UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Airton Ribeiro de Moura Gomes Filho

Cache de Atributos Oportunista: Melhorando a eficiência do ABAC com o uso de uma política de distribuição de identidades em redes multinível para névoas computacionais

Juiz de Fora
2021

Airton Ribeiro de Moura Gomes Filho

Cache de Atributos Oportunista: Melhorando a eficiência do ABAC com o uso de uma política de distribuição de identidades em redes multinível para névoas computacionais

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Edelberto Franco Silva

Coorientador: Prof. Dr. Alex Borges Vieira

Juiz de Fora

2021

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Gomes Filho, Airton Ribeiro de Moura.
Cache de Atributos Oportunista: Melhorando a eficiência do ABAC com o uso de uma política de distribuição de identidades em redes multinível para névoas computacionais / Airton Ribeiro de Moura Gomes Filho. – 2021.
55 f. : il.

Orientador: Edelberto Franco Silva
Coorientador: Alex Borges Vieira
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2021.

1. Controle de acesso. 2. Controle de acesso baseado em atributos. 3. Cache de atributos. 4. Internet das coisas. I. Franco Silva, Edelberto, orient. II. Borges Vieira, Alex, coorient. III. Título.

Airton Ribeiro de Moura Gomes Filho

Cache de Atributos Oportunista: melhorando a eficiência do ABAC com o uso de uma política de distribuição de identidades em redes multinível para névoas computacionais

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 29 de novembro de 2021.

BANCA EXAMINADORA

Prof. Dr. Edelberto Franco Silva - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Alex Borges Vieira - Coorientador
Universidade Federal de Juiz de Fora

Prof. Dr. Luciano Jerez Chaves
Universidade Federal de Juiz de Fora

Prof. Dr. José Augusto Miranda Nacif
Universidade Federal de Viçosa

Juiz de Fora, 10/03/2022.

Documento assinado eletronicamente por **Alex Borges Vieira**, **Professor(a)**, em 10/03/2022, às 10:48, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **José Augusto Miranda Nacif**, **Usuário Externo**, em 10/03/2022, às 10:49, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Edelberto Franco Silva**, **Professor(a)**, em 10/03/2022, às 16:08, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luciano Jerez Chaves**, **Professor(a)**, em 10/03/2022, às 19:06, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **0705564** e o código CRC **08E34EC9**.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me dar força, coragem e sabedoria em todos os momentos da minha vida. Que o Pai Celestial continue me abençoando com Sua Graça ontem, hoje e sempre.

A toda minha família principalmente à Maria Adélia, minha mãe, que com seu amor incondicional, esforço e apoio sempre colaborou e incentivou para que todos os seus filhos pudessem estudar.

Agradeço a Taisy Santanna, minha esposa, que sempre me incentivou a seguir adiante, me escutando e ajudando nas decisões difíceis da vida. Agradeço a Malu, minha filha, que pela sua ingenuidade e alegria me faz achar energia para continuar seguindo.

Agradeço a todos os colegas que conheci nessa jornada, aos colegas do laboratório Netlab e aos colegas do curso que ajudaram nos estudos, trabalhos e aos nossos momentos de descontração. Em especial aos amigos que levarei para a vida toda: Mayara Amanda, Genilson Israel, Frederico Sales, Jorge Rafael, Pedro Henrique e Eduardo Sousa.

À Universidade Federal de Juiz de Fora pela estrutura e pelo ensino de qualidade e principalmente aos professores do Programa de Pós Graduação em Ciência da Computação.

Aos professores e pesquisadores do grupo de pesquisa CUIDA - WBAN, Professor Doutor José Augusto Nacif - UFV, Professora Doutora Michele Nougueira - UFMG e ao Doutorando Bruno Cremonezi - UFPR. A ajuda e conhecimentos compartilhados por vocês foi fundamental para o desenvolvimento desta pesquisa.

Ao meu orientador, Professor Doutor Edelberto Franco Silva, por dispor do seu tempo para me orientar na produção deste trabalho e me orientar como aluno. Ao meu coorientador, Professor Doutor Alex Borges Vieira. Agradeço a vocês pela oportunidade e confiança.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, pelo apoio financeiro sem o qual este trabalho não poderia ser realizado.

RESUMO

O Controle de Acesso Baseado em Atributos (*Attribute-based Access Control* - ABAC) é um dos métodos de controle de acesso mais populares. Apesar de sua popularidade, apenas alguns trabalhos abordam o gerenciamento de atributos na Internet das Coisas (*Internet of Health Things* - IoT). A maioria dos atributos necessários para uma avaliação de política em IoT vem de uma fonte externa. Portanto, o gerenciamento de atributos através da rede requer comunicação entre o ponto de decisão da política e o ponto de informação da política para cada atributo, impactando o desempenho do ABAC. Os *caches* de atributos podem atenuar esse problema. No entanto, devido à natureza dinâmica dos atributos, o custo para manter *caches* atualizados aumenta para cada nova réplica. Este trabalho apresenta um método que prevê solicitações de atributo e antecipa o posicionamento do atributo mais próximo do solicitante, equilibrando o custo de criação de uma nova réplica dando benefícios para o desempenho do ABAC. O método lida com a solicitação de atributo atual e prevê onde ocorrerá a solicitação subsequente. Através de simulações com uma base de dados real, o método proposto reduziu acima de 80% o número de requisições na nuvem utilizando os atributos nos *caches* e entrega até 55% dos atributos no primeiro salto.

Palavras-chave: Controle de Acesso, Controle de acesso baseado em atributos, Cache de atributos, Internet das coisas.

ABSTRACT

Attribute-based Access Control (ABAC) is one of the most popular access control methods. Despite its popularity, a few works address attribute management in the Internet of Things (IoT). Most of the attributes needed for an IoT policy evaluation come from an external source. Therefore, managing attributes across the network requires communication between the policy decision point and the policy information point for each attribute, impacting ABAC performance. Attribute caches can mitigate this problem; however, due to the dynamic nature of the attributes, the cost of keeping caches up to date increases for each new replica. This work presents a method that predicts attribute requests and anticipates the attribute placement closer to the requester, balancing the cost of creating a new replica and giving ABAC performance benefits. The method handles the current attribute request and predicts where the subsequent request will occur. Based on simulations with a real dataset, the proposed method reduces above 80% the number of requests in the cloud using attributes' caches and delivers up to 55% of the attributes in the first hop.

Keywords: Access Control, Attribute-based Access Control, Attributes Cache, Internet of Things.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

A&A	Autenticação e a Autorização
ABAC	*Attribute-based Access Control*
APs	*Access Points*
CC	*Cloud Computing*
DCs	*Data Centers*
EAP	*Extensible Authentication Protocol*
FC	*Fog Computing*
FIFO	*First-In, First-Out*
FNs	*Fog Nodes*
GId	Gestão de Identidade
HR	*Hit Ratio*
IaaS	*Infrastructure as a Service*
IAM	*Identity and Access Management*
IdM	*Identity Management*
IdPs	*Identity Providers*
IoHT	*Internet of Health Things*
IoT	*Internet of Things*
kNN	*k-Nearest Neighbors*
LCD	*Leave Copy Down*
LCE	*Leave Copy Everywhere*
LEAF	*Leave a Copy at LEAF*
LGPD	Lei Geral de Proteção de Dados
LRU	*Least Recently Used*
OHHR	*One Hop Hit Ratio*
PaaS	*Platform as a Service*
PAP	*Policy Administration Point*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
PIP	*Policy Information Point*
RBAC	*Role-based Access Control*
RR	*Random Replacement*
SaaS	*Software as a Service*
SLRU	*Segmented Least Recently Used*
SPROT	Segmentado com Predição no espaço Protegido
SPs	*Service Providers*
UAVs	*Unmanned Aerial Vehicles*
UFJF	Universidade Federal de Juiz de Fora
VANETs	*Vehicular Ad Hoc Network*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*

WPA	*Wi-Fi Protected Access*
XACML	*Extensible Access Control Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Desde os primeiros registros pela busca por uma conexão ubíqua e pervasiva, são propostas soluções que auxiliem na transparência e abrangência de conectividade, como pode ser comprovado em (32). Um exemplo de tecnologia de sucesso que auxilia na cobertura e expansão das redes, são as redes móveis e sem fio. E um dos exemplos de maior abrangência e adoção são as redes sem fio locais (*Wireless Local Area Network* - WLAN, e *Wireless Fidelity* - Wi-Fi da *Wi-Fi Alliance*) apoiadas no padrão IEEE 802.11, provendo larga escala, além de cobertura de espaços físicos, oferta de banda larga e acesso a diversos serviços da Internet. Porém, é importante considerar como o acesso a esse tipo de rede e a seus serviços são garantidos sob a perspectiva da segurança da informação, *i.e.,* confidencialidade, integridade, disponibilidade, autenticidade e não repúdio. Assim, para garantir um acesso mais seguro a tais serviços e ambientes, deve-se levar em consideração requisitos como a Autenticação e a Autorização (A&A) (31). Conforme (29), os processos relacionados à A&A estão intimamente ligados à Gestão de Identidade (GId ou IdM - *Identity Management*), e são utilizados para garantir o uso seguro de recursos arbitrários e definir os ciclos de vida de uma identidade e a composição delas, com seus atributos. Da mesma forma, é comum encontrarmos o termo IAM (*Identity and Access Management*) para definir os mesmos conceitos.

Para que a A&A possa ser realizada em ambientes de larga escala sem fio é necessário considerar alguns pontos cruciais. Em relação à autenticação e autorização, usualmente, cria-se políticas de acesso que consideram dados relacionados à identidade do objeto, do ambiente e, principalmente, da identidade do usuário para determinar se um acesso é ou não válido (26). Este mecanismo é conhecido como parte da A&A provendo o suporte a métodos de controle de acesso ao ambiente. Porém, devido ao possível atraso na recuperação das identidades e seus atributos, os atuais processos de A&A podem não ser capazes de proporcionar a sensação de transparência no acesso a serviços pelo usuário, necessário para diversas aplicações em redes móveis (17). Logo, é de grande importância avaliar e propor técnicas que reduzam o tempo de recuperação dessas identidades a fim de que o processo de A&A ocorra eficientemente. Por exemplo, redução da latência na autenticação em ambientes como Internet das Coisas (*Internet of Things* - IoT), Internet das Coisas Aplicadas na Saúde (*Internet of Health Things* - IoHT), Rede Veicular *Ad Hoc* (*Vehicular Ad Hoc Network* - VANETs), Veículos Aéreos Não Tripulados (*Unmanned Aerial Vehicles* - UAVs), entre outros, para que a entrega da identidade e os atributos que a compõem tão logo sejam recuperados quando requisitados. Neste caso, o requisito temporal pode ser crucial para o funcionamento ou não de um serviço que utilize A&A através de redes móveis (9).

Neste trabalho, apresentaremos a possibilidade da aplicação de políticas de *cache* de identidade sob o conceito de névoa computacional, a fim de avaliar o ganho em relação à redução de número de saltos, e consequentemente, redução da latência na recuperação da identidade. Tal redução de latência representa um melhor desempenho para respostas em tomadas de decisões em processos de A&A como um todo, como comentado anteriormente. A fim de mostrar a aplicabilidade da proposta, sugerimos uma nova política de *cache* de identidade baseada na predição de mobilidade do usuário sob uma rede sem fio de larga escala. Através de técnicas de inteligência computacional, propomos e validamos uma política de *cache* de identidades multinível para névoas computacionais, além de compará-la com o estado da arte em relação às políticas de replicação e substituição em *cache*. O ganho da política proposta em relação ao estado da arte é de 2,9%, ficando apenas à 6,4% da solução ótima. O ambiente de experimentação utiliza dados de uma rede sem fio real de uma grande universidade, com mais de 36.000 usuários e reforça a validade da proposta.

1.1 PROBLEMA

Atualmente há atraso no processo de autenticação e autorização, dado que os objetos necessários para identificação dos usuários estão distantes dos pontos de acesso. Em geral, esses dados estão na nuvem computacional, o que inviabiliza seu uso, por exemplo, no controle de acesso em ambientes dinâmicos e com requisitos temporais de recuperação dos atributos.

1.2 OBJETIVO

Como objetivo, propomos explorar a computação em névoa para diminuir a latência dos processos de autenticação e autorização através do uso de cache de atributos das identidades. A arquitetura em névoa utilizará de políticas de replicações e substituições para distribuir os atributos reativamente e proativamente. Assim, propomos adaptar o modelo de controle de acesso para permitir que aplicações com requisitos temporais utilizem os benefícios dos sistemas de Gestão de Identidades em ambientes de rede sem fio.

1.3 CONTRIBUIÇÕES

Como contribuições, temos a proposta e validação de uma arquitetura para ambiente de névoa computacional com políticas de replicação e substituição de identidade validada sob dados coletados de uma rede real de uma universidade (*i.e.*, Universidade Federal de Juiz de Fora - UFJF), com mais de 36.000 (trinta e seis mil) usuários únicos e 108 APs. Uma outra contribuição do nosso trabalho é a caracterização dos acessos à rede sem fio acadêmica por esses usuários, e a identificação dos seus padrões de mobilidade. Assim, são

avaliadas, além das políticas clássicas de *cache* encontradas no estado da arte, o impacto do número de saltos entre o AP associado pelo cliente até o servidor de identidades na nuvem, considerando tais saltos no incremento da latência de recuperação de uma identidade. Por fim, propomos e validamos uma política de replicação oportunista e proativa com utilização de aprendizado de máquina, de modo a considerar as características de mobilidade dos usuários para tomada de decisão na replicação e substituição de identidades e atributos. Além disso, desenvolvemos um oráculo com o maior percentual de acerto possível para *cache* de objetos de identidade dos usuários a fim de verificar o quão distante do melhor caso de replicação e substituição de atributos nos APs a política proposta se encontra.

1.4 ORGANIZAÇÃO DO TEXTO

O restante deste trabalho está organizado da seguinte forma: no Capítulo 2 os fundamentos teóricos, no Capítulo 3 a discussão dos trabalhos relacionados, no Capítulo 4 uma caracterização dos dados utilizados nesse trabalho, no Capítulo 5 a descrição do método de *cache* de atributos oportunista proposto, no Capítulo 6 a metodologia de avaliação da pesquisa, no Capítulo 7 a avaliação e apresentação dos resultados e no Capítulo 8 a conclusão.

2 FUNDAMENTOS

Neste capítulo são apresentados os principais conceitos relacionados à GId no contexto deste trabalho. Além disso, como as *caches* de objetos podem ser utilizadas como auxílio à GId no contexto da arquitetura de névoas e nuvens computacionais. Portanto, temos primeiramente uma seção sobre GId, seguida por nuvem e névoa computacionais, e por fim a gerência de *cache* de objetos no contexto estudado.

2.1 GESTÃO DE IDENTIDADES

Conforme (19), a GId pode ser entendida como o conjunto de processos e tecnologias usados para garantir a identidade de uma entidade ou de um objeto, garantir a qualidade das informações de uma identidade (identificadores, credenciais e atributos) e para prover procedimentos de autenticação, autorização, contabilização e auditoria.

Uma única entidade pode ter vários atributos de identidade e pode ser solicitada em diferentes locais. Portanto, é responsabilidade do GId fornecer recursos que permitam a troca segura de informações entre entidades, como partes confiáveis e Provedores de Identidades (*Identity Providers* - IdPs) e essa troca de informações é baseada em políticas estabelecidas e na confiança estabelecida entre essas entidades em um ambiente de múltiplos Provedores de Serviços (*Service Providers* - SPs) (19).

2.1.1 Identidades, Atributos e Políticas

Em GId, as identidades representam as provas e credenciais de uma entidade fornecidas aos aplicativos e serviços em seus com contextos específicos (6). Tanto as informações da entidade como o seu contexto podem ser utilizados para distinguir tais entidades umas das outras, fornecendo ou revogando privilégios. Entidades solicitam acesso a objetos, sendo que uma entidade pode ser também chamada de usuário, que por sua vez pode representar uma pessoa humana, um serviço, um aplicativo e até mesmo um dispositivo de rede. Independentemente da natureza do usuário em questão, esse possui um conjunto de credenciais e atributos descritivos. Tais atributos variam entre um conjunto discreto de valores, e são exemplos, a data de nascimento, estado civil, papel que assume no ambiente (*e.g.*, gerente, coordenador). Já as políticas definem quais atributos com seus respectivos valores um usuário deve possuir para acessar um certo objeto.

Alguns dos sistemas que auxiliam a GId são os de autenticação, autorização, controle de acesso e auditoria. Para a autenticação, atributos da identidade são utilizados para auxiliar na tomada de decisão, já na autorização e controle de acesso, tais atributos podem ser utilizados para permitir ou não acesso a certos recursos. Por fim, a auditoria registra cada ações que ocorre desde a autenticação até a utilização do sistema, passando

pela autorização, a fim de permitir análises posteriores no uso dos sistemas. A seção a seguir dá mais detalhes sobre autenticação e autorização.

2.1.2 Autenticação e Autorização

O processo de A&A para usuários, em diversos contextos, só é possível a partir de dados que identificam o usuário. Por exemplo, para que um usuário seja identificado e avaliada sua permissão a um dado serviço de rede, é necessário que ele apresente suas credenciais. Após a aplicação do método de autenticação associado (*e.g.*, um par usuário-senha, um certificado digital), é realizado o processo de autorização.

Diversos métodos de autorização associados ao controle de acesso são propostos na literatura, dentre os mais reelvantes destacam-se o Controle de Acesso Baseado em Papel (*Role-based Access Control* - RBAC) e o Controle de Acesso Baseado em Atributos (*Attribute-based Access Control* - ABAC) (29). Porém, só é possível executar os métodos citados a partir da identidade – ou o conjunto de atributos – associados ao usuário. Outro ponto relevante é a origem da identidade e os atributos associados a este usuário. Tradicionalmente, esses dados estão armazenados em uma árvore de diretório localizada, por exemplo, na instituição de origem do usuário, os IdPs, que por sua vez, se encontra, em geral, em servidor distante fisicamente do usuário. Isso faz com que a cada solicitação dos atributos do usuário para a realização da A&A naquele SP seja necessário consultar esse diretório remoto, o que pode ser custoso e limitar a aplicação a alguns cenários (17).

