UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE INFORMÁTICA

REORGANIZAÇÃO INTELIGENTE DE CLUSTERS EM COMPUTAÇÃO EM NÉVOA UTILIZANDO APRENDIZADO POR REFORÇO

Trabalho apresentado como parte da avaliação da disciplina INF 496, ministrada pelo professor: Marcos Henrique Fonseca Ribeiro, realizado pelo aluno do curso de Ciência da Computação, Moises Henrique Pereira (83390) sob orientação do professor Vitor Barbosa Carlos de Souza

Viçosa, Minas Gerais
Abril, 2019

SUMÁRIO

SUMÁRIO DE IMAGENS

LISTA DE SIGLAS

IA: Inteligência Artificial

QoS: Qualidade de Serviço

NIST: National Institute of Standards and Technology

SaaS: Software as a Service

PaaS: Platform as a Service

IaaS: Infrastructure as a Service

IDE: Integrated Development Environment ou Ambiente de Desenvolvimento Integrado

MDP: Markov Decision Processes ou Processo de Decisão de Markov

INTRODUÇÃO

Nas últimas décadas, uma grande evolução na execução de serviços sob demanda tem ocorrido. Por um lado, vimos o aumento na oferta de recursos de hardware concentrados em grandes centros de dados capazes de processamentos de alto desempenho a um preço relativamente baixo. Por outro lado, o grande avanço nas tecnologias de comunicação com e sem fio permitiu o oferecimento de uma comunicação banda larga, ubíqua e pervasiva. Neste contexto, o paradigma da computação em nuvem surge com o objetivo de unir a mobilidade desejada por usuários finais conectados e a capacidade de processamento e armazenamento demandada por aplicações cada vez mais exigentes.

Segundo NIST (National Institute of Standards and Technology) [11] dos Estados Unidos, a Computação em Nuvem apresenta como principais características: serviço sob demanda (qualquer usuário deve ser capaz de ter suas necessidades computacionais devidamente atendidas de forma automática), amplo acesso à rede (recursos disponíveis devem ser acessíveis a inúmeros clientes com diversas plataformas), agrupamento de recursos (os recursos dos provedores são agrupados em um mesmo ambiente virtual), flexibilidade (capacidade de rápida e fácil alocação e liberação de recursos para e do usuário) e seus serviços devem ser mensuráveis (uma vez que alocação e liberação de recursos deve ser feita, é necessário que se saiba o quanto alocar ou liberar).

Os principais modelos adotados para oferecimento de serviços na nuvem são: software como um serviço (SaaS), onde os usuários têm acesso a aplicações rodando na Nuvem e por meio de um navegador Web podem fazer uso dessas aplicações; plataforma como um serviço (PaaS), onde uma plataforma completa é oferecida aos usuários onde os mesmos podem desenvolver e executar aplicações próprias na Nuvem; e infraestrutura como um serviço (IaaS), onde os usuários podem alocar uma “sub-infraestrutura” e executar, por exemplo, processos que demandam muitos recursos de hardware, sem a necessidade de montar uma infraestrutura local que será utilizada apenas eventualmente.

Considerando as características apresentadas pelo conceito de nuvem, bem como seus distintos modelos de serviço, variadas estruturas organizacionais surgiram no decorrer dos anos. Logo apareceram as ideias de Nuvem privada (criada por alguma organização visando uso exclusivo ou ambiente de negócios), Nuvem comunitária (criada por alguma comunidade especifica, podendo ser de indivíduos ou até de organizações, visando utilização interna a

comunidade ou também o ambiente de negócios), Nuvem publica (construída para uso da população em geral), Nuvem híbridas (criada pela composição de dois ou mais modelos de Nuvem que permanecem disjuntas, porém compatíveis entre si) [11].

Desta forma, fica claro que a utilização da Computação em Nuvem é algo interessante, porém ela também apresenta algumas desvantagens como: potencial congestionamento da rede (uma vez que o transito de dados gerado por muitas aplicações é cada vez mais alto), alto consumo energético (tanto no *backbone* da rede quanto nos diversos servidores localizados nos centros de dados na nuvem), alta latência (pois os servidores podem estar muito distantes do usuário final) e preocupações com segurança e privacidade (uma vez que dados sensíveis de usuários e empresas são potencialmente acessados a partir da internet). Por causa desses infortúnios surge a ideia de Computação em Névoa.

A Computação em Névoa [1] é uma arquitetura para computação, comunicação, armazenamento e controle localizada próxima ao usuário final. O conceito de Névoa permitirá agrupar, orquestrar, gerenciar e garantir segurança dos recursos computacionais disponíveis na borda da rede de modo a abranger múltiplos domínios e aplicações além de ser algo capaz de trabalhar bem tanto com conexões por fio quanto por Wireless [2].

Algo que caracteriza bem a Névoa é o fato de possuir tamanho flexível, o que torna bem ajustável ao ambiente e as disponibilidades de recursos presentes. Outra coisa que se pode destacar é que não se deve confundir Névoa com mini-Nuvem, uma vez que mini-Nuvens passam a ideia de plataformas de computação isoladas e parte do conceito de Névoa é a integração entre as Névoas [2].

