# Uma Política de Cache de Identidades Multinível para Névoas Computacionais

**Airton Ribeiro de Moura Gomes Filho[1], Bruno Cremonezi[2], Edelberto Franco Silva[1], Alex Borges Vieira[1], José Nacif[3], Michele Nogueira[4]**

[1]Depto. de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)
[2]Depto. de Informática – Universidade Federal do Paraná (UFPR)
[3]Instituto de Ciências Exatas e Tecnológicas – Universidade Federal de Viçosa (UFV)
[4]Depto. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
{armgfilho, edelberto}@ice.ufjf.br, alex.borges@ufjf.edu.br,
cremonezi@ufpr.br, jnacif@ufv.br, michele@dcc.ufmg.br

***Abstract.*** *Network services rely upon Authentication and Authorization mechanisms to determine a user's identity to ensure a secure and personalized service provision. However, in current virtual environments, the retrieval of this identity usually relies on distant servers, which can be harmful to an application when it has strict time requirements. Therefore, to minimize the latency of identity retrieval, the fog computing paradigm has to be considered, and retrieval and maintenance policies of identity caches. This paper presents a multi-level cache identity policy with priority based on machine learning techniques to foresee the users' mobility in wi-fi networks. The proposal was evaluated using actual data of a wi-fi network with more than 36 thousand users. The results showed a* $64\%$ *of hit ratio to retrieve an identity on the fog. It also reduces by* $3.42\%$ *the number of hops to recover an identity, staying behind only* $6.35\%$ *of the best solution obtained through the implementation of an oracle.*

***Resumo.*** *Autenticação e autorização em serviços de redes dependem da identidade e dos atributos relacionados. Entretanto, os ambientes atuais para recuperação dessas informações são geralmente baseados em servidores distantes a alguns saltos do usuário, o que pode prejudicar sua aplicação quando há requisitos temporais. Portanto, o paradigma de névoa computacional deve ser levado em consideração, assim como políticas de recuperação e manutenção em cache de identidades. Este trabalho apresenta uma proposta de política de cache de identidade multinível com prioridade baseada em técnicas de aprendizado de máquina para prever a mobilidade de usuários em redes sem fio. A proposta foi avaliada com dados reais de uma rede sem fio com mais de trinta e seis mil usuários. Os resultados demonstram uma taxa de* $64\%$ *de acerto para recuperação de identidades na névoa, e ainda reduz em* $3,42\%$ *o número de saltos para a recuperação de uma identidade, ficando apenas a* $6,35\%$ *da solução ótima, obtida através da implementação de um oráculo.*

## 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 [Weiser 1999]. Um exemplo de tecnologia de sucesso que auxilia na cobertura e expansão das redes, são

as redes móveis. Redes sem fio locais de larga escala provêm, além de cobertura, grande oferta de banda e acesso a diversos serviços. Porém, para fornecer um acesso seguro para estes serviços e ambientes, deve-se levar em consideração requisitos obrigatórios de segurança, como a autenticação e a autorização (A&A) [Trnka et al. 2018].

Para que a A&A possa ser realizada em ambientes de larga escala sem fio é necessário considerar alguns pontos. 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 [Ranjith and Srinivasan 2013]. Devido ao possível atraso na recuperação das identidades, os atuais processos de autenticação e autorização não são capazes de proporcionar a sensação de transparência no acesso a serviços pelo usuário, necessário para diversas aplicações em redes móveis [Hu et al. 2013]. Logo, são de grande importância levantar técnicas que reduzam o tempo de recuperação dessas identidades e que o processo de A&A ocorra eficientemente, por exemplo, como a entrega da identidade tão logo ela for requisitada. Principalmente se for considerado que, para diversos serviços oferecidos através das redes móveis, possuem um forte requisito temporal [Fang et al. 2020].

Neste artigo, será mostrado que políticas de *cache* de identidade podem ser aplicadas através do conceito de névoa computacional, e será apresentado um ganho em relação à redução de número de saltos. Consequentemente, essa redução do número de salto implica em uma melhor latência na recuperação das identidades. Essa redução de latência representa um melhor desempenho para respostas em tomadas de decisões em processos de A&A como um todo. Portanto, com a proposta apresentada será possível a criação de novas soluções de A&A em diversos ambientes e para diversos tipos de aplicações, como Internet das Coisas (*Internet of Things* - IoT), ou para Internet das Coisas Aplicadas na Saúde (*Internet of Health Things* - IoHT), dentre outras.