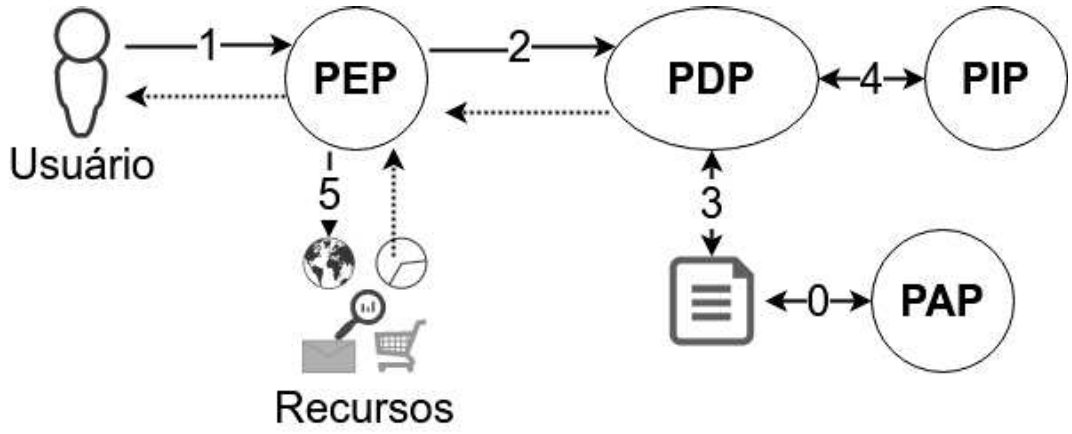
O RBAC é um controle de acesso relativamente simples, que segue a utilização de atributos do usuário de forma a associa-lo a um papel ou outro. Um exemplo clássico de controle de acesso baseado em papéis é a visualização de um sistema baseado na *web* por um administrador ou usuário comum. No ambiente de redes e serviços, podemos exemplificar o RBAC para um acesso mais, ou menos, limitado a serviços de rede (*e.g.*, servidor de arquivos, sistemas) conforme os atributos da identidade do usuário.

Se comparado ao RBAC, o ABAC é considerado um método mais escalável e granular para a realização do controle de acesso. Isso se deve, principalmente, pelo fato de atributos poderem ser tratados de maneira mais granular e de políticas de controle de acesso serem suportadas de maneira distribuída e dinâmica, como podemos comprovar em (29, 17). O ABAC, geralmente, controla suas políticas por meio da Linguagem de Marcação de Controle de Acesso eXtensível (*Extensible Access Control Markup Language* - XACML). XACML é um modelo que oferece mecanismos, arquitetura e linguagem, para avaliar as solicitações de requisições de recursos por meio de políticas de acesso previamente definidas (2). XACML foi originalmente criado para o ABAC, mas o RBAC também pode utilizar essa linguagem, obviamente, limitando claramente os potenciais do XACML para um acesso mais direto baseado apenas em papéis.

A Figura 1 mostra o diagrama da implementação da arquitetura XACML com as

direções com que as mensagens são trocadas entre as entidades. As entidades são: Ponto de aplicação da política (*Policy Enforcement Point* - PEP), Ponto de decisão da política (*Policy Decision Point* - PDP), Ponto de Informação da Política (*Policy Information Point* - PIP) e Ponto de Administração de Política (*Policy Administration Point* - PAP). Em (1) um usuário tenta acessar um recurso protegido. Em (2) o PEP intercepta a solicitação e envia ao PDP para verificar se o usuário está autorizado. Em (3) o PDP avalia as políticas de acesso que foram anteriormente criadas pelo PAP e implantadas no PDP em (0). Em (4) o PDP recupera atributos relacionados à solicitação no PIP. Em (5) com base na política e nos atributos, o usuário recebe acesso ao recurso protegido.

Figura 1 – Arquitetura XACML.



Fonte: Adaptado de (2).

2.2 ARQUITETURAS COMPUTACIONAIS

Com a intenção de apresentar as arquiteturas computacionais envolvidas neste trabalho, ou seja, àquelas distribuídas e que deram insumos à possibilidade de uma busca e recuperação de objetos para A&A mais próxima ao usuário, temos esta seção. Além disso, é necessário introduzir conceitos que motivam a nuvem e névoas computacionais, como a IoT e a IoHT.

Como sabemos, a IoT e a IoHT, para que sejam úteis aos seus usuários e aplicáveis no ambiente real, necessitam de um número relevante de serviços inovadores e dados coletados. Porém, em geral, esses dispositivos não apresentam recursos suficientes para hospedar diretamente tais serviços (25). Desta forma, a grande quantidade de dados coletados por esses dispositivos precisam ser armazenados e processados em ambientes mais distantes, que possuem mais recursos computacionais para realizar todas as tarefas envolvidas, como: obtenção, recuperação, visualização ou atuação.

2.2.1 Nuvem Computacional

Como já comentado, a IoT precisa de suporte a recursos poderosos de computação para seus serviços de aplicações sejam úteis, e o mais comum é que esses recursos estejam no que conhecemos como Computação na Nuvem, ou (*Cloud Computing* - CC) (5). É importante notar que as nuvens podem ser públicas, privadas ou uma combinação híbrida de ambas (3). A computação em nuvem é um modelo para fornecer virtualização de recursos (por exemplo, redes, servidores, armazenamento e serviços) com alta flexibilidade, eficiência de custos e gerenciamento centralizado (21). Os modelos de serviço de computação em nuvem são geralmente categorizados em Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS), Plataforma como Serviço (*Platform as a Service* - PaaS) e Software como Serviço (*Software as a Service* - SaaS). Esses serviços oferecem, respectivamente, recursos virtuais (computação, armazenamento e rede), *software* e plataformas de desenvolvimento (fornecidas pela infraestrutura em nuvem) e aplicativos baseados na Internet (hospedados na nuvem) (3). No contexto deste trabalho, a CC é onde o IdP estará, ou seja, a n saltos distantes do usuário final que dependerá dele para realizar sua autenticação, e consequentemente, receber seus atributos relacionados à sua identidade para a autorização.

2.2.2 Névoa Computacional

Como os recursos de Computação na Nuvem, em geral, estão concentrados em poucos Centros de Dados (*Data Centers* - DCs), que estão consideravelmente distantes da grande maioria dos produtores e consumidores de dados, surgem conceitos de arquiteturas que visam aproximar os dados das bordas da rede. Essa premissa é especialmente verdadeira para Nuvens públicas. Tal distância pode ser contabilizada em número de saltos e enlaces, aumentando o atraso, e perdas de pacotes, por exemplo. Sendo assim, uma característica não desprezível para aplicações nos dispositivos finais. Em (4) foi proposto o paradigma de Computação na Névoa (*Fog Computing* - FC) (4), a fim de estender os recursos e capacidades baseadas em nuvem para a borda da rede. Dessa forma distribui-se os recursos e serviços de computação, armazenamento e rede ao longo da Nuvem-para-Coisas (*Cloud-to-Things*), aumentando a proximidade topológica dos dados e recursos em relação aos dispositivos IoT.

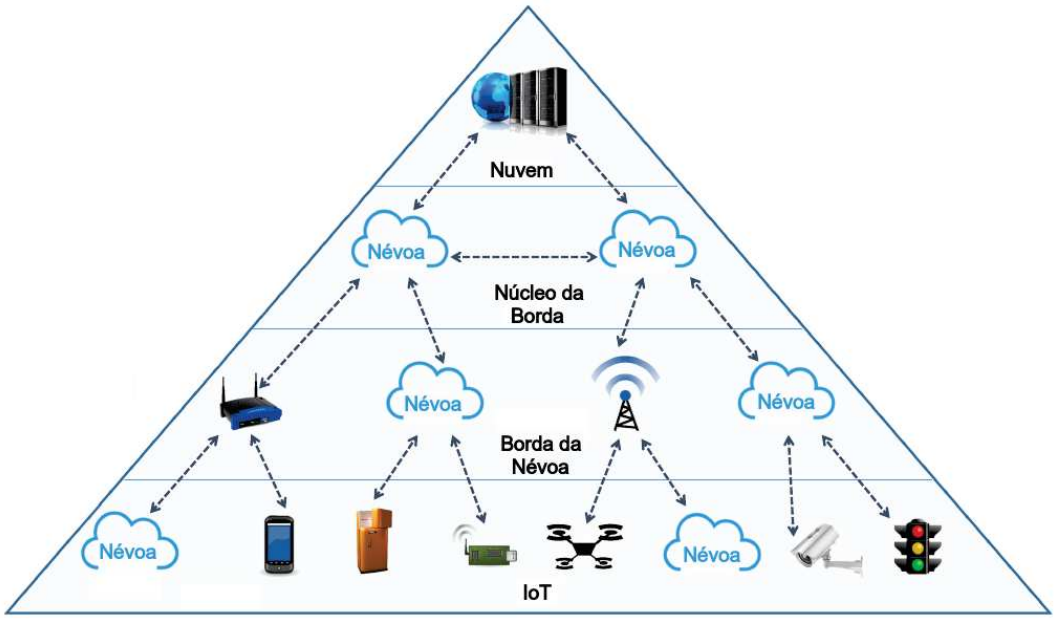
Mesmo adotando a *Fog Computing*, os principais benefícios do *Cloud Computing* devem ser preservados, incluindo virtualização de recursos, transparência e elasticidade (12). Além disso, quanto aos recursos e serviços da nuvem, também os de névoas podem ser fornecidos gratuitamente ou mediante pagamento. Por exemplo, um município pode explorar parte de seus próprios recursos da sua névoa gratuitamente e conceder, mediante pagamento, o restante a desenvolvedores terceirizados (25).

A névoa é um paradigma de nível de sistema no sentido de que se estende pelas bordas da rede, pela nuvem e por várias camadas de protocolos, não apenas em uma parte

de um sistema ponta a ponta, mas é um todo que se estende entre as coisas e a nuvem (25). Consequentemente, o FC promove o desenvolvimento de sistemas onde o serviço geral geralmente não é fornecido por um único computador rico em recursos. Em vez disso, o serviço é normalmente decomposto e fornecido por uma hierarquia de nós de névoa (*Fog Nodes* - FNs) de modo que cada um deles execute uma parte específica do serviço geral, enquanto coopera com os outros FNs (25). Essa organização em forma de pirâmide é um dos princípios orientadores da Arquitetura de Referência de (12). No entanto, conforme declarado por (12), a hierarquia computacional e do sistema não é necessária para todas as Arquiteturas de Névoa, mas ainda é expresso na maioria das implantações.

Conforme pode ser visto na Figura 2, a camada mais baixa na hierarquia compreende as coisas e os dispositivos finais, em geral, que podem se comportar como FNs se forem poderosos o suficiente. As camadas superiores vão da borda da rede até o núcleo, e seu número de saltos e composição dependem do domínio, do cenário e propósito da aplicação (12). Finalmente, a camada superior pode ser representada pela nuvem. De fato, a névoa não substitui a nuvem, mas normalmente coexiste e coopera com ela, já que muitos serviços exigem as características da névoa e da nuvem (4). As interações em tais sistemas hierárquicos podem ser de qualquer tipo, tanto dentro da mesma camada quanto entre nós pertencentes a camadas diferentes (12). Cada nó dá sua própria contribuição para o serviço geral, e a natureza de sua função depende muito de sua posição na pirâmide (25).

Figura 2 – Hierarquia Organizacional da Computação em Névoa.



Fonte: Traduzido de (25).

2.3 CACHE DE OBJETOS

A fim de auxiliar que a névoa e a nuvem se comuniquem e de colaborar com os dispositivos em uma recuperação mais rápida da informação, e possibilidade de aplicação de métodos de controle de acesso, introduzimos conceitos relacionados à *cache* de objetos. *Cache* de objetos, de forma geral, tem seu conceito envolvo no suporte à recuperação da informação de uma forma mais rápida. Isso é possível através da manutenção de cópias temporárias de objetos, geralmente àqueles consultados com maior frequência. Grandes espaços de *caches* são necessários em sistemas que demandam alto desempenho, esses espaços em *cache* oferecem alta largura de banda das memórias em sistemas operacionais (1). Além de que o sistema deve ser caracterizado com precisão para encontrar os melhores tamanhos de *caches*, visto que uma pequena diminuição no desempenho da *cache* pode resultar em degradação significativa do desempenho do sistema (1).

Já as políticas de distribuições são formas lógicas de difundir os objetos na *cache*. Há duas etapas básicas no processo de disseminação e manutenção dos objetos em uma *cache* hierárquica e distribuída, são elas, a replicação e a substituição. As replicações são feitas ao finalizar uma requisição, passando pelos diversos nós no caminho de uma árvore, *e.g.*, comutadores de rede, por exemplo. Já as substituições são feitas sempre que o espaço de armazenamento interno a esses dispositivos alcançam o seu limite físico. Sendo assim, apresentamos nas próximas seções as políticas de replicação e substituição utilizadas neste trabalho.

2.3.1 Políticas de Replicação

Para cada requisição feita ao servidor de identidades, uma política de replicação é utilizada para distribuir as identidades no sistema e seus elementos de rede. Neste trabalho foram avaliadas as políticas de distribuições Deixe uma cópia em todos os lugares (*Leave Copy Everywhere* - LCE), Deixe uma cópia para baixo (*Leave Copy Down* - LCD) e Copiar na Folha (*Leave a Copy at LEAF* - LEAF).

Na política LCE os objetos são replicados por todo o caminho entre o dispositivo até a nuvem (20). Dessa forma, distribuindo as identidades rapidamente até a névoa. Esta política é considerada de alto custo, por gerar muitos dados no caminho da rede, adicionar redundâncias e, por sua vez, ocupar rapidamente o espaço em *cache* disponível nos dispositivos intermediários.

A política LCD deixa uma cópia do objeto requisitado apenas no nível logo abaixo de onde foi encontrado (20). Considerada uma abordagem conservadora, permite que os objetos mais requisitados se aproximem da borda da rede.

Por fim, na política LEAF, ou replicação em folhas, é sugerido que apenas as folhas da árvore de topologia da rede construída armazenem dados em *cache*, ou seja, somente

elementos comutadores de rede que estão na névoa podem ter armazenamento. *e.g.*, *switches* ou APs (10). Essa política de replicação atende a cenários onde os dispositivos da névoa não têm grande capacidade de armazenamento, como em ambientes com *switches* legados ou de prateleira, e apenas as folhas teriam tal capacidade para manter o armazenamento de objetos em *cache*.

2.3.2 Políticas de Substituição

Por sua vez, as políticas de substituições são aplicadas quando o espaço em *cache* está totalmente ocupado com objetos já utilizados, mas novos objetos precisam ser inseridos. Assim, os objetos existentes em *cache* devem ser substituídos, e tais políticas são aplicadas para essa tomada de decisão. Neste trabalho, foram utilizadas as políticas Menos usado recentemente (*Least Recently Used* - LRU), Primeiro a entrar Primeiro a sair (*First-In, First-Out* - FIFO), Substituição aleatória (*Random Replacement* - RR), Baseada em Acertos *Hit*, Baseada em Tempo, e a Segmentado menos usado recentemente (*Segmented Least Recently Used* - SLRU).

O entendimento das políticas de substituição é simples, e portanto, apresentamos apenas uma visão objetiva da funcionalidade de cada uma delas. A primeira dela, é a política LRU, que descarta os objetos menos utilizados primeiro (20). Para tanto, seu algoritmo organiza uma lista e adiciona o novo objeto na primeira posição. Objetos que são utilizados e já estavam na lista são colocados na primeira posição da fila. Quando a lista está cheia é removido o último objeto.

Já a política FIFO é um clássico e obedece a mesma lógica de um algoritmo de fila tradicional, onde todos os objetos entram na última posição da fila, e quando a fila se torna cheia, o primeiro objeto é descartado (23).

Para a política RR os objetos são descartados de forma randômica. Esse algoritmo usa o menor processamento para realizar a remoção dos objetos quando o espaço de armazenamento fica cheio, uma vez que não há necessidade de manter nenhum estado associado aos objetos (23).

Por sua vez, a política baseada em *Hit* (frequência de acerto) faz uma analogia ao *hit* e *miss* de políticas clássicas de *cache* de objetos. Essa política utiliza uma estratégia de persistente para a contabilidade de acessos realizados aos objetos. Ou seja, aqueles objetos que receberam maior número de acertos (*hit*) em requisições recentes devem demandar acessos futuros, e devem ser preservados na *cache*. Assim, esses são considerados objetos que devem ser mantidos no cache, ainda quando o espaço de armazenamento está em seu limite. Então àqueles objetos com maior número de acesso e que existem na *cache* sempre se manterão nela, e outros com menor quantidades de acertos são removidos (24).

Outras políticas distintas e mais complexas que as já apresentadas foram também analisadas. Primeiro temos a política baseada em tempo. Essa política é implementada

com a variável de referência de tempo, que pode representar o tempo de utilização, tempo entre reutilização ou qualquer outro atributo temporal (23). Objetos com menores tempos podem ser escolhidos para serem mantidos ou excluídos da *cache* a depender da lógica necessária. Os objetos são armazenados na *cache* e quando a *cache* está cheia é analisado o objeto que tem o maior tempo de reconexão, e esse objeto é removido.

