são mapeados em regras difusas e mais de uma regra difusa pode ser ativada ao mesmo tempo. As ações são escolhidas de acordo com a estratégia de exploração e a ação resultante é uma média das ações escolhidas, ponderada pela ativação das ações. A quantidade de estados da Tabela Q é, portanto, limitada pela quantidade de regras difusas e a quantidade de ações é limitada pelo número de funções de pertinência consequentes.

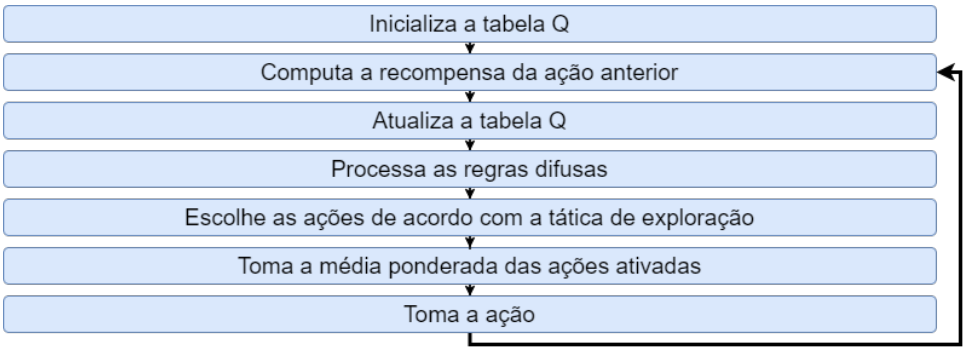


Figura 2.5: Esquema de funcionamento do aprendizado por reforço difuso.

2.5.1 *Q-learning* difuso

No aprendizado por reforço difuso, cada estado é mapeado para uma regra difusa, consequentemente mais de um estado pode ser ativado a cada iteração. A ação resultante é composta por uma combinação das ações escolhidas em cada linha ativada da Tabela Q, ponderadas com a função de ativação θ das regras difusas, conforme Equação (2.4).

$$a(s_{t+1}) = \frac{\sum_{i=1}^{N} a_i \theta_i(s_{t+1})}{\sum_{i=1}^{N} \theta_i(s_{t+1})} \tag{2.4}$$

Nem todas as regras são ativadas, mas para cada regra ativada há uma linha da Tabela Q correspondente, com ações a serem escolhidas pela estratégia de exploração escolhida. Uma vez escolhidas as ações a_i^*, é necessário atualizar a Tabela Q com as informações vindas do ambiente.

FQL, por ser baseado em *Q-learning*, também é um algoritmo fora da política. Diferentemente do *Q-learning* puro, o FQL não possui a limitação de retornar apenas valores discretos, a saída se estende a qualquer combinação linear entre as ações existentes. O processo de atualização da Tabela Q ocorre como segue. Há uma função V, também chamada de função valor, que determina o quão bom é um estado de acordo com a ação

tomada. A função V é recalculada de acordo com a Equação (2.8).

$$V(s_{t+1}) = \frac{\sum_{i=1}^{N} \theta_i(s_{t+1}).\max_a q[i,a]}{\sum_{i=1}^{N} \theta_i(s_{t+1})} \tag{2.5}$$

Calcula-se, então, o erro de diferença temporal ΔQ, obtido pela Equação (2.6).

$$\Delta Q_{FQL} = r + \gamma V(s_{t+1}) - Q(s_t, a_t) \tag{2.6}$$

Por fim, uma vez calculado o erro, o processo de atualização da Tabela Q para as ações escolhidas ativadas pelas regras é dado pela Equação (2.7).

$$q[i, a_i^*] \leftarrow q[i, a_i^*] + \alpha\theta_i(s_{t+1})\Delta Q \tag{2.7}$$

2.5.2 SARSA difuso

O processo de atualização da Tabela Q para o FSL é semelhante ao do FQL, exceto pelo cálculo do erro de variação temporal. Como SARSA é um algoritmo dentro da política, utiliza-se no cálculo cada ação escolhida a_i^* ativada por cada regra i, como mostra a Equação (2.8).

$$Q(s_{t+1}, a(s_{t+1})) = \frac{\sum_{i=1}^{N} q[i, a_i^*]\theta_i(s_{t+1})}{\sum_{i=1}^{N} \theta_i(s_{t+1})} \tag{2.8}$$

Consequentemente, o erro de diferença temporal para o SARSA difuso é dado pela Equação (2.9).

$$\Delta Q_{FSL} = r + \gamma Q(s_{t+1}, a(s_{t+1})) - Q(s_t, a_t) \tag{2.9}$$

2.6 Trabalhos Relacionados

Esta seção aborda os trabalhos relacionados, divididos em duas subseções. A primeira subseção apresenta abordagens para provisionamento de recursos em centro de

dados multi-inquilino, utilizando alocação de máquinas virtuais. A segunda subseção foca em propostas para provisionamento de largura de banda, utilizando técnicas como escalonamento de fluxos, lógica difusa e aprendizado por reforço.

2.6.1 Provisionamento de recursos através de alocação de máquinas virtuais

A alocação de máquinas virtuais é um processo que pode ser feito manualmente, por inquilinos e provedores. As aplicações utilizadas por diferentes inquilinos possuem requerimentos variados, que precisam ser atendidos simultaneamente. Alocar manualmente máquinas virtuais para os requerimentos para um inquilino incorre em alta probabilidade de conflito com requerimentos de outros inquilinos. Consequentemente há a busca por algoritmos para a otimização da alocação de máquinas virtuais de forma dinâmica para suprir a demanda dos inquilinos [52]. O aprendizado por reforço, a lógica difusa e o aprendizado por reforço difuso já foram propostos na literatura para o provisionamento de recursos computacionais em centros de dados multi-inquilino. Para realizar este provisionamento, os autores geralmente se baseiam nas ações de alocação de máquinas virtuais por ser mais simples para o provedor, variando estratégias, modelagens e objetivos do provisionamento.

Benifa e Dejey [53] propõem o *Reinforcement Learning-based Proactive Auto-Scaler* (RLPAS), uma estratégia de aprendizado por reforço multi-agente, que usa SARSA como algoritmo. As ações possíveis para os agentes são "alocar mais máquinas virtuais", "reduzir o número de máquinas virtuais" e "manter o cenário atual". Na modelagem dos estados para o algoritmo SARSA, o número de requisições de usuários, a taxa de atualização de máquinas virtuais, o tempo de resposta e a vazão para cada aplicação por um período específico são levados em consideração. A vazão e o tempo de resposta por requisição são tomados para o cálculo da recompensa, de modo que descumprimentos nos SLAs causam uma punição aos agentes. Os autores propõem o uso de política de aprendizado paralelo, na qual, além da atualização comum do algoritmo SARSA para as Tabelas Q, essas tabelas também são atualizadas durante a aquisição das ações. Wang *et al.* [31] também usam uma técnica de Aprendizado por Reforço Profundo (DRL) para provisionar recursos computacionais em nuvens IaaS, alocando instâncias de máquinas virtuais concentrando esforços em processamento visando diminuir os custos associados ao uso de recursos computacionais. São utilizadas ações de adicionar até duas instâncias e remover até duas instâncias, e os estados são obtidos da utilização de Unidade de Processamento

Central (CPU), quantidade de pacotes, latência e quantidade de requisições. Para as recompensas, define-se uma punição que aumenta de acordo com o uso de CPU, para estimular o agente a tomar medidas caso o uso seja exacerbado. Shaw *et al.* [54] focam na eficiência energética do centro de dados multi-inquilino. Os autores propõem o *Agente de Consolidação de Aprendizado por Reforço* (ARLCA) que usa o algoritmo SARSA para distribuir recursos para inquilinos. A quantidade de estados é uma porcentagem que relaciona o total do número de hospedeiros habilitados com a quantidade total de hospedeiros dentro do centro de dados. Para as ações, os autores optam por um conjunto composto no qual cada ação consiste alocação de capacidade de CPU para as máquinas virtuais dos inquilinos multiplicada pela utilização atual. A estratégia de exploração utilizada é o *softmax*. Um dos diferenciais do trabalho dos autores é a utilização de *Modelagem Baseada na Recompensa Potencial* (PBRS) [55], uma técnica de modelagem de recompensa que adiciona conhecimento prévio para aumentar a velocidade de convergência das políticas agentes de aprendizado por reforço. A abordagem de aprendizado por reforço de Rao *et al.* [56] leva em consideração os parâmetros de tempo de uso de CPU, a quantidade de CPUs virtuais alocadas e a quantidade de memória física alocada por máquina virtual. O espaço de ações é definido como a configuração de *hardware* para as máquinas virtuais que são visíveis dentro do domínio de um *driver*. As ações são definidas como a mudança na combinação de configurações para as máquinas virtuais. Para a função de recompensas, é avaliada a quantidade de largura de banda total em relação ao SLA, de forma que quanto maior for a largura de banda atingida, maior é a recompensa. A estratégia de exploração utilizada é a ϵ-*greedy*, por ser a mais comum. O uso de aprendizado por reforço implica em limitações no número de ações e estados pelo processamento e memória aumentar exponencialmente. Este trabalho utiliza aprendizado por reforço difuso, que possui a característica de assumir qualquer valor dentro de um intervalo definido e que converge mais rápido por fazer a atualização de mais de um elemento da Tabela Q por vez.

A solução de Mateen *et al.* [57] para alocação de máquinas virtuais tem como base a lógica difusa. Os autores têm como objetivo provisionar recursos de CPU em um centro de dados visando aumentar a utilização e consequentemente o lucro do provedor. Os autores utilizam lógica difusa dentro do laço (*loop*) de controle *Monitor, Analysis, Plan, and Execute* (MAPE), composto por quatro fases: monitoramento, análise, planejamento e execução. A lógica difusa é utilizada na etapa de planejamento, para determinar o coeficiente do controlador de CPU com o intuito de escolher a quantidade de núcleos para otimizar a utilização de CPU. Rajagopal *et al.* [58] propõem a utilização de conjuntos difusos suaves para realizar a migração de máquinas virtuais em centros de dados com

o objetivo de diminuir o consumo de energia. Os conjuntos difusos utilizados são o número de hospedeiros que se encontram sobrecarregados, qual recurso está sendo usado em excesso, agregando CPU, memória e a correlação entre recursos e as máquinas virtuais que estão alocadas dentro do hospedeiro sobrecarregado. Abordagens que utilizam lógica difusa podem ser tão eficientes quanto as regras escolhidas. Este trabalho utiliza aprendizado por reforço difuso, que é capaz de ajustar as próprias regras de acordo com as variações dinâmicas do ambiente pela integração com a Tabela Q, provendo adaptabilidade à lógica difusa.

Jamshidi *et al.* [48] propõem uma solução para alocação de recursos computacionais em nuvens usando uma abordagem de aprendizado por reforço difuso nomeada de FQL4KE. Posteriormente os autores estendem o FQL4KE [59] com o objetivo de integrar a solução aos componentes relevantes do *OpenStack*, como *Heat* para gerenciamento de recursos, o *Ceilometer* para serviços de telemetria utilizados para monitoramento e o *Keystone* para prover autenticação. As ações existentes na solução proposta pelos autores focam na adição e remoção de máquinas virtuais para prover recursos para os inquilinos. A proposta dos autores possui o aprendizado por reforço difuso comum com este trabalho, porém foca no provisionamento de largura de banda, uma vez que a alocação de máquinas virtuais não promove garantias para largura de banda.

Diferentemente dos outros autores, Son *et al.* [15] focam tanto em recursos computacionais quanto em recursos de rede. Os autores propõem uma combinação de alocação de máquina virtuais, em conjunto com um esquema de alocação de largura de banda usando um controlador SDN. O objetivo dos autores é minimizar o tempo de resposta para as requisições dos inquilinos. Comparada à solução em que apenas máquinas virtuais são alocadas, a proposta que usa tanto alocação de máquinas virtuais quanto de largura de banda alcança melhores resultados para o tempo de resposta às solicitações dos inquilinos. O mecanismo utilizado para priorização e distribuição de largura de banda usado é *Hierarchy Token Bucket* (HTB) e a proposta é implementada com o CloudSimSDN, um arcabouço para simulações de nuvens com suporte à SDN. Zhuh *et al.*[60] propõem o *Oktopus*, um mecanismo de posicionamento de máquinas virtuais baseado na demanda de largura de banda de cada inquilino, de forma a isolar o desempenho da rede de cada inquilino. Todas as máquinas virtuais são conectadas a um comutador com enlaces de largura de banda fixa, baseados nos SLAs dos inquilinos. Os autores utilizam a técnica de alocação de máquinas virtuais em conjunto com monitoramento de largura de banda para os inquilinos, mas não utilizam nenhuma técnica de adaptação automática a demanda de largura de banda dos inquilinos como é feito neste trabalho.

Este trabalho se diferencia dos demais apresentados nesta seção posto que se concentra exclusivamente no provisionamento de largura de banda de forma separada, sem levar em consideração outros recursos computacionais como uso de CPU e memória, uma vez que a largura de banda não é considerada nos demais e tem impacto direto nas aplicações dos inquilinos. Além disso, a ação escolhida para os trabalhos desta seção é de alocação de máquinas virtuais, enquanto a ação neste trabalho é a alocação de banda diretamente para os inquilinos.

2.6.2 Provisionamento de largura de banda

Diversos trabalhos na literatura propõem o uso de escalonadores de fluxos para o provisionamento de largura de banda. Este tipo de abordagem tem o objetivo de minimizar o tempo de encerramento dos fluxos, cada autor escolhendo uma métrica para decidir quais tipos de fluxos precisam ser tratados com prioridade e um método para a execução da priorização. Bai *et al.* [61, 62, 63] propõem um escalonador de fluxos agnóstico à informação chamado *Practical Information-Agnostic Flow Scheduling* (PIAS), que utiliza *Multilevel Feedback Queue* (MLFQ) para priorizar fluxos curtos em um algoritmo de *Least Attained Service* (LAS). Os autores utilizam DRL na evolução do trabalho chamada *Automatic Traffic Optimization* (AuTO) [64] para modificar os limiares das filas de fluxos dinamicamente. Tang *et al.* [65] propõem um escalonador de fluxos baseado em DRL para *Hybrid optical-electrical switching based data center network* (HOE-DCN) chamado *Flow Splitter*. Um único agente é utilizado no *Flow Splitter* para tomar a decisão de qual limiar define um fluxo longo, sabendo a origem e destino de um fluxo. De maneira mais detalhada, o estado para o agente é o par origem-destino e a ação é corrigir as definições do escalonador sobre o é um fluxo longo. Lei *et al.* [66] propõem o *Distributed Information-agnostic Flow Scheduling* (DIAS), onde informações sobre os fluxos são agregadas dentro dos próprios pacotes. No DIAS, as prioridades dos pacotes decrescem ao passo que o tempo de espera nos *buffers* de transmissão se eleva. Dessa forma, quanto maior é a idade de um fluxo, menor será a sua prioridade. Wang *et al.* [67] propõem o escalonador de fluxos *Aemon*, que se baseia na urgência dos fluxos para evitar congestionamentos através de um algoritmo de controle de congestionamento nomeado como *Urgency-based Congestion Control* (UCP). A urgência é definida pelos autores como a quantidade de tempo despendida na transmissão de um fluxo e a quantidade de tempo restante para o fluxo expirar. Diferente do proposto pelos autores supracitados, este trabalho possui o objetivo de maximizar o uso de largura de banda para o centro de dados e para tal utiliza um mecanismo de controle de admissão, o que não impede que posteriormente um mecanismo de escalonamento de

fluxos seja integrado para melhorar a eficiência das aplicações dos inquilinos.