A fim de mostrar a aplicabilidade da proposta, será sugerido uma nova política de *cache* de identidade baseada na predição de mobilidade do usuário. Através de técnicas de inteligência computacional, será proposto e validado uma política de *cache* de identidades multinível para névoas computacionais. O ganho desta política proposta em relação ao estado da arte é de 2,3%, ficando apenas à 6,35% 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.

O restante deste trabalho está organizado da seguinte forma: na Seção 2 os fundamentos teóricos, na Seção 3 a arquitetura utilizada, na Seção 4 as políticas de distribuições aplicadas, na Seção 5 a proposta da nova política, na Seção 6 a avaliação dos resultados, na Seção 7 a discussão dos trabalhos relacionados e na Seção 8 a conclusão.

## 2. Fundamentação Teórica

A gestão de identidades, ou GId (*IdM - Identity Management*), 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 [ITU 2009]. A seguir serão apresentados os principais conceitos relacionados à Gestão de Identidade e o contexto deste trabalho, mais especificamente

como a *cache* de objetos pode ser utilizada como auxílio à GId no contexto da arquitetura de névoas e nuvens computacionais.

### 2.1. Identidade e Atributos

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 são propostos na literatura, dentre os mais famosos destaca-se o RBAC (*Role-based Access Control*) e ABAC (*Attribute-based Access Control*) [Silva et al. 2018]. 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 esse usuário. Tradicionalmente, esses dados estão armazenados em uma árvore de diretório localizada, por exemplo, na instituição de origem do usuário, também chamados de IdPs (*Identity Provider*). Isso faz com que a cada solicitação dos atributos do usuário para a realização da A&A seja necessário consultar esse diretório remoto, o que pode ser custoso e limitar a aplicação a alguns cenários [Silva et al. 2018].

O ABAC, geralmente, controla as políticas por meio da linguagem XACML (*Extensible Access Control Markup Language*). 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 [Anderson et al. 2003]. XACML foi originalmente criado para o ABAC, mas o RBAC também pode utilizar essa linguagem. A Figura 1 mostra o diagrama da implementação da arquitetura XACML com as mensagens que são trocadas entre as entidades. As entidades são: PEP (*Policy Enforcement Point*), PDP (*Policy Decision Point*), PIP (*Policy Information Point*) e PAP (*Policy Administration Point*). 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 (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.

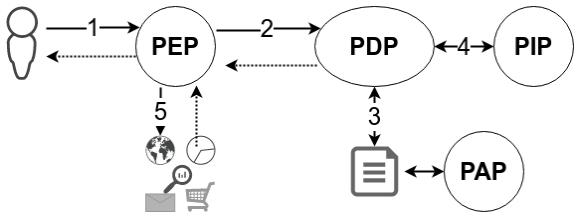



**Figura 1. Arquitetura XACML.**

### 2.2. Cache

Um dos requisitos para um sistema de controle de acesso é a prontidão. A pronta entrega de atributos e identidades logo que o pedido é requisitado. Utilizar uma *cache* para essas identidades é uma forma de atender esse requisito. Além de que 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 [Hu

et al. 2019]. A partir da *cache* de identidades é possível recuperar dados diretamente, sem a necessidade de consultar o diretório de origem do usuário. Em um contexto onde pode-se visualizar o IdP há diversos saltos e enlaces do usuário e o serviço ao qual ele deseja acessar, é interessante que o *cache* de sua identidade esteja o mais próximo possível. Por exemplo, um usuário está caminhando em um ambiente hospitalar com restrição de acesso. Usuários podem ter em sua identidade atributos que o identificam como profissional de saúde, visitante, paciente etc. Assim, para acesso a alguns setores, somente profissionais de saúde, por exemplo, deveriam ter acesso. Esse requisito de autorização consulta a identidade comentada anteriormente, e dependendo da aplicação, é de suma importância que esteja disponível ao serviço para avaliação o mais rápido possível. Da mesma forma, o usuário pode, repetidamente, acessar o mesmo ambiente, o que agilizaria sua avaliação caso sua identidade esteja no *cache* do serviço. A partir dos exemplos ficam claros alguns benefícios em relação à *cache* de identidades, e como o conceito de *cache* é aplicado nesta pesquisa. Sem esquecer que com relação a *cache* é necessário sempre avaliar o impacto em relação às métricas como *hit ratio* (acerto), quantidade de réplicas (cópias), requisições à nuvem (maior custo) e acerto no primeiro salto (menor custo).