Outra característica da Névoa que vale destaque é sua busca por novas formas de decomposição de tarefas para que possam ser executadas em máquinas com hardware heterogêneo, possivelmente voláteis em questões de disponibilidade, mobilidade e segurança, além de limitada capacidade de processamento, armazenamento ou bateria [2].

Mas apesar desses desafios, que podem parecer desvantagens, pode-se destacar claros pontos positivos no uso de Névoa, por exemplo: segurança (reduzindo a distância no trafego de informações sensíveis, verificações de identidade e autenticações podem ser fortalecidos), cognição (também pela proximidade do usuário final, torna-se mais fácil a decisão de onde e como realizar as tarefas, de modo a atender melhor às necessidades e refletir com mais precisão a disponibilidade de recursos na borda da rede), agilidade (inovação rápida e escalável), latência (extremamente útil para aplicações de tempo real, uma vez que estando perto do usuário o

tempo de propagação é muito menor resultando em um impacto positivo no tempo de resposta), eficiência (pode distribuir computação, armazenamento e controle de modo a otimizar o uso dos recursos disponíveis) [2].

Para exemplificar o uso de computação em Névoa, podemos tomar: veículos conectados (rico cenário de conectividade e interações tanto carro a carro, quanto carro a ponto de acesso ou ponto de acesso a ponto de acesso) ou sensores sem fio (os sensores em sua maioria são extremamente limitados a pouca energia, pouco poder de processamento, pouca memória, portanto são usados normalmente para capturar dados e envia-los para quem de fato vai processa-los). O uso da Névoa é ideal para ambos pois atua com baixa latência, grande heterogeneidade, e suporta tarefas de tempo real [1].

Tendo exposto algumas importantes características da Névoa, fica claro que a implementação de mecanismos eficientes para tomadas de decisões em vários contextos são fundamentais para que se alcance um funcionamento próximo ao ótimo. Alguns exemplos bem estudados são as tomadas de decisões relativas a alocação de recursos [3,4], obtenção de eficiência energética [5,6], orquestração de serviços [7,8], para citar apenas alguns.

Uma outra tomada de decisão que certamente tem impacto significativo na QoS (Qualidade de Serviço) é a reorganização eficiente dos recursos que fazem parte de distintas Névoas, por exemplo, devido à mobilidade de dispositivos subjacentes. Como a arquitetura da Névoa é baseada em um modelo colaborativo, onde recursos, mesmo que móveis, devem estar localizados em uma área geograficamente próxima ao usuário final, é fundamental otimizar o tempo em que cada recurso estará disponível para os clientes em um domínio da Névoa, minimizando as mudanças de domínios ao mesmo tempo que mantendo uma baixa latência na comunicação.

Diversos pesquisas na área de redes de computadores, principalmente em redes sem fio, têm feito uso de técnicas de Machine Learning para avaliar o ganho de eficiência obtido em decisões em tempo real quando comparadas com os mecanismos utilizados pelas arquiteturas de rede tradicionais [9]. Uma técnica que tem ganhado campo é a de aprendizagem por reforço. Essa técnica é uma fusão de técnicas de tentativa e erro com teoria de controle ótimo, o que permite aos pesquisadores de Inteligência Artificial (IA) desenvolverem algoritmos que maximizem o retorno (recompensa) de longo prazo do agente ao realizar alguma tarefa [10].

A função de recompensa é o retorno que o ambiente dá ao agente como resposta e pode estar associada a um estado ou um par ação-estado (onde estado representa algum fator que o

agente pode considerar para realizar uma ação e ação representa alguma decisão que o agente aprende a tomar) definindo o objetivo do agente em uma dada situação, dessa forma a função de recompensa define qual o objetivo desse agente, indicando para o mesmo o que é bom e o que é ruim para ele [10].

Partindo disso, o presente trabalho visa utilizar a técnica de aprendizado por reforço para ajudar (orientar) na tomada de decisão de organização e reorganização de clusters de forma inteligente, buscando maximizar o tempo em que elementos distribuídos em distintos domínios da Névoa permaneçam em um mesmo domínio, minimizando o número de trocas de clusters devido a mobilidade (ou outros fatores).

REVISÃO BIBLIOGRÁFICA

HANDOVER

O processo de Handover consiste na troca de um ponto de acesso para outro ponto de acesso e é composto por duas etapas: a primeira trata-se da escolha do melhor ponto de acesso para a nova associação e a segunda trata-se da autenticação e associação de fato com o novo ponto de acesso escolhido na etapa anterior. Devido a necessidade de alguns serviços, como jogos de tempo real e serviços de vídeo e voz por exemplo, por conexão constante ou o mais próximo possível disso, é obrigatório que o Handover seja rápido e eficiente, visando a continuidade dos serviços com o mínimo possível de interferência para que todo o processo fique transparente ao usuário [16].