Qu Fu *et al.* [68] utilizam a técnica de enfileiramento HTB com SDN para prover garantias de largura de banda em redes virtuais. Utilizando esta técnica os autores conseguem prover garantias de priorização para fluxos de tráfego com diferentes prioridades. Entretanto as métricas são inseridas de maneira estática, a solução não prevê comportamento dinâmico do tráfego na rede. Diferente de Qu Fu *et al.*, Mera-Gomez *et al.* [69] propõem uma abordagem baseada em *Q-learning* para gerenciar o uso da largura de banda. Os estados são modelados de acordo com a proporção de inquilinos com solicitações na fila e a priorização é realizada de acordo com a quantidade de tempo restante até o período de cobrança. Habib e Khan [70] adotam uma proposta de ação semelhante à de Mera-Gomez *et al.* Os autores focam no problema de alocação de recursos em nuvens, mas comparando *Q-learning* com outro algoritmo de aprendizagem por reforço, o SARSA. O trabalho conclui que *Q-learning* tem melhor desempenho em relação ao tempo de convergência. A otimização da utilização de largura de banda falha na utilização de HTB para ambientes dinâmicos, e os autores que utilizam essa técnica em conjunto aprendizado por reforço puro não garantem velocidade na adaptação do provisionamento de largura de banda. Este trabalho ao propor provisionamento de largura de banda usando aprendizado por reforço difuso, garante maior velocidade em relação a aprendizado por reforço puro.

Struhár *et al.* [71] propõem o *Dynamic Bandwidth Distribution* (DART), um arcabouço baseado em uma arquitetura virtualizada de SDN com múltiplos controladores para a distribuição da largura de banda em um sistema de *Internet das Coisas Industrial* (IIoT). O arcabouço se apoia em virtualização de redes, de modo que existe uma topologia física única e cada rede lógica possui o próprio controlador, responsável por gerenciar o tráfego nela. Os controladores se comunicam com o objetivo de manter o tráfego constante na rede física compartilhada, ao mesmo tempo que prioriza fluxos de tráfego. Guo *et al.* [72] propõem o SecondNet, um sistema de virtualização de centros de dados que utiliza um roteamento de origem baseado em comutação de portas. O sistema se aproveita do fato de a topologia do centro de dados ser conhecida, e implementa esta topologia utilizando *Multiprotocol Label Switching* (MPLS). Para tal, reserva-se largura de banda entre cada par de máquinas virtuais alocadas, com base em uma heurística para mapear especificações de centros de dados virtuais em centros de dados físicos, com foco nas demandas de recursos. Embora tenham a utilização de SDN em comum com este trabalho, os autores não incrementam adaptabilidade a diferentes cenários em suas propostas, algo que é realizado neste trabalho decorrente do aprendizado por reforço difuso.

Drummond *et al.* [73] propõem um modelo de particionamento de largura de banda multiobjetivo para redes autoajustáveis baseado em lógica difusa. Os autores investigam como aumentar a acurácia da alocação de largura de banda de maneira ótima, a fim de reduzir o desperdício de recursos, levando em consideração fatores como tamanho das amostras coletadas e a frequência com que são coletadas. Para tal, o custo de alocação de recursos de rede, a quantidade de largura de banda alocada e a quantidade de alocações feitas em cada iteração são levadas em consideração. Singla *et al.* [74] propõem o Proteus, um sistema de alocação de largura de banda dinâmico utilizando aglomerados computacionais (*clusters*) virtuais intercalados no tempo para modelar e explorar a variação no tempo das demandas de largura de banda pelos inquilinos e reduzir os custos para o provedor. A alocação é baseada na observação e reserva de largura de banda para tarefas delegadas para o centro de dados. Diferente do trabalho dos autores, este trabalho visa a maximização do lucro para o provedor pelo aumento da utilização pelos inquilinos e não pela redução do custo. Além disso, a utilização de lógica difusa é tão boa quanto a escolha das regras. O aprendizado por reforço difuso proposto neste trabalho modifica as regras a cada iteração, buscando encontrar as melhores regras para o ambiente a cada momento.

Lam *et alm.* [75] propõem o *NetShare*, um mecanismo de compartilhamento hierárquico ponderado no qual a largura de banda não utilizada por um serviço é atribuída a outros serviços. Simultaneamente, a solução cumpre as garantias de largura de banda total para cada inquilino, mas não garante a largura de banda por máquina virtual do inquilino. Popa *et al.* propõem o FairCloud [16], uma alocação de largura de banda baseada na quantidade de máquinas virtuais para cada inquilino, e não de acordo com o número de fluxos. Os autores buscam a proporcionalidade, tomando como base a topologia para prover garantias de largura de banda mínima para os inquilinos. Em seguida propõem o *ElasticSwitch* [76], que segue o princípio das garantias mínimas estabelecendo garantias de largura de banda entre pares de máquinas virtuais. A proposta também garante a conservação de trabalho, alocando a largura de banda não utilizada para outros enlaces através de limitadores de largura de banda dinâmicos, buscando a maior utilização possível. Diferente dos autores, a utilização de um sistema multi-agente garante a este trabalho que cada instância de cada inquilino seja gerenciada de forma independente, ágil pela utilização de lógica difusa e adaptativa pela integração com aprendizado por reforço.

Carvalho *et al.* [77] propõem um sistema de controle de SLAs para redes virtuais baseado em um controlador difuso. A proposta verifica os recursos consumidos por cada roteador virtual em um hipervisor de virtualização e estabelece uma função de recom-

pensa que pondera o uso de recursos e o SLA contratado. O controlador difuso calcula a recompensa e a punição. Jeyakumar *et al.* [78] propõem o *EyeQ*, um mecanismo que contém um conjunto de limitadores de tráfego na transmissão, para garantir os acordos de nível de serviço de largura de banda de egresso das máquinas virtuais de e detectores de congestionamento para a recepção, para evitar receptores congestionados. Caso uma máquina virtual exceda a largura de banda contratada, envia uma realimentação para o transmissor dos fluxos, de modo a reduzir o tráfego. Diferente do trabalho dos autores, este trabalho não busca apenas o cumprimento das SLAs dos inquilinos mas também a maximização do lucro do provedor visando a maior utilização possível. Além disso, este trabalho garante que cada instância do inquilino possua seus recursos garantidos uma vez que cada instância é gerenciada por um agente independente.

Rodrigues *et al.* [79] propõem o *GateKeeper*, um mecanismo de controle de admissão que limita a soma de garantias para a largura de banda disponível pelo enlace físico, tanto de ingresso quanto de egresso. O principal objetivo do trabalho é obter robustez, em relação à flexibilidade no nível do serviço, à escalabilidade e à resistência a inquilinos maliciosos. Paredes et al. [80] propõem um controle de admissão de largura de banda para uma rede em um campus, com o objetivo de priorizar usuários que visitam sítios educacionais. Assim, aloca-se mais largura de banda para os usuários que estão visitando mais sítios educacionais do que outros tipos de sítios, e diminui-se a largura de banda alocada para usuários que acessam outros sítios não relacionados à área acadêmica, como redes sociais e plataformas de *streaming* de vídeo. Siripongwutikorn *et al.* [81] propõem o uso de um Controle Adaptativo de Largura de Banda (ABC) utilizando lógica difusa com o objetivo de garantir os requisitos de perda de pacotes e consequentemente prover uma maior Qualidade de Serviço (QoS) aos usuários. Para as regras difusas, as funções de pertinência utilizadas são a *Média Móvel Exponencial Ponderada* (EWMA) do tamanho das filas de pacotes e a variação normalizada do tamanho das filas de pacotes. As ações do algoritmo de inferência difusa utilizado são o aumento ou diminuição da largura de banda alocada para cada usuário, de acordo com o tamanho das filas. Embora os autores também proponham um controle de admissão de largura de banda como este trabalho, a maioria implementa um controle com regras fixas, o que implica em uma adaptabilidade baixa se comparada com aprendizado reforço difuso.

Diferente dos trabalhos que propõem escalonadores de fluxo [66, 65, 61, 62, 63, 64, 67, 82] e trabalhos que propõem uma topologia adaptativa [72, 74] com o objetivo de cumprir os SLAs, este trabalho propõe um controle de admissão de largura de banda que objetiva a maximização do lucro do provedor ao mesmo tempo que cumprindo os SLAs. E diferente

dos trabalhos que também propõe um controle de admissão para o provisionamento de recursos de rede em centros de dados [79], este trabalho utiliza um controle de admissão adaptativo, se ajustando dinamicamente às mudanças no comportamento do tráfego no centro de dados.

Capítulo 3

O mecanismo proposto para provisionamento de largura de banda

A ideia-chave deste trabalho é desenvolver um mecanismo para provisionamento de largura de banda em nuvens multi-inquilino que permita aos inquilinos excederem a largura de banda contratada de maneira proporcional e justa. O mecanismo deve permitir priorização entre os inquilinos, mas o uso de largura de banda adicional não deve afetar o desempenho da rede para outros inquilinos. O uso do mecanismo proposto aumenta a utilização da nuvem, reduzindo a capacidade ociosa. Este capítulo descreve a proposta deste trabalho e a implementação do protótipo de controlador.

O mecanismo para provisionamento de largura de banda proposto neste trabalho é desenvolvido através da implementação de um protótipo de controlador multiagente modular que define políticas de alocação dinâmica de largura de banda para otimizar o uso dos recursos disponíveis da nuvem. A Figura 3.1 mostra os principais componentes do mecanismo proposto e como eles interagem entre si. Os agentes gerenciam a largura de banda individual de cada inquilino de acordo com a prioridade. A configuração leva em consideração os interesses dos inquilinos e a carga atual na infraestrutura em nuvem. A carga total da nuvem não pode ultrapassar um limiar predefinido. Esse limiar deve estar abaixo da capacidade total da nuvem para garantir o bom funcionamento do centro de dados. Utiliza-se uma abordagem que implementa múltiplos agentes, na qual um agente é alocado para cada inquilino. Essa abordagem permite a criação de um Sistema Multi-Agente (*Multiagent System* – MAS), sendo mais vantajosa do que uma abordagem de agente único, porque permite economizar recursos ao ativar ou desativar agentes na entrada e saída de inquilinos.

As informações do ambiente são recuperadas pelos agentes realizando requisições para

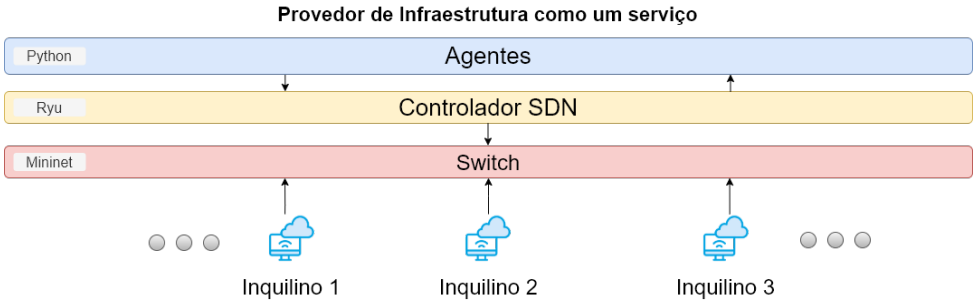


Figura 3.1: Esquema de funcionamento do mecanismo proposto, com os elementos que o compõe e as interações entre eles.

um controlador SDN. O controlador SDN é responsável por coletar informações da rede e limitar o tráfego. As informações disponíveis para o controlador SDN são coletadas diretamente de um elemento de interconexão, como um comutador, ao qual os inquilinos estejam conectados. As informações consistem na carga atual da nuvem, no ganho na taxa de transmissão de cada inquilino e na taxa de transmissão agregada de todos os inquilinos. Essas informações servem como entrada para os agentes, que determinam se o controlador deve aumentar ou diminuir a largura de banda de um inquilino específico. No sistema multiagente proposto neste trabalho, todos os agentes são iguais, exceto pela informação de entrada da largura de banda contratada por cada inquilino. Cada agente executa ações de maneira independente em relação aos demais e não há comunicação entre os agentes. No entanto, um agente pode optar por diminuir a largura de banda disponível para seu inquilino caso outro inquilino seja prejudicado. Vale ressaltar também que, tanto os agentes quanto o controlador são executados na infraestrutura do provedor e, portanto, ficam isolados das aplicações ou ações deliberadas praticadas pelos inquilinos. Assume-se nesse cenário que o controlador se comporta de maneira confiável e sempre executa corretamente. Destaca-se que o uso do controlador SDN para validar os resultados não limita a proposta, uma vez que pode ser substituída por qualquer entidade centralizadora capaz de receber informação de agentes e agir sobre os comutadores.

O mecanismo proposto é construído de forma que quando o tráfego ultrapasse o limiar permitido para a nuvem, a largura de banda alocada para cada inquilino é reduzida. Caso um inquilino solicite o uso de mais do que a largura de banda contratada se houver capacidade não utilizada na infraestrutura e se não afetar um inquilino mais prioritário, o mecanismo atende a solicitação. A prioridade é definida neste trabalho de acordo com a largura de banda contratada pelo inquilino.

O mecanismo proposto é modular, e permite que sejam utilizados diferentes algoritmos de decisão pelos agentes. Atualmente são implementados 5 algoritmos distintos para

provisionar a largura de banda em uma nuvem multi-inquilino: Sistema de inferência difuso [83], *Q-learning* [30, 84], SARSA, *Q-learning* difuso e SARSA difuso. Os parâmetros de cada algoritmo foram modelados para obter o melhor desempenho, de forma empírica, como descrito nas seções seguintes.

3.1 Sistema de inferência Difusa (*Fuzzy Inference System* - FIS)

A priorização do tráfego é feita por meio de regras difusas. Os três parâmetros levados em consideração pelos agentes, nesse algoritmo [83], estão na forma das seguintes variáveis linguísticas:

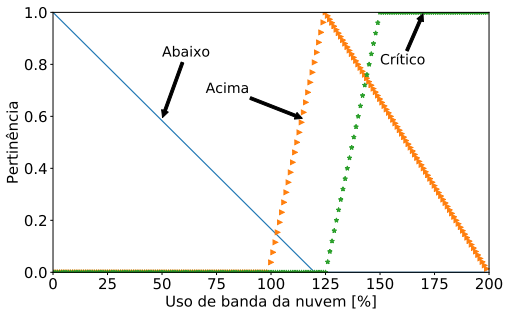
- **Uso da nuvem:** indica o total de tráfego direcionado para a nuvem somado por todos os inquilinos. Essa variável é utilizada porque um dos objetivos do mecanismo proposto é manter o uso total da nuvem o mais próximo possível de um limiar predefinido. Caso o uso total da nuvem esteja acima do limiar, reduz-se a alocação de largura de banda para os inquilinos. Caso contrário, aumenta-se a largura de banda alocada.

- **Existência de inquilinos abaixo do limiar contratado:** indica se há, ou não, inquilinos que estejam abaixo do limiar e qual é a quantidade de inquilinos que estão buscando utilizar a largura de banda contratada mas não conseguem porque os outros inquilinos estão consumindo toda a largura de banda disponível. Essa variável é usada porque um dos objetivos do mecanismo proposto é manter os SLAs de todos os inquilinos que estão utilizando a nuvem no momento.

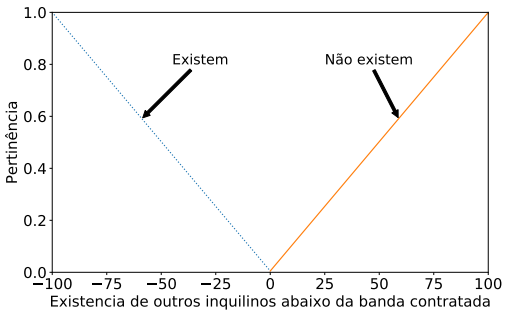
- **Uso pelo inquilino:** indica o total de banda em relação a contratada em uso pelo inquilino pelo qual o agente é responsável. Assim, essa variável é usada porque permite avaliar o quanto se deve aumentar ou diminuir a largura de banda, de acordo com a quantidade de recurso disponível.