## 3. Arquitetura

Este trabalho usa uma topologia hierárquica para a computação em névoa. As camadas são definidas como: nuvem, névoa e dispositivos, conforme Figura 2. A camada Nuvem contém o Servidor de Identidades que possui todas as identidades, sendo considerado o último nível que poderá responder às solicitações de recuperação dos atributos das identidades para a camada da Névoa.

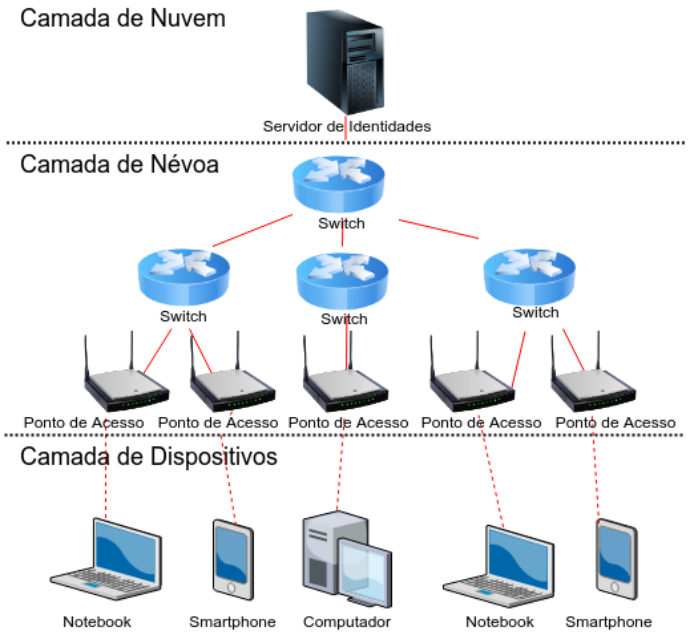



**Figura 2. Topologia de Rede.**

A camada Névoa é composta por múltiplas redes contendo diferentes níveis hierárquicos. Os dispositivos dessa camada terão a capacidade de armazenar temporariamente uma pequena fração das identidades que serão requisitados pelas camadas inferiores. Os dispositivos dessa camada podem ser *switches* e pontos de acessos, porém a

borda inferior da névoa sempre será um ponto de acesso. A camada dos Dispositivos não acessa diretamente a Nuvem e fazem requisições de identidades aos pontos de acesso da camada de névoa. Os dispositivos dessa camada são celulares, notebook e computadores pessoais que precisam identificar seus usuários para realizar o acesso à rede.

As requisições de identidades são feitas de uma camada inferior a uma camada superior. A camada de dispositivos requisita a identidade à camada de Névoa que irá buscar sempre por um mesmo caminho dentro da rede até encontrar a camada de Nuvem. Quando a identidade for encontrada, em alguma das redes da névoa ou no servidor na nuvem, ela será encaminhada ao usuário que requisitou. Sempre que uma identidade for encontrada o sistema usará uma política para replicação para que seja copiado para um local mais próximo do usuário que realizou a requisição. Na Figura 3 ao ser requisitada uma Identidade X pelo celular esse pedido é encaminhado ao ponto de acesso, que verifica se esse objeto está em *cache*, não estando ele verifica no nível acima, no exemplo ao *switch* R2, fazendo isso sucessivamente até encontrar a identidade requisitada.