Por fim, a SLRU é uma variação da LRU, descartando os objetos menos utilizados primeiro. Diferentemente da política *Hit*, nessa política o espaço da *cache* é dividido em dois segmentos. Um deles é um segmento protegido e o outro é um segmento de oportunidade (24). Todos os novos objetos são adicionados no segmento de oportunidade, colocando novos objetos no início desse espaço. Quando o espaço de armazenamento está cheio, remove-se o último objeto do segmento oportunidade. O algoritmo do SLRU se diferencia também da política LRU considerando que quando há um acerto de acesso a algum objeto a sua posição é atualizada para o início do segmento protegido. Quando esse espaço protegido está cheio, o objeto menos utilizado é enviado ao espaço oportunidade. Com essa política os objetos que foram reutilizados na *cache* recebem uma oportunidade extra de serem (re)utilizados, uma vez que por estarem em um espaço reservado ganham mais tempo de vida antes de serem eliminados/substituídos na *cache*. O tamanho entre os segmentos de oportunidade e reservado podem variar, por exemplo, em 25% para o espaço reservado e 75% para o espaço oportunidade. Porém, o ideal é que o tamanho entre esses segmentos atenda aos padrões de comportamento do uso da *cache* no ambiente ao qual a política está sendo aplicada.

3 TRABALHOS RELACIONADOS

O problema da ineficiência dos métodos A&A para recuperação das identidades foi primeiramente apresentado por (17), ao relatar que o ABAC pode perder desempenho quando da recuperação dos objetos referentes a essa identidade, como no caso dos atributos. Em seu trabalho, (17) descreve os problemas de desempenho do ABAC, e mostra que o *cache* de atributos pode aliviar esses problemas. No entanto, espalhar atributos em *caches* não é uma abordagem direta, necessitam de investigações e propostas que atendam aos requisitos do próprio padrão ABAC, mas também das aplicações e serviços que pretendem utilizá-lo. No mesmo trabalho os autores também descrevem a compensação em relação à atualização dos atributos e seu impacto na segurança do sistema. A atualização de atributos com menor frequência diminui a segurança do sistema, se comparado com a atualização de atributos em tempo real, embora haja um custo associado para mantê-los atualizados.

Considerando que na IoT utiliza-se um grande número de dispositivos e usuários associados, cada um com seu próprio conjunto de atributos, é necessário avaliar o número de atributos que devem ser armazenados em *cache*, a fim de manter essa relação de custo de atualização e benefício ao ambiente balanceado. Ainda no trabalho de (17), os autores argumentaram que quando um número mínimo de atributos é armazenado em *cache*, ocorre uma simplificação da atualização e, consequentemente, incremento sob a ótica da segurança. Isso torna o controle de acesso viável para diversos casos de uso de IoT, principalmente onde o desempenho é um requisito obrigatório. Desde então, outros trabalhos possuem como alvo essa ineficiência na recuperação das identidades, como em um dos primeiros trabalhos na área, apresentado por (16), onde sugere-se o uso de *caches* de identidades como forma de se alcançar o desafio apresentado neste trabalho. Este capítulo apresenta os trabalhos relacionados, e ainda complementa com uma tabela de comparação entre o estado da arte com relação à proposta a ser descrita neste trabalho.

Em (22) foi apresentado um mecanismo de *caching* multinível baseado em análises estatísticas de um *campus* universitário. Os autores propuseram um sistema de três camadas de A&A, sendo as camadas: de Sistema de Autenticação de Usuários, Sistemas de Controle de Acesso e Função de Controle de Acesso. Nele, dois mecanismos de *cache* foram desenvolvidos para otimizar o tempo de recuperação das identidades. Ainda que eficiente em se realizar o que se propõe, esse trabalho está limitado para o modelo de autorização RBAC, que é definido como um modelo já datado e que não oferece uma autorização granular.

Em (11) se propôs uma arquitetura névoa-nuvem, mostrando seus benefícios para os métodos de controle de acesso. Para reduzir o tempo de recuperação, os autores fragmentam os atributos em dados minúsculos e usam os nós de névoa como dispositivos

agregados para aproximar os atributos do ponto onde as políticas são avaliadas e os atributos são necessários. Seus resultados mostraram que o uso de nós de névoa para armazenar os atributos impacta positivamente no desempenho do método de controle de acesso. No entanto, os autores concluíram que os atributos devem ser colocados em pontos estratégicos da rede para atenderem as diversas solicitações de atributos.

No trabalho de (7) são apresentadas modificações necessárias em uma arquitetura IoT para agilizar a avaliação de políticas de controle de acesso. Os autores propõem um sistema de controle de acesso híbrido, que avalia parcialmente as políticas nos dispositivos IoT, além de um mecanismo de *cache* de políticas. Eles argumentaram que os *gateways* estão presentes na maioria das soluções de IoT para mediar a comunicação entre o usuário e seus aplicativos que possuem vários objetos inteligentes. Portanto, os *gateways* podem armazenar chaves de acessos e processar políticas de acesso devido ao seu poder computacional e de memória. No entanto, embora o trabalho tenha se concentrado no armazenamento em *cache* e na distribuição de políticas, os autores não discutiram ou investigaram o custo relacionado ao armazenamento e recuperação dos atributos necessários para aplicação das políticas.

No trabalho de (28) é apresentada uma nova arquitetura para A&A com simplificação do processo de autorização. Os autores propõem proteger os dados dentro do domínio da rede de uma forma mais eficiente, oferecendo mais controle sobre o acesso aos dados. O ganho real da arquitetura pode ser visto como a possibilidade de mudar as tecnologias dentro do domínio sem precisar reescrever toda a lógica de A&A. Além disso, o domínio pode usar atributos aos quais são controlados e mantidos por outros sistemas, sem precisar conhecer a implementação de terceiros. Em nosso trabalho não foi proposto uma mudança na arquitetura como no (28), porém foram utilizados conceitos parecidos ao propor que a entidade PDP e PIP, do XACML, atuem dentro dos nós da rede.

Em (8) propõe o uso de armazenamento nos nós da rede usando identidades como serviços. Assim, os objetos são persistidos por meio de políticas de *cache*. Foi avaliado o impacto na distribuição em diferentes capacidades de armazenamento e comprovou-se que algoritmos probabilísticos possuem, em geral, um desempenho melhor que os demais. É importante destacar que o trabalho de (8) deu origem a pesquisa aqui apresentada, onde identificou-se a necessidade de avaliar com maior profundidade algoritmos de replicações e substituições. Ao longo do texto será possível identificar diversos avanços desta pesquisa com relação à apresentada por (8), desde a utilização de dados reais frente à entrada simulada pela distribuição Zipf à época, quanto a novas políticas de replicação e substituição.

Como forma de deixar mais clara as contribuições deste trabalho em relação ao estado da arte, apresentamos a tabela a seguir. A Tabela 1 sintetiza uma comparação entre os trabalhos relacionados e a pesquisa desenvolvida. Todos os trabalhos são baseados

em redes hierárquicas, exceto em (17) e em (16), onde não há requisito de uma estrutura de rede para ser implementado. Todos os trabalhos sugerem, ou usam, algum sistema de persistência de dados em formato de *cache*. A maioria dos trabalhos usa o ABAC como método de controle de acesso. Alguns trabalhos relacionados não realizaram avaliações com experimentações, apresentando apenas uma visão teórica sobre as *caches* de atributos. Os trabalhos que avaliaram suas propostas com auxílio de experimentos, ou usaram a base de dados simulada, ou não foi informada a origem dos dados. Desta forma, essa pesquisa se difere das demais ao realizar comparações com vários métodos de substituições e por utilizar uma base de dados com comportamento real utilizando *logs* de acesso de controladoras sem fio de um ambiente institucional com mais de 36.000 usuários.

Tabela 1 – Comparação dos trabalhos relacionados.

Trabalho	**Rede Hierárquica**	**Cache**	**Método de Controle de Acesso**	**Políticas Distribuições e Substituições**	**Avaliação**	**Base de dados**
(17)	Não informado	Sugere	ABAC	-	-	-
(16)	Não informado	Sugere	ABAC	-	-	-
(22)	Sim	Políticas de Grupos e de Papel	RBAC	Baseado em Grupos e Papel	-	-
(11)	Sim	Sim, fragmentos das IDs	ABAC	Não informado	Sim, simulador	Sem informação se real ou simulado
(7)	Sim	Sim, contexto dos usuários	ABAC	Não informado	Sim, testbed	Simulado
(28)	Sim	Sim, IDs autorizados	ABAC	Não informado	-	-
(8)	Sim	Sim, IDs autorizados	-	LRU, FIFO, RR, LCE, LCD, Probabilístico 70% e 30%	Sim, simulador	Simulado
Esta Pesquisa	Sim	Sim, IDs autorizados	ABAC	LCE, LCD, LEAF, LRU, FIFO, RR, SLRU, SProt	Sim, simulador	Real

Fonte: Elaborado pelo autor (2021)

4 CARACTERIZAÇÃO

Este capítulo apresenta e analisa a caracterização da base de dados utilizada neste trabalho. Seu objetivo é apresentar e compreender os efeitos da recuperação de atributos, e como um método de *cache* pode melhorar o desempenho em um cenário de redes sem fio de larga escala real.

4.1 EXTRAÇÃO DE INFORMAÇÕES

Para entender como as *caches* de atributos podem reduzir a latência na recuperação de atributos em uma rede de uma universidade requer a coleta de dados reais. O objetivo principal é capturar os acessos dos usuários e os padrões de mobilidade quando utilizam os recursos da rede. A base de dados criada é composta por dados da política de acesso para alunos, funcionários administrativos e professores da Universidade Federal de Juiz de Fora, com uma população diária de cerca de 20 mil pessoas.

Para compreender o comportamento de acesso dos seus usuário exigiu-se o rastreamento do mecanismo de autorização da universidade por um período de quatro meses, em 108 APs. Os dados vieram da inspeção dos registros, e incluíram, para cada solicitação de acesso, a hora do evento, a identidade do usuário e os atributos necessários para essa solicitação de acesso. Resumindo, as estações de trabalho sem fio em escritórios administrativos e de pesquisa e os APs espalhados pelo *campus* podem gerar a origem das solicitações dos usuários. Foram monitorados 36 mil usuários únicos que fizeram 14,2 milhões de solicitações de acesso durante esses quatro meses.

O mecanismo de autenticação da rede sem fio em questão avalia uma solicitação de acesso sempre que um usuário se conecta a um AP. O método coleta os registros desses APs que fornecem o tempo de conexão e desconexão de cada usuário conectado. Os registros foram convertidos de texto simples para uma base de dados com informações de hora da requisição, identificação do AP, identificação do usuário e funções de autenticação utilizadas. Dentre as funções existem operações de associação, desassociação relacionadas ao padrão IEEE 802.11, resposta de requisição Acesso protegido por Wi-Fi (*Wi-Fi Protected Access* - WPA), resposta de requisição Protocolo de Autenticação Extensível (*Extensible Authentication Protocol* - EAP) e resposta de acesso negado. Sendo que as funções utilizadas nesse trabalho foram as de associação e desassociação do usuário, uma vez que nosso objetivo é identificar onde é mais conveniente que os objetos atributos dos usuários estejam mais próximos a ele na névoa.

A fim de respeitar a Lei Geral de Proteção de Dados (LGPD)[1], os dados nominais dos usuários foram anonimizados nessa etapa de conversão, ficando irreversível sua associação aos dados reais. Com os dados convertidos, novas informações puderam ser extraídas

[1] `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`

da base de dados como o tempo que o usuário permanece em um AP, o intervalo de tempo entre as solicitações de acesso subjacentes em diferentes AP e quantos APs o usuário conecta durante os quatro meses. Finalmente, com o comportamento de acesso e mobilidade, pode se estimar como as *caches* de atributos fazem o sistema utilizado chegar à decisão mais rápido, correlacionando o comportamento dos usuários com o potencial das políticas de *caches* de atributos propostas.

4.2 CARACTERIZAÇÃO DAS REQUISIÇÕES DE ACESSOS

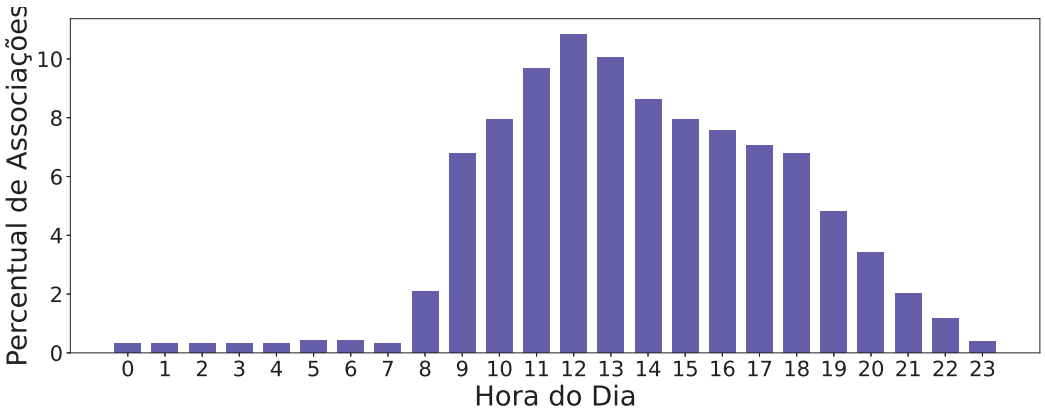
Com os dados obtidos e com as novas essas informações podem ser descritos os perfis dos usuários e o comportamento da rede. Ao todo são 108 APs, 36.169 usuários únicos e 14.289.207 requisições durante um período de 4 meses. Os dados mostram tanto requisições que obtiveram sucesso na conexão e permanecem um período conectado como também conexões com curtos períodos de tempo, como se o usuário estivesse se locomovendo. Em média, cada AP recebeu 132.307 requisições, os usuários passaram em média 303 segundos conectados nesses rádios e levaram em média 172 segundos para fazerem uma reconexão. Cada um dos APs foram utilizados, em média, por 5813 usuários diferentes. Os usuários da rede fizeram em média 395 requisições e passaram em média 423 segundos conectados. Os usuários tiveram acesso a uma média de 17 APs diferentes, e por meio dessa característica, APs conhecidos, podem ser agrupados os usuários definindo seu nível de mobilidade.

O primeiro passo para entender o cenário é identificar o padrão de acesso e mobilidade no *campus*. Portanto, os resultados se concentram em identificar quando, onde e como ocorre a maioria dos acessos. Para iniciar a análise, será examinado o comportamento de forma geral, para identificar os horários de picos da aplicação. A Figura 3 mostra o número percentual de solicitações de todos os acessos ao longo do dia. O pico de solicitações de requisições da rede está no horário das 12h. O número de solicitações de acesso é mais significativo durante o horário comercial, onde a maior parte dos acessos ocorre entre 9h e 18h. Como 80% das solicitações de acesso ocorrendo nessa janela de tempo, todas as análises realizadas consideram esses acessos.

A frequência de requisições da rede durante os dias da semana estão na Figura 4. As quantidades de requisições são maiores nos primeiros dias da semana, segunda a quarta, diminuem um pouco na quinta e sexta, e tem uma baixa utilização nos dias de sábado e domingo. Com esse gráfico fica evidente que os dias úteis, de segunda a sexta-feira, são os de maiores utilizações dos recursos da rede e correspondem a 96% das requisições. Todas as análises realizadas consideram apenas esses acessos dos dias úteis.

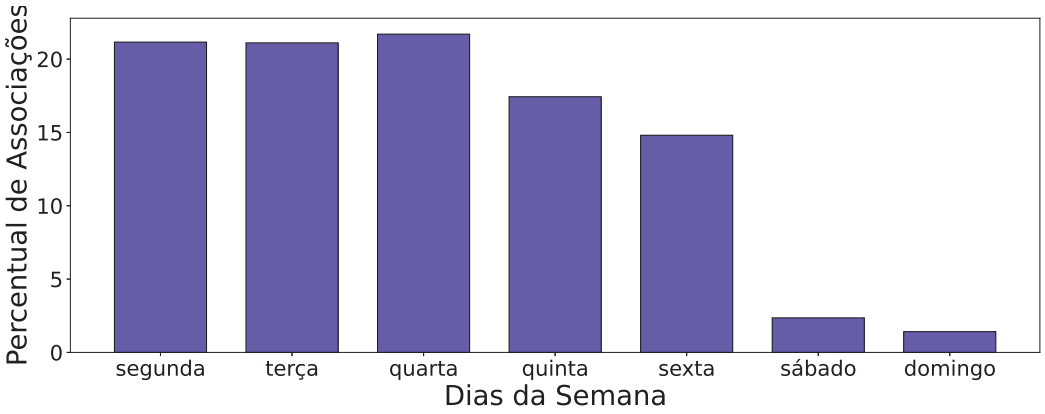
A análise mostra que uma parte significativa das solicitações de acesso dos usuários ocorrem em diferentes locais ao longo do dia. Esse padrão pode ocorrer porque a maioria dos usuários tem aulas em vários locais espalhados pelo *campus* e possuem uma rotina

Figura 3 – Requisições por hora.



Fonte: Elaborado pelo autor (2021)

Figura 4 – Requisições por dias da semana.



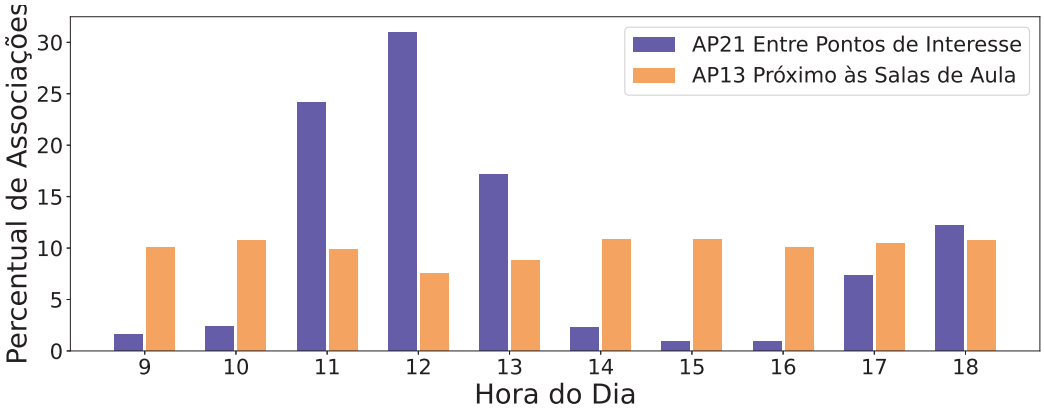
Fonte: Elaborado pelo autor (2021)

bem definida nos pontos de interesse, como departamentos, biblioteca e refeitório (RU – Restaurante Universitário). No geral, percebe-se que os usuários ficam mais conectados em APs localizados perto do caminho das salas de aula e em outros poucos pontos de interesse conhecidos. No entanto, os usuários tendem a permanecer conectados apenas por curtos períodos. Em média, cada AP nessas rotas conhece cerca de 6 mil usuários, e esses usuários permanecem conectados por aproximadamente cinco minutos. Quando um usuário se desconecta, ele pode se reconectar em 3 minutos e geralmente usa um AP adjacente, caracterizando os padrões de rota e mobilidade.