Existe uma variável linguística consequente que é a decisão de quanto da largura de banda do inquilino deve ser suprimida ou liberada. O método de defuzzificação utilizado é o centroide por não possuir zonas proibidas [85], de forma que qualquer ação possa ser escolhida com qualquer intensidade presente nas funções de pertinência consequentes. Todas as funções de pertinência usadas para os antecedentes e os consequentes são

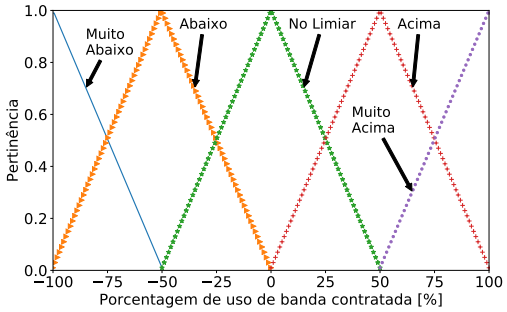
mostradas na Figura 3.2, modeladas por triângulos e trapézios, uma vez que atingem o mesmo desempenho que gaussianas [86].



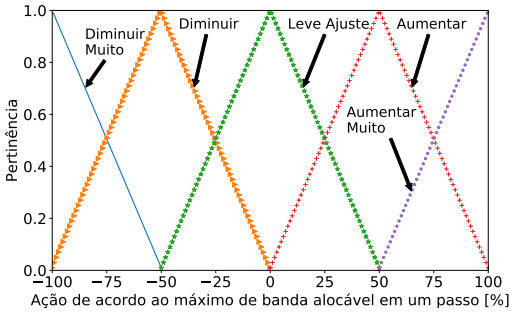
(a) Funções de pertinência antecedentes do uso da nuvem, onde 100% é o limiar da nuvem.



(b) Funções de pertinência antecedentes da existência de inquilinos abaixo do limiar contratado.



(c) Funções de pertinência antecedentes do uso pelo inquilino.



(d) Funções de pertinência consequentes das ações.

Figura 3.2: As funções de pertinência para o antecedente ou o consequente são compostas por triângulos e trapézios. Utiliza-se o centroide como algoritmo de defuzzificação para levar em consideração a associação gerada por cada regra e não possuir zonas proibidas.

O primeiro conjunto de funções de pertinência, mostrado na Figura 3.2(a), é relativo à variável linguística de uso da nuvem. Uma das funções de pertinência, indicada por "abaixo", é um triângulo que cobre mais da metade dos valores possíveis para o uso da nuvem, de forma que quanto menor for o uso da nuvem, maior deve ser o estímulo para aumentar a largura de banda fornecida aos inquilinos. As inclinações das outras duas funções, indicadas por "acima" e "crítico", são maiores comparadas à da função "abaixo" para que os agentes parem rapidamente de aumentar a largura de banda a fim de acentuar a diminuição do tráfego caso a carga na nuvem ultrapasse o limiar predefinido.

As funções de pertinência que contemplam a existência de inquilinos abaixo do limiar contratado são constituídas por dois triângulos, conforme Figura 3.2(b). A função rotulada como "existem" representa a porcentagem de inquilinos que estão buscando utilizar os recursos contratados, mas estão sendo impedidos por outros inquilinos. A função rotulada

como “não existem” representa o caso em que todos os inquilinos estão utilizando acima da banda contratada. As funções são construídas de forma que quanto mais inquilinos existirem com SLA em não conformidade, mais os agentes devem preferir escolher ações que diminuam a largura de banda provisionada ao seu inquilino. Os triângulos não se cruzam nesse caso porque as premissas são mutuamente exclusivas.

O último conjunto de funções de pertinência das variáveis linguísticas antecedentes é mostrado na Figura 3.2(d). Essas funções representam o quanto o inquilino está usando da largura de banda contratada. A função rotulada como “no limiar” é relativa ao inquilino estar na margem de largura de banda contratada, as duas funções “acima” e “muito acima” são referentes ao caso de o inquilino estar acima do contratado, e as duas funções “abaixo” e “muito abaixo” são para o caso de o inquilino estar abaixo do contratado.

O processamento das regras no mapeamento das entradas nas funções de pertinência das Figuras 3.2(a), 3.2(b) e 3.2(d) culmina no conjunto único de funções de pertinência consequentes representado na Figura 3.2. As funções indicam a ação tomada pelos agentes de acordo com as regras e o mapeamento das entradas. A função “leve ajuste” representa o caso em que não é necessária uma ação brusca pelo agente. Essa função provoca aumento leve, diminuição leve ou nenhuma ação sobre a regulação da largura de banda, dependendo do resultado das regras. Além da função “leve ajuste”, existem mais quatro funções que representam outras intensidades para as ações de diminuir ou aumentar a largura de banda para o inquilino, “diminuir muito”, “diminuir”, “aumentar”, “aumentar muito”. As funções que terminam com “muito” são um aumento ou diminuição com maior magnitude para os casos em que há inquilinos impedidos de utilizar a sua banda contratada ou quando o limiar da nuvem é excedido. A largura de banda alterada pelos agentes é proporcional ao SLA dos inquilinos, de forma a este mecanismo ser aplicável para qualquer valor de banda.

A Tabela 3.1 mostra um conjunto de 12 regras difusas escolhidas de forma empírica dentro da base de conhecimento difuso para produzir o consequente. As regras 1 e 2 têm como objetivo diminuir drasticamente a largura de banda do inquilino caso o uso da nuvem ultrapasse o limiar. As demais regras constituem todas as outras combinações possíveis das saídas das funções de pertinência antecedentes, controlando as ações executadas pelos agentes. Por exemplo, no caso de haver largura de banda ociosa na nuvem e se não existirem inquilinos de prioridade mais alta solicitando recursos, as regras definidas fazem com que os agentes tendam a escolher aumentar a largura de banda dos inquilinos para reduzir a ociosidade da nuvem.

Tabela 3.1: Conjunto de regras na base de dados difusa.

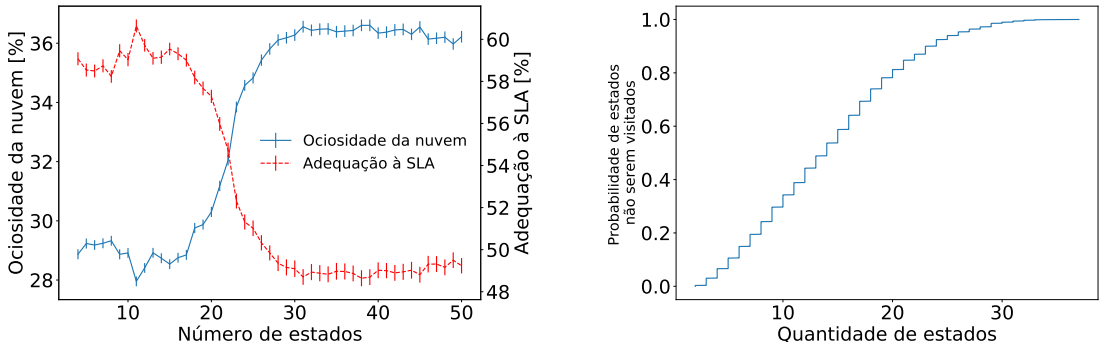
#	Uso da Nuvem	Uso pelo inquilino	Existência de inquilinos abaixo do limiar contratado	Ação
1	Crítico	Qualquer	Qualquer	Diminuir Muito
2	Acima	Qualquer	Qualquer	Diminuir
3	Abaixo	Muito Abaixo	Não Existem	Aumentar Muito
4	Abaixo	Muito Abaixo	Existem	Aumentar
5	Abaixo	Abaixo	Não Existem	Aumentar
6	Abaixo	Abaixo	Existem	Aumentar Muito
7	Abaixo	No Limiar	Não Existem	Aumentar
8	Abaixo	No Limiar	Existem	Leve Ajuste
9	Abaixo	Acima	Não Existem	Aumentar
10	Abaixo	Acima	Existem	Diminuir Muito
11	Abaixo	Muito Acima	Não Existem	Leve Ajuste
12	Abaixo	Muito Acima	Existem	Diminuir Muito

3.2 Q-learning e SARSA

Os algoritmos *Q-learning* e SARSA são algoritmos de diferença temporal para aprendizado por reforço (RL) e utilizam os conceitos de estados e ações. Neste trabalho, cada estado representa um intervalo $[L_i, L_f)$ referente ao uso da nuvem. Os valores L_i e L_f variam de um estado para o outro. Uma ação, por sua vez, é definida como a variação na largura de banda do inquilino de uma fração de seu valor contratado. Cada ação consiste no aumento ou diminuição de uma fração da largura de banda contratada pelo inquilino. Há a ação em que o agente não realiza qualquer variação no tráfego do inquilino. O número de ações e a quantidade de estados na Tabela Q são definidos da seguinte forma. Sabendo que a nuvem suporta uma determinada demanda de tráfego, define-se um limiar M que deve ser inferior à capacidade total da nuvem. Dessa forma, divide-se o valor M em frações iguais, dando origem a n_{fs} estados. Um estado adicional deve ser considerado para o caso em que o tráfego total é maior que M. Logo, o número total de estados é dado por $Q_s = 1 + n_{fs}$. Como o número de linhas na Tabela Q é igual ao número de estados, existem Q_s linhas. Quanto maior for a quantidade de linhas da Tabela Q, mais lentamente os agentes convergem para tomar decisões otimizadas [87].

Uma vez que a quantidade de estados é determinada, o espaço para ações potenciais deve ser caracterizado. Os agentes alteram a taxa de transmissão máxima para cada uma das portas do elemento de interconexão ao qual os inquilinos estão conectados. As ações são modeladas como a magnitude de tráfego que deve ser limitado ou liberado para a porta à qual o agente está atribuído. Para que o método seja utilizado considerando qualquer limite M para tráfego na nuvem, a quantidade de tráfego depende do número de M frações dependentes n_{fa}. Considerando que existe o mesmo número de ações tanto

para aumentar quanto para diminuir a largura de banda, para cada ação de aumento há uma ação de diminuição, então o número de ações seria $2n_{fa}$. Por exemplo, para uma ação de aumentar 20% da largura de banda, há também uma ação de diminuir 20% da largura de banda. Existe ainda o caso em que o agente não realiza nenhuma intervenção na largura de banda do inquilino. Portanto, número total de ações é $Q_a = 1 + 2n_{fa}$.



(a) Adequação dos inquilinos à largura de banda contratada e ociosidade da nuvem.

(b) Distribuição acumulada da probabilidade de estados não serem visitados.

Figura 3.3: Adequação do SLA nos algoritmos de RL de 4 estados até 50, com as ações fixadas em 5. Para cada número de estados o teste é feito 20 vezes. Segundo os testes realizados, 26 estados levam a um melhor cumprimento das SLAs em relação a quantidades menores de estados, e implica em uma convergência mais rápida devido à probabilidade de seus estados serem visitados. Além disso, a probabilidade de os 26 estados serem visitados está entre as maiores.

O tamanho final da Tabela Q em RL depende do número de estados e ações, sendo $Q_t = Q_a \times Q_s$. Portanto, o tamanho final, Q_t, dessa tabela varia de acordo com n_{fa} e n_{fs}, substituindo as expressões de Q_a e Q_s tem-se sendo $Q_t = 1 + 2n_{fa} + n_{fs} + 2n_{fs}n_{fa}$. Nesta dissertação, a escolha dos parâmetros n_{fa} e n_{fs} é feita de forma empírica através de um teste variando o número de estados de 4 a 50. Considerando $n_{fa} = 2$ para limitar o tamanho da Tabela Q e acelerar o tempo de convergência, os melhores resultados são quando $n_{fs} = 25$, conforme mostra a Figura 3.3. Para esse número de estados, os inquilinos possuem uma maior adequação à banda contratada, que é definida como a largura de banda utilizada dividido pela largura de banda contratada. Assim, menos recursos de rede ficam ociosos na nuvem e a probabilidade de todos os estados serem visitados é próxima de 100%. A existência de estados não visitados significa memória ocupada desnecessariamente e maior lentidão para a convergência da política para uma política ótima.

Ambos os algoritmos *Q-learning* e SARSA requerem a definição de uma função recompensa. A banda de todos os inquilinos é utilizada para calcular a recompensa. Nesse

cálculo utiliza-se o ganho da largura de banda que é calculado como a largura de banda do inquilino depois da ação do agente dividido pela largura de banda do inquilino antes da ação do agente. Além do ganho de largura de banda, é utilizado a largura de banda agregada de toda a nuvem, como é mostrado na Equação (3.1):

$$r_n(L, L_{c1}, L_{c2}, ..., L_{cn}, g'_n) = \begin{cases} -\frac{L}{M} & , \text{ se } L > M \\ g_n & , \text{ se } L \leq M \text{ e } g'_n \geq 1 \\ -g_n & , \text{ se } L \leq M \text{ e } g'_n < 1 \end{cases}, \tag{3.1}$$

em que L é a carga total que oferecida à nuvem, L_{cn} é a largura de banda atual do inquilino n, r_n é a função de recompensa dada ao agente do inquilino n, g_n é o ganho de largura de banda atual em relação à anterior para o inquilino n e g'_n é o ganho da iteração anterior para o inquilino n. O ganho é calculado dividindo a banda atual pela banda antes da ação tomada pelo agente. Caso o ganho de utilização gerado pela ultima ação do agente tenha sido maior que 1, ele recebe uma recompensa, caso este ganho seja menor que 1, ele recebe uma punição para que o mecanismo tenda a aumentar a largura de banda permitida para cada inquilino. O mecanismo proposto deve garantir a priorização dos inquilinos, de forma que solicitações de uso de banda adicional de inquilinos mais prioritários sejam atendidas mesmo se essa solicitação é feita quando esses inquilinos não estão utilizando integralmente a largura de banda contratada. Além disso, o mecanismo deve garantir o cumprimento aos SLAs. Consequentemente, a solicitação por largura de banda adicional feita pelo inquilino mais prioritário não é atendida quando houver outros inquilinos cujos SLAs não estiverem sendo cumpridos. Por exemplo, caso não existam recursos ociosos e outros inquilinos menos prioritários estejam usando uma largura de banda adicional, os agentes atribuídos aos inquilinos menos prioritários devem começar a executar ações que restrinjam a largura de banda oferecida. Ao mesmo tempo, o agente atribuído ao inquilino mais prioritário que solicitou a largura de banda adicional deve executar ações que aumentem a largura de banda disponibilizada para esse inquilino. No entanto, independente da prioridade do inquilino, sempre que o SLA do inquilino não estiver sendo cumprido, os agentes atribuídos a inquilinos que estão usando largura de banda adicional devem executar ações para reduzir a largura de banda provisionada.

A ação de reduzir a largura de banda provisionada quando há inquilinos com SLAs não cumpridos pode prejudicar o desempenho do algoritmo de aprendizado. Assim, é necessário adotar uma estratégia que evite atrasos na tratativa de casos onde o tráfego na nuvem está acima do limiar permitido. Neste trabalho, a solução utilizada foi o modo

de auto-coibição [84].

No modo de auto-coibição a taxa de aprendizado α segue a Equação (3.2):

$$\alpha(L) = \begin{cases} 0.5 & , \text{ se } L < M \\ 0.1 & , \text{ c.c} \end{cases}, \tag{3.2}$$

onde L é a carga na nuvem e M é o limiar da nuvem. A taxa de aprendizado é diminuída para evitar que as ações do modo de auto-coibição tenham pouca relevância na construção da política do agente, uma vez que neste modo os agentes só podem realizar ações de diminuição de tráfego para os inquilinos. A modelagem é feita dessa forma com o intuito de reduzir rapidamente o tráfego na nuvem para evitar que esteja superior ao limiar.

3.3 Q-learning difuso e SARSA difuso

Tabela 3.2: Conjunto de regras na base de dados de aprendizado por reforço difuso.