Devido a essas necessidades por Handovers mais rápidos e eficientes, muitas pesquisas têm buscado soluções utilizando métodos mais sofisticados por meio de algoritmos de aprendizado e heurísticas para fazer predições. Algumas buscam uma melhor escolha do novo ponto de acesso enquanto outras buscam a redução do tempo gasto na busca por esse novo ponto de acesso.

Para predições de *Handover* podem ser usadas algumas abordagens: abordagem de Célula usa informações globais sobre o fluxo de trafego dos usuários, abordagem de Usuário faz uso de informações mais especificas ao considerar cada usuário e seus comportamentos [14], também tem-se a abordagem que leva em conta o grafo de vizinhos (trata-se de uma estrutura de dados que abstrai as relações de Handover entre os pontos de acesso agrupando as informações de canais vizinhos ao ponto de acesso atual e os vizinhos de cada um desses canais) usando também o conceito de células [15], abordagem baseada em WiMap [16].

A abordagem de Célula apresenta algumas vantagens pois não exige nenhum controle de mensagens e apenas gerencia suas estatísticas de forma independente devido a sua observação global do sistema e do fluxo dos usuários. Para um usuário em uma dada Célula que deseja mudar para outra, basta que a mesma indique a seus vizinhos o tipo de volume de trafego que será passado (pela mudança de Célula), dessa forma a melhor alocação será

escolhida para o usuário. Porém essa abordagem é tida como interessante quando o fluxo é continuo [14].

A abordagem de Usuário é mais complexa pois exige um aprendizado (seus movimentos são observados e medidos buscando origem, destino e tempo em cada Célula) sobre a mobilidade do usuário para lhe fornecer o *Handover* mais adequando e periodicamente algum algoritmo tem que ser rodado para garantir que a aprendizagem esteja atualizada [14].

Para desenvolver e testar essa técnica é utilizado um agente (representante do usuário) que se movimenta (normalmente utilizando métodos probabilísticos – probabilidade de uma mudança de uma Célula para o seu vizinho) na rede passando pelas Células e cumprindo tarefas nas mesmas, dessa forma os dados necessários para o aprendizado são extraídos, vale ressaltar que cada agente deve ser considerado de forma separada [14].

Além disso essa abordagem considera como ciclos os hábitos de cada indivíduo e como esses ciclos mantém uma sequência que representa toda a movimentação, o que inclui identificação das Células visitadas e tempo dispendido em cada. Para evitar que haja supertreinamento em vez de aprendizado, aleatoriamente ciclos podem ser sobrescritos por movimentações aleatórias para simular comportamentos imprevisíveis dos indivíduos. Dessa forma a predição leva em consideração até mesmo essas imprevisibilidades, porém não é tão boa quanto se tem novos agentes inseridos de forma repentina [14].

A abordagem de grafo de vizinhos pode utilizar 3 métodos: centralizado, distribuído e orientado ao usuário. No método centralizado o grafo de vizinhos é armazenado em um servidor chamado NG-server, dessa forma os Handovers são apenas informados aos pontos de acesso. No método distribuído cada ponto de acesso possui seu próprio grafo de vizinhos e uma lista com os pontos de acesso vizinhos [15].

A abordagem baseada em WiMap segue a seguinte ideia: quando se perde a conexão com a estação, um algoritmo de predição é executado para se obter um conjunto de prováveis pontos de acesso. Com esse conjunto de possíveis pontos de acesso, a localização atual e um raio arbitrário outro algoritmo é executado, dessa vez retornando um conjunto com os potenciais pontos de acesso para conexão. Com esse novo conjunto de pontos de acesso mais limitado a estação reduz o tempo de busca do melhor ponto de acesso [16].

METODOLOGIA

Esta seção irá abordar detalhes da metodologia utilizada para o desenvolvimento do presente trabalho. Será visto o uso da técnica de aprendizado por reforço na tentativa de aprimorar a organização e/ou reorganização de elementos que compõem a Névoa. Ao utilizar aprendizado por reforço, espera-se que a inteligência atribuída por essa técnica contribua para uma clusterização mais eficaz e eficiente. Para avaliar os resultados obtidos, será utilizado um simulador (OMNeT++) para extração de dados relativos a mobilidade e consequente desconexão/conexão a distintos pontos de acesso. Tal simulador apresenta uma rede tradicional onde a escolha do ponto de acesso se dá de forma automática onde o ponto de acesso apresentando a melhor relação sinal-ruído (SNR) é selecionado e a continuidade no mesmo se dá até que o host saia de sua área de cobertura (o que as vezes pode ser algo ruim, pois podem haver melhores opções de pontos de acesso mesmo ainda estando na área de cobertura do ponto de acesso da primeira conexão).

APRENDIZADO POR REFORÇO

Aprendizado por reforço é uma das três grandes técnicas de aprendizado de máquina existentes, sendo elas: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço. Cada uma apresenta diferenças conceituais embora normalmente sejam tratadas como a mesma coisa [18].