Para APs mais próximos às salas de aula, e outros pontos de interesse, o comportamento é diferente. Esses APs conhecem uma parte significativa dos usuários, visto que são centros de convivência para alunos, professores e equipe administrativa, apresentando tempos de conexão mais longos em horários específicos do dia. Por exemplo, a Figura 5 mostra o comportamento de conexões em dois APs diferentes. O primeiro, AP21, é uma

rota entre dois pontos relevantes de interesse da universidade comentados anteriormente, esse AP apresenta um comportamento considerado mais estável e esperado, com um padrão recorrente quanto ao número de usuários conectados e aos os horários de uso. Este comportamento ocorre porque a sua localização compreende um caminho que cruza pontos de interesses, e os usuários não permanecem muito tempo conectados nesta localidade. Para o segundo AP, o AP13, ele está localizado próximo às salas de aula e laboratórios, o que significa que este AP registra a atividade de alguns usuários e permanecem mais tempo conectados, e seus picos de carga correspondem à programação de aulas.

Figura 5 – Comportamento de mobilidade.



Fonte: Elaborado pelo autor (2021)

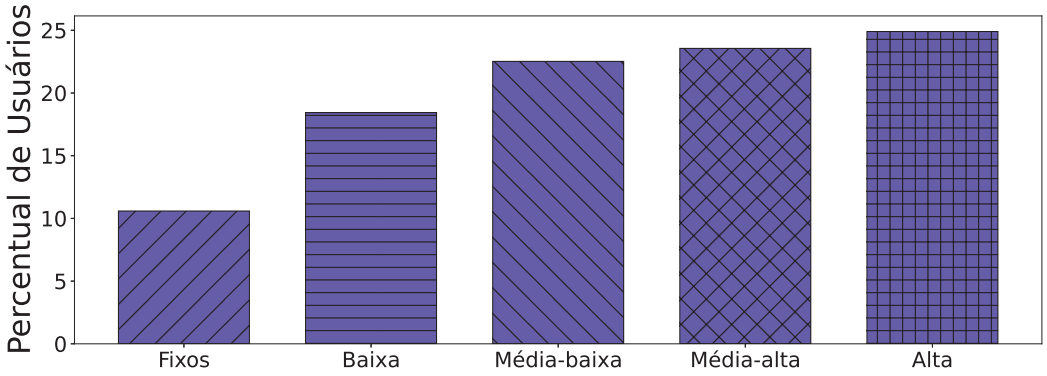
Pode-se usar uma abordagem reativa ou pró-ativa para armazenar em *cache* os dados do usuário neste cenário. A primeira abordagem é mais direta, onde o sistema pode reagir ao padrão de acesso, colocando os atributos mais próximos dos APs em pontos de interesse ao longo de sua rota sobre o *campus*. Embora essa abordagem possa funcionar para quase todos os padrões de acesso, ela pode ficar um passo atrás quando o usuário se move dentro do *campus*, uma vez que armazenamos em *cache* os atributos necessários somente após a movimentação ocorrer, atrasando o sistema constantemente para aumentar o desempenho dessa solicitação de acesso. Por outro lado, a segunda abordagem usa o padrão de movimento do usuário e prevê movimentos futuros. Em seguida, ele replica proativamente os dados do usuário nas *caches*. O desempenho dessa abordagem se deteriora quando os usuários apresentam padrões de mobilidade não determinísticos.

Nesse sentido, foi caracterizada a diversidade de padrões de mobilidade encontrada no *campus* universitário. Por exemplo, a Figura 6 ilustra quantos APs um usuário conecta durante sua vida útil. A seguir, usamos essa distribuição para rastrear perfis de mobilidade.

Conforme a Figura 6, cerca de 10% dos usuários do *campus* se conectam a apenas um único AP. Em outras palavras, esses usuários praticamente não apresentam padrão de mobilidade e sua localização futura é simples de determinar. Por este motivo, os usuários deste grupo são marcados como “usuários fixos” e os removemos de nossa análise, dado que

o problema de *cache* de atributos é trivial para eles. Assim, o primeiro quartil, excluindo os usuários fixos, conecta-se a um número de 2 a 4 APs. Portanto, os usuários desse grupo apresentam um pequeno padrão de mobilidade, que é um padrão direto que auxilia na abordagem proativa. Por esse motivo, o método rotulou esses usuários como usuários de "baixa mobilidade".

Figura 6 – Perfil de mobilidade.



Fonte: Elaborado pelo autor (2021)

O segundo quartil da Figura 6 apresenta usuários que se conectam por 5 a 12 APs. Esses usuários se movem pelo *campus*. Porém, na maioria dos casos, os usuários deste grupo ainda possuem um caminho organizado e de fácil conhecimento, o que também pode favorecer a abordagem proativa. Por esse motivo, foram rotulados como usuários de "mobilidade média-baixa". O terceiro quartil tem usuários com movimentos mais confusos e difíceis de determinar. O método classificou os usuários desse grupo como usuários de "mobilidade média-alta". Nesse caso, os usuários se conectam, em média, de 13 a 28 APs. Observe que esse padrão de movimento é mais aleatório do que os apresentados antes, o que significa que pode tornar mais difícil a abordagem proativa da *cache*. Finalmente, os usuários que apresentam um padrão de acesso quase caótico conectando pelo menos por 29 APs no último quartil. Para este caso, está claro que a abordagem de *cache* proativa deve contar com um algoritmo de previsão de mobilidade altamente complexo e preciso para atender a esses usuários e entregar os atributos de forma eficiente.

5 CACHE DE ATRIBUTOS OPORTUNISTA

Neste capítulo apresentamos os conceitos relacionados à *cache* de atributos oportunistas e os relacionamos à nossa proposta. Destacamos que nossa proposta cobre os requisitos listados anteriormente para o ABAC, propõe e avalia uma arquitetura de nuvem-névoa para os elementos componentes do XACML e realiza o *cache* de atributos considerando replicação e substituição de forma mais eficiente. Para tanto, além da adoção e adaptação do XACML e ABAC no ambiente nuvem-névoa, propomos uma política de *cache* de atributos oportunista proativa baseada em um modelo de predição de mobilidade.

Assim, temos que o armazenamento em *cache* de atributos oportunistas é um método que identifica a melhor oportunidade para armazenar atributos em *caches*, minimizando o tempo de recuperação do atributo para as solicitações de acesso subsequentes. O método se baseia em duas abordagens que se complementam para replicar os atributos nos nós de névoa da arquitetura de rede: *cache* de atributo oportunista reativo; e cache de atributo oportunista proativo. A seguir detalhamos essas técnicas.

Uma rede é considerada configurada em uma topologia de árvore, o que significa que o nó raiz representa um serviço em nuvem (ou um conjunto de serviços em nuvem externos) e que armazena atributos dos usuários. De acordo com a arquitetura XACML, esse nó raiz representa um enorme PIP, com capacidade suficiente para armazenar todas as informações sobre todos os usuários associados ao IdP. Todos os nós internos desta árvore representam nós de névoa. Esses nós de névoa atuam como um PEP com um PDP embutido da arquitetura XACML.

Embora isso possa parecer impreciso, o principal desafio é reduzir possíveis alterações de arquitetura quando há uma fusão dessas duas entidades do XACML. Esse tipo de junção pode funcionar apenas quando os aplicativos são pequenos ou com poucas solicitações e para um problema de autorização unificada, porque mesmo que não seja escalonável e não seja extensível, unificar as autorizações sem perder a generalização do sistema pode transformar em uma arquitetura sem representatividade (18). A proposta aqui não é juntar e fundir as entidades modificando a arquitetura do XACML, mas que elas funcionem lado a lado numa mesma estrutura que são os nós da névoa. Uma vez que a maioria dos nós de névoa tem capacidade de computação e armazenamento, eles podem atuar como PDP e como um PIP. Porém esse PIP poderá armazenar apenas um subconjunto de atributos/objetos. Embora o poder computacional seja suficiente para avaliar as políticas, e atuar como PDPs, esses nós de névoa, em geral, têm várias limitações sobre sua capacidade de armazenamento.

Neste trabalho, nosso foco é melhorar a qualidade da comunicação PDP-PIP. Logo, para abstrair a comunicação PEP-PDP, será considerado que ambos elementos estão funcionando em uma única estrutura física, os nós da névoa. Ainda que essa junção

PEP-PDP não seja ideal em alguns cenários, visto que a escalabilidade pode ser afetada por essa junção, ela não afeta nosso problema e se mostra uma arquitetura generalista.

O cenário abordado neste trabalho é uma aplicação IoT em execução em uma grande área. Esta área (por exemplo, um *campus* universitário) utiliza um servidor em nuvem para manter a coleção completa de atributos de todos os seus usuários. Por lidar com um grande *campus*, o aplicativo IoT espera muitos usuários e dispositivos móveis distribuídos em uma grande área geográfica, solicitando acesso a vários objetos em qualquer local e a qualquer hora. Esta aplicação requer autorização contínua e constante dos usuários que acessam os objetos. Para manter essa autorização contínua, vale a pena reduzir o tempo de avaliação da política para manter o impacto da latência da avaliação da política o mínimo possível. Visto que, para cada avaliação de política, o PDP deve enviar uma solicitação de atributo pela rede para chegar à nuvem, obtendo os atributos para avaliar uma política sofre de uma latência significativa que impacta diretamente na avaliação da política. Além de diminuir a latência temos outros benefícios como ter disponível as identidades quando os repositórios originais não podem ser acessados por questões de emergências como baixa largura de banda ou perda de serviço (15).

Para resolver isso, a universidade utilizaria vários nós de névoa e os distribuiria na em sua área geográfica para atuar como *caches* de atributos, funcionando como fonte de atributos além da nuvem e mais próximos ao usuário. Embora este trabalho se concentre em uma aplicação IoT em execução em uma grande área, sua aplicabilidade não se limita apenas a este cenário. Diversas aplicações IoT, como cidades inteligentes (14) e saúde (27), compartilham algumas características com o cenário abordado. Seja pela nuvem usada para armazenar atributos, a arquitetura distribuída ou o requisito de latência.

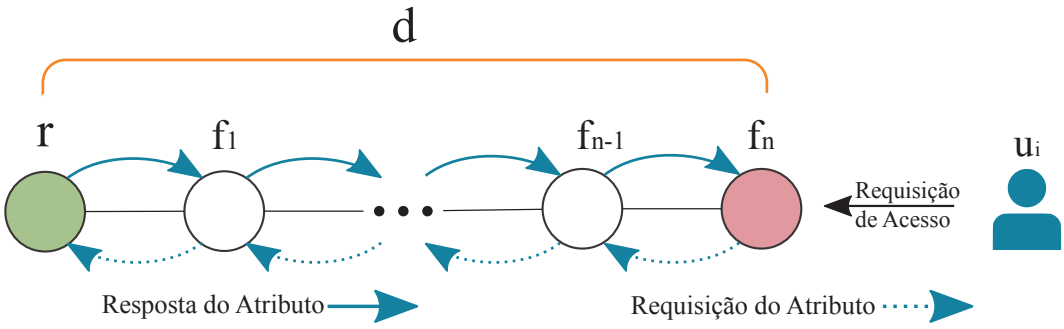
Este trabalho segue o relatório da CISCO (30) que define um conjunto de candidatos potenciais a se tornarem dispositivos de névoa, incluindo roteadores, *switches*, APs e outros dispositivos. Portanto, todos os roteadores e *switches* se tornam nós de névoa implantados com uma capacidade de armazenamento para manter parte dos atributos necessários mais perto dos usuários para minimizar o impacto da recuperação de atributos. No entanto, quando adotados os nós de névoa como *caches* de atributos, o PIP é descentralizado e, em contrapartida, é acrescentado o custo de manter os atributos atualizados e seguros. Além disso, os *caches* por si só não garantem um melhor desempenho na recuperação de atributos, pois alguns atributos podem não estar presentes em seu *cache* durante uma solicitação de atributo. Para esses casos, é um "erro" (*miss*) e, quando ocorre, significa que os atributos do usuário não estão presentes no melhor ponto naquele momento. Como resultado, a avaliação da política sofre uma latência para recuperar os atributos, culminando em maior latência para o usuário acessar o recurso desejado.

Portanto, o problema consiste em encontrar o subconjunto de atributos para armazenar em cada nó de névoa para minimizar o número de falhas, assumindo um

conjunto de usuários solicitando acesso a recursos em vários pontos da universidade em momentos diferentes. Portanto, devemos encontrar onde e quando esses atributos devem ser replicados nos nós de névoa para resolver este problema. Como cada replicação de atributo aumenta o custo para manter a consistência do sistema, é necessário encontrar o custo-benefício entre uma réplica e o custo para mantê-la atualizada e segura.

O modelo de solicitação resposta correspondente a arquitetura da rede está na Figura 7, onde um usuário poderá enviar uma solicitação de acesso a um objeto específico ou recurso protegido que não está representado nesta figura. O nó raiz r é o servidor na nuvem, enquanto os outros $(f_1, f_2, ..., f_n)$ são nós na névoa da rede. Por exemplo, os roteadores e *switches*. Uma vez que os nós de névoa podem atuar como PDP e PEP, esta solicitação de acesso é interceptada por um nó da névoa. Com a política instalada em cada nó de névoa, este nó intercepta a solicitação de acesso e envia uma solicitação de atributo na direção da nuvem para obter os atributos necessários para avaliar a política. Visto que assumimos que esses nós de névoa podem atuar como PIP se eles tiverem algum atributo solicitado, esse atributo é enviado de volta diretamente em resposta à solicitação. Caso a solicitação de atributo não encontre os atributos necessários nos nós da névoa, ela chega à nuvem que garante a resposta dos atributos necessários.

Figura 7 – Modelo requisição-resposta.



Fonte: Elaborado pelo autor (2021)

5.1 CACHE DE ATRIBUTOS OPORTUNISTA REATIVO

A *cache* de atributo oportunista reativo é uma abordagem que replica os atributos com base apenas na resposta local a uma solicitação de atributo. A Figura 7 ilustra uma solicitação de acesso que ocorre no solicitante de nó r. Assim que a solicitação de atributo chega ao parceiro p que contém os atributos, p envia de volta ao solicitante os atributos de que precisa. Como esta resposta de atributo atinge vários nós no caminho de volta para o solicitante, cada nó nesta rota $(f_1, ..., f_{n-1}, f_n)$ tem a oportunidade de replicar esses atributos. Portanto, de acordo com este *cache* de atributo oportunista reativo, do primeiro ao último nó na rota entre o solicitante e o parceiro p, todo nó pode oportunamente

salvar uma cópia do atributo na *cache*. Este trabalho considera três estratégias reativas de replicação de atributos oportunistas, a saber: LCE, LCD e LEAF.

Como já apresentado, o LCE é a estratégia mais gananciosa onde todos os nós no caminho entre o solicitante r e o parceiro p replicam o atributo e armazenam uma cópia em seu PIP. Assim, no exemplo da Figura 7, todos os nós $(f_1, ..., f_{n-1}, f_n)$ mantêm uma cópia dos atributos necessários em seus *caches*. Dado que essa estratégia sempre gera uma cópia do atributo solicitado em todos os nós, é a estratégia mais abrangente, criando muita redundância no sistema. Embora essa redundância possa parecer ótima para diminuir a latência para encontrar os atributos necessários (diminuindo d para solicitações de atributos populares), ela aumenta o custo de manter todos esses atributos seguros.

O LCD é uma estratégia onde as cópias dos atributos se movem gradualmente na direção de seu ponto ideal. Nessa estratégia, durante a resposta do parceiro p ao solicitante r, apenas o primeiro nó em seu caminho replica os atributos. Na Figura 7, apenas o nó f_1 replica os atributos necessários em seu PIP. Se esses atributos necessários forem exigidos novamente em uma solicitação de atributo subsequente, se esses atributos ainda estiverem armazenados em *cache* no nó f_1, apenas o nó f_2 os replica. Essa estratégia conservadora implica que os atributos necessários se movem gradualmente em direção ao solicitante r, movendo-se um passo mais perto para cada solicitação de atributo subsequente. Assim, nesta estratégia, enquanto os atributos populares ficam mais próximos do usuário (diminuindo d para solicitação de atributos populares), os menos necessários se mantêm distantes, deixando mais espaço para os nós mais próximos armazenarem os atributos populares, o que pode aumentar a proporção geral de acertos.

A estratégia LEAF é a mais arriscada entre as três. Essa estratégia adota uma abordagem mais tudo ou nada, onde os *caches* fornecem o desempenho ideal ou pior na solicitação de um atributo. Assim, durante a resposta do parceiro p ao solicitante r, apenas o último nó em seu caminho replica os atributos. Nesse sentido, na Figura 7, apenas o nó f_n, que é o nó solicitante, replica os atributos necessários no PIP. Se os atributos necessários ainda estiverem em *cache* para uma solicitação de atributo subsequente, essa estratégia fornecerá o melhor desempenho; caso contrário, ele precisará alcançar a nuvem.