#	Uso da Nuvem	Uso pelo inquilino	Existência de inquilinos abaixo do limiar contratado	Ação
1	Crítico	Qualquer	Qualquer	$q_1(t)$
2	Acima	Qualquer	Qualquer	$q_2(t)$
3	Abaixo	Muito Abaixo	Não Existem	$q_3(t)$
4	Abaixo	Muito Abaixo	Existem	$q_4(t)$
5	Abaixo	Abaixo	Não Existem	$q_5(t)$
6	Abaixo	Abaixo	Existem	$q_6(t)$
7	Abaixo	No Limiar	Não Existem	$q_7(t)$
8	Abaixo	No Limiar	Existem	$q_8(t)$
9	Abaixo	Acima	Não Existem	$q_9(t)$
10	Abaixo	Acima	Existem	$_{10}(t)$
11	Abaixo	Muito Acima	Não Existem	$q_{11}(t)$
12	Abaixo	Muito Acima	Existem	$q_{12}(t)$

O aprendizado por reforço difuso adiciona ao mecanismo proposto as vantagens do aprendizado por reforço e da lógica difusa. As regras difusas utilizadas para os algoritmos de aprendizado por reforço difuso são as mesmas usadas na modelagem do FIS proposta neste trabalho (Tabela 3.1), com a diferença de que a ação escolhida é variante no tempo como mostrado na Tabela 3.2, onde t é um instante de tempo e q_n é a função que define a ação para a regra n no tempo t. A ação $q_n(t)$ escolhida depende dos valores da linha Tabela Q destinada a regra n e da estratégia de exploração. As funções de pertinência também são aproveitadas da modelagem proposta para o FIS (Figura 3.2). A diferença entre a modelagem proposta para o FIS e a proposta para o *Q-learning* difuso e o SARSA difuso se relaciona aos consequentes. No *Q-learning* difuso e no SARSA difuso os consequentes das

regras variam de acordo com as recompensas. Por sua vez, a modelagem das recompensas segue a Equação 3.1, a mesma função empregada na abordagem que usa os algoritmos *Q-learning* e SARSA (Seções 2.3.1 e 2.3.2). Devido ao uso do conjunto de regras com 12 regras, o tamanho da Tabela Q é limitado em 12 linhas por propriedade do aprendizado por reforço difuso.

A estratégia de exploração utilizada neste trabalho segue o algoritmo *softmax*. A escolha se deve ao fato de o *softmax* precisar de um único parâmetro T. O valor de T adotado neste trabalho é $T = 1$, pois está entre os valores que garantem as maiores recompensas acumuladas no problema clássico de aprendizado por reforço dos bandidos multi armados [88]. Além disso, há trabalhos que concluem que a estratégia *softmax* supera a estratégia $\epsilon - greedy$ no quesito de recompensa acumulada [42, 89].

Capítulo 4

Ambiente de Emulação para Validação e Avaliação

O mecanismo proposto é validado e avaliado através de emulação, utilizando tráfego constante e um conjunto de dados que contém informações de tráfego reais. A validação é feita inicialmente utilizando tráfego constante, para verificar o funcionamento do mecanismo. Em seguida, o mecanismo é avaliado utilizando um conjunto de dados que contém informação de tráfego real. Este capítulo descreve o conjunto de dados utilizado, bem como o ambiente emulado, incluindo as características desse ambiente e as ferramentas utilizadas.

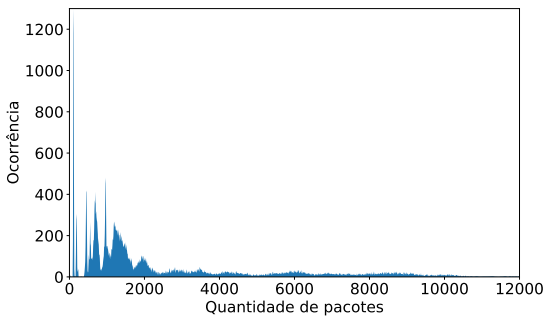
4.1 O conjunto de dados NetForager

A avaliação do mecanismo proposto neste artigo é feita em ambiente emulado. A informação de entrada fornecida ao ambiente para representar o tráfego na rede é obtida de um conjunto de dados, disponível publicamente[1], composto pelo tráfego proveniente de 15 aplicações. O tráfego é coletado de vários locais distribuídos geograficamente a cada hora durante 2 meses, sendo os pontos de monitoramento para coleta localizados na Universidade do Havaí, na Universidade de Wisconsin, na Universidade do Tennessee, na *Corporation For Education Network Initiatives In California* (CENIC) e na Universidade Politécnica de Nova Iorque (NYU) em Chattanooga. O tráfego é coletado usando o arcabouço de coleta de dados NetForager [90], que coleta dados de aplicações para prever o seu desempenho para diferentes cargas de trabalho inseridas. Primeiramente são coletadas as informações de tráfego de cada aplicação, depois as informações são analisadas de

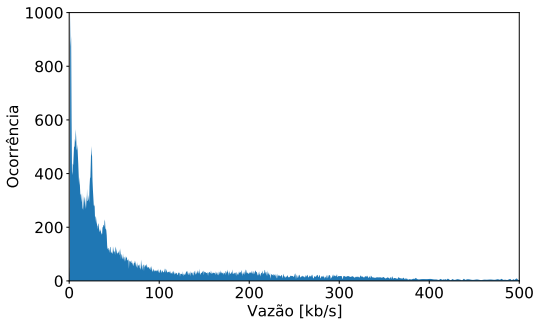
[1] https://dataverse.tdl.org/dataset.xhtml?persistentId=doi:10.18738/T8/OPWBMN

forma a dar um peso para os trechos que mais se repetem. Usando este peso, é predito o desempenho das aplicações para uma dada carga de trabalho.

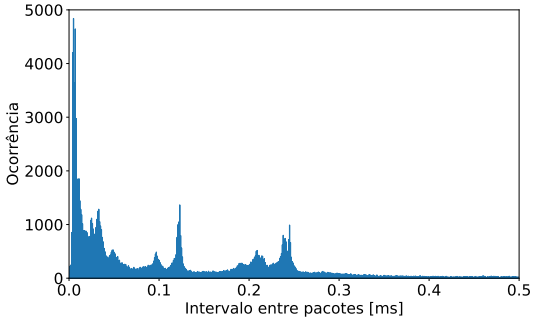
A conjunto de dados NetForager é escolhido para avaliar o mecanismo proposto devido à variedade de aplicações que utilizam *Hypertext protocol* (HTTP), que é um dos protocolos da Internet mais amplamente usados [91]. No conjunto de dados existem aproximadamente 177 milhões de pacotes, segregados em 57.197 arquivos que somam no total 30GB. Cada arquivo é referente a uma medição feita durante um período de 54 segundos em média. Cada arquivo possui em média 3.091 pacotes, com vazão média de 125 kb/s. A Figura 4.1(a) mostra a distribuição de pacotes por arquivo, a Figura 4.1(b) ilustra a distribuição da vazão por arquivo e a Figura 4.1(c) expõe a distribuição do intervalo entre a chegada de pacotes.



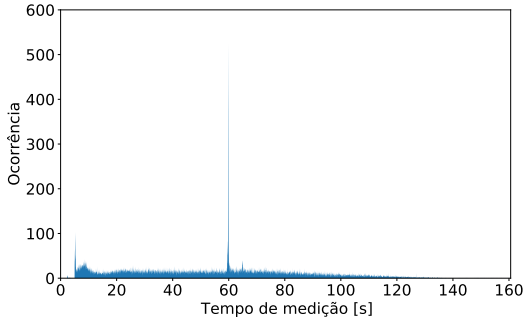
(a) Distribuição da quantidade de pacotes por arquivo em todo conjunto de dados



(b) Distribuição da vazão por arquivo em todo conjunto de dados



(c) Distribuição do intervalo entre pacotes



(d) Distribuição do tempo de medição por arquivo em todo conjunto de dados

Figura 4.1: Caracterização do conjunto de dados NetForager. a) Distribuição do atraso na chegada entre cada pacote para o tráfego de 100.000 pacotes retirados do conjunto de dados utilizado nessa dissertação. b) Distribuição da quantidade de pacotes por arquivo no conjunto de dados. c) Distribuição da vazão por arquivo no conjunto de dados. d) Distribuição do tempo de medição por arquivo no conjunto de dados.

Neste trabalho, cada inquilino gera um fluxo composto pela reprodução de 100.000 pacotes retirados do conjunto de dados, enviados em *loop*. Cada inquilino utiliza os

pacotes relativos a uma das 15 aplicações. As origens e destinos são modificados para se adequar à emulação, de forma que são apenas do inquilino para a nuvem. O fluxo é reproduzido através do *TCPReplay* [2], utilizando suas funções para aumentar a largura de banda até atingir a banda contratada pelo inquilino. O *TCPReplay* é um *software* desenvolvido para replicar o tráfego de arquivos de captura de pacotes. A Figura 4.1(c) mostra a distribuição do intervalo de chegada entre pacotes para o tráfego gerado pelo *TCPReplay*.

4.2 Ambiente de Emulação

Os cenários de avaliação e validação do mecanismo proposto são implementados em um ambiente emulado. A diferença entre eles é o tipo de tráfego utilizado e a quantidade de inquilinos conectados à nuvem. No cenário de *avaliação* existem 3 inquilinos e cada inquilino gera um tráfego de (*Constant Bit Rate* – CBR) UDP de 100 Mb/s, gerado utilizando a ferramenta *Iperf*. A execução do cenário de avaliação tem dois objetivos principais. O primeiro é analisar o funcionamento do mecanismo proposto utilizando FIS como estratégia de provisionamento, a fim de validar as regras difusas (Tabela 3.1). O segundo objetivo é verificar a possibilidade de utilização de algoritmos de aprendizado por reforço, como *Q-learning*, para provisionamento de largura de banda. Também se avalia o efeito provocado na eficiência do mecanismo ao variar o tempo de escalonamento entre ações. Os valores testados são 1, 2 e 5 segundos. Uma vez confirmado o correto funcionamento do mecanismo e o bom desempenho da proposta, a validação é feita em um cenário de maior escala. Assim, o cenário de *validação* possui 10 inquilinos que geram tráfego realístico reproduzido pelo *TCPReplay*, utilizando informações de tráfego contidas no conjunto de dados NetForager.

O ambiente emulado utilizado para os dois cenários conta com um controlador SDN, que se comunica com um comutador virtual para definir a largura de banda atribuída a cada porta do comutador. O controlador SDN utilizado é o *Ryu*[3], que se comunica através do protocolo OpenFlow 1.3. A existência de uma Interface de programação de aplicação (*Application Pogramming Interface* – API) `HTTP` *RESTful* no controlador *Ryu* permite o desenvolvimento de aplicações para gerenciamento e controle de aplicativos suportados por SDN. A emulação é criada com o auxílio do *Mininet* [92] no cenário de validação e do *Maxinet* [93] no cenário de avaliação. O *Maxinet* é uma versão distribuída

[2]https://github.com/appneta/tcpreplay
[3]https://ryu.readthedocs.io/en/latest/index.html

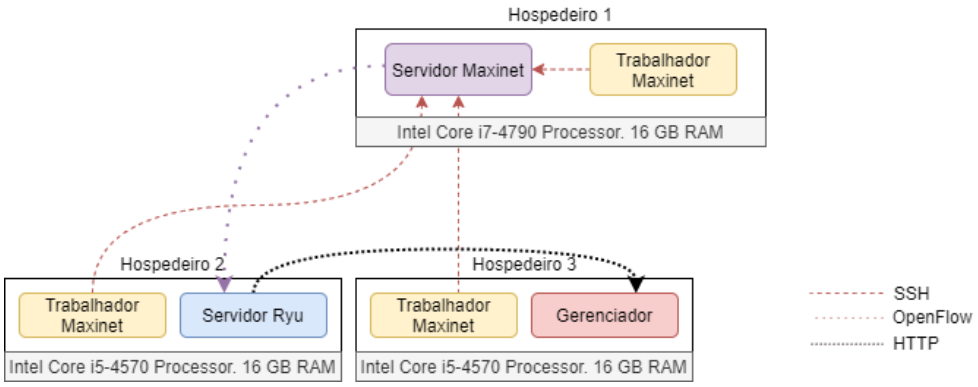


Figura 4.2: Esquema do relacionamento entre máquinas físicas e os respectivos módulos em execução, com a descrição do *hardware* de cada máquina.

do Mininet [92] que permite compartilhar recursos entre máquinas físicas distintas para usá-los em topologias mais complexas. A utilização do *Mininet* em conjunto com o Ryu facilita o desenvolvimento uma vez que ambos são escritos na linguagem Python [94]. A avaliação é feita utilizando 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. A validação conta com conjunto de três máquinas físicas, equipadas com 16 GB de RAM e diferentes capacidades de processamento. Os modelos dos processadores das máquinas podem ser vistos na Figura 4.2. As máquinas executam o sistema operacional Ubuntu 18.06 e cada uma hospeda um trabalhador (*worker*) Maxinet e um módulo adicional. A Figura 4.2 mostra um esquema que relaciona uma maquina física com os respectivos módulos em execução, bem como as especificações de *hardware*.

Na Figura 4.2, o Hospedeiro 1 contém o servidor *Maxinet*, a configuração da topologia e um trabalhador *Maxinet* que representa a nuvem. Esses módulos são alocados nessa máquina devido aos recursos de processamento superiores. Os outros módulos são distribuídos aleatoriamente entre as máquinas restantes, uma vez que possuem a mesma especificação de *hardware*. Dessa forma, além de hospedarem um trabalhador *Maxinet* que representa um inquilino, o Hospedeiro 2 hospeda o servidor do controlador *Ryu*, e o Hospedeiro 3 hospeda um *software* construído em Python 3.6 que gerencia o cenário. Neste cenário, um inquilino é representado por um nó Mininet 2.3.0d6 reproduzindo o tráfego do conjunto de dados NetForager [90]. O gerenciador aloca agentes para os inquilinos e armazena suas bases de dados. O controlador é manipulado pelos agentes de dentro do gerenciador, com os agentes de cada inquilino utilizando a API. A comunicação entre os agentes e o controlador SDN ocorre por meio de solicitações Transferência de Estado Representacional (REST) `HTTP` com estrutura `JSON`.

Capítulo 5

Resultados e Discussão

Este capítulo apresenta os resultados obtidos, divididos em um cenário de validação e outro de avaliação. No cenário de validação são tomados resultados preliminares utilizando uma configuração com menos inquilinos e tráfego com Taxa de Bit Constante (CBR) para verificar a viabilidade do mecanismo. Já na avaliação utiliza-se o conjunto de dados NetForager [90] para geração do tráfego de 10 inquilinos. Dessa forma, verifica-se a eficiência do mecanismo em um cenário de maior escala, comparando os algoritmos *Q-learning*, SARSA, FIS, FSL e FQL para verificar qual possui uma maior eficiência.

5.1 Validação

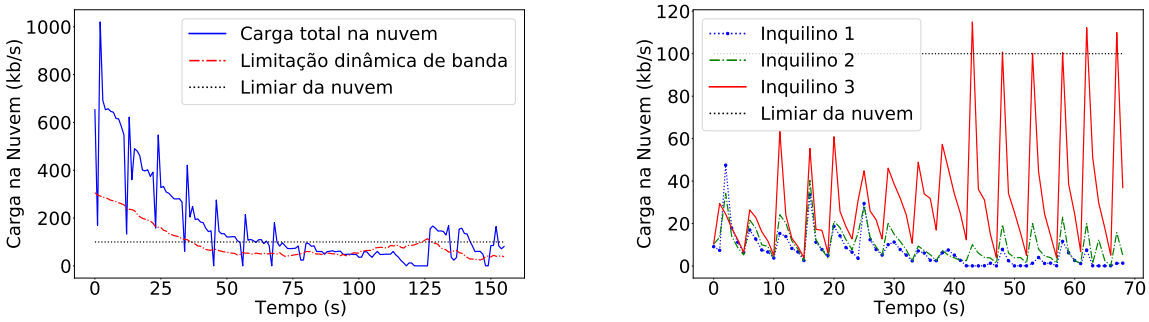
Na validação preliminar, investiga-se a viabilidade do uso de técnicas de aprendizado por reforço e de lógica difusa para implementação do mecanismo proposto. Para tanto, valida-se a eficiência do mecanismo quanto à capacidade de controlar o uso de largura de banda, e seu consumo de recursos computacionais durante a execução. Valida-se, ainda, a viabilidade de usar métricas indiretas, como o uso de capacidade de processamento em cada inquilino, para controlar o uso de largura de banda.