No aprendizado supervisionado é fornecido um conjunto de exemplos devidamente catalogados por um supervisor humano onde cada exemplo é uma descrição de uma situação e sua catalogação quanto ás suas características de modo a informar à máquina qual a ação correta para aquela dada situação. Nesses sistemas a ideia central é que o próprio seja capaz de generalizar suas respostas [18].

No aprendizado não supervisionado normalmente é tratado a busca por padrões ocultos em coleções de dados não catalogados [18].

Enquanto o aprendizado por reforço consiste na ideia de fazer com que um agente seja capaz de cumprir algum conjunto de tarefas sem que haja a necessidade de programa-las explicitamente. Decorrente disso, pesquisas em IA tem buscado meios de ensinar esse conjunto

de tarefas ao agente por meio de aprendizado supervisionado, não supervisionado ou por reforço [12].

No aprendizado por reforço o agente aprende estratégias ao interagir com o meio em que está inserido por meio do aprendizado das condições, ou seja, que ele seja capaz de entender como funcionam as associações entre o estado e a escolha das ações possíveis e que conduzem a recompensas positivas ou negativas [12]. Uma vez que o agente seja punido ou agraciado pelas recompensas ele irá tentar maximiza-las ao preferir ações que tenham sido avaliadas no passado e produzam boas recompensas [18].

Como exemplo, pode-se considerar um agente que jogue Xadrez. Quando ele coloca o oponente em checkmate, ele é informado que isso é algo bom, porém quando ele é colocado em checkmate, ele é informado que isso é algo ruim. Esse retorno a ele é a recompensa pela ação [13]. Dessa forma o agente deve descobrir, por meio de tentativa e erro, quais ações são mais proveitosas [12].

Porém o agente em fase de treinamento não se atém apenas as melhores ações, pois as vezes pode ser interessante para ele tentar algo novo por ter a chance de obter recompensas ainda melhores que as conhecidas, tal processo é conhecido como exploração, dessa forma pode

estar uma variedade de ações e assim determinar com mais propriedade quais são de fato as melhores ações para determinado estado do ambiente [18].

A título de curiosidade vale observar a forte interação que aprendizado por reforço apresenta com psicologia e neurociência na inspiração desses algoritmos, pois é muito próximo do tipo de aprendizado que é utilizado por animais e pelos humanos [18].

Voltando ás características de aprendizado por reforço, pode-se destacar os elementos que compõem esta técnica, sendo eles, política, recompensa, função valor e em alguns casos um modelo do ambiente: [18]

- Política: pode ser comparada a um conjunto de respostas a estímulos que algum indivíduo recebe, ou seja, ao comportamento do indivíduo ou agente ao longo do tempo;
- Recompensa: pode ser comparada as sensações de dor ou de prazer momentâneos, ou seja, diz ao agente o que é bom e o que é ruim, dessa forma é responsável por definir os objetivos do agente;
- Função valor: pode ser comparada à capacidade de julgamento de uma dada situação devido a experiências anteriores, ou seja, é o montante de recompensas do agente em um dado período de tempo;
- Modelo do ambiente: pode ser uma representação aproximada do ambiente real que o agente deve ou deveria interagir, com esses modelos é possível fazer inferências sobre o ambiente e seus comportamentos.

Uma vez que a ideia de política, recompensa e função valor tenham sido mencionadas pode-se introduzir os conceitos de Processo de Decisão de Markov (MDP), pois esse tipo de problema envolve tanto avaliação de retornos do ambiente quanto a ideia de tomada de diferentes ações para diferentes situações [18].

Em MDPs são utilizados valores $q_*(s, a)$ que determina o valor de um estado e uma determinada ação e $v_*(s)$ que determina a melhor ação dado um estado, com essa interdependência é possível pensar no aprendizado [18].

Para o caso de talvez não ter ficado tão claro quem seria o agente, pode-se defini-lo como o ser que irá aprender com as interações com o ambiente, ele que tomará as decisões e receberá os retornos do ambiente em que estará inserido e será ele quem tentará maximizar esses retornos do ambiente.

De forma mais descritiva a cada iteração o agente visualiza uma representação do ambiente, essa representação é o estado normalmente chamado de S e para esse dado S ele deve tomar uma ação A dentre todas as possíveis para esse estado e como consequência de sua ação o agente recebe uma recompensa R e se desloca para um novo estado S' [18].

O que descreve a mudança de um estado para outro e a recompensa recebida é uma distribuição de probabilidade que depende apenas de um estado e ação fornecidos, dessa forma é definida a dinâmica do MDP [18].

Algo que merece ser observado é a propriedade de Markov, que diz que cada estado deve conter informações sobre todos os aspectos relevantes das iterações passadas do agente com o ambiente [18].