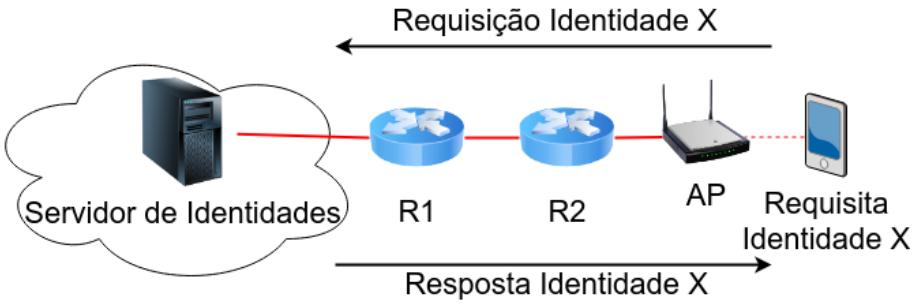



**Figura 3. Requisição de identidade.**

## 4. Políticas de Distribuições

As políticas de distribuições são formas lógicas de difundir os objetos na *cache*. As replicações são feitas ao finalizar uma requisição, e as substituições são feitas sempre que o espaço de armazenamento utilizado alcança o seu limite.

### 4.1. Políticas de replicação

Para cada requisição feita ao servidor de identidades, uma política de replicação é utilizada para distribuir as identidades no sistema. Nesse trabalho foram implementadas políticas de distribuições LCE (*Leave Copy Everywhere*), a LCD (*Leave Copy Down*) e a LEAF (folhas). Essas políticas foram escolhidas por serem estratégias já utilizadas e legadas em relação à distribuição de objetos em *cache*. A política LCE os objetos são replicados por todo o caminho entre o dispositivo até a nuvem [Laoutaris et al. 2006]. Dessa forma distribui as identidades rapidamente na névoa. Ela tem alto custo por gerar muitos dados, redundâncias e ocupa rapidamente o espaço em *cache*. A política LCD deixa uma cópia do objeto requisitado apenas no nível logo abaixo de onde foi encontrado [Laoutaris et al. 2006]. Considerada uma abordagem conservadora, permitindo que objetos mais requisitados se aproximem da borda da rede. Na política LEAF, ou replicação em folhas, sugere que apenas as folhas da névoa podem ter armazenamento [Gadde et al. 2001]. No caso da topologia proposta neste trabalho, apenas os pontos de acesso poderiam armazenar objetos. Essa política de replicação atende cenários em que os dispositivos da névoa não tenham capacidade de armazenamento como ambientes com *switches* legados.

### 4.2. Políticas de substituição

Computação na névoa é uma extensão da computação na nuvem com menos espaço de armazenamento e mais segurança que a nuvem [Hassan et al. 2018]. Neste trabalho o espaço de armazenamento para a *cache* de identidades na névoa é bastante reduzido em relação ao servidor de identidades na nuvem. Assim, quando o espaço em *cache* está totalmente ocupado com objetos já utilizados, mas novos objetos precisam ser inseridos, objetos devem ser substituídos, e isso acontece conforme alguma das políticas de substituição. Nesta seção será comentado sobre as política LRU (*Least Recently Used*), FIFO (*First-In, First-Out*), RR (*Random Replacement*), Baseada em *Hit*, Baseada em Reconexão, e a SLRU (*Segmented Least Recently Used*).

A política LRU descarta os objetos menos utilizados primeiro [Laoutaris et al. 2006]. O 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, obedece a mesma lógica de uma fila normal, onde todos os objetos entram na última posição da fila e quando a fila fica cheia o primeiro objeto é descartado [Panda et al. 2016]. Na 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 [Panda et al. 2016]. Por sua vez, a política baseada em *Hit* (frequência de acerto) é usado uma estratégia para contabilizar os objetos que recebem acertos de utilização, ou seja, objetos que foram reaproveitados ainda quando estavam no espaço de armazenamento são qualificados para se manterem, e quando o espaço fica cheio os objetos com menores quantidades de acertos são removidos [Podlipnig and Böszörmenyi 2003].