5.2 CACHE DE ATRIBUTOS OPORTUNISTA PROATIVO

O principal objetivo desta abordagem consiste em encontrar um subconjunto de atributos a serem armazenados para minimizar o número de perdas na operação de decisão de política de diferentes locais ao longo do tempo. Uma vez que essas solicitações variam entre os locais, lidar com esse problema de uma forma míope pode não ser a melhor abordagem, dado que lida apenas com a solicitação atual. Para reduzir o número de falhas, se a abordagem prever onde ocorrerá a solicitação subsequente, os atributos necessários podem ser replicados proativamente naquele local, reduzindo a taxa de falhas.

Um algoritmo de previsão de mobilidade eficaz é essencial para determinar a posição da solicitação subsequente.

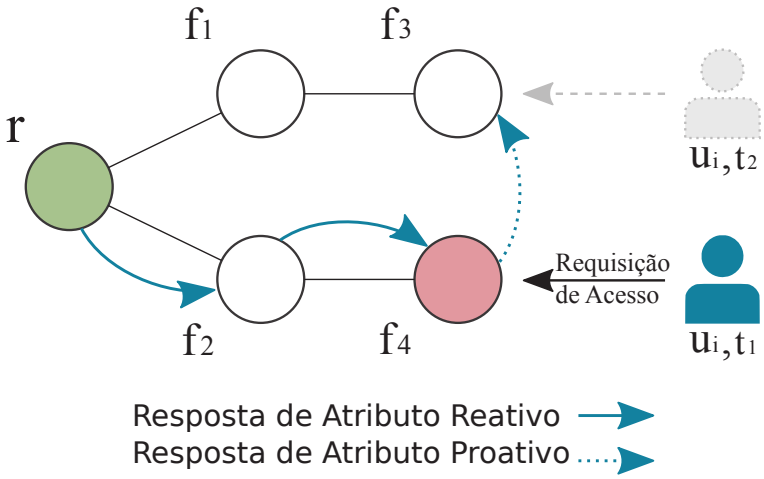
Nesse trabalho foi utilizado o classificador k vizinhos mais próximos (*k-Nearest Neighbors - kNN*), que atingiu uma acurácia geral de 45%. Todavia a destacamos que a previsão de mobilidade não é o foco deste trabalho, e portanto, serve como avaliação e oportunidade de investigação para futuras pesquisas na área. Ainda destacamos que a estratégia de replicação de atributos proativa e oportunista assume também a implementação para avaliação por meio de um oráculo. Tal oráculo conhece o padrão de mobilidade do usuário e está ciente da localização da solicitação atual e da próxima. Assim, o oráculo serve como base de comparação do ótimo possível para o ambiente, no auxílio da avaliação dos algoritmos, e principalmente, do algoritmo de previsão de mobilidade. Neste caso, para as avaliações conduzidas, ao invés de usarmos abordagens complexas, *e.g.*, cadeias de Markov complexas ou redes neurais para prever a mobilidade e aumentar a latência de nossa abordagem, o oráculo as substitui como uma caixa preta que fornece a previsão correta e auxilia na avaliação de parte da proposta. Para esses casos, a suposição é que um oráculo fornecerá a previsão com 100% de precisão para o ambiente de mobilidade e *cache* de atributos. Destacando que o enfoque principal do trabalho não é a previsão de mobilidade, e sim propor uma arquitetura nuvem-névoa baseada em XACML e ABAC para redução da latência e recuperação de objetos da identidade dos usuários.

Para exemplificar esta abordagem, considere uma rede com quatro nós de névoa (f_1, f_2, f_3, f_4), um servidor de nuvem no nó raiz r e um usuário u_1, como representado na Figura 8. Supondo que no momento t_1, o usuário u_1 solicite acesso em f_4, a solicitação de atributo flui do solicitante r para o parceiro p e volta, permitindo que todos os nós no caminho repliquem esses atributos. No entanto, como no instante t_2, esse mesmo usuário solicita acesso ao nó na névoa f_3, o método proposto replica proativamente os atributos para esse nó na névoa se ele não contiver esse atributo. Assim, no instante de tempo t_2, é possível fazer a solicitação de acesso (e a eventual solicitação de atributo subsequente que pode originar) com $d = 0$ em relação distância entre a requisição resposta da Figura 7.

5.3 OPERANDO COM UM PIP COMPLETO

Como comentado, cada nó de névoa tem capacidade para armazenar apenas uma quantidade limitada de atributos. Portanto, esses nós podem ficar sem espaço de armazenamento para *cache* em algum momento. Supondo que este nó deva armazenar um novo atributo, ele então deve descartar um ou mais atributos antigos primeiro. Esse ato é chamado de substituição de *cache* e este trabalho emprega cinco políticas diferentes para determinar qual atributo deve ser descartado, apresentadas já neste trabalho. São elas: a RR, que escolhe aleatoriamente um atributo para ser substituído; a FIFO, onde o atributo mais antigo armazenado é o primeiro a ser descartado; a LRU, que descarta o atributo no

Figura 8 – Modelo requisição-resposta proativo.



Fonte: Elaborado pelo autor (2021)

final da lista, assumindo que o atributo usado mais recentemente está na primeira posição e o atributo menos usado recentemente na última posição da lista; a *Hit*, que descarta o atributo que tem o menor contador de acertos entre os objetos armazenados. Esse contador é acrescentado sempre que um atributo é reutilizado; a política Baseada em Tempo, que seleciona o objeto com maior tempo entre reconexões naquele AP; A política do Menos Usado Recentemente Segmentado, ou SLRU, que é uma variação do LRU clássico, que separa a *cache* em dois segmentos – um de oportunidade e outro segmento de protegidos; por fim, a política final é a Menos Usado Recentemente Segmentado com Predição no espaço Protegido (SPROT), que é uma variação do SLRU.

A SPROT é uma nova política, e proposta desse trabalho. Ela usa do espaço protegido para os objetos que passaram pela predição de mobilidade. Com a *cache* segmentada em dois espaços, a abordagem proativa aciona o modelo de predição para adicionar a identidade prevista no segmento protegido, dessa forma esse objeto tem uma oportunidade maior de ser reutilizado numa futura solicitação da identidade.

Essa política, SPROT, organiza os objetos no espaço oportunidade com a lógica da estratégia LRU. Se a quantidade de objetos em *cache* fossem de dez itens e a proporção de espaço protegido for de 50%, a organização do espaço em *cache* seria conforme Figura 9. Quando ocorre uma requisição e esta necessitar adicionar um novo objeto, ele é então colocado na posição 6 da lista representada pela Figura 9. Se esse objeto já estava em memória ele então irá para o topo do espaço oportunidade ocupando, também, a posição 6.

Ao finalizar essa alocação considerando o dispositivo de névoa um AP de onde ocorreu a requisição, o algoritmo verifica onde será o próximo AP a ser visitado pelo usuário seguindo o algoritmo de predição de mobilidade. Uma vez identificado o próximo

AP, a política coloca essa identidade no início do espaço protegido daquele AP, que na Figura 9 corresponde à posição 1. O algoritmo neste caso obedece a uma mesma lógica para cada objeto. Se for um novo objeto que já estava em memória atualiza apenas a posição dele para 1. Em todos os casos, quando o espaço de armazenamento está cheio, é removido o último objeto, que na Figura 9 seria o objeto 10. Para fins de referência, o **oráculo** que também foi implementado será chamado neste trabalho de **Sprot100**.

Figura 9 – Menos Usado Recentemente Segmentado com Predição.

Fonte: Elaborado pelo autor (2021)

A construção do trabalho foi para reorganizar as estruturas da arquitetura do XACML, por mais que tenha sido criado para dar suporte ao ABAC, nossa proposta pode ser utilizada para o RBAC e para outros modelos de autorização que venham a surgir.

Por fim, quanto às contribuições deste trabalho em relação às políticas de replicação e substituição, avalia-se e propõe-se uma nova política de substituições baseada em segmentação e predição de mobilidade. Desta forma, auxiliando na redução do problema de atrasos em entrega de identidades em ambientes hierárquicos para névoas computacionais. Tal política é, portanto, considerada uma nova política de *cache* oportunista e proativa de identidades, baseada na predição de mobilidade do usuário, que une o benefício da SLRU com o uso de aprendizado de máquina, reduzindo a exclusão de objetos e abrindo a oportunidade para novas pesquisas.

6 METODOLOGIA

Este capítulo descreve a metodologia de avaliação utilizada, apresentando os cenários de topologia de rede considerados, o método de aprendizado de máquina aplicado. Além disso, introduz brevemente o simulador implementado para avaliar os cenários e políticas e gerar os resultados deste trabalho.

6.1 CENÁRIOS DE TOPOLOGIA DE REDE

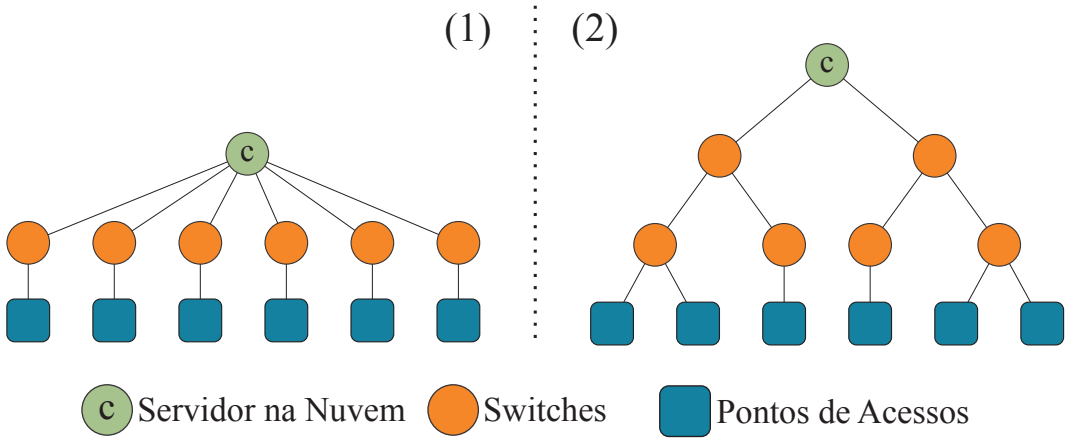
Como comentado anteriormente na seção 4.1, a base de dados utilizada contém as informações das autenticações dos usuários nos APs da rede. Porém, essa base de dados não informa o caminho percorrido dentro da rede. Portanto, foram gerados cenários de topologia de redes para representar redes com profundidades e larguras diferentes. Todos os cenários possuem o máximo de 62 *switches* e foram organizados por meio de um algoritmo de crescimento de árvore balanceada. A única restrição foi para não ocorrer mais de dois *switches* interligados sem que eles não tenham outros filhos. A escolha da quantidade de 62 *switches* foi para que as redes projetadas pudessem ter a mesma quantidade de espaço de armazenamento de *cache*. Os cenários gerados consideram pontos de acesso IEEE 802.11. Esses dispositivos podem ser usados, com sucesso, para aplicações IoT (13).

A topologia de rede simulada segue uma árvore hierárquica, como em um *campus* universitário. Para avaliação, foram geradas diferentes topologias de rede em forma de árvore hierárquica, onde variam a altura da árvore de 2 a 7. As topologias também variam com a altura de cada camada. Em todas as topologias, a camada inferior da rede apresenta 108 APs espalhados geograficamente pela universidade. A raiz da topologia de rede é semelhante a uma árvore, ou seja, a camada superior da rede, representa um servidor de nuvem privada. Todos os níveis intermediários restantes da rede (ou seja, as camadas entre a raiz da árvore e a camada inferior) apresentam 62 *switches* que interconectam toda a rede do *campus*.

Topologias de redes específicas, por exemplo, como a estrutura de rede da UFJF, poderiam deixar o trabalho mais restritivo e diminuir o impacto das comparações de alturas e larguras diferentes. Assim, essas topologias geradas reforçam a análise das estruturas de rede para uma análise geral. Estes cenários seguem o relatório da Cisco já citado, que define o conceito de computação em nuvem como dispositivos com recursos computacionais integrados, armazenamento e rede, essenciais para facilitar a execução de aplicativos IoT (30). De acordo com o relatório, um conjunto de candidatos potenciais são dispositivos de névoa, incluindo roteadores, *switches*, APs, etc. (30).

A Figura 10 ilustra isso e mostra duas topologias em uma escala menor. Ambos têm o mesmo número de *switches* e Aps. No entanto, a organização dos *switches* é diferente, implicando duas árvores com alturas diferentes, mas com o mesmo número de folhas.

Figura 10 – Exemplos de topologias de redes.



Fonte: Elaborado pelo autor (2021)

Enquanto o primeiro (1) tem altura dois e é mais largo, o segundo (2) tem altura três e é mais estreito quando comparado ao outro. Portanto, este trabalho segue a mesma ideia, mas com 108 APs e 62 *switches*.

O servidor de nuvem privada funciona como um IdP, e armazena os atributos de todos os 36 mil usuários únicos, enquanto os *switches* e APs (nós de névoa) podem conter uma fração desse número. Assim, nas simulações apresentadas a capacidade de armazenamento dos nós de névoa também irá mudar. Para avaliar um cenário com abundância e escassez de recursos, foi definido arbitrariamente, mas com base no trabalho de (8) a variação da capacidade de armazenamento de forma proporcional ao total de identidades armazenadas na nuvem. Assim, os nós de névoa têm capacidade de armazenamento de 0,1% ou 1% do total número total de atributos, o que corresponde a 36 e 360 atributos por nó de névoa, respectivamente.

6.2 APRENDIZADO DE MÁQUINA

Como comentado na seção 5.2, uma *cache* oportunista proativa com predição de mobilidade foi proposta neste trabalho, avaliando seu impacto sobre a manutenção dos atributos na névoa. Para tanto, a base de dados foi dividida de forma a evitar sobre-ajustes e adaptações aos modelos de aprendizado testados e de forma que não interfira nos dados da simulação que será apresentada na próxima seção. A base de dados possui quatro meses de requisições, sendo que os três primeiros meses foram separados para avaliar em modelos de aprendizado estatísticos e o último mês usado na avaliação do simulador. Em todas as avaliações de aprendizado foi utilizado a proporção de 70 e 30, sendo 70% dos dados usados no treinamento e 30% na avaliação e teste dos modelos.

Nessa etapa alguns dados da base foram considerados desnecessários, como todas as respostas WPA, EAP e acesso negado. Essas informações não estavam completas e

iriam trazer ruídos não necessários às análises, portanto o processamento do modelo foi realizado utilizando a característica de associação e desassociação aos APs que deram origem a outros dados como comentado na seção 4.1 de Extração de Informações. A base foi reduzida em características ficando com os seguintes atributos: data e hora (time1 em formato timestamp), tempo conectado (timet em segundos), usuário (client – identificador numérico), AP (radio – identificador numérico), AP alvo (targ-1 – identificador numérico), AP anterior (prev-1 – identificador numérico), data e hora em valores numéricos (mês, dia, hora, semana). Nesse trabalho foi utilizado o classificador k Vizinhos mais Próximos (*k-Nearest Neighbors - kNN*), que atingiu uma acurácia geral de 45%.

As estruturas das redes geradas na seção 6.1 não influencia os algoritmos de predições, pois a predição é realizada com as informações dos acessos dos usuários ao se associarem a um determinado AP. Para tanto, as topologias de redes geradas não atuam nos modelos de predições avaliados neste trabalho.

6.3 SIMULADOR

Para avaliar a proposta deste trabalho foi implementado um simulador de eventos discreto em Python. Ele recebe as requisições através de um arquivo de entrada e executa os algoritmos a serem avaliados para replicação e substituição.

Com o propósito de avaliar o momento mais requisitado da rede, e com possíveis congestionamentos e grandes mudanças de posições dos usuários, as simulações foram executadas com as requisições que ocorreram no horário comercial. Este período corresponde à maior utilização da rede, que conforme a Figura 3 na seção 4.2 compreende os horários entre 9h e 18h. Também foram selecionados os dias úteis da semana que conforme a Figura 4 na seção 4.2 compreende os dias entre a segunda e sexta-feira.

As simulações foram executadas com validade dos atributos replicados apenas para o dia corrente. Logo, a cada novo dia todos os atributos deveriam ser requisitados do servidor na nuvem novamente de modo a manter um tempo de vida máximo de 24h.

A cada solicitação realizada pelo cliente, simulações avaliam o método oportunista de replicação dos atributos. Portanto, as simulações consideram os cenários usando a replicação de atributo oportunista reativo e proativo com todas as estratégias de replicação (LCE, LCD e LEAF) e todas as políticas de substituição: o menos recentemente usado (LRU), primeiro a entrar primeiro a sair (FIFO), descarte aleatório (RD), baseado em tempo de reconexão, baseado em acerto, Segmentado Menos Usado Recentemente (SLRU) e Menos Usado Recentemente Segmentado com Predição (SPROT). Como a última política de substituição depende muito da abordagem proativa, as simulações comparam essa política de substituição de duas formas: a primeira com o preditor do kNN com uma precisão de aproximadamente 45% e a segunda com um oráculo perfeito com 100% de precisão.