Pela sua alta abstração e flexibilidade, o MDP pode ser aplicado a vários problemas e de diversas formas. Um exemplo pode ser o de um robô de reciclagem:

Um robô precisa coletar latas vazias, porém é limitado pela sua capacidade de bateria. Para modelar esse exemplo, pode-se considerar os estados como o nível disponível de bateria, dessa forma o conjunto de estados pode ser limitado em nível alto e baixo (para simplificar). Logo para cada estado o robô pode decidir entre procurar (gasta bateria) uma lata vazia, ficar parado (não gasta bateria) ou ir recarregar sua bateria (caso o nível da bateria esteja cheio a recarga seria algo desnecessário), portanto o conjunto de ações para cada estado pode ser dado como:

- Nível alto: procurar lata ou esperar;
- Nível baixo: procurar lata, esperar ou recarregar.

As recompensas podem ser definidas como sendo positivas quando o robô encontra uma lata, sendo nulas quando ele não faz nada e sendo fortemente negativas quando a bateria dele acaba, pois nesse caso ele precisará de uma intervenção externa para resgata-lo (o que não é desejável).

Considerando esses estados e ações, é possível imaginar que quando ele começar a procurar ele pode em algum momento mudar de estado pois o nível de sua bateria irá diminuir ao fazer a procura, mas para isso existe a probabilidade de acontecer, ou seja, não é algo imediato. Dessa forma o robô pode transitar entre os estados dado as ações que ele for tomando por meio das iterações com o ambiente.

Voltando as características dos MDPs é interessante que o agente ao decorrer das iterações aprenda quais são as melhores ações, dessa forma aprendendo a política que maximize as recompensas (política ótima), porém na pratica dificilmente algo assim é alcançado, pois em

alguns casos a quantidade de estados é muito grande (o que iria requerer uma quantidade absurda de memória) para ser computada, portanto aproximações podem ser bem vistas.

OMNET++

Trata-se de uma plataforma modular (porém não é um simulador em si) e extensível desenvolvida em C++ e utilizada para simulação de redes de comunicação, podendo abranger redes de comunicação cabeada ou sem fio, em chips e demais tipos de redes. Fornece um IDE baseado no Eclipse, modo de execução com interface gráfica interativa e permite uso de extensões para simulações de tempo real, emulação de rede, integração com banco de dados e outras funcionalidades [17].

De forma mais descritiva pode-se dividir o OMNeT++ em: kernel de simulação feito em C++, linguagem NED para descrição da topologia, ambiente de desenvolvimento integrado, interface interativa para execução das simulações, acesso à linha de comando para execução e ferramentas. Permite integração de inúmeros projetos com modelos de simulação e frameworks de modelos (em sua maioria de código aberto) desenvolvidos de forma independente. Dentre esses frameworks vale destacar o INET, pois é tido como a biblioteca de modelo de protocolo padrão do OMNeT++, contendo vários modelos, protocolos e componentes. O framework INET é mantido pela própria equipe de OMNeT++ e é comumente utilizado como base para outros frameworks [17].

Partindo disto, tanto o OMNeT++ quanto o INET tem sido utilizados no presente trabalho na geração de dados que alimentarão o algoritmo de aprendizado por reforço e também poderão ser usados para as comparações dos resultados obtidos posteriormente.

Os dados em sua forma bruta já apresentam informações interessantes, pois com eles é possível saber onde o host está conectado em um dado instante (qual ponto de acesso), quanto tempo ele ficou conectado nesse ponto de acesso, quando ele se desconectou, quanto tempo ficou desconectado, e com dados auxiliares extraídos por meio de alterações feitas nas classes do INET, na classe responsável pela mobilidade, mais especificamente, é possível saber a posição do host em cada segundo de sua movimentação.

MODELO

Uma vez que o problema se trata de otimizar escolha da organização/reorganização de clusters na Névoa, assumir uma abordagem que dê mais inteligência a esse processo de decisão parece ser uma ideia no mínimo interessante. Partindo disto, foi escolhida a técnica de aprendizado por reforço, que permite que o agente aprenda com o ambiente sem necessariamente conhecer todas as dinâmicas do mesmo. Com isto, espera-se que o agente possa escolher melhores conexões se comparado à técnica tradicional (escolha do ponto de acesso mais próximo e mudança de ponto apenas ao sair da área de cobertura), dessa forma aumentando o tempo conectado em um dado ponto de acesso e minimizando também o número de trocas de pontos de acesso.

Por meio do OMNeT++ e do INET foram realizadas simulações usando uma rede convencional, onde um host representando um indivíduo de carro, por exemplo, se desloca em um grid quadriculado de tamanho 600x400 metros se movendo com velocidade constante. Nesse percurso, o host alterna suas conexões entre os pontos de acessos disponíveis e em alguns casos fica sem conectar por não apresentar nenhum ponto de acesso ao seu alcance.