5.1.1 Abordagem baseada em aprendizado por reforço

A primeira implementação do mecanismo proposta utiliza apenas *Q-learning* para gerenciar o uso da largura de banda. Os resultados são obtidos para um cenário com 3 inquilinos e tráfego UDP CBR. A Figura 5.1(a) mostra a medição da carga máxima na nuvem em kb/s no decorrer de um período de 150 segundos. Esse resultado é obtido utilizando o algoritmo *Q-learning* com Tabela Q com valores zerados no tempo zero. O

objetivo é verificar a capacidade do modo de auto-coibição [84] de reduzir a carga na nuvem caso esteja maior que o limiar permitido. Nesse cenário, o limiar é de 100 kb/s. Com a Tabela Q zerada, no tempo zero a carga na nuvem é de 1.000 kb/s, que é 10 vezes o limiar de 100 kb/s. Os 3 agentes atribuídos aos 3 inquilinos são capazes de direcionar o tráfego agregado na nuvem para o limiar após cerca de 50 iterações, mantendo o tráfego próximo do limiar nas ações subsequentes. Isso é possível graças ao comportamento do modo de auto-coibição que só permite ações de limitação de largura de banda, uma vez que a carga na nuvem está acima do permitido, e deixa de agir quando há uma estabilização.

A Figura 5.1(b) mostra a capacidade de priorização do mecanismo proposto, para cada inquilino. O resultado é obtido utilizando *Q-learning* com Tabela Q com valores zerados no tempo zero. A limitação dinâmica de banda é a média das limitações feitas pelos agentes na banda de seus inquilinos em cada instante. O Inquilino 3 possui uma prioridade maior, consequentemente os agentes de outros inquilinos ativam o modo de auto-coibição caso o Inquilino 3 não consiga utilizar a largura de banda contratada. Embora possibilite a priorização de forma rápida, o modo de auto-coibição causa um efeito de dente-de-serra no tráfego dos inquilinos devido a variações bruscas nos limites de largura de banda impostos causando imprevisibilidade. Contudo, esse efeito é amenizado conforme os agentes ajustam suas políticas.



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

(b) Priorização de inquilinos quando qualquer inquilino é impedido de usar a banda contratada.

Figura 5.1: O mecanismo proposto a) leva à convergência da carga de tráfego recebida. Quando a carga é maior que o limiar, a convergência é rápida devido ao modo de auto-coibição que permite apenas ações de limitação de largura de banda. b) Quando os inquilinos demandam o uso da largura de banda contratada, os agentes entram em modo de auto-coibição, coordenando a limitação de largura de banda dos inquilinos, a fim de garantir a priorização adequada entre os inquilinos.

O mecanismo proposto deve ser capaz de se adaptar à entrada e saída de inquilinos, pois esse comportamento é uma realidade nos centros de dados. A Figura 5.2 mostra o

comportamento do mecanismo quando o Inquilino 3 entra e sai da nuvem. No momento da entrada do Inquilino 3, os agentes entram em modo de auto-coibição para permitir que o novo inquilino utilize a largura de banda contratada. Quando o Inquilino 3 deixa de gerar tráfego para a nuvem, os agentes reajustam o tráfego agregado para que fique mais próximo ao limiar. Mais de 300 iterações foram tomadas posteriores a entrada do Inquilino 3, tempo suficiente para os agentes convergirem as suas políticas, possibilitando ajustes mais ágeis e assertivos na carga total da nuvem, de modo a respeitar a priorização.

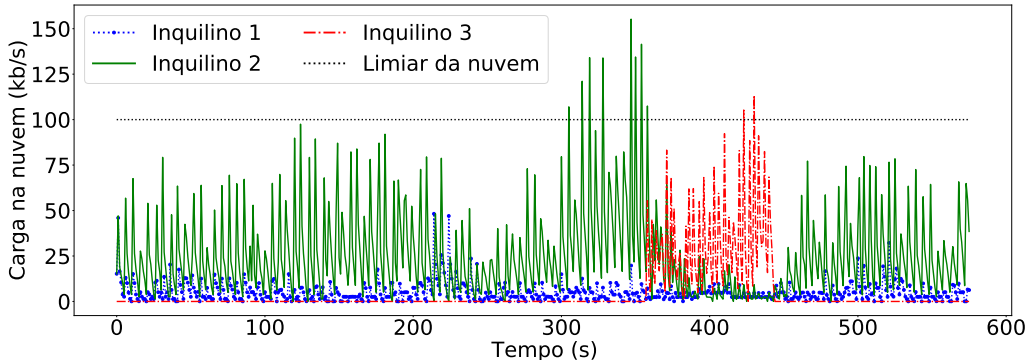


Figura 5.2: O mecanismo proposto é capaz de se adequar à entrada de um inquilino prioritário que não estava usando a largura de banda contratada, priorizando esse inquilino. Quando o inquilino deixa de usar a largura de banda, os agentes dos outros inquilinos se adaptam ao novo cenário.

5.1.2 Abordagem baseada em métricas indiretas

Em casos nos quais não seja possível calcular o tráfego agregado na nuvem, é necessário utilizar outras métricas para gerenciar a largura de banda dos inquilinos. Por exemplo, centros de dados que fazem uso de dispositivos legados, sem acesso a tecnologias como SDN ou outras tecnologias para fazer medidas de largura de banda. Nesses contextos, uma das possibilidades é verificar a porcentagem de processamento total utilizado pela nuvem e mapear essa porcentagem em um valor de largura de banda. Neste trabalho, para cada faixa percentual de processamento, o mapeamento entre processamento e largura de banda é feito através da Equação 5.1:

$$B_{t+1}(CPU^{a,b}) = (1-\phi)B_t(CPU^{a,b}) + (\phi)B_{real}(t) \tag{5.1}$$

em que $B(CPU^{a,b})$ é o valor de largura de banda quando a utilização de CPU está entre $a\%$ e $b\%$, $B_{real}(t)$ é o valor real de largura de banda no tempo t e ϕ é uma taxa de atualização. Neste trabalho, o caso em que é possível medir a carga na nuvem diretamente é chamado

de *observável*, e o caso no qual é feito o mapeamento entre processamento e largura de banda é chamado de *parcialmente observável*. Existe, ainda, um caso *híbrido*, no qual se utiliza a medição do uso de largura de banda e o mapeamento proposto, de forma alternada.

O objetivo dos resultados dessa seção é confirmar que é possível utilizar o mecanismo mapeando processamento com largura de banda. A Figura 5.3 mostra os resultados da emulação realizada para os casos *observável*, *parcialmente observável* e *híbrido*, com e sem a utilização do modo de auto-coibição. A emulação é repetida 20 vezes para os casos *observável* e *parcialmente observável*, com e sem o modo de auto-coibição. Esse resultado tem como objetivos (i) mostrar a eficácia do modo de auto-coibição em realizar a manutenção do tráfego agregado abaixo do limiar, (ii) mostrar que é possível utilizar o mecanismo proposto utilizando métricas indiretas, como o consumo de CPU através do mapeamento entre esse consumo e o uso de largura de banda. Observa-se na Figura 5.3 que, em todos os casos em que o modo de auto-coibição é utilizado, os agentes são capazes de manter de maneira mais eficiente a carga na nuvem abaixo do limiar. Há pouca diferença entre os resultados obtidos para os casos observável e parcialmente observável. Isso mostra que é possível utilizar o mecanismo sem medir diretamente o uso de largura de banda, como no caso parcialmente observável.

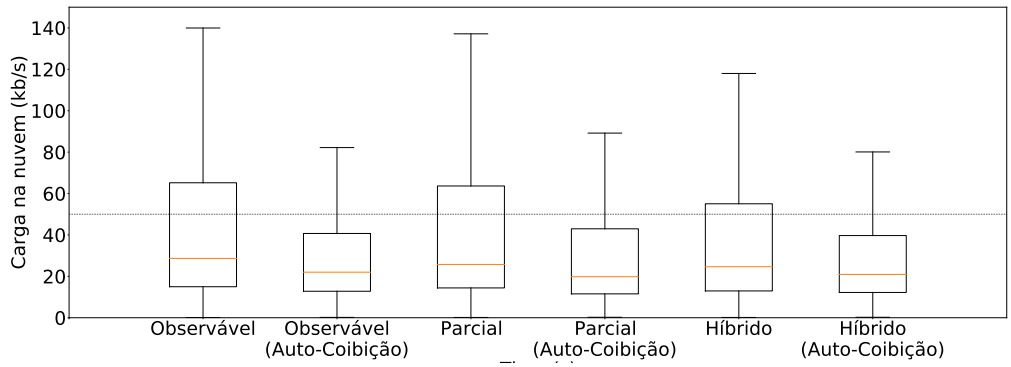


Figura 5.3: Resultados da emulação para os casos *observável*, *parcialmente observável* e *híbrido*, com e sem o modo de auto-coibição. É possível utilizar o mecanismo para o caso *parcialmente observável*, porque não há grandes diferenças na mediana e na posição dos quartis entre os *boxplots* dos dois casos. Casos em que o modo de auto-coibição é utilizado apresentam uma maior eficiência em manter o tráfego abaixo do limiar da nuvem.

5.1.3 Abordagem baseada em lógica difusa

Ainda no cenário com 3 inquilinos conectados à nuvem e gerando tráfego UDP CBR, avalia-se o desempenho do mecanismo proposto utilizando agentes baseados em lógica di-

fusa. O desempenho é comparado à implementação da proposta que usa agentes baseados em aprendizado por reforço. A Figura 5.4 mostra a comparação das duas abordagens quanto ao uso de largura de banda. A emulação é repetida 20 vezes e o tráfego CBR gerado é de 100 Mb/s para cada inquilino. Nessa emulação, a abordagem baseada em aprendizado por reforço utiliza o algoritmo *Q-learning* com uma Tabela Q que possui valores inicialmente zerados. Por sua vez, a abordagem baseada em lógica difusa implementa o sistema de inferência difuso (FIS) descrito na Seção 2.4, que possui um conjunto de 12 regras difusas. Os resultados mostram que, no cenário avaliado, ambas as abordagens, *Q-learning* e FIS, são capazes de priorizar a largura de banda entre os inquilinos, uma vez que a mediana é muito próxima da largura de banda contratada. No entanto, a abordagem baseada em FIS regula a largura de banda mais rapidamente do que a baseada em *Q-learning*, considerando que o número de ações realizadas pelos agentes em ambas as abordagens é igual a 100. Isso ocorre porque a abordagem FIS não requer um tempo de aprendizado para determinar as políticas ótimas. A maior rapidez da abordagem FIS é percebida no gráfico comparando a altura da mediana das caixas. Observa-se que as medianas de largura de banda provisionada pelos agentes acionados pelo FIS estão mais próximas do SLA de cada inquilino, com uma distância interquartil menor, enquanto os agentes acionados pelo *Q-learning* não atingem o SLA e possuem uma distância interquartil maior. Assim, o resultado mostra que o algoritmo FIS prioriza inquilinos e atende aos SLAs, uma vez que os inquilinos usam maior largura de banda do que a contratada durante 50% do tempo, enquanto a carga total da nuvem é mantida dentro do limiar permitido.

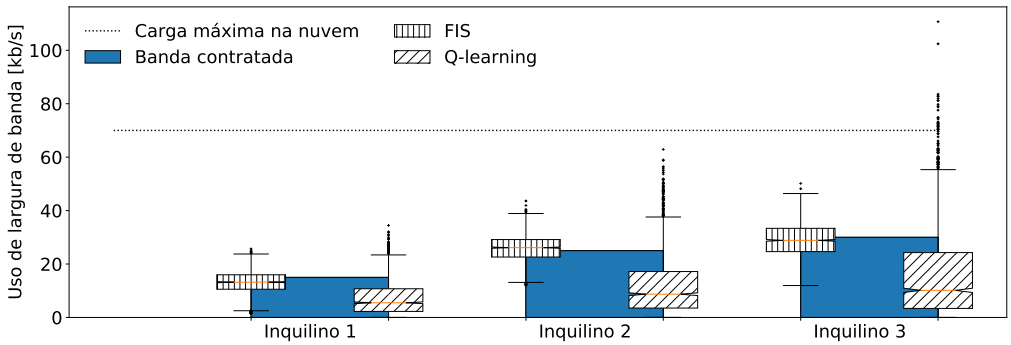
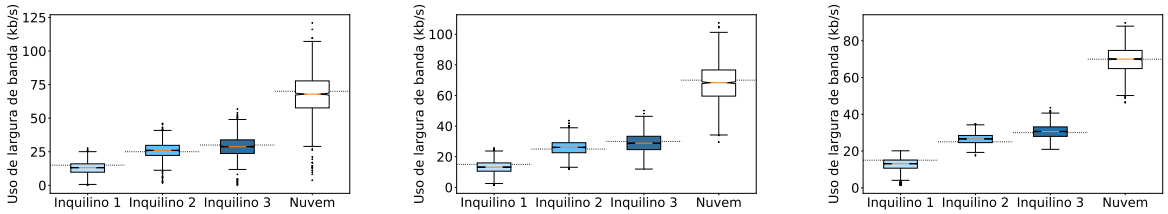


Figura 5.4: Comparação entre a prioridade e a capacidade de estabilidade do mecanismo proposto usando FIS e *Q-learning*. Ambos os algoritmos conseguem manter a prioridade entre os inquilinos, porém, o FIS consegue um resultado mais expressivo, pois a largura de banda contratada de cada inquilino é ultrapassada em cerca de 50% do tempo, enquanto a carga total da nuvem está sob controle.

Varia-se o tempo entre as ações realizadas pelos agentes para avaliar o quanto a vari-

ação de tempo entre ações influencia sobre a eficiência em manter os SLAs. A Figura 5.5 mostra resultados do cumprimento do SLA para intervalos de decisão entre os agentes de 1, 2 e 5 segundos. Os resultados são obtidos para o mesmo cenário anterior, que conta com 3 agentes provisionando a largura de banda de 3 inquilinos gerando tráfego UDP CBR de 100 Mb/s. A emulação é executada 20 vezes e em cada execução o controlador atua 100 vezes regulando a largura de banda provisionada aos inquilinos. As linhas pontilhadas representam a largura de banda contratada de cada inquilino. A linha pontilhada marca o limite máximo de carga da nuvem. Observa-se que quanto maior é o intervalo de tempo para os agentes consultarem o comutador, menor é a distância interquartil. Além disso, para 1, 2 e 5 segundos de intervalo, tem-se respectivamente a mediana a 95,80%, 96,22% e 98,56% da banda contratada por inquilino na média e a mediana de carga na nuvem a 97,20%, 97,61% e 100,05% do limiar da nuvem. Estes valores permitem concluir que aumentar o intervalo de tempo entre as consultas faz com que os agentes tendam a executar ações mais assertivas, diminuindo a ociosidade da nuvem e cumprindo os SLAs. Entretanto, aumentar o intervalo de tempo pode fazer com que o controlador perca possíveis picos entre consultas consecutivas, levando os agentes a permitirem que o limiar da nuvem seja ultrapassado, prejudicando o provisionamento de largura de banda.



(a) O controlador consulta o comutador a cada 1 segundo.

(b) O controlador consulta o comutador a cada 2 segundos.