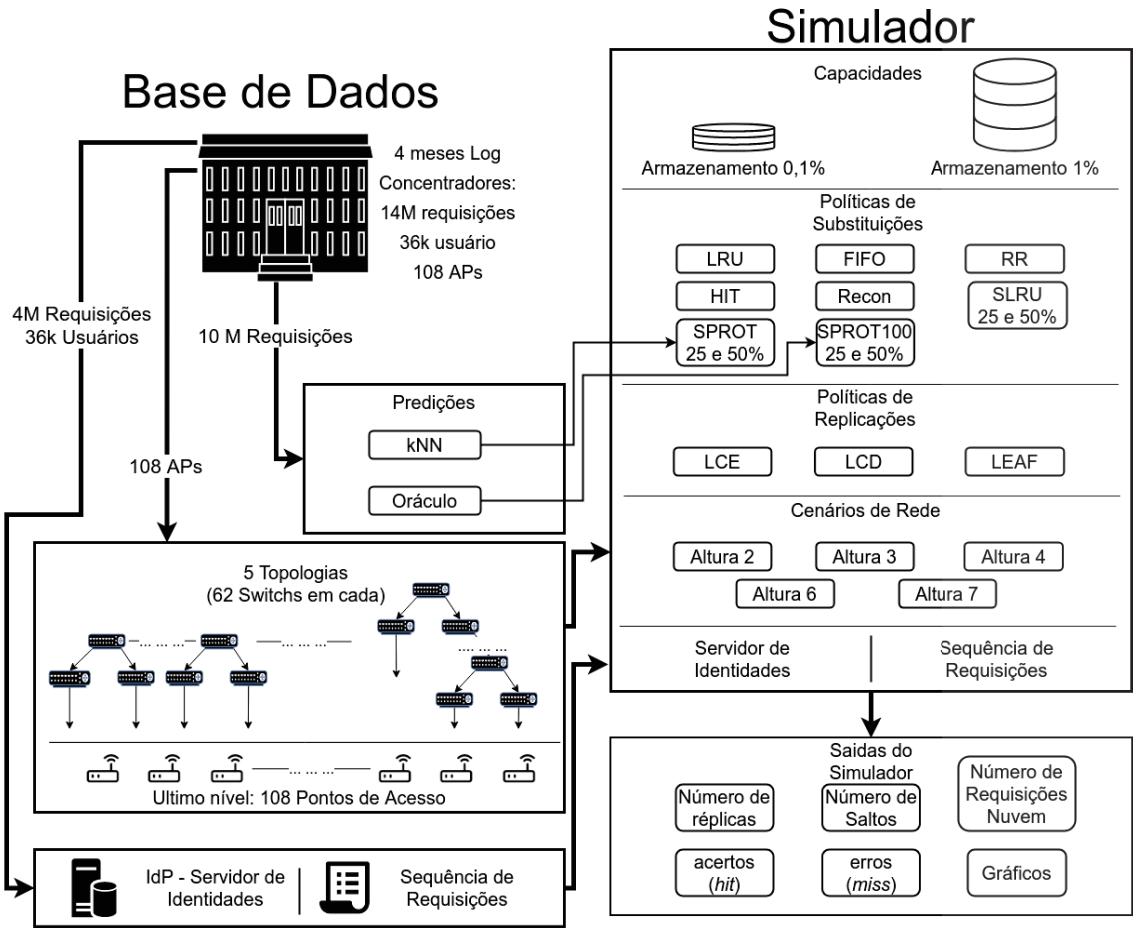
7 RESULTADOS

Este capítulo apresenta a avaliação de desempenho dos métodos de *cache* de atributos oportunistas proativo e reativo. Em primeiro lugar, descrevemos as métricas, e nomenclaturas usadas nos gráficos de desempenho. Em seguida, apresentamos os resultados de várias estratégias de *cache* e suas combinações com as políticas de substituições consideradas no trabalho.

7.1 AVALIAÇÃO

A avaliação da proposta foi realizada através de um simulador de eventos discretos implementado em Python conforme comentado na seção 6.3. A Figura 11 apresenta um resumo das estruturas, variáveis de entrada, variáveis de saída e configurações utilizadas no simulador. Nesse trabalho foram replicados todos os atributos das identidades autorizadas, considerando que não se tem a informação de qual outro recurso poderia ser solicitado na próxima requisição do mesmo usuário.

Figura 11 – Avaliação com Simulador.



Fonte: Elaborado pelo autor (2021)

As métricas que serão observadas são: taxa de acertos (*hit ratio* - HR), taxa de acertos de um salto (*one hop hit ratio* - OHHR), taxa de requisição ao servidor na nuvem e o número de réplicas de atributos. A taxa de acertos é uma métrica para determinar quantas solicitações de atributos os *caches* podem atender com sucesso, em comparação com quantas solicitações recebem. Portanto, essa métrica indica o quão bem os atributos replicados localizados nos nós de névoa estão. A equação 7.1 formaliza a definição da métrica RH.

$$HR = \frac{total\ de\ acertos\ na\ cache}{total\ de\ acertos\ na\ cache\ +\ total\ de\ erros\ na\ cache} \tag{7.1}$$

A taxa de acertos de um salto é uma métrica para determinar quantas solicitações de atributos os *caches* mais próximos do usuário podem atender em comparação com quantas solicitações ele requisitou. Esta métrica mostra quantas solicitações de atributo são respondidas instantaneamente pelos nós de névoa. Esse é o melhor caso para o sistema e gera o menor custo em utilização da rede. A equação 7.2 determina como calcular a métrica de taxa de acerto de um salto.

$$OHHR = \frac{total\ de\ acertos\ na\ cache\ com\ um\ salto}{total\ de\ acertos\ na\ cache\ +\ total\ de\ erros\ na\ cache} \tag{7.2}$$

A quantidade de requisições à nuvem é uma informação que mostra a quantidade de requisições que necessitaram percorrer todos os níveis da rede até alcançar o servidor de identidades na nuvem. Este sendo o pior caso, e de maior custo ao sistema.

Por fim, para calcular o número de réplicas de atributos, as simulações armazenam todos os atributos replicados em todos os nós de névoa e consideram que cada nova réplica aumenta o custo de mantê-los seguros e atualizados.

As políticas que usam espaço de armazenamento segmentado foram implementadas com proporções de 25% e 50%. Por exemplo, as políticas marcadas como 25% usa 25 porcento do espaço disponível em *cache* para o armazenamento protegido e 75 porcento para o espaço de oportunidade. Então, quando configurado com 50% a segmentação será em proporções iguais para o espaço protegido e de oportunidade. Os cenários de topologia de redes gerados para representar as profundidades e larguras diferentes possuem as seguintes configurações: altura de 2 níveis com até 9 irmãos, altura de 3 níveis com até 4 irmãos, altura de 4 níveis com até 3 irmãos, altura de 6 e de 7 níveis com até 2 irmãos. A disposição dos irmãos foi dessa forma para atender a quantidade total de 62 *switches*.

7.2 RESULTADOS NUMÉRICOS

Nesta seção são apresentados os resultados para a recuperação de atributos do ABAC usando *caches* oportunistas com a política de distribuição de identidades proposta. A política proposta respeita o conceito de redes multinível de recuperação de objetos

para névoas computacionais. Nossas avaliações consideram as diferentes estratégias para geração de uma réplica de atributo, em combinação com as diversas estratégias de *cache* reativas e proativas e políticas de replicação e substituição.

Na Figura 12a tem-se a métrica *Hit*, ou *hit ratio* (HR), em combinação com a política de replicação LCE e as políticas de substituições: baseada em acertos, baseado em tempo de reconexão e substituição randômica (RR). Nessa figura, a política de substituição baseada em acerto *hit* e a baseada em tempo de reconexão destoam em relação às outras políticas avaliadas, obtendo um desempenho muito abaixo em relação às demais políticas.

Comparando o método de substituição considerado o mais simples – *i.e.*, a política de substituição RR – as duas políticas citadas ficam até 5% abaixo para um tamanho da *cache* de 1%, e 10% abaixo quando a capacidade de armazenamento é 0,1%. O mesmo comportamento acontece quando utilizamos as demais políticas de replicações, *i.e.*, LCD e LEAF. Essas duas últimas foram suprimidas dos gráficos, uma vez que apresentam comportamentos muito próximos.

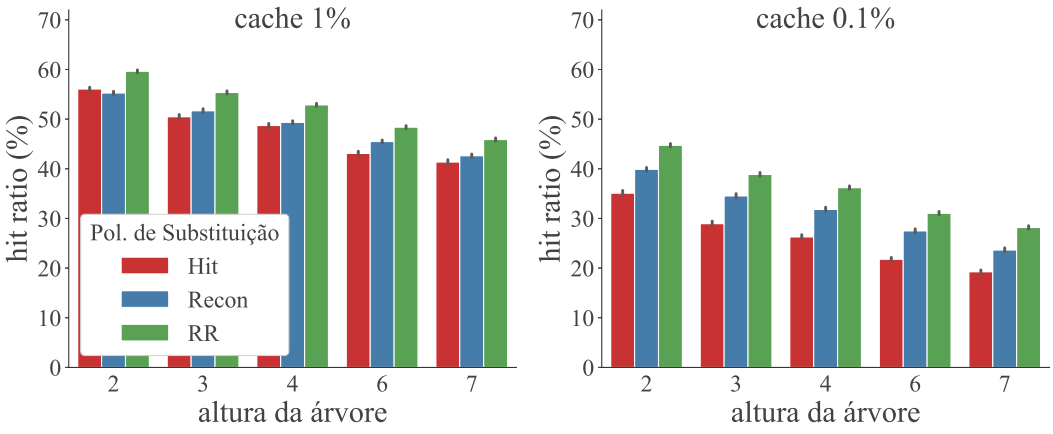
Assim, observamos nas simulações resultados desfavoráveis para as políticas de substituição baseadas em *Hit* e em Tempo de Reconexão. Acreditamos que isso ocorra devido ao fato de que essas políticas geram um viés, persistindo dados nem sempre necessários ao comportamento da rede. Por esse motivo, nos próximos gráficos e tabelas de resultados daqui para frente, serão suprimidos os resultados das duas políticas de substituição citadas. Assim, a política substituição randômica (RR) será usada como coluna inicial nas apresentações dos resultados nas próximas figuras como base de comparação (*baseline*).

Ao analisar as proporções das políticas que efetuam segmentação da *cache* em dois espaços, pode-se verificar que a diferença na métrica *hit ratio* não é muito grande. Isso fica claro na Figura 12b, que compara as políticas que usam segmentação para a replicação do tipo LCE. Nesse caso, a diferença de desempenho entre o SLRU_25 e SLRU_50 é muito pequena, não trazendo análises significativas[2]. Isso ocorre por conta das características particulares da base de dados utilizada (como quantidade de usuários, quantidade de requisições entre outras) sendo necessária uma adaptação para esse parâmetro surtir efeitos relevantes. O mesmo ocorre para a política proposta neste trabalho, a SProt, e o oráculo de avaliação SProt100. Desta forma, os próximos resultados serão apresentadas apenas com as proporções de 25% para a área protegida do *cache*.

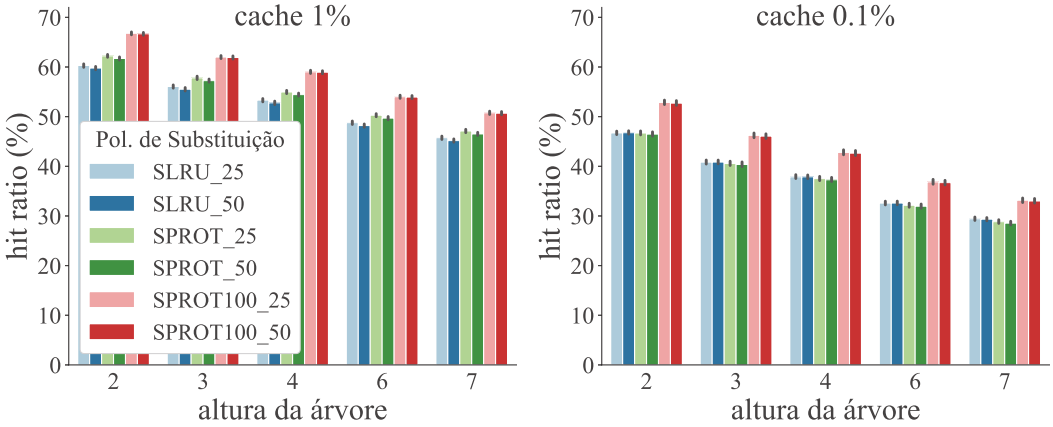
A Figura 13 mostra a **taxa de acertos** para as combinações de estratégias de *cache* selecionadas. As figuras 14, 15 e 16 seguem a mesma metodologia de apresentação dos resultados para o número de acertos em um único salto, número de requisições ao servidor na nuvem e o número de réplicas de atributo, respectivamente. Cada conjunto

[2] Ressaltando que os valores 25 e 50 em SLRU_25 e SLRU_50, correspondem ao percentual de espaço protegido na *cache* exemplificado na descrição da Figura 9.

Figura 12 – *Hit Ratio* com Replicação LCE para comparação.



(a) *Hit* e Reconexão



(b) Segmentados em 25% e 50%

Fonte: Elaborado pelo autor (2021)

de gráficos representa uma estratégia reativa diferente. Sendo o primeiro par **(a)** com a estratégia LCE, o segundo **(b)** com a estratégia LCD e o último **(c)** com a estratégia LEAF.

Para cada par, o gráfico da esquerda apresenta nós de névoa com mais recursos (*i.e.*, 1% do número total de atributos), e o gráfico da direita mostra nós de névoa mais limitados quanto à capacidade de armazenamento (0,1% do número total de atributos). Além disso, cada plotagem apresenta todas as políticas de substituição de *cache* em colunas, e cada conjunto de seis colunas refere-se a diferentes topologias de rede. Os conjuntos à esquerda representam árvores menores e mais amplas, e os da direita referem-se a topologias de rede de árvores mais profundas e estreitas, essas configurações seguem as descrições sobre cenários de topologia de rede da seção 6.1 e pode ficar mais claro pela Figura 10 da mesma seção.

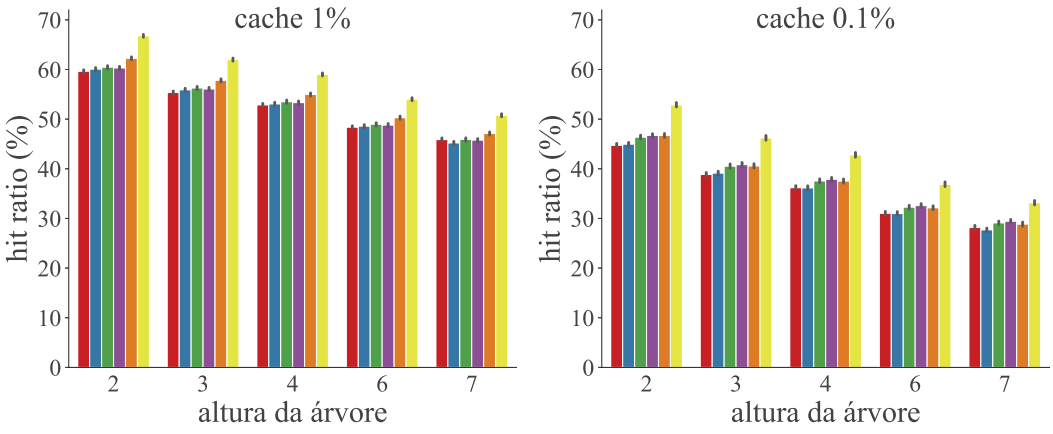
Como comentado anteriormente, este trabalho considera políticas tradicionais de substituição de cache como RR, FIFO, LRU, SLRU, Baseado em Acerto, Baseado em tempo de reconexão. Além disso, duas políticas relacionadas à proposta, a SPROT tradicional (com predição de mobilidade) e o SPROT100 (oráculo). Ou seja, ambos SPROT e SPROT100 são a mesma política de substituição de SPROT apresentada na seção 5. Porém, o SPROT usa a predição com precisão de 45% (alcançada como resultado dos experimentos com aprendizado de máquina), e o SPROT100 usa o oráculo com precisão de 100%.

A Figura 13 mostra as **estratégias oportunistas** reativas que são os fatores mais relevantes que afetam a **taxa de acerto**. Por exemplo, ao comparar os resultados apresentados pelo LCE (Figura 13a), LCD (Figura 13b) e LEAF (Figura 13c), o LCE alcança cerca de 10% mais acertos nos nós de névoa quando usado em conjunto com uma política de substituição se comparar com o LCD ou LEAF para a *cache* de 1% e alcança cerca de 5% mais acertos quando a cache é 0,1%. Esse resultado é esperado visto que o LCE é uma política que cria a maior quantidade de réplicas, aumentando a chance de um acerto ocorrer nos nós de névoa. As estratégias LCD e LEAF fornecem quase o mesmo número de acertos nos nós de névoa quando aplicadas em conjunto com as políticas de substituição RR, FIFO, LRU e SLRU. No entanto, ao aplicar a política de substituição SPROT, a taxa de acerto aumenta 5%, mesmo com uma predição baixa de mobilidade, e aumenta em 10% quando a predição é perfeita, através do oráculo. Esse resultado demonstra que a estratégia LEAF não inunda as *caches*, dando mais oportunidades ao SPROT de aumentar a eficácia da estratégia. A estratégia LEAF combinada com a política de substituição SPROT dá quase os mesmos resultados que LCE combinada com SPROT. Portanto, como esperado, quanto mais precisamente o algoritmo de predição de mobilidade empregado (representado pela predição imperfeita SPROT e perfeita SPROT_100), maior o impacto do SPROT nas taxas de acerto.

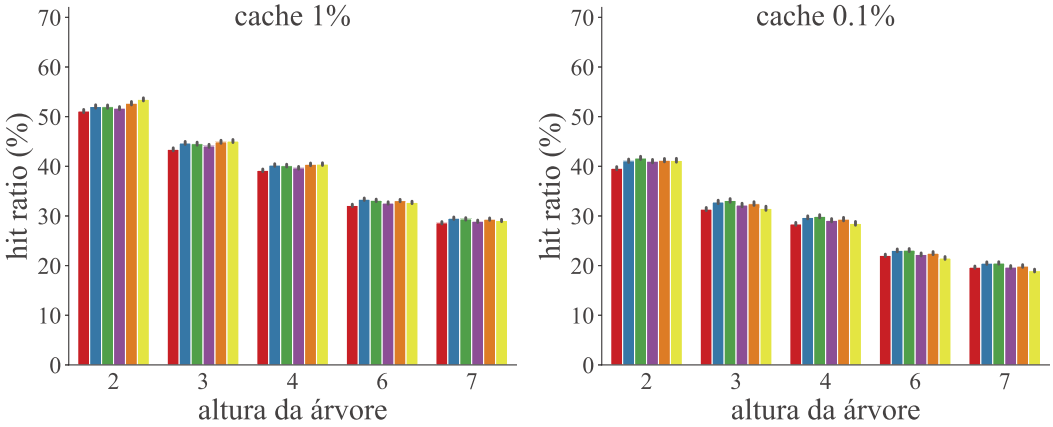
O tamanho da *cache* segue um padrão bem definido, que mostra que o número de acertos aumenta proporcionalmente ao tamanho da *cache*. Em geral, quando temos uma *cache* com 1% do número total de atributos dos usuários, ele apresenta 10% mais acertos do que a *cache* com tamanho de 0,1%. Obviamente, isso ocorre porque quanto mais tamanho tivermos, maior será o número de atributos armazenados e melhores serão as chances de encontrar o atributo demandado, sugerindo que investir em nós de névoa com maior capacidade de armazenamento pode aumentar o desempenho de recuperação de atributos.