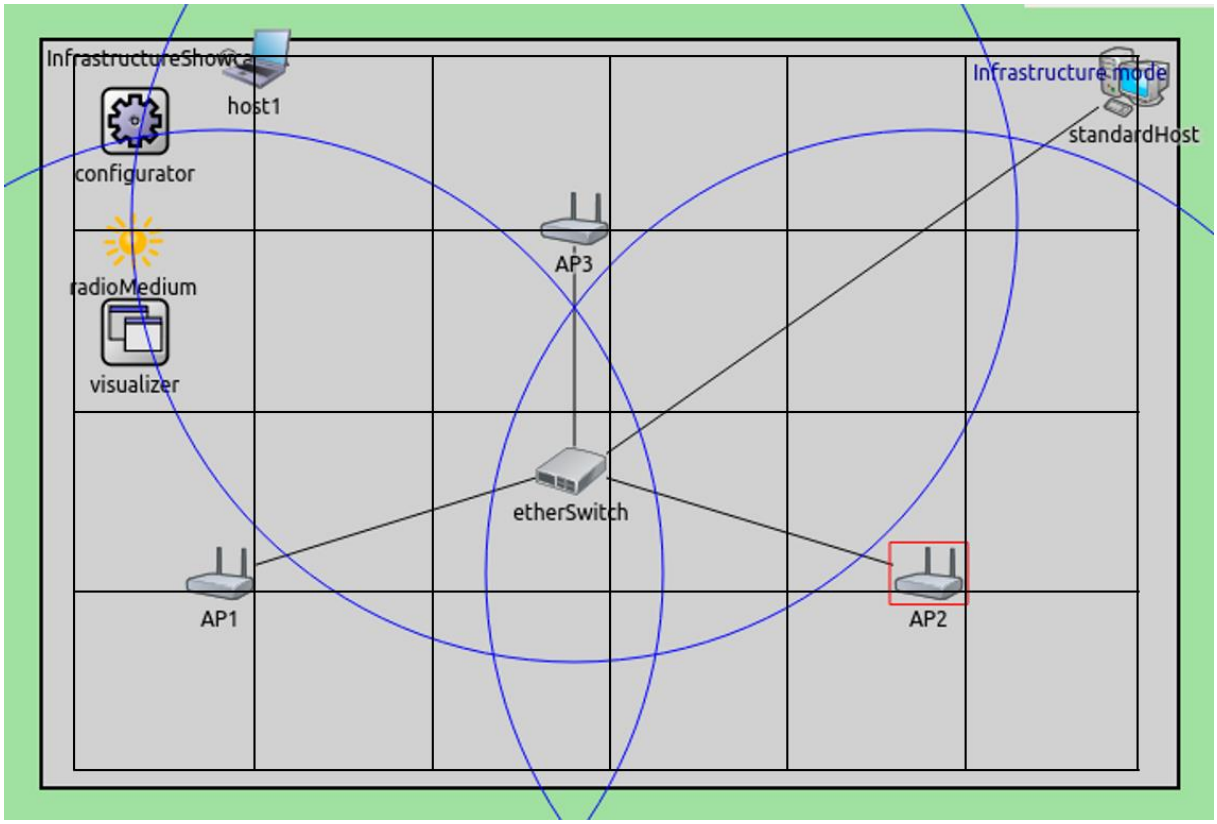

Figura 1: grid que orienta a movimentação do host1

Com essa simulação é possível extrair informações sobre onde e quando o host se conecta, quanto tempo ele fica conectado em um dado ponto de acesso, quando ele se desconecta e quanto tempo ele fica desconectado. Vale ressaltar que a movimentação foi definida de forma aleatória ao atribuir probabilidades aos cruzamentos ou vértices desse grid. Dessa forma, uma rota aleatória é definida e utilizada a cada simulação.

```
1 -- T InfrastructureShowcaseA.host1.wlan[0].radio 0 M ProbeReq
8841 S 126.162041881385 (120, 48.3796, 0) m -> E
126.162079881385 (120, 48.3796, 0) m
2 -- T InfrastructureShowcaseA.AP3.wlan[0].radio 1 M AssocResp-
OK 8863 S 126.464565018686 (300, 100, 0) m -> E 126.464607018686
(300, 100, 0) m
3 -- R InfrastructureShowcaseA.host1.wlan[0].radio 0 M
AssocResp-OK 8863 S 126.46456564616 (120, 45.3543, 0) m -> E
126.46460764616 (120, 45.3543, 0) m
4 -- T InfrastructureShowcaseA.host1.wlan[0].radio 0 M ProbeReq
9516 S 135.619933098053 (63.8007, 10, 0) m -> E 135.619971098053
(63.8007, 10, 0) m
5 -- T InfrastructureShowcaseA.host1.wlan[0].radio 0 M ProbeReq
9848 S 145.369933098053 (20, 63.6993, 0) m -> E 145.369971098053
(20, 63.6993, 0) m
6 -- T InfrastructureShowcaseA.AP1.wlan[0].radio 3 M AssocResp-
OK 9868 S 145.672057211393 (100, 300, 0) m -> E 145.672099211393
(100, 300, 0) m
```

Figura 2: exemplo dos dados extraídos do INET

No exemplo vale ressaltar que os dados extraídos estão em sua forma bruta, ainda sem processamento e adequações. Considerando os dados nessa forma, as informações mais relevantes são: InfrastructureShowcaseA.host1.wlan[0].radio ou InfrastructureShowcaseA.AP*.wlan[0].radio, que informa se a mensagem é emitida pelo host ou por um ponto de acesso qualquer ao qual o mesmo se encontra conectado; ProbeReq; AssocResp-OK; e os valores numéricos que sucedem os caracteres S e E (com uma diferença tão pequena que pode ser considerada irrelevante), que representam o instante da mensagem.