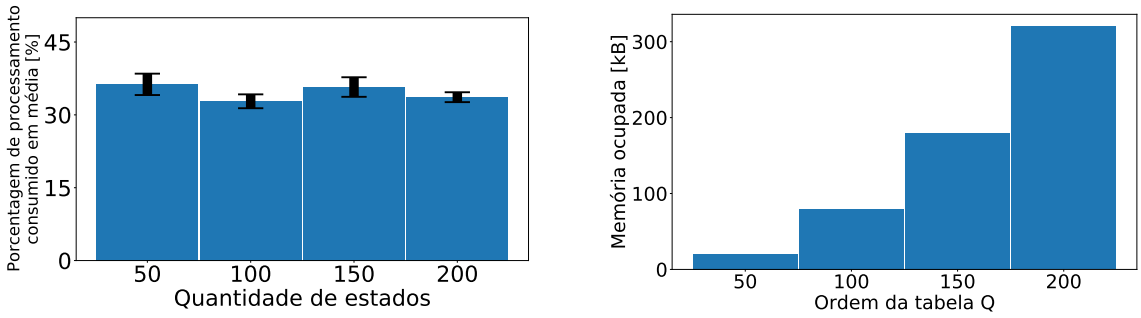
(c) O controlador consulta o comutador a cada 5 segundos.

Figura 5.5: O cumprimento do SLA de cada inquilino é avaliado variando o intervalo de tempo do agente para consultar o comutador sobre as condições da rede. As linhas pontilhadas são as larguras de banda contratadas de cada inquilino. O controlador consulta o comutador a) a cada segundo, b) a cada 2 segundos e c) a cada 5 segundos. Aumentar o intervalo de tempo entre as consultas tende a diminuir a ociosidade da nuvem, mas pode ocultar picos entre as consultas.

5.1.4 Consumo de recursos computacionais pelo mecanismo proposto

A escalabilidade é um dos principais requisitos para a computação em nuvem. Portanto, é necessário conhecer a capacidade do mecanismo de escalar, verificando como o consumo de recursos evolui de acordo com o número de estados e ações. A Figura 5.6

mostra a variação do consumo de memória e de processamento em função do tamanho da Tabela Q, para um cenário com 3 inquilinos gerando tráfego de 100kb/s. Considerando uma Tabela Q de k ações por k estados, o aumento de k implica um aumento exponencial no consumo de memória, conforme mostra a Figura 5.6(a). O processamento se mantém constante, em torno de 36% no cenário avaliado, independentemente do tamanho da Tabela Q. O fato de o processamento se manter constante, independente do tamanho da Tabela Q utilizada, permite modificar a configuração dos agentes, variando o número de estados e de ações sem restrições de desempenho de processamento. No entanto, ao modificar o número de estados e ações, ou seja, a ordem da Tabela Q, o mecanismo tem a praticabilidade prejudicada, perdendo a escalabilidade, porque cada novo inquilino requer a criação de uma Tabela Q.



(a) Consumo de processamento.

(b) Consumo de memória.

Figura 5.6: 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 k ações por k estados. A variação em k aumenta exponencialmente a memória consumida.

5.2 Avaliação

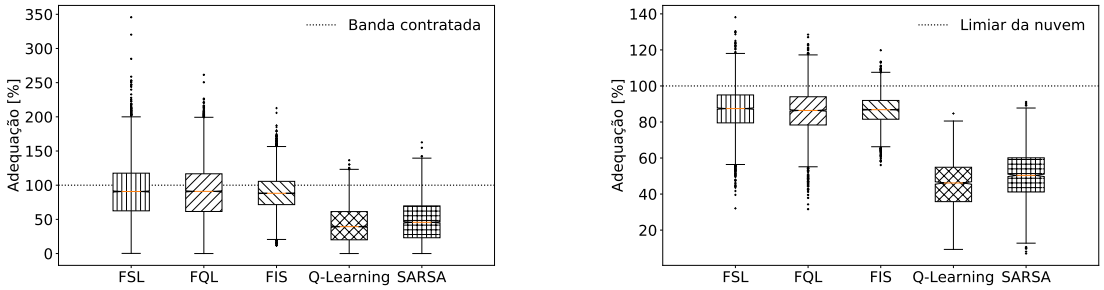
Verificada a viabilidade do mecanismo proposto tanto baseado em aprendizado por reforço como em lógica difusa, segue-se a avaliação do mecanismo em um cenário de maior escala. Nesse cenário, o mecanismo é modificado para acionar os agentes de acordo com 3 abordagens: (i) lógica difusa, (ii) aprendizado por reforço, e (iii) aprendizado por reforço difuso. Na primeira abordagem utiliza-se o FIS, conforme modelagem apresentada na Seção 3.1. A segunda abordagem é implementada de duas formas: a primeira usa *Q-learning* e a segunda usa SARSA, conforme a modelagem apresentada na Seção 3.2. Por fim, a terceira abordagem também é implementada de duas formas, usando FQL e FSL, conforme a modelagem apresentada na Seção 3.3. Assim, implementa-se o mecanismo

proposto utilizando 5 algoritmos distintos.

Inicialmente, a adequação dos inquilinos à banda contratada é verificada para cada implementação, a fim de analisar o cumprimento dos SLAs. Os resultados são obtidos a partir de 20 execuções do cenário emulado com 10 inquilinos, usando Maxinet e tráfego baseado no conjunto de dados NetForager replicado pelo TCPReplay. A adequação dos inquilinos à banda contratada é calculada dividindo a largura de banda usada pela largura de banda contratada. A Figura 5.7(a) mostra os resultados de adequação para cada implementação. Investiga-se, também, a adequação do mecanismo quanto à capacidade de manter a carga na nuvem abaixo do limite pré-estabelecido, conforme Figura 5.7(b). A adequação da carga da nuvem ao limiar pré-definido é calculada dividindo a largura de banda usada pelo limite da nuvem. A altura da mediana e o 95º percentil são os parâmetros utilizados para comparar a eficiência dos algoritmos. Quanto maior a mediana e o 95º percentil, maior a capacidade de aumentar o uso de largura de banda até o limiar definido. A análise conjunta das Figuras 5.7(a) e 5.7(b) mostra que o *Q-learning* e o SARSA não convergem com rapidez suficiente para cumprir os SLAs, nem direcionam com eficiência a carga da nuvem até o limite pré-estabelecido visando reduzir a ociosidade dos recursos de rede. Portanto, a abordagem que utiliza algoritmos RL puros não se apresentam como solução viável para o mecanismo de provisionamento de largura de banda proposto. A convergência lenta ocorre porque apenas um elemento da Tabela Q é atualizado a cada iteração e não há conhecimento prévio oferecido ao mecanismo para reduzir o tempo de adaptação. Esse tempo de aprendizagem não existe na abordagem implementada usando FIS devido à definição do conjunto de regras difusas. Com essa abordagem, é possível priorizar os inquilinos e permitir que eles usem mais largura de banda do que a contratada, potencialmente aumentando a receita do provedor de nuvem. Assim, a abordagem que implementa o FIS compreende um espaço contínuo de ações e não requer tempo de aprendizagem, implicando tomadas de decisão mais rápidas, mas depende de regras difusas fixas. Essa dependência reduz a adaptabilidade do mecanismo a cenários variados. A abordagem que implementa FQL e FSL supera as outras abordagens porque ambos os algoritmos não têm a limitação de espaço de ações discreto. As regras podem ser modificadas dependendo do cenário e o conhecimento especializado pode ser incorporado como regras iniciais. Esse melhor desempenho pode ser verificado pela altura da mediana e do terceiro quartil nas Figuras 5.7(a) e 5.7(b). Isso implica permitir que os inquilinos utilizem mais largura de banda do que a contratada.

O FIS é mais rápido que os algoritmos RL para manter a carga da nuvem mais próxima de seu limite porque o FIS não precisa de tempo para convergir. Esse tempo

não é necessário porque o FIS pode ajustar diretamente a carga para a largura de banda máxima permitida usando as regras difusas. Embora os algoritmos FRL precisem de mais tempo do que o FIS para ajustar a carga de tráfego, os algoritmos FRL têm um desempenho semelhante ao do FIS devido ao conhecimento especializado inserido nas regras difusas, com a vantagem de serem adaptáveis às mudanças na rede.

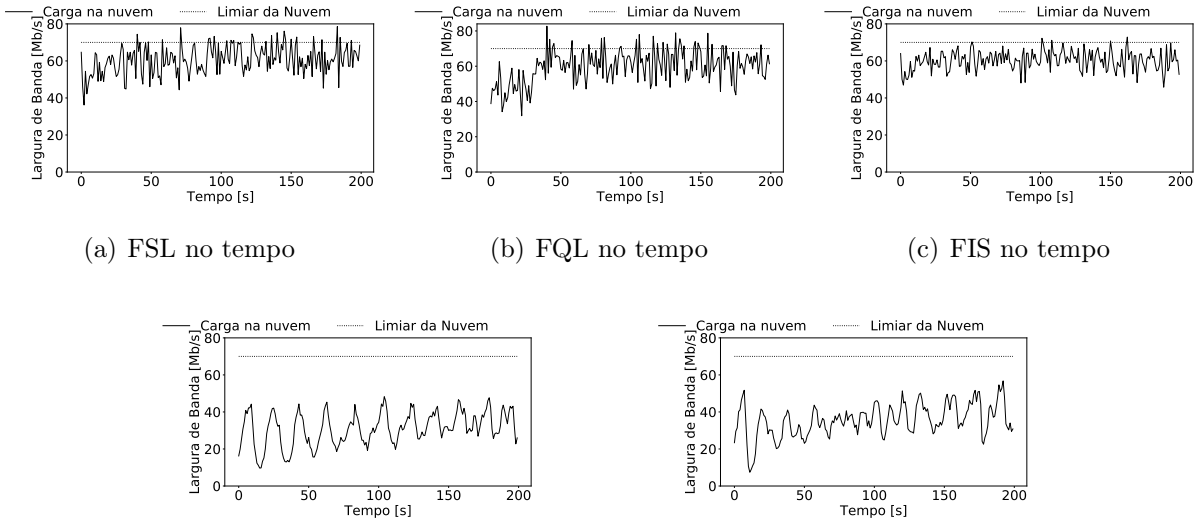


(a) Adequação dos inquilinos à banda contratada.

(b) Adequação da carga da nuvem ao limiar pré-definido.

Figura 5.7: Comparação entre as adequações aos SLAs e a manutenção da carga na nuvem para os 5 algoritmos implementados, calculadas pela banda utilizada pelo inquilino dividida pela banda contratada. Maior altura da mediana indica melhora no cumprimento das SLAs, com consequente potencial par aumentar a receita do provedor. (a) FSL apresenta melhor desempenho, permitindo que os inquilinos usem 35% mais largura de banda do que com o FIS, por ser capaz de modificar as próprias regras difusas. (b) FIS apresenta um desempenho melhor porque não possui um tempo de aprendizado, sendo assim capaz de ajustar diretamente os limiares dos inquilinos para se adequarem ao limiar da nuvem.

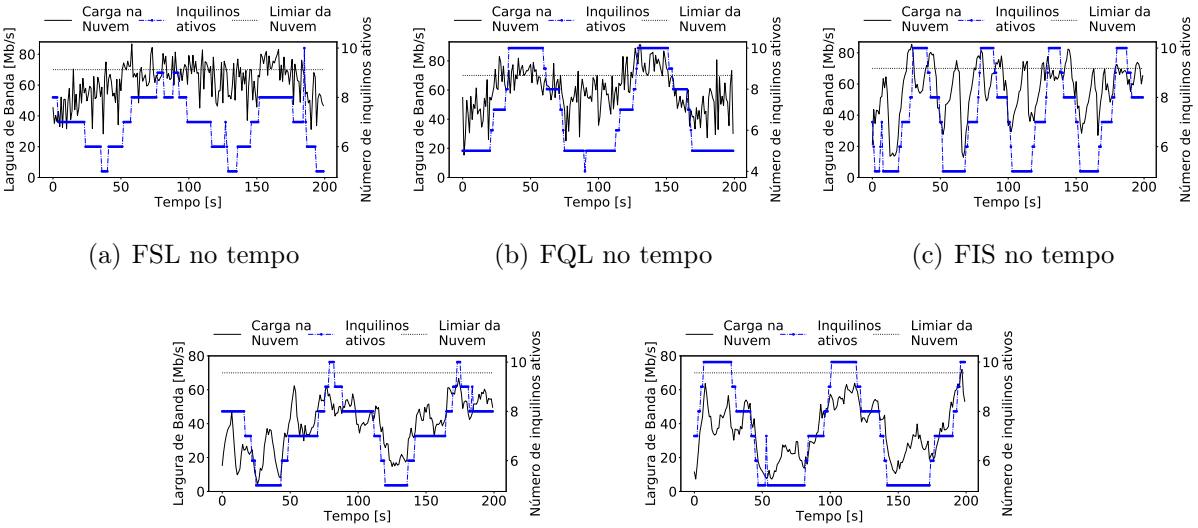
Compara-se também a evolução da carga na nuvem ao longo do tempo, quando o número de inquilinos é constante e cada inquilino tenta exceder sua largura de banda contratada. A Figura 5.8 mostra o comportamento da carga na nuvem obtido nessa situação. Observa-se forte comportamento de dente-de-serra para a carga da nuvem quando algoritmos RL puros são usados. Esse comportamento ocorre devido ao espaço de ações discreto e ao modo de auto-coibição usado para evitar que a carga da nuvem se exceda em mais de 20% do limiar definido. O FIS, Figura 5.8(c), é capaz de regular a largura de banda sem permitir variações bruscas, mantendo a carga da nuvem abaixo do limiar. Analogamente, a Figura 5.9 mostra a carga da nuvem ao longo do tempo quando inquilinos param de usar a nuvem por 30 segundos. O FIS, representado na Figura 5.9(c), e o FRL, que está na Figura 5.9, mantêm uma maior utilização da nuvem em comparação com os algoritmos RL puros. A maior utilização se deve ao conhecimento inserido através das regras difusas, o que implica convergência mais rápida. Esse resultado confirma que usar RL puro não é a abordagem mais adequada para o cenário emulado, devido à variabilidade de inquilinos, com a curva de carga de nuvem quase seguindo a curva de número de inquilinos.



(a) FSL no tempo (b) FQL no tempo (c) FIS no tempo

(d) Q-Learning no tempo (e) SARSA no tempo

Figura 5.8: Comparação do uso da nuvem ao longo do tempo para um número constante de inquilinos. Todos os algoritmos difusos são capazes de deslocar a carga da nuvem para mais perto do uso máximo mais rápido do que os algoritmos RL puros. O conhecimento presente nas regras difusas acelera o processo de convergência. Os algoritmos FRL melhoram essas regras usando o RL para modificar a saída de cada regra, o que implica adaptabilidade.



(a) FSL no tempo (b) FQL no tempo (c) FIS no tempo

(d) Q-Learning no tempo (e) SARSA no tempo

Figura 5.9: Comparação do uso da nuvem ao longo do tempo para um número variável de inquilinos. O FRL e o FIS são mais ágeis na recuperação da variabilidade no número de inquilinos. Esses algoritmos alcançam mais rapidamente a alta utilização da nuvem, devido ao conhecimento inserido através das regras difusas.