Outra política analisada foi a baseada em tempo de Reconexão. Essa política foi implementada com a variável de referência de tempo entre requisições [Panda et al. 2016], então os objetos com menores tempos de reconexão eram mantidos na *cache*. Esse tempo de reconexão foi extraído de uma caracterização da base de dados, porém poderia ser extraído em tempo de execução ou utilizar outra variável temporal. 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, pois descarta os objetos menos utilizados primeiro. Nessa política o espaço da *cache* é dividido em dois segmentos, um deles é um segmento protegido e outro é um segmento de oportunidade [Podlipnig and Böszörmenyi 2003]. Todos os novos objetos são adicionados no segmento de oportunidade colocando novos objetos no início do espaço. Quando o espaço de armazenamento está lotado é removido o último objeto do segmento oportunidade. O algoritmo diferencia do LRU quando tem um acerto em algum objeto, quando ocorre o acerto a posição desse objeto é atualizada para o início do segmento protegido. Quando o 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 reutilizados por usar um espaço reservado e levar mais tempo para serem descartados da *cache*. O tamanho entre os segmentos de oportunidade e reservado podem variar, por exemplo, em 25% para reservado e 75% para oportunidade. Porém, o tamanho entre esses segmentos tem que atender aos padrões de comportamento do uso da *cache*.

## 5. Proposta

Para avaliar a aplicação da arquitetura apresentada anteriormente na Seção 3, foi proposto e validado uma nova política de *cache* de identidade multinível. Essa política é baseada na predição de mobilidade do usuário para as névoas computacionais. A proposta deste trabalho é aplicar o benefício da SLRU com o uso de aprendizado de máquina criando uma nova política de substituição que evita excluir objetos que passaram pela predição do modelo de aprendizado de máquina. Com a *cache* segmentada em dois espaços, o modelo de predição adiciona a identidade prevista no segmento protegido, dessa forma esse objeto tem uma oportunidade maior de ser reutilizado.


**Figura 4. Algoritmo Segmentado Protegido.**

Essa política 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 4. E quando ocorrer uma requisição e for adicionar um novo objeto ele é colocado na posição 6 da Figura 4, e se esse objeto já estava em memória ele vai para o topo do espaço oportunidade ocupando, também, a posição 6. Ao finalizar essa alocação no ponto de acesso que ocorreu a requisição, o algoritmo verifica onde será o próximo ponto de acesso do usuário, e coloca essa identidade no espaço Protegido, que na Figura 4 seria a posição 1. O algoritmo obedece a mesma lógica, 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 4 seria o objeto 10. Nesse trabalho foi utilizado o classificador kNN (*k-Nearest Neighbors*) que atingiu uma acurácia geral de 45%. O algoritmo Segmentado Protegido foi abreviado para Sprot. O oráculo, com acerto de 100%, foi simulado e será chamado de Sprot100.

## 6. Avaliação

Essa seção apresenta a avaliação do simulador de distribuições de identidades e os cenários de topologia de rede. As métricas que serão observadas são o total de réplicas utilizadas, quantidade de requisições à nuvem, quantidade de requisição à borda da rede e o *hit ratio*. Essas métricas serão usadas para analisar os resultados obtidos nas simulações.

A métrica quantidade de réplicas é apresentada em forma de porcentagem e pode ter valores acima de 100%. Isso ocorre porque alguns algoritmos criam muitas copias quando a capacidade de armazenamento é limitada. Com essa métrica pode-se avaliar a eficiência de uma simulação em distribuição dos objetos no contexto geral porque contabiliza a quantidade de réplicas em toda a rede. Essa métrica é calculada pela razão da quantidade de réplicas criadas pela quantidade de solicitações de requisições.

A quantidade de requisições à nuvem é similar com a métrica anterior, apresentada em forma de porcentagem, porém limitada a 100%. Essa informação mostra a quantidade de requisições que necessitaram percorrer todos os níveis da rede até alcançar o servidor de identidades na nuvem, sendo esse o pior caso e de maior custo ao sistema. Essa métrica é calculada pela razão da quantidade requisições realizadas no servidor de identidades pela quantidade de solicitações de requisições.

A quantidade de requisição à borda da rede também é apresentada em porcentagem e mostra a quantidade de requisições que conseguiram a resposta da requisição no primeiro salto da rede, ou seja, a requisição encontrou o objeto requisitado no ponto de acesso. Esse é o melhor caso do sistema, que gera o menor custo e objetivo principal.

O *hit ratio* avalia quão bem o sistema de *cache* funciona como um todo e se o desempenho está otimizado. Um acerto de *cache* é quando a *cache* consegue entregar o objeto requisitado e uma falha de *cache* é quando o objeto requisitado não foi encontrado na *cache*. O *hit ratio* é calculado com a quantidade de acertos de *cache* dividido pela quantidade de acertos somados com a quantidade de erros de *cache*.