Quando o host deseja se conectar, ele envia um ProbeReq solicitando conexão, dessa forma sabe-se que ele está desconectado (linha 1). Uma vez que se associa a algum ponto de acesso, eles trocam mensagens de AssocResp-OK (linha 2 e 3), dessa forma é possível saber quando e com quem o host se conectou. Após uma conexão, o ProbeReq que se segue, por aproximação, representa o momento em que o host sai da área de cobertura do ponto de acesso com isso pode-se estimar o tempo que ficou conectado nesse ponto de acesso (linha 4). Dois ProbeReq seguidos representam a situação em que o host ficou desconectado um tempo significativo, dessa forma é possível estimar o

tempo que ficou desconectado de todos os pontos de acessos apenas olhando a diferença de tempo entre o primeiro e o segundo ProbeReq.

Devido a necessidade de mais dados e de dados mais detalhados, foi necessário a modificação da classe do INET responsável pela mobilidade do host. Nenhuma mudança em sua dinâmica de funcionamento foi feita, entretanto um código para extrair a posição do host em função do tempo foi acrescentado.

```
0 --- (220, 10, 0) m
1 --- (220, 20, 0) m
2 --- (220, 30, 0) m
3 --- (220, 40, 0) m
4 --- (220, 50, 0) m
5 --- (220, 60, 0) m
6 --- (220, 70, 0) m
7 --- (220, 80, 0) m
8 --- (220, 90, 0) m
9 --- (220, 100, 0) m
10 --- (220, 110, 0) m
11 --- (220, 120, 0) m
12 --- (220, 130, 0) m
13 --- (220, 140, 0) m
14 --- (220, 150, 0) m
15 --- (220, 160, 0) m
16 --- (220, 170, 0) m
17 --- (220, 180, 0) m
18 --- (220, 190, 0) m
19 --- (220, 200, 0) m
20 --- (220, 210, 0) m
21 --- (210, 210, 0) m
22 --- (200, 210, 0) m
23 --- (190, 210, 0) m
24 --- (180, 210, 0) m
25 --- (170, 210, 0) m
26 --- (160, 210, 0) m
27 --- (150, 210, 0) m
28 --- (140, 210, 0) m
29 --- (130, 210, 0) m
30 --- (120, 210, 0) m
31 --- (110, 210, 0) m
```

Figura 3: exemplo de dados da mobilidade

O valor numérico inicial representa o tempo em segundos e a tupla seguinte representa a posição, em metros, do host em termos das coordenadas X, Y, Z dado o tempo.

MODELO PROPOSTO

Outro passo importante é a modelagem do problema de modo que seja possível a sua implementação. Dessa forma, considerando os dados disponíveis e os objetivos do projeto, pensou-se inicialmente em uma matriz de valores $q(S, a)$ onde cada estado é representado por uma tupla (P, AP), P representa a posição do agente em um dado instante do tempo e AP representa onde esse agente está conectado nesse mesmo instante, e cada ação é representada por uma tupla (AP, M) onde

AP representa o ponto de acesso disponível para conexão e M representa a movimentação do agente sobre o grid.

	(AP_0, M)	(AP_1, M)	(AP_2, M)	(AP_3, M)
(P_0, AP_0)				
(P_0, AP_1)				
(P_0, AP_2)				
(P_0, AP_3)				
...				
(P_n, AP_0)				
(P_n, AP_1)				
(P_n, AP_2)				
(P_n, AP_3)				

Figura 4: matriz q(S,a)

Além disso pensou-se na função de recompensa que será responsável por direcionar o objetivo do agente. Inicialmente está definida como:

- Cada vez que o agente se desconectar de um ponto de acesso ele será punido (recompensa fortemente negativa);
- A cada instante que o agente ficar desconectado ele será punido (recompensa negativa);
- A cada instante que ele ficar conectado em um dado ponto de acesso ele não será punido (recompensa 0).

Como o agente tenta maximizar sua recompensa final, espera-se que ele entenda e aprenda que mais tempo conectado em um mesmo ponto de acesso é algo bom, enquanto que ficar trocando

de pontos de acesso ou ficar sem conectar em nenhum ponto de acesso são coisas extremamente ruins.

REFERÊNCIAS

[1] BONOMI, FLAVIO, ET AL. "FOG COMPUTING AND ITS ROLE IN THE INTERNET OF THINGS." PROCEEDINGS OF THE FIRST EDITION OF THE MCC WORKSHOP ON MOBILE CLOUD COMPUTING. ACM, 2012.