Por fim, calcula-se o lucro que cada algoritmo fornece ao provedor IaaS. Utiliza-se o método de precificação *burstable billing* [95], que usa o 95º percentil do uso do tráfego

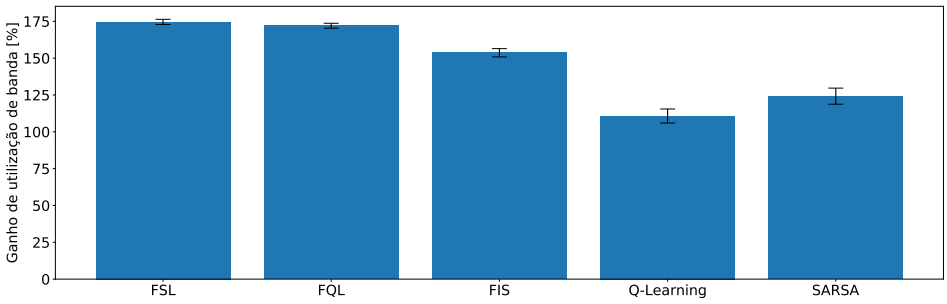


Figura 5.10: Ganho médio de largura de banda para cada algoritmo. O FIS aumenta o uso de largura de banda mais rapidamente do que os algoritmos de aprendizado por reforço puros. Por outro lado, o FQL e o FSL aumentam ainda mais esse uso, tendo a capacidade de modificar as regras difusas para se adaptarem às mudanças do ambiente.

Tabela 5.1: Conjunto de regras na base de dados difusa.

#	Algoritmo	Lucro adicional
1	FSL	$+(0.75 \pm 0.02)RI_n$
2	FQL	$+(0.72 \pm 0.02)RI_n$
3	FIS	$+(0.54 \pm 0.03)RI_n$
4	SARSA	$+(0.24 \pm 0.05)RI_n$
5	*Q-Learning*	$+(0.11 \pm 0.05)RI_n$

para calcular as tarifas dos inquilinos. O lucro é dado pela Equação (5.2),

$$F_n(r_n) = \begin{cases} RI_n, & \text{se } r_n < 0 \\ RI_n + Rr_n, & \text{c.c.} \end{cases} \tag{5.2}$$

em que, F_n é a tarifa do inquilino n, I_n é a quantidade de largura de banda contratada pelo inquilino n, r_n é o 95º percentil da largura de banda consumida menos a largura de banda contratada, e R é o preço para cada unidade de largura de banda dentro do contratado, medido em megabits por segundo. Nesse modelo, quanto maior o 95º percentil de utilização do inquilino, maior a sua tarifa, consequentemente quanto maior o ganho de largura de banda dado pelo mecanismo, maior o lucro para o provedor. A Figura 5.10 mostra o ganho médio de uso de largura de banda para cada inquilino, calculado através do 95º percentil de utilização de largura de banda do inquilino dividia pela banda contratada. O ganho de uso de largura de banda para o FSL é 39% maior que o ganho para o FIS, provendo, portanto, uma receita maior para o provedor de nuvem. O algoritmo FSL apresenta melhor desempenho, pois o 95º percentil de ganho de uso é superior ao dos demais algoritmos, sendo consequência da capacidade de o FSL alterar as regras difusas dependendo do ambiente. Baseado no mesmo resultado mostrado na Figura 5.10, a Tabela 5.1 mostra o aumento de lucro para o provedor na unidade I_n, ordenados de acordo com o lucro.

Capítulo 6

Conclusão

Este trabalho propôs e implementou um mecanismo para provisionamento de largura de banda em um centro de dados multi-inquilino. O objetivo do mecanismo proposto é reduzir a ociosidade dos recursos de rede. O mecanismo prioriza inquilinos, a fim de manter a conformidade com os SLAs. Para tanto, utilizou-se uma estratégia de aprendizado por reforço difuso multiagente para gerenciar a largura de banda utilizada pelos inquilinos a cada instante, de forma a reduzir ou aumentar a disponibilidade de largura de banda para os inquilinos de acordo com o estado atual de consumo dos recursos de rede. O mecanismo proposto automatiza o gerenciamento da largura de banda de forma ágil e adaptável. Essas características são devidas aos algoritmos utilizados para implementar o mecanismo.

Três estratégias foram utilizadas para desenvolver o mecanismo, (i) lógica difusa, (ii) aprendizado por reforço e (iii) aprendizado por reforço difuso. Na primeira, o mecanismo é implementando utilizando um sistema de inferência difuso, que agrega agilidade na tomada de decisão devido à existência de conhecimento prévio na forma de um conjunto de regras difusas. No entanto, essa estratégia se mostrou menos adaptável a variações no cenário estudado. A segunda estratégia lida melhor com essas variações, sendo mais adaptável devido ao uso dos algoritmos de diferença de tempo para Aprendizado por Reforço, SARSA e *Q-Learning*, em duas implementações distintas. Esses algoritmos, no entanto, apresentam maior tempo de convergência para tomada de decisão, devido ao tempo necessário para aprender a melhor política de provisionamento em cada momento. Isso se reflete em um comportamento de dente-de-serra para o gerenciamento do uso de largura de banda. Por fim, a estratégia baseada em aprendizado por reforço difuso reduz a intensidade do comportamento de dente-de-serra, e apresenta maior agilidade no processo de tomada de decisão. Isso ocorre porque, nessa estratégia, o mecanismo é implementado

utilizando *Q-learning* difuso ou SARSA difuso, que usam uma Tabela Q para modificar a saída de regras difusas, mais de uma regra na mesma iteração, e pelo conhecimento prévio inserido nas regras difusas.

A escolha de parâmetros para modelagem dos algoritmos utilizados nas estratégias de aprendizado por reforço e aprendizado por reforço difuso leva em consideração as características do cenário, assim como as regras difusas e funções de pertinência usadas na modelagem da estratégia baseada em sistema de inferência difuso. A estratégia da exploração utilizada foi *softmax*, já utilizado por trabalhos anteriores. O mecanismo proposto utilizando as estratégias baseadas em sistema de inferência difuso e aprendizado por reforço é avaliado em uma análise preliminar para verificação de viabilidade. Essa análise foi feita em um cenário emulado com 3 inquilinos que geram tráfego UDP CBR de baixa taxa. Os resultados mostraram que é possível utilizar o mecanismo para aumentar a largura de banda disponível para os inquilinos e consequentemente aumentar o lucro para o provedor. Os resultados mostraram também que é possível utilizar o mecanismo de provisionamento de largura de banda utilizando o uso de processamento como entrada e que para priorização entre os inquilinos utilizar um sistema de inferência difuso é mais eficiente que aprendizado por reforço.

Verificada a viabilidade do uso de aprendizado por reforço e de lógica difusa para provisionamento de largura de banda no cenário estudado, o mecanismo proposto é validado usando tráfego realístico baseado em um conjunto de dados real. O conjunto de dados NetForager contém aproximadamente 30 GB de dados, contendo fluxos HTTP. O cenário de validação emula a comunicação entre 10 inquilinos e a nuvem. Os inquilinos reproduzem parte do tráfego HTTP, cada inquilino reproduz parte do tráfego de uma aplicação específica. Foram realizadas 20 repetições da emulação para cada algoritmo avaliado. Cada emulação teve a duração necessária para que o mecanismo proposto executasse 200 iterações de tomada de decisão. Verificou-se um aumento de até 72% no 95º percentil do tráfego utilizado pelos inquilinos durante a execução da emulação. A medida do 95º percentil é usada na tarifação do inquilino, e, portanto, o seu aumento implica em um aumento no lucro do provedor de nuvem. Os inquilinos foram capazes de utilizar mais largura de banda do que a contratada, em mais da metade do tempo de emulação. Dessa forma, o mecanismo é benéfico para os inquilinos porque permite aos inquilinos utilizarem mais recursos de rede e é benéfico para o provedor porque reduz a ociosidade da rede e o consumo adicional de largura de banda pelos inquilinos pode ser revertido em receita. Em relação ao estado da arte, o mecanismo inova por utilizar aprendizado por reforço difuso para o gerenciamento de banda em centros de dados, normalmente proposto por outros

autores de maneira estática.

Ao variar a quantidade de inquilinos no cenário, o aprendizado por reforço difuso provou ser robusto, pois quando o número de inquilinos diminui, a largura de banda alocada para os demais inquilinos aumenta. Analogamente, quando o número de inquilinos aumenta, a largura de banda dos outros inquilinos é limitada para que os SLAs dos novos inquilinos sejam cumpridos. Esse comportamento também é verificado para a estratégia que utiliza aprendizado por reforço puro e lógica difusa pura, mas a adaptação às mudanças ocorre mais lentamente. Assim, os resultados mostraram que o mecanismo proposto é capaz de priorizar inquilinos e cumprir os SLAs. Além disso, dentre as estratégias utilizadas, a que apresentou o melhor desempenho foi a baseada em aprendizado por reforço difuso, somando vantagens da lógica difusa e do aprendizado por reforço. Conclui-se, então, que uma implementação em ambiente realístico deve utilizar a abordagem baseada em aprendizado por reforço difuso, pois possui modelagem simples, agilidade e capacidade de se adaptar de acordo com as mudanças do ambiente.

A proposta discutida nesse trabalho apresenta uma contribuição na área de provisionamento de recursos em centros de dados multi-inquilino. O trabalho apresentou um mecanismo de provisionamento de recursos que se baseia em uma estratégia multiagente utilizando aprendizado por reforço [84] e lógica difusa [83]. O trabalho mostrou que o mecanismo apresentado pode ser utilizado mapeando processamento em largura de banda [30]. O trabalho obteve como resultado direto a publicação de um artigo em congresso nacional e dois artigos em congressos internacionais.

Como trabalhos futuros, identifica-se de imediato a necessidade de escalar o cenário de validação, tornando o ambiente mais realístico. Em paralelo, vislumbra-se a inclusão de outras funcionalidades no mecanismo proposto, como o provisionamento de recursos computacionais e análise de tráfego para detecção de anomalias. O uso de ferramentas de emulação mais novas e robustas que o *Maxinet* como o CloudSimSDN [1] deve ser considerado na validação de cenários com maior escala.

[1] https://github.com/Cloudslab/cloudsimsdn

Referências

[1] MELL, P.; GRANCE, T. The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National, 2011.

[2] HERBST, N. R.; KOUNEV, S.; REUSSNER, R. Elasticity in cloud computing: What it is, and what it is not. In: *10th International Conference on Autonomic Computing ({ICAC} 13)*. [S.l.: s.n.], 2013. p. 23–27.

[3] KHAJEH-HOSSEINI, A.; GREENWOOD, D.; SOMMERVILLE, I. Cloud migration: A case study of migrating an enterprise it system to IAAS. In: IEEE. *2010 IEEE 3rd International Conference on cloud computing*. [S.l.], 2010. p. 450–457.

[4] SHEN, H.; LI, Z. New bandwidth sharing and pricing policies to achieve a win-win situation for cloud provider and tenants. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 27, n. 9, p. 2682–2697, 2015.

[5] GIRI, S.; SHAKYA, S. Cloud computing and data security challenges: A nepal case. *International Journal of Engineering Trends and Technology, 67 (3), 146*, v. 150, 2019.

[6] MUÑOZ-ESCOÍ, F. D.; BERNABÉU-AUBÁN, J. M. A survey on elasticity management in paas systems. *Computing*, Springer, v. 99, n. 7, p. 617–656, 2017.

[7] LOUKIS, E.; JANSSEN, M.; MINTCHEV, I. Determinants of software-as-a-service benefits and impact on firm performance. *Decision Support Systems*, Elsevier, v. 117, p. 38–47, 2019.

[8] GUO, J.; SONG, Z.; CUI, Y.; LIU, Z.; JI, Y. Energy-efficient resource allocation for multi-user mobile edge computing. In: IEEE. *GLOBECOM 2017-2017 IEEE Global Communications Conference*. [S.l.], 2017. p. 1–7.

[9] NINE, M. S. Z.; AZAD, M. A. K.; ABDULLAH, S.; RAHMAN, R. M. Fuzzy logic based dynamic load balancing in virtualized data centers. In: IEEE. *2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.], 2013. p. 1–7.

[10] WANG, T.; MA, H.; ZHOU, Y.; ZHANG, R.; SONG, Z. Fully accountable data sharing for pay-as-you-go cloud scenes. *IEEE Transactions on Dependable and Secure Computing*, IEEE, 2019.

[11] DUTREILH, X.; KIRGIZOV, S.; MELEKHOVA, O.; MALENFANT, J.; RIVIERRE, N.; TRUCK, I. Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. In: *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*. Venice: [s.n.], 2011. p. 67–74.

[12] BARRETT, E.; HOWLEY, E.; DUGGAN, J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 25, n. 12, p. 1656–1674, 2013.

[13] CHEN, Z.; HU, J.; MIN, G. Learning-based resource allocation in cloud data center using advantage actor-critic. In: *ICC -2019 IEEE International Conference on Communications*. Singapore: [s.n.], 2019. p. 1–6.

[14] COUTO, R. S.; CAMPISTA, M. E. M.; COSTA, L. H. M. XTC: a throughput control mechanism for xen-based virtualized software routers. In: IEEE. *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*. [S.l.], 2011. p. 1–6.

[15] SON, J.; BUYYA, R. Priority-aware vm allocation and network bandwidth provisioning in software-defined networking (SDN)-enabled clouds. *IEEE Transactions on Sustainable Computing*, IEEE, v. 4, n. 1, p. 17–28, 2018.

[16] POPA, L.; KUMAR, G.; CHOWDHURY, M.; KRISHNAMURTHY, A.; RATNASAMY, S.; STOICA, I. Faircloud: Sharing the network in cloud computing. In: *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. [S.l.: s.n.], 2012. p. 187–198.

[17] GLORENNEC, P. Y.; JOUFFE, L. Fuzzy q-learning. In: IEEE. *Proceedings of 6th international fuzzy systems conference*. [S.l.], 1997. v. 2, p. 659–662.

[18] HENG, S.; NEITZEL, S.; STOBBE, A.; AG, D. B.; MAYER, T. Cloud computing. *Freundliche Aussichten für die Wolke, Deutsche Bank DB Research, Economics. Digitale Ökonomie und struktureller Wandel, Frankfurt am Main*, 2012.

[19] SHEN, Z.; SUBBIAH, S.; GU, X.; WILKES, J. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: *Proceedings of the 2nd ACM Symposium on Cloud Computing*. [S.l.: s.n.], 2011. p. 1–14.

[20] RIMAL, B. P.; MAIER, M. Workflow scheduling in multi-tenant cloud computing environments. *IEEE Transactions on parallel and distributed systems*, IEEE, v. 28, n. 1, p. 290–304, 2016.

[21] MCDOLE, A.; ABDELSALAM, M.; GUPTA, M.; MITTAL, S. Analyzing cnn based behavioural malware detection techniques on cloud IAAS. In: SPRINGER. *International Conference on Cloud Computing*. [S.l.], 2020. p. 64–79.

[22] CHEN, L.; LI, B.; LI, B. Allocating bandwidth in datacenter networks: A survey. *Journal of Computer Science and Technology*, v. 29, n. 5, p. 910–917, set. 2014.

[23] MATTOS, D. M.; FERRAZ, L. H. G.; COSTA, L. H. M.; DUARTE, O. C. M. Evaluating virtual router performance for a pluralist future Internet. In: *Proceedings of the 3rd international conference on information and communication systems*. Irbid, Jordan: [s.n.], 2012. p. 1–7.

[24] SHIEH, A.; KANDULA, S.; GREENBERG, A. G.; KIM, C. Seawall: Performance isolation for cloud datacenter networks. In: *HotCloud*. [S.l.: s.n.], 2010.

[25] ISMAEEL, S.; KARIM, R.; MIRI, A. Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres. *Journal of Cloud Computing*, Springer, v. 7, n. 1, p. 10, 2018.

[26] ALI-ELDIN, A.; TORDSSON, J.; ELMROTH, E. An adaptive hybrid elasticity controller for cloud infrastructures. In: IEEE. *2012 IEEE Network Operations and Management Symposium*. [S.l.], 2012. p. 204–212.

[27] LORIDO-BOTRAN, T.; MIGUEL-ALONSO, J.; LOZANO, J. A. A review of auto-scaling techniques for elastic applications in cloud environments. *J. Grid Comput.*, Springer-Verlag, Berlin, Heidelberg, v. 12, n. 4, p. 559–592, dez. 2014.