As políticas que usam espaço de armazenamento segmentado foram implementadas com proporções de 25% e 50%. Por exemplo, as políticas marcadas como "_25" usam 25% do espaço disponível em *cache* para o armazenamento protegido e 75% para o espaço de oportunidade. Quando marcadas com "_50" a segmentação será em proporções iguais para o espaço protegido e para o espaço oportunidade.

Nesse trabalho foram gerados cenários de topologia de redes para representar redes com profundidades e larguras diferentes. Os cenários possuem o máximo de 62 *switches* que foram organizados por meio de um algoritmo de crescimento de árvore balanceada. A única restrição criada foi para que não ocorresse 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*. O nó raiz da árvore é o servidor de atributos da rede. E os seus filhos são um dos 62 *switches*. Para balancear o crescimento dessa rede e obter a altura de dois níveis, os filhos do nó raiz poderiam variar de 8 a 62 irmãos. Para altura de três níveis os filhos poderiam variar de 4 a 7 irmãos. E a cada nova altura a rede fica com menos liberdade em expandir a largura, visto que possui apenas 62 *switches* para distribuir. Os cenários utilizados nesse trabalho serão identificados como alt2_irm8 e alt6_irm2. Onde "alt" é a abreviação de altura e "irm" é a abreviação de irmãos.

Para avaliar o desempenho das políticas foi implementado um simulador de eventos discretos com capacidade de receber vários pedidos de requisição de acessos. A base de dados utilizada foi a de um *campus* de universidade federal com requisição de acessos de todos os tipos de usuários. Ao todo foram rastreados quatro meses dos sistemas de autenticação da rede sem fio com 108 pontos de acesso. Foram adquiridos 14 milhões de requisições de 36 mil usuários. Os dados nominais dos usuários e dos pontos de acessos foram anonimizados no momento da aquisição para tornar irreversível sua associação aos dados reais. Por não ter a localização física dos pontos de acessos, nem os *switches* aos quais eles estão interligados, foram utilizados os cenários de topologias anteriormente explicados. A última camada da névoa possui 108 pontos de acesso. A base de dados possui tanto requisições que obtiveram sucesso na conexão e permanecem um período

conectado como também conexões com curtos períodos, como se o usuário estivesse se locomovendo ou não estivesse conseguindo realizar a conexão.

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. Nas 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. As técnicas de aprendizado de máquina foram utilizados na política proposta por este trabalho e também foi simulado um oráculo para avaliar a política proposta considerando que este sempre acerta o local da identidade.

Nas simulações a capacidade de armazenamento dos *switches* e dos pontos de acessos para o tamanho da *cache* das identidades dos usuários foi de 0.1% ou 1% do total de identidades no servidor na nuvem. Esses dois cenários permitem avaliar uma capacidade de armazenamento muito pequena e outra grande.