[2] CHIANG, MUNG, ET AL. "CLARIFYING FOG COMPUTING AND NETWORKING: 10 QUESTIONS AND ANSWERS." IEEE COMMUNICATIONS MAGAZINE55.4 (2017): 18-20.

[3] XU, JINLAI, ET AL. "ZENITH: UTILITY-AWARE RESOURCE ALLOCATION FOR EDGE COMPUTING." 2017 IEEE INTERNATIONAL CONFERENCE ON EDGE COMPUTING (EDGE). IEEE, 2017.

[4] SKARLAT, OLENA, ET AL. "TOWARDS QOS-AWARE FOG SERVICE PLACEMENT." 2017 IEEE 1ST INTERNATIONAL CONFERENCE ON FOG AND EDGE COMPUTING (ICFEC). IEEE, 2017..

[5] DENG, RUILONG, ET AL. "TOWARDS POWER CONSUMPTION-DELAY TRADEOFF BY WORKLOAD ALLOCATION IN CLOUD-FOG COMPUTING." 2015 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC). IEEE, 2015.

[6] SHOJAFAR, MOHAMMAD, NICOLA CORDESCHI, AND ENZO BACCARELLI. "ENERGY-EFFICIENT ADAPTIVE RESOURCE MANAGEMENT FOR REAL-TIME VEHICULAR CLOUD SERVICES." IEEE TRANSACTIONS ON CLOUD COMPUTING (2016).

[7] SIMOENS, PIETER, ET AL. "CHALLENGES FOR ORCHESTRATION AND INSTANCE SELECTION OF COMPOSITE SERVICES IN DISTRIBUTED EDGE CLOUDS." 2015 IFIP/IEEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT (IM). IEEE, 2015.

[8] VELASQUEZ, KARIMA, ET AL. "FOG ORCHESTRATION FOR THE INTERNET OF EVERYTHING: STATE-OF-THE-ART AND RESEARCH CHALLENGES." JOURNAL OF INTERNET SERVICES AND APPLICATIONS 9.1 (2018): 14.

[9] JAGANNATH, JITHIN, ET AL. "MACHINE LEARNING FOR WIRELESS COMMUNICATIONS IN THE INTERNET OF THINGS: A COMPREHENSIVE SURVEY." ARXIV PREPRINT ARXIV:1901.07947 (2019).

[10] SUTTON, RICHARD S., AND ANDREW G. BARTO. "REINFORCEMENT LEARNING." JOURNAL OF COGNITIVE NEUROSCIENCE 11.1 (1999): 126-134.

[11] MELL, P., & GRANCE, T. (2011). THE NIST DEFINITION OF CLOUD COMPUTING.

[12] FARIA, G., & ROMERO, R. A. F. (2002). NAVEGAÇÃO DE ROBÔS MÓVEIS UTILIZANDO APRENDIZADO POR REFORÇO E LÓGICA FUZZI. SBA: CONTROLE & AUTOMAÇÃO SOCIEDADE BRASILEIRA DE AUTOMATICA, 13(3), 219-230

[13] RUSSELL, S. J., & NORVIG, P. (2016). ARTIFICIAL INTELLIGENCE: A MODERN APPROACH. MALAYSIA; PEARSON EDUCATION LIMITED

[14] PERATO, L., & AL AGHA, K. (2002). HANDOVER PREDICTION: USER APPROACH VERSUS CELL APPROACH. IN 4TH INTERNATIONAL WORKSHOP ON MOBILE AND WIRELESS COMMUNICATIONS NETWORK (PP. 492-496). IEEE.

[15] SHIN, M., MISHRA, A., & ARBAUGH, W. A. (2004, JUNE). IMPROVING THE LATENCY OF 802.11 HAND-OFFS USING NEIGHBOR GRAPHS. IN PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON MOBILE SYSTEMS, APPLICATIONS, AND SERVICES (PP. 70-83). ACM.

[16] KULKARNI, A., MENEZES, S., VU, H., & ARACHCHIGE, C. L. (2008, JUNE). WIMAP: FAST HANDOVER FOR 802.11 MOBILE DEVICES. IN 2008 INTERNATIONAL SYMPOSIUM ON A WORLD OF WIRELESS, MOBILE AND MULTIMEDIA NETWORKS (PP. 1-4). IEEE.

[17] WHAT IS OMNET++?. DISPONÍVEL EM: <HTTPS://OMNETPP.ORG/INTRO/>. ACESSO EM: 13 JUN. 2019.

[18] SUTTON, R. S., & BARTO, A. G. (2018). REINFORCEMENT LEARNING: AN INTRODUCTION. MIT PRESS.

Cronograma 2019-1

	Março	Abril	Maio	Junho
1.	**X**	**X**	**X**	**X**
2.	**X**	**X**	**X**	**X**
3.		**X**	**X**	**X**
4.			**X**	**X**
5.				**X**

Legenda:

1. Revisão bibliográfica
2. Estudo do simulador
3. Desenvolvimento do ambiente
4. Definição do modelo
5. Aplicação do aprendizado por reforço