A topologia da rede também impõe um resultado bem definido, em todos os casos. O desempenho do mecanismo de *cache* oportunista diminui à medida que a árvore se

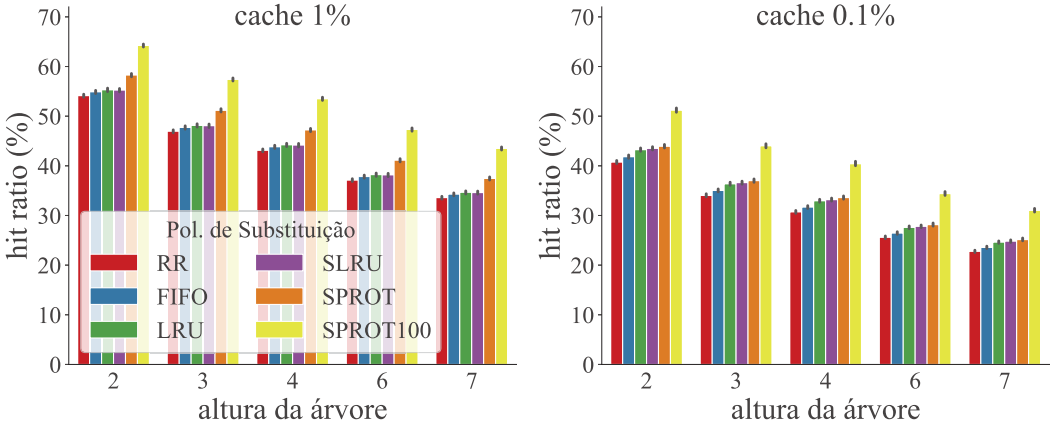
Figura 13 – *Hit Ratio* por estratégia de replicação reativa.



(a) LCE



(b) LCD



(c) LEAF

Fonte: Elaborado pelo autor (2021)

aprofunda. Este resultado também pode ser observado quando – conforme mostrado na Figura 10 – o número de folhas permanece o mesmo para todas as arquiteturas. O que

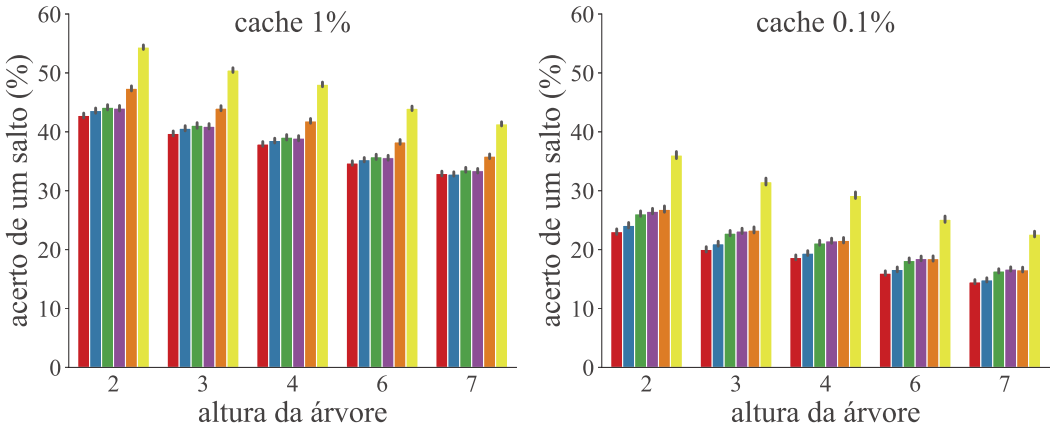
significa que as árvores mais profundas têm mais folhas conectadas por roteador, criando gargalos e uma disputa por atributos nesses nós concentradores.

A Figura 14 mostra a **taxa de acerto de um salto** para cada combinação de estratégias de *cache* reativas, políticas de substituição, tamanhos de *cache* e topologias de rede. As estratégias de *cache* LCE (Figura 14a) e LEAF (Figura 14c) apresentam melhor desempenho do que LCD (Figura 14b) para todas as combinações equivalentes de estratégias de *cache* e políticas de substituição. Por exemplo, tanto LCE quanto LEAF entregam mais de 55% das solicitações de atributo instantaneamente para casos onde a *cache* tem o tamanho de 1% do número total de atributos e 35% quando as *caches* têm o tamanho de apenas 0,1% do número total de atributos. Esses resultados são próximos de serem iguais, não dependendo da política de substituição e da topologia de rede utilizada. Por outro lado, para LCD, observamos que o tamanho do *cache* afeta o desempenho, mas a topologia da rede também o afeta. Isso ocorre porque o LCD tem uma abordagem mais conservadora, diminuindo lentamente os atributos. Portanto, é possível observar que quanto maior a altura da árvore, pior o resultado dessa estratégia. Ao comparar LCE e LEAF, observa-se que ambas possuem estratégias fixas, colocando atributos em toda a extensão da rede até alcançar as folhas. Consequentemente, ambas as estratégias apresentam resultados semelhantes e nenhum impacto sobre a altura da árvore.

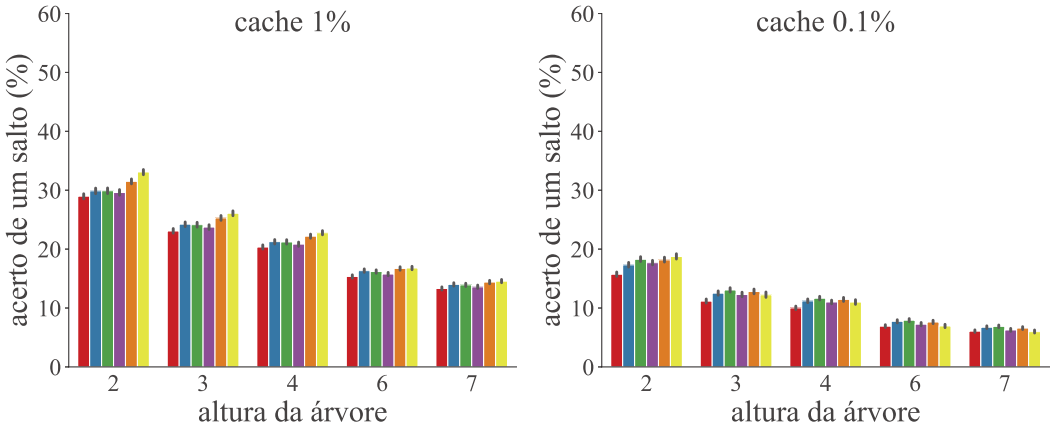
Ainda para a Figura 14, as políticas tradicionais de substituição de *cache*, como RR, FIFO, LRU, SLRU, apresentam resultados semelhantes para todos os casos das estratégias LCE e LEAF (com o mesmo tamanho de *cache*). A estratégia de substituição do SPROT com LCE e LEAF apresenta melhor desempenho do que as políticas de substituição tradicionais. O oráculo (SPROT100) teve um desempenho 12% melhor na taxa de acerto de um salto comparando com as outras politicas avaliadas, e fica 5,5% melhor que o SPROT quando a *cache* é 1% e até 9,2% quando a *cache* é 0,1%. Esses resultados mostram que a proatividade dessas estratégias de substituição coloca os atributos nos lugares certos, aumentando a eficácia da *cache*, resultando em mais acertos de um salto. No LCD (Figura 14b), o SPROT e o SPROT100 não melhoram o desempenho geral, dado seu comportamento conservador em distribuir os objetos as concorrências maiores ocorrem nos primeiros níveis da árvore, não permitindo difundir rapidamente os objetos. Esse comportamento deixa até mesmo as políticas proativas trabalhando muito no tudo ou nada. Com isso SPROT fica até 5% melhor que as outras políticas reativas no LCD, e o SPROT100 melhora esse valor em 2% considerando a *cache* em 1%.

A Figura 15 mostra o **número de requisições realizadas ao servidor na nuvem** para cada combinação de estratégias de *cache* reativas, políticas de substituição, tamanhos de *cache* e topologias de rede. Essa métrica representa o pior caso nas simulações, uma vez que a requisição de um atributo tem que percorrer toda a árvore de topologia da rede até alcançar o servidor na nuvem de identidades. O cenário com maior capacidade de armazenamento, no caso *cache* com 1%, tem valores melhores do que o cenário com menos

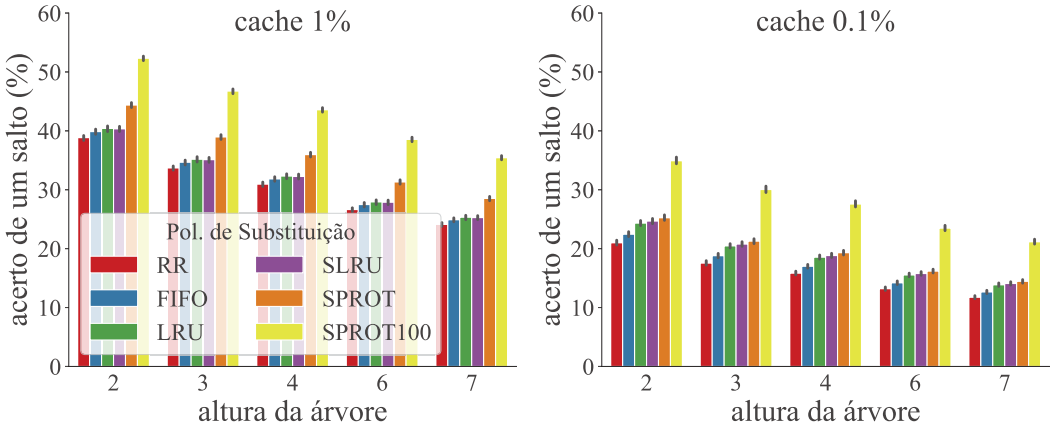
Figura 14 – *One Hop* por estratégia de replicação reativa.



(a) LCE



(b) LCD



(c) LEAF

Fonte: Elaborado pelo autor (2021)

recursos, porém a topologia com alturas diferentes não causam grandes mudanças. As estratégias LCE (Figura 15a) e LEAF (Figura 15c) apresentam melhor desempenho do que

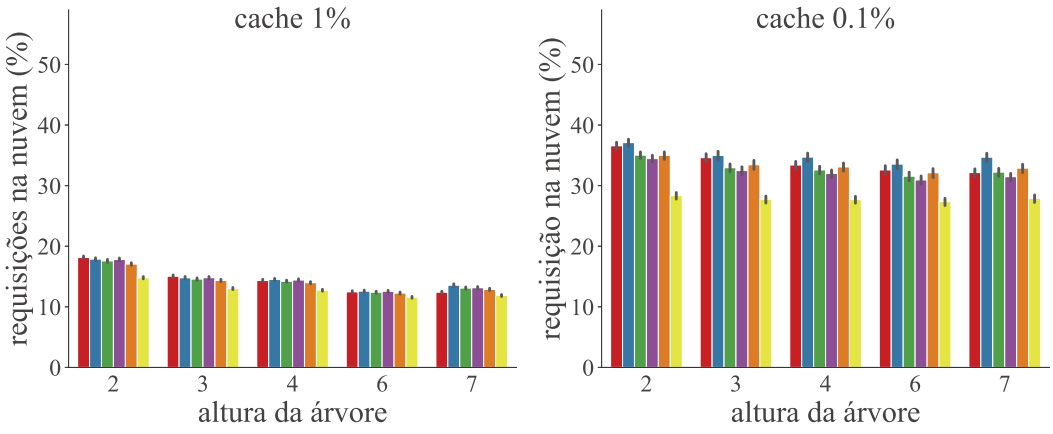
LCD (Figura 15b). Isso ocorre porque o LCD, *sendo uma abordagem mais conservadora*, dissemina menos atributos na rede e assim os nós mais próximos da nuvem precisam realizar mais trocas para atender os níveis abaixo deles e a replicação só alcança o próximo nível.

Ainda para a Figura 15, a política de substituição de *cache* SPROT apresentou um desempenho melhor em relação às estratégias convencionais apenas na replicação LEAF (Figura 15c). Mesmo a política SPROT100/oráculo, que teve o melhor desempenho na LCE (Figura 15a) e LCD (Figura 15b), não teve um bom desempenho na LCD (Figura 15b) quando a capacidade de armazenamento era de apenas 0,1%. Esse comportamento ocorre uma vez que ao realizar a distribuição de forma proativa numa política conservadora, a estratégia SPROT100 inibe outras replicações na árvore da rede. Por exemplo, quando o atributo requisitado não é encontrado no nó, o sistema continua subindo nos níveis da rede em busca desse atributo, contudo os primeiros nós da árvore já fizeram o descarte desse objeto para atender a outras requisições do sistema. Em contrapartida, o SPROT e SPROT100 apresentam desempenho superior às demais estratégias de substituição quando a política de substituição é LEAF. Identificamos que, portanto, políticas com inteligência, priorização e predição de mobilidade para manutenção de atributos, permitem que o *cache* de objetos na folha (névoa) otimize a recuperação desses objetos, reduzindo o custo na sua recuperação e trazendo para mais próximo do usuário seus atributos.

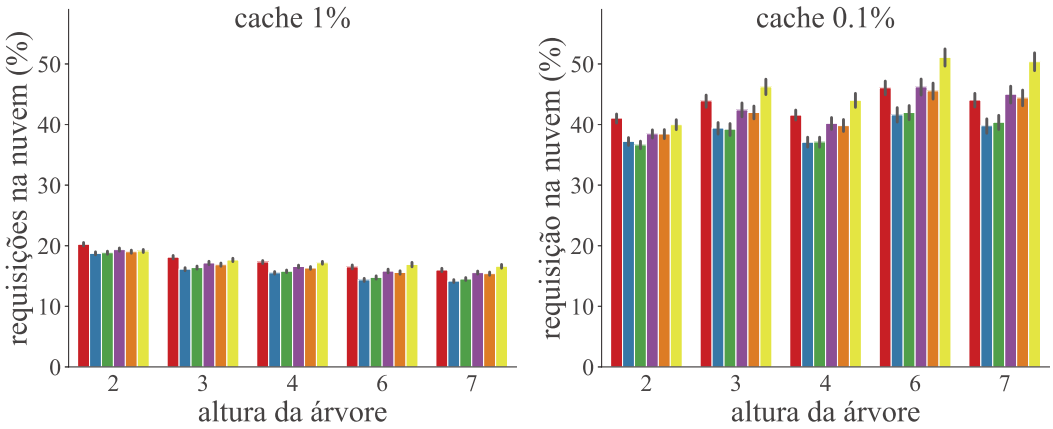
A Figura 16 mostra o **número de réplicas de atributos** que distinguem a estratégia reativa, política de substituição, topologia e tamanho da *cache* usado. Por exemplo, olhando para a estratégia reativa, pode-se observar que o LCE (Figura 16a) cria mais réplicas do que as estratégias LEAF (Figura 16c) e LCD (Figura 16b) ao usar a mesma política de substituição e tamanho de *cache*. Esse resultado é esperado dado que o LCE cria uma cópia em cada nó, enquanto o LCD e o LEAF criam apenas uma por solicitação de atributo, justificando seu menor número de réplicas.

As políticas de substituição de *cache* SPROT e SPROT100 apresentam um comportamento distinto e interessante na Figura 16. Enquanto as políticas tradicionais de substituição de RR, FIFO, LRU e SLRU apresentam resultados semelhantes, o SPROT100 aumenta o número de réplicas quando combinado com LCD e LEAF. No entanto, quando usado em conjunto com o LCE, as réplicas diminuem independentemente do tamanho da *cache* e da altura da árvore. Isso ocorre porque quando o mecanismo cria proativamente uma réplica no LCE, essa réplica é colocada no lugar certo, diminuindo a necessidade de novas réplicas. Ainda assim, quando as estratégias criam apenas algumas réplicas, como LCD e LEAF, ela faz uma adicional no local certo, acelerando a descida dos atributos ao custo de aumentar o número de réplicas. Quando o SPROT usa uma predição imperfeita, observamos que ele apenas aumenta o número de réplicas sem nenhum benefício aparente. No entanto, como se pode ver, mesmo aumentando o número de réplicas com ou sem predição perfeita, a estratégia proativa, em conjunto com o LEAF, custa muito menos para

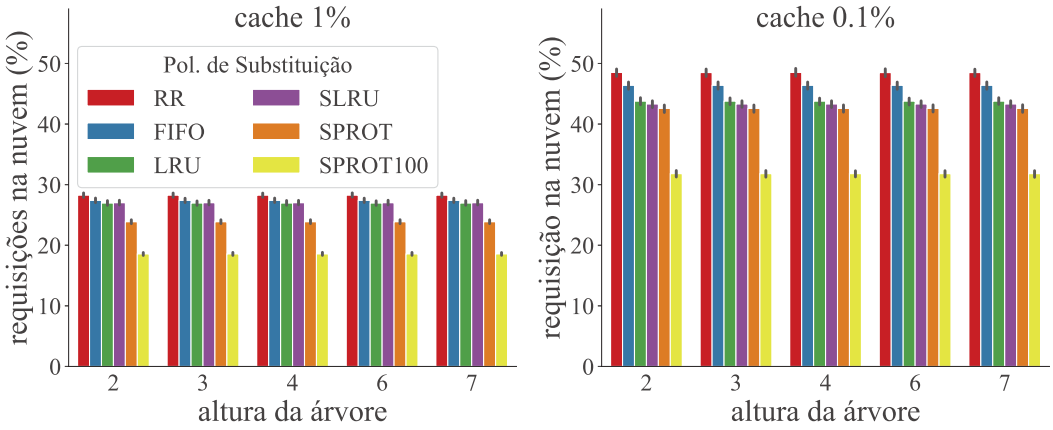
Figura 15 – Requisições na nuvem por estratégia de replicação reativa.