[28] PATIKIRIKORALA, T.; COLMAN, A. Feedback controllers in the cloud. In: SN. *Proceedings of Asia-Pacific Software Engineering Conference (APSEC)*. Los Alamitos, CA, USA, 2010. p. 1–6.

[29] HEINZE, T.; PAPPALARDO, V.; JERZAK, Z.; FETZER, C. Auto-scaling techniques for elastic data stream processing. In: *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS)*. New York, NY, USA: Association for Computing Machinery, 2014. p. 318–321. ISBN 9781450327374.

[30] FILHO, R. H. S.; FERREIRA, T. N.; MATTOS, D. M.; MEDEIROS, D. S. A lightweight reinforcement-learning-based mechanism for bandwidth provisioning on multitenant data center. In: IEEE. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.], 2020. p. 331–336.

[31] WANG, Z.; GWON, C.; OATES, T.; IEZZI, A. Automated cloud provisioning on AWS using deep reinforcement learning. *arXiv preprint arXiv:1709.04305*, 2017.

[32] ISBELL, C.; SHELTON, C. R.; KEARNS, M.; SINGH, S.; STONE, P. A social reinforcement learning agent. In: ACM. *Proceedings of the fifth international conference on Autonomous agents*. Montreal, 2001. p. 377–384.

[33] KIUMARSI, B.; VAMVOUDAKIS, K. G.; MODARES, H.; LEWIS, F. L. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems*, IEEE, v. 29, n. 6, p. 2042–2062, 2017.

[34] HASSELT, H. V.; GUEZ, A.; SILVER, D. Deep reinforcement learning with double q-learning. In: *Proc. Thirtieth AAAI conference on artificial intelligence*. Phoenix: [s.n.], 2016. p. 2094–2100.

[35] BOUTABA, R. e. a. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, v. 9, n. 1, p. 16, Jun 2018.

[36] KAMRANI, M.; SRINIVASAN, A. R.; CHAKRABORTY, S.; KHATTAK, A. J. Applying markov decision process to understand driving decisions using basic safety messages data. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 115, p. 102642, 2020.

[37] GOSAVI, A. A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning*, Springer, v. 55, n. 1, p. 5–29, 2004.

[38] VIDYASAGAR, M. Recent advances in reinforcement learning. In: IEEE. *2020 American Control Conference (ACC)*. [S.l.], 2020. p. 4751–4756.

[39] FUJIMOTO, S.; HOOF, H.; MEGER, D. Addressing function approximation error in actor-critic methods. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2018. p. 1587–1596.

[40] FAKOOR, R.; CHAUDHARI, P.; SMOLA, A. J. P3o: Policy-on policy-off policy optimization. In: PMLR. *Uncertainty in Artificial Intelligence*. [S.l.], 2020. p. 1017–1027.

[41] SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

[42] TIJSMA, A. D.; DRUGAN, M. M.; WIERING, M. A. Comparing exploration strategies for q-learning in random stochastic mazes. In: IEEE. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. [S.l.], 2016. p. 1–8.

[43] VERMOREL, J.; MOHRI, M. Multi-armed bandit algorithms and empirical evaluation. In: SPRINGER. *European conference on machine learning*. [S.l.], 2005. p. 437–448.

[44] ARABNEJAD, H.; PAHL, C.; JAMSHIDI, P.; ESTRADA, G. A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling. In: IEEE. *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. [S.l.], 2017. p. 64–73.

[45] WANG, P.; CHAN, C.-Y.; FORTELLE, A. de L. A reinforcement learning based approach for automated lane change maneuvers. In: IEEE. *2018 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2018. p. 1379–1384.

[46] NAVIN, N. K.; SHARMA, R. A fuzzy reinforcement learning approach to thermal unit commitment problem. *Neural Computing and Applications*, Springer, v. 31, n. 3, p. 737–750, 2019.

[47] KOFINAS, P.; DOUNIS, A.; VOUROS, G. Fuzzy q-learning for multi-agent decentralized energy management in microgrids. *Applied energy*, Elsevier, v. 219, p. 53–67, 2018.

[48] JAMSHIDI, P.; SHARIFLOO, A.; PAHL, C.; ARABNEJAD, H.; METZGER, A.; ESTRADA, G. Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures. In: IEEE. *2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*. [S.l.], 2016. p. 70–79.

[49] OMIDZADE, F.; GHODOUSI, H.; SHAHVERDI, K. Comparing fuzzy sarsa learning and ant colony optimization algorithms in water delivery scheduling under water shortage conditions. *Journal of Irrigation and Drainage Engineering*, American Society of Civil Engineers, v. 146, n. 9, p. 04020028, 2020.

[50] ZHOU, L.; SWAIN, A.; UKIL, A. Q-learning and dynamic fuzzy q-learning based intelligent controllers for wind energy conversion systems. In: IEEE. *2018 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*. [S.l.], 2018. p. 103–108.

[51] ZHAO, X.; WANG, X.; MA, L.; ZONG, G. Fuzzy approximation based asymptotic tracking control for a class of uncertain switched nonlinear systems. *IEEE Transactions on Fuzzy systems*, IEEE, v. 28, n. 4, p. 632–644, 2019.

[52] KIM, S.; CHOI, Y.-r. Constraint-aware vm placement in heterogeneous computing clusters. *Cluster Computing*, Springer, v. 23, n. 1, p. 71–85, 2020.

[53] BENIFA, J. B.; DEJEY, D. Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment. *Mobile Networks and Applications*, Springer, v. 24, n. 4, p. 1348–1363, 2019.

[54] SHAW, R.; HOWLEY, E.; BARRETT, E. An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers. In: IEEE. *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. [S.l.], 2017. p. 61–66.

[55] ECK, A.; SOH, L.-K.; DEVLIN, S.; KUDENKO, D. Potential-based reward shaping for finite horizon online pomdp planning. *Autonomous Agents and Multi-Agent Systems*, Springer, v. 30, n. 3, p. 403–445, 2016.

[56] RAO, J.; BU, X.; XU, C.-Z.; WANG, L.; YIN, G. VCONF: a reinforcement learning approach to virtual machines auto-configuration. In: ACM. *Proceedings of the 6th international conference on Autonomic computing*. Wuerzburg, 2009. p. 137–146.

[57] MATEEN, M.; HAYAT, S.; TEHREEM, T.; AKBAR, M. A. A self-adaptive resource provisioning approach using fuzzy logic for cloud-based applications. *International Journal of Computing and Digital Systems*, v. 9, n. 03, 2020.

[58] RAJAGOPAL, E.; BASKARAN, N. Fuzzy softset based vm selection in cloud datacenter. In: IEEE. *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. [S.l.], 2019. p. 462–467.

[59] ARABNEJAD, H.; JAMSHIDI, P.; ESTRADA, G.; IOINI, N. E.; PAHL, C. An auto-scaling cloud controller using fuzzy q-learning-implementation in openstack. In: SPRINGER. *European Conference on Service-Oriented and Cloud Computing*. [S.l.], 2016. p. 152–167.

[60] ZHU, J.; LI, D.; WU, J.; LIU, H.; ZHANG, Y.; ZHANG, J. Towards bandwidth guarantee in multi-tenancy cloud computing networks. In: IEEE. *2012 20th IEEE International Conference on Network Protocols (ICNP)*. [S.l.], 2012. p. 1–10.

[61] BAI, W.; CHEN, L.; CHEN, K.; HAN, D.; TIAN, C.; WANG, H. Information-agnostic flow scheduling for commodity data centers. In: *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. [S.l.: s.n.], 2015. p. 455–468.

[62] BAI, W.; CHEN, L.; CHEN, K.; HAN, D.; TIAN, C.; SUN, W. Pias: Practical information-agnostic flow scheduling for data center networks. In: *Proceedings of the 13th ACM workshop on hot topics in networks*. [S.l.: s.n.], 2014. p. 1–7.

[63] BAI, W.; CHEN, L.; CHEN, K.; HAN, D.; TIAN, C.; WANG, H. Pias: practical information-agnostic flow scheduling for commodity data centers. *IEEE/ACM Transactions on Networking*, IEEE, v. 25, n. 4, p. 1954–1967, 2017.

[64] CHEN, L.; LINGYS, J.; CHEN, K.; LIU, F. Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. [S.l.: s.n.], 2018. p. 191–205.

[65] TANG, Y.; GUO, H.; YUAN, T.; GAO, X.; HONG, X.; LI, Y.; QIU, J.; ZUO, Y.; WU, J. Flow splitter: A deep reinforcement learning-based flow scheduler for hybrid optical-electrical data center network. *IEEE Access*, IEEE, v. 7, p. 129955–129965, 2019.

[66] LEI, K.; LI, K.; XING, J.; JIN, B.; WANG, Y. Distributed information-agnostic flow scheduling in data centers based on wait-time. In: IEEE. *2018 IEEE Global Communications Conference (GLOBECOM)*. [S.l.], 2018. p. 1–7.

[67] WANG, T.; XU, H.; LIU, F. Aemon: Information-agnostic mix-flow scheduling in data center networks. In: *Proceedings of the First Asia-Pacific Workshop on Networking*. [S.l.: s.n.], 2017. p. 106–112.

[68] FU, Q.; QING, L.; YINGZHU, A.; YAMEI, F. A priority based virtual network bandwidth guarantee method in software defined network. In: IEEE. *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. [S.l.], 2015. p. 153–156.

[69] MERA-GÓMEZ, C.; RAMÍREZ, F.; BAHSOON, R.; BUYYA, R. A debt-aware learning approach for resource adaptations in cloud elasticity management. In: SPRINGER. *International Conference on Service-Oriented Computing*. Kanazawa, Japan, 2017. p. 367–382.

[70] HABIB, A.; KHAN, M. I. Reinforcement learning based autonomic virtual machine management in clouds. In: IEEE. *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. Dhaka, Bangladesh, 2016. p. 1083–1088.

[71] STRUHÁR, V.; ASHJAEI, M.; BEHNAM, M.; CRACIUNAS, S. S.; PAPADOPOULOS, A. V. Dart: Dynamic bandwidth distribution framework for virtualized software defined networks. In: IEEE. *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*. [S.l.], 2019. v. 1, p. 2934–2939.

[72] GUO, C.; LU, G.; WANG, H. J.; YANG, S.; KONG, C.; SUN, P.; WU, W.; ZHANG, Y. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In: *Proceedings of the 6th International Conference*. New York, NY, USA: [s.n.], 2010. p. 1–12.

[73] DRUMMOND, A. C.; FONSECA, N. L. da; DEVETSIKIOTIS, M. A multiobjective fuzzy bandwidth partitioning model for self-sizing networks. *European journal of operational research*, Elsevier, v. 191, n. 3, p. 1161–1174, 2008.

[74] SINGLA, A.; SINGH, A.; RAMACHANDRAN, K.; XU, L.; ZHANG, Y. Proteus: a topology malleable data center network. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. [S.l.: s.n.], 2010. p. 1–6.

[75] LAM, V. T.; RADHAKRISHNAN, S.; PAN, R.; VAHDAT, A.; VARGHESE, G. Netshare and stochastic netshare: predictable bandwidth allocation for data centers. *ACM SIGCOMM Computer Communication Review*, ACM New York, NY, USA, v. 42, n. 3, p. 5–11, 2012.

[76] POPA, L.; YALAGANDULA, P.; BANERJEE, S.; MOGUL, J. C.; TURNER, Y.; SANTOS, J. R. Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing. In: *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. [S.l.: s.n.], 2013. p. 351–362.

[77] CARVALHO, H. E.; FERNANDES, N. C.; DUARTE, O. C. M.; PUJOLLE, G. Slapv: A service level agreement enforcer for virtual networks. In: IEEE. *2012 International Conference on Computing, Networking and Communications (ICNC)*. [S.l.], 2012. p. 708–712.

[78] JEYAKUMAR, V.; ALIZADEH, M.; MAZIÈRES, D.; PRABHAKAR, B.; GREENBERG, A.; KIM, C. Eyeq: Practical network performance isolation at the edge. In: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. [S.l.: s.n.], 2013. p. 297–311.

[79] RODRIGUES, H.; SANTOS, J. R.; TURNER, Y.; SOARES, P.; GUEDES, D. O. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. *WIOV*, v. 1, n. 3, p. 784–789, 2011.

[80] PAREDES, R. K.; FAJARDO, A. C.; SISON, A. M. A fuzzy-based dynamic bandwidth allocation approach for campus area networks. In: IEEE. *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*. [S.l.], 2018. p. 1–6.

[81] SIRIPONGWUTIKORN, P.; BANERJEE, S.; TIPPER, D. Fuzzy-based adaptive bandwidth control for loss guarantees. *IEEE transactions on neural networks*, IEEE, v. 16, n. 5, p. 1147–1162, 2005.

[82] MODI, T.; SWAIN, P. Flowdcn: Flow scheduling in software defined data center networks. In: IEEE. *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. [S.l.], 2019. p. 1–5.

[83] FILHO, R. H. S.; FERREIRA, T. N.; MATTOS, D. M. F.; MEDEIROS, D. S. V. A rapid fuzzy controller for decentralized bandwidth provisioning on a multitenant data center. In: *Proceedings of Network of Future(NOF)*. [S.l.: s.n.], 2020. p. 1–8. To appear.

[84] FILHO, R. H. S.; FERREIRA, T. N.; MATTOS, D. M.; MEDEIROS, D. S. Localização de usuários móveis baseada em fingerprint de rádio frequência: Redução de espaço de busca usando parâmetros de atraso de onda das redes celulares. In: *Anais do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2020. ISSN 2177-9384.

[85] RUNKLER, T. A. Selection of appropriate defuzzification methods using application specific properties. *IEEE transactions on fuzzy systems*, IEEE, v. 5, n. 1, p. 72–79, 1997.

[86] RAHIM, R. Comparative analysis of membership function on mamdani fuzzy inference system for decision making. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2017. v. 930, n. 1, p. 012029.

[87] DAS, P.; BEHERA, H.; PANIGRAHI, B. Intelligent-based multi-robot path planning inspired by improved classical q-learning and improved particle swarm optimization with perturbed velocity. *Engineering science and technology, an international journal*, Elsevier, v. 19, n. 1, p. 651–669, 2016.

[88] TOKIC, M. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In: SPRINGER. *Annual Conference on Artificial Intelligence*. [S.l.], 2010. p. 203–210.

[89] VAMPLEW, P.; DAZELEY, R.; FOALE, C. Softmax exploration strategies for multiobjective reinforcement learning. *Neurocomputing*, Elsevier, v. 263, p. 74–86, 2017.

[90] DANE, L.; GURKAN, D. Netforager: Geographically-distributed network performance monitoring of web applications. In: IEEE. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2020. p. 0142–0149.

[91] ZHANG, S.; SUN, Y.; MENG, F.; FU, Y.; JIA, B.; WU, Z. XWM: a high-speed matching algorithm for large-scale url rules in wireless surveillance applications. *Multimedia Tools and Applications*, Springer, p. 1–19, 2019.

[92] LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York, NY, USA: [s.n.], 2010. (Hotnets-IX), p. 19:1–19:6. ISBN 978-1-4503-0409-2.

[93] WETTE, P.; DRÄXLER, M.; SCHWABE, A.; WALLASCHEK, F.; ZAHRAEE, M. H.; KARL, H. Maxinet: Distributed emulation of software-defined networks. In: IEEE. *2014 IFIP Networking Conference*. [S.l.], 2014. p. 1–9.

[94] GONÇALVES, D. S. de M.; COUTO, R. de S.; RUBINSTEIN, M. G. Sistema de proteção contra ataques de http flood usando redes definidas por software. In: SBC. *Anais do XXV Workshop de Gerência e Operação de Redes e Serviços*. [S.l.], 2020. p. 29–42.

[95] NETWORKING, C. V. Cisco global cloud index: Forecast and methodology, 2015-2020. white paper. *Cisco Public, San Jose*, 2016.