(a) LCE



(b) LCD

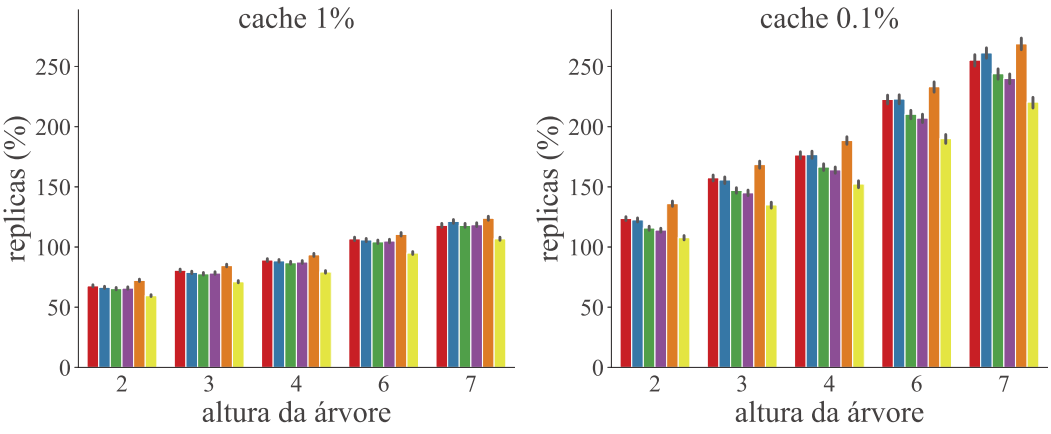


(c) LEAF

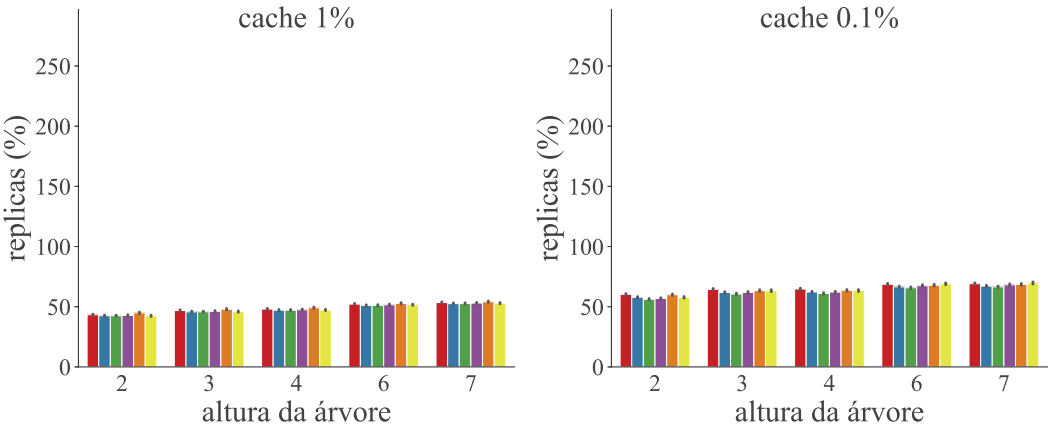
Fonte: Elaborado pelo autor (2021)

se manter segura, com apenas 10% menos *Hit Ratio* RH e ocorre semelhante *One-Hop Hit Ratio* OHHR.

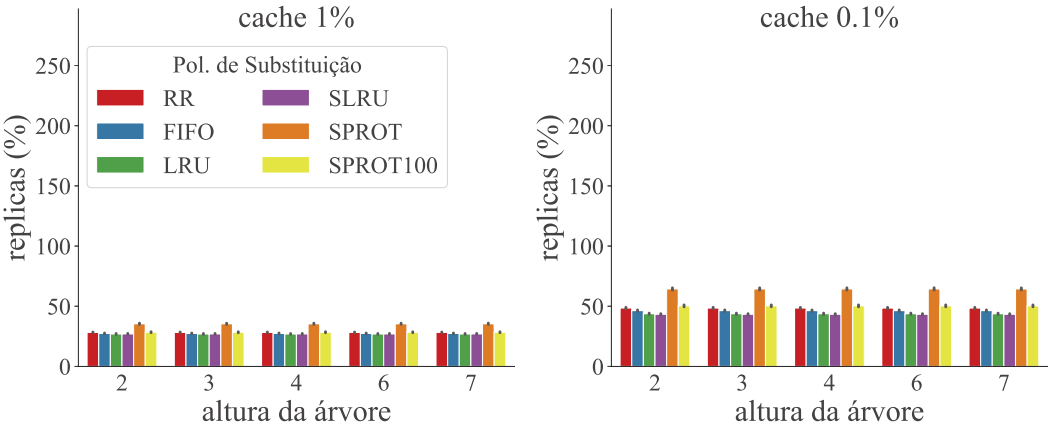
Figura 16 – Replicas por estratégia de replicação reativa.



(a) LCE



(b) LCD



(c) LEAF

Fonte: Elaborado pelo autor (2021)

Ainda na Figura 16, observamos que tamanhos maiores de *cache* podem incorrer em menos réplicas, pois ocorrem menos substituições. Este efeito ocorre para todas as

estratégias reativas. No entanto, é mais evidente para o LCE, onde *caches* de 0,1% criam duas vezes réplicas do que a *cache* de 1%. Além disso, enquanto as outras estratégias mantêm um número linear de réplicas de atributos para todas as alturas das árvores, a estratégia LCE aumenta o número de réplicas de atributos com base na altura das árvores. Ao adotar o LCE, as árvores com mais altura criam mais réplicas do que aquelas com alturas mais curtas.

Em suma, segundo os resultados apresentados, a estratégia reativa LEAF, utilizada em conjunto com a política de substituição SPROT100/oráculo, apresenta o melhor custo-benefício de todas as combinações, como esperado. Indicando que estratégias de replicação de atributos na folha combinados às políticas de substituição que utilizem espaço reservado de *cache* de maneira oportunista é um caminho interessante de investigação. Como hipótese levantada anteriormente, políticas com predição de mobilidade para manutenção de *cache*, por exemplo, tendem a melhorar significantemente a recuperação de objetos no cenário de nuvem-névoa.

A combinação de LEAF-SPROT e LEAF-SPROT100/oráculo, oferecem respectivamente até 10% e 7% menos taxas de acerto em relação ao LCE, mas custa muito menos do que o LCE. Por exemplo, conforme mostrado na Figura 13, o LCE e o LEAF, com *caches* de tamanho 0,1% e árvore de altura 7, usados em conjunto com a política de substituição SPROT100, apresentam os melhores resultados. Nesse caso, o LCE tem uma taxa de acerto cerca de 2% melhor para SPROT100 e 3,5% melhor que o SPROT, em relação a LEAF. Por outro lado, conforme mostrado na Figura 16, o cenário de altura 7 custa 4,41 vezes mais para o LCE do que para o LEAF com SPRTO100 e 4,18 vezes mais com SPROT. Isso mostra que a LCE faz quatro vezes mais replicas no sistema para conseguir poucos pontos percentuais em relação ao LEAF.

A política LCD não funciona bem em nenhuma estratégia oportunista, então a sugestão para quem já tem sistemas implantados nessa forma de replicação devem avaliar trocar para uma política que dissemine mais objetos. Para novas implementações, essa estratégia conservadora, LCD não entrega melhores desempenhos em distribuição de *cache* de atributos. A política LCE entrega melhores resultados ao custo de gerar mais cópias em todo o sistema. Essa sobrecarga de dados pode ser interessante se a busca for os melhores resultados sem se preocupar com os custos empregados na quantidade de dados e informações descentralizadas no sistema. Já a política LEAF entrega bons resultados com menos custo em distribuição de informações no sistema. Se o objetivo for encontrar resultados balanceados tanto de implantação, visto que a *cache* só existe nas folhas, como para manter o sistema com menos dados de atributos compartilhados, a melhor escolha é utilizar o LEAF.

8 CONCLUSÃO

Neste trabalho foram exploradas diversas estratégias de distribuição e substituição do espaço de armazenamento temporário de objetos. Foram utilizados os dados de um sistema de autenticação e autorização de uma grande rede sem fio para avaliar as políticas propostas. Foram conduzidas as investigações do comportamento dos usuários, das combinações de políticas de replicação e substituição, e analisados os impactos práticos por meio de métricas, como: *cache hit*, número de saltos, número de réplicas, e acerto de um salto; a fim de propor melhorias no cenário nuvem-névoa para recuperação de atributos de identidade.

Neste trabalho também foi apresentado um método para incrementar o desempenho da recuperação dos atributos de identidade a serem aplicados no método ABAC, adaptado a um sistema baseado em névoa. O método segue abordagens reativas e proativas, em que usuários e dispositivos são altamente móveis e exigem acesso a vários recursos. O método adota uma arquitetura com pontos de decisão, PDP, distribuídos, e remove a centralidade de um PIP (diretório de atributos e identidades, IdP) hospedado em uma nuvem, distribuindo sua funcionalidade por vários nós de névoa. Em cada solicitação de atributo, o método vê uma oportunidade de replicar os atributos para reduzir a latência de uma decisão de política do sistema de autorização e assim melhorar a eficiência ABAC.

As avaliações empregaram dados de um cenário universitário real com um sistema distribuído por todo o ambiente do *campus*. Os resultados mostraram que o método pode entregar mais de 55% dos atributos instantaneamente devido à estratégia de replicação proposta. Os resultados também mostraram ganhos mais relevantes quando a abordagem proativa utiliza uma predição perfeita, por meio de oráculo (SPROT_100). Em uma predição imperfeita (SPROT), o número de solicitações na nuvem é o mesmo que sem ele, mas o número de réplicas de atributos aumenta quando temos capacidade de armazenamento reduzida. Contudo, com a capacidade de armazenamento maior a política proativa com a predição alcançada mostrou realizar uma boa troca entre réplicas geradas e resultados de entrega de identidades. Assim, o método remove a centralidade PDP e PIP, aumenta a eficiência da rede ABAC e reduz os problemas de atraso que as decisões de política podem causar em aplicações de internet das coisas sem aumentar o custo para manter a segurança e a atualização dos atributos armazenados em *cache*.

No trabalho foi utilizado como base o ABAC na intenção de que ao conseguir dar suporte a ele, conseguiria dar suporte a outros métodos de autorização também. Como o ABAC é um sistema distribuído e granular, realizar a cache de atributos para esse método abre oportunidades para implantar o mesmo sistema de caches para outros métodos, e até mesmo outras aplicações possam utilizar dos mesmos conceitos.

As contribuições desse estudo foram publicadas em conferência internacional do

IEEE *International Conference on Communications* 2021 e em simpósio nacional de importante relevância no tema, como o Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais 2021. Ainda no SBSeg 2021, o trabalho apresentado foi premiado como melhor artigo na trilha principal. Esse estudo também criou a oportunidade de apresentar em *workshop* como o XI Workshop de Gestão de Identidades Digitais trazendo a comunidade acadêmica para a discussão sobre o assunto. Como trabalhos futuros, podemos citar a necessidade da investigação de métodos que alcancem melhores resultados de predição de mobilidade, verificar o comportamento em redes com profundidades diferentes (mais saltos), topologias com formatos diferentes, espaço de armazenamentos diferentes para cada nível da rede, avaliar outros algoritmos de replicação e substituição, e fazer um estudo para sugerir melhores disposições dos APs para a rede analisada.

Ao realizar essa pesquisa também foram encontrados alguns desafios nas áreas de segurança, de mobilidade e implementação que necessitam de uma maior investigação. Na área de segurança temos exemplos, como: o armazenamento de atributos em ambientes distribuídos, principalmente quando os equipamentos utilizados possuem baixo poder de processamento; a comunicação entre os nós para a disseminação dos atributos, que pode ser interceptada ou manipulada por terceiros; a origem dos objetos que pode ser diferentes fontes de identidades e atributos na nuvem; e a privacidade dos objetos armazenados principalmente em relação ao consentimento dos usuários. Foram levantados também desafios em relação à mobilidade como: cenários com alta mobilidade dos usuários; e cenários com eventos fora do padrão como feriados, festas e rotinas incomuns. A experimentação e avaliação dessa pesquisa foi realizada por um simulador de eventos discretos e poderia ser melhor analisada com a utilização de um *testbed*, cuja implantação representa um desafio do ponto de vista de implementação, visto que demanda um ambiente complexo e bem definido na área de autenticação e autorização.

REFERÊNCIAS

1 Agarwal, A., Hennessy, J., and Horowitz, M. (1988). Cache performance of operating system and multiprogramming workloads. *ACM Transactions on Computer Systems (TOCS)*, 6(4):393–431.

2 Anderson, A., Nadalin, A., Parducci, B., Engovatov, D., Lockhart, H., Kudo, M., Humenn, P., Godik, S., Anderson, S., Crocker, S., et al. (2003). extensible access control markup language (xacml) version 1.0. *OASIS*.

3 Bele, S. B. (2018). An empirical study on 'clou key words: Cloud computing, architecture, vm, sla, saas, paas, iaas, daas, cloud service provider, cloud computing metaphor.

4 Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16.

5 Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700.

6 Cao, Y. and Yang, L. (2010). A survey of identity management technology. In *2010 IEEE International Conference on Information Theory and Information Security*, pages 287–293.

7 Castro, T. O., Caitité, V. G., Macedo, D. F., and dos Santos, A. L. (2019). Casa-iot: Scalable and context-aware iot access control supporting multiple users. *International Journal of Network Management*, 29(5):e2084.

8 Cremonezi, B., Nogueira, M., dos Santos, A. L., Vieira, A. B., and Nacif, J. A. M. (2019). Um sistema multinível de distribuição de identidades em névoas computacionais. In *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 543–555. SBC.

9 Fang, H., Qi, A., and Wang, X. (2020). Fast authentication and progressive authorization in large-scale iot: How to leverage ai for security enhancement. *IEEE Network*, 34(3):24–29.

10 Gadde, S., Chase, J., and Rabinovich, M. (2001). Web caching and content distribution: A view from the interior. *Computer Communications*, 24(2):222–231.

11 Gómez-Cárdenas, A., Masip-Bruin, X., Marin-Tordera, E., Kahvazadeh, S., and Garcia, J. (2018). A resource identity management strategy for combined fog-to-cloud systems. In *2018 IEEE 19th International Symposium on"A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 01–06. IEEE.

12 Group, O. C. A. W. et al. (2017). Openfog reference architecture for fog computing. *OPFRA001*, 20817:162.

13 Hazmi, A., Rinne, J., and Valkama, M. (2012). Feasibility study of ieee 802.11ah radio technology for iot and m2m use cases. In *2012 IEEE Globecom Workshops*, pages 1687–1692.

14 Hossain, S. A., Rahman, M. A., and Hossain, M. A. (2018). Edge computing framework for enabling situation awareness in iot based smart city. *Journal of Parallel and Distributed Computing*, 122:226–237.

15 Hu, C. T., Ferraiolo, D. F., and Kuhn, D. R. (2019). Attribute considerations for access control systems. *Special Publication*.

16 Hu, V., Ferraiolo, D. F., Kuhn, D. R., Kacker, R. N., and Lei, Y. (2015). Implementing and managing policy rules in attribute based access control. In *2015 IEEE International Conference on Information Reuse and Integration*, pages 518–525. IEEE.

17 Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al. (2013). Guide to attribute based access control (abac) definition and considerations (draft). *Special Publication*.

18 Hussein, D., Bertin, E., and Frey, V. (2017). A community-driven access control approach in distributed iot environments. *IEEE Communications Magazine*, 55(3):146–153.

19 ITU (2009). Ngn identity management framework. Recommendation Y.2720.

20 Laoutaris, N., Che, H., and Stavrakakis, I. (2006). The lcd interconnection of lru caches and its analysis. *Performance Evaluation*, 63(7):609–634.

21 Le, N.-T., Arif Hossain, M., Islam, A., Kim, D.-Y., Choi, Y.-J., and Jang, Y. M. (2016). Survey of promising technologies for 5g networks. *Mobile Information Systems*, 2016:25.

22 Liu, B., Yang, Y., and Zhou, Z. (2018). Research on hybrid access control strategy for smart campus platform. In *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 342–346. IEEE.

23 Panda, P., Patil, G., and Raveendran, B. (2016). A survey on replacement strategies in cache memory for embedded systems. In *IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pages 12–17. IEEE.

24 Podlipnig, S. and Böszörmenyi, L. (2003). A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)*, 35(4):374–398.

25 Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., and Rana, O. (2019). Fog computing for the internet of things: A survey. *ACM Trans. Internet Technol.*, 19(2).

26 Ranjith, D. and Srinivasan, J. (2013). Identity security using authentication and authorization in cloud computing. *International Journal of Computer & Organization Trends*, 3(4):122–129.

27 Ray, P. P., Dash, D., and De, D. (2019). Edge computing for internet of things: A survey, e-healthcare case study and future direction. *Journal of Network and Computer Applications*, 140:1–22.

28 Siebach, J. and Giboney, J. (2021). The abacus: A new architecture for policy-based authorization. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 7055.

29 Silva, E. F., Muchaluat-Saade, D. C., and Fernandes, N. C. (2018). Across: A generic framework for attribute-based access control with distributed policies for virtual organizations. *Future Generation Computer Systems*, 78:1–17.

30 Systems, C. (2015). Fog computing and the internet of things: Extend the cloud to where the things are. *White Paper*.

31 Trnka, M., Cerny, T., and Stickney, N. (2018). Survey of authentication and authorization for the internet of things. *Security and Communication Networks*, 2018.

32 Weiser, M. (1999). The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review*, 3(3):3–11.