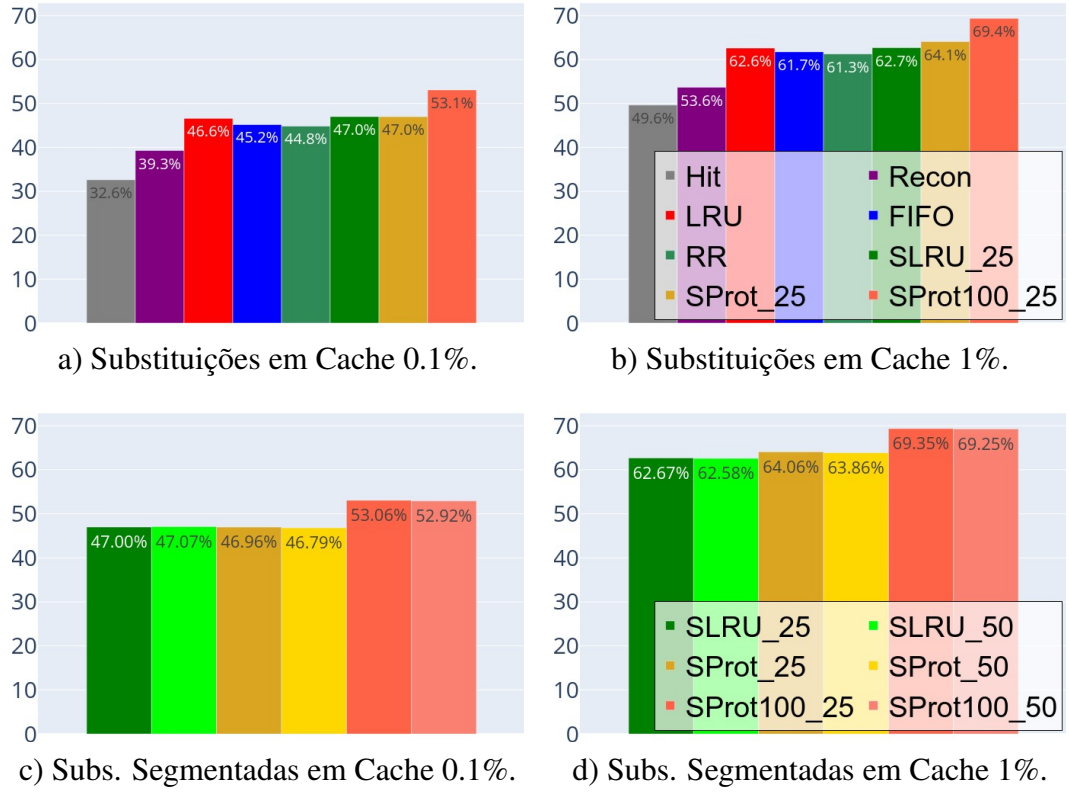
### 6.1. Resultados

Na Figura 5a tem-se a métrica *hit ratio* para todas as políticas de substituições quando a política de replicação é LCE e o tamanho da *cache* é 0.1%. Nessa figura a política de substituição baseado em *hit* e a baseada em tempo de reconexão destoa em relação as outras políticas avaliadas e ficou com um desempenho muito abaixo em relação as outras políticas. Esse resultado também ocorre para quando o espaço de armazenamento é maior em 1% na Figura 5b. O mesmo comportamento acontecem nas demais políticas de replicações, LCD e LEAF, e não foram apresentados em gráficos, pois são comportamentos repetidos. As simulações apresentaram resultados desfavoráveis para as políticas de substituição baseado em *Hit* e em Tempo de Reconexão porque essas políticas geram um viés persistindo dados nem sempre necessários ao comportamento da rede. Por esse motivo nos próximos resultados serão apresentados sem as duas políticas de substituição.

Ao analisar as proporções das políticas que fazem segmentação da *cache* em dois espaços pode-se verificar que a diferença na métrica *hit ratio* não é muito grande como pode ser visualizado na Figura 5c e 5d que são as comparações das políticas que usam segmentação quando a replicação é LCE. A diferença de desempenho entre o SLRU_25 e SLRU_50 é muito pequena tanto em Figura 5c como 5d. O mesmo ocorre para a política proposta neste trabalho a SProt e SProt100 para tanto os próximos resultados serão apresentadas apenas com as proporções de 25%.

As tabelas estão organizadas da seguinte forma: na coluna "Rede" com o nome da topologia de rede utilizada; na coluna "Pol. Substituição" com as políticas analisadas; e resultados das métricas separados pelos tamanhos das *caches* de 0,1% e 1%. As abreviações das métricas utilizadas são "prep" para porcentagem de replicações utilizadas, "pnuv" para porcentagem de requisição para a nuvem e "ps0" porcentagem requisição à borda da rede. Estão em negrito os valores que obtiveram o melhor resultado e sublinhados os valores que estão próximos ao maior em até 0,5% para mais ou para menos.

Ao analisar os resultados pode-se verificar que ao aumentar o tamanho do espaço em *cache* terá um crescimento do *hit ratio* e uma redução no prep comparando as capacidades de armazenamento de 0,1% e 1%. Ao aumentar a altura da rede, comparando entre altura 2 e a altura 6, verifica-se uma redução no *hit ratio* e o incremento dos valores de prep ocorrendo em todas as situações simuladas.




a) Substituições em Cache 0.1%. b) Substituições em Cache 1%.

c) Subs. Segmentadas em Cache 0.1%. d) Subs. Segmentadas em Cache 1%.

**Figura 5. Hit Ratio com Replicação LCE.**

O resultado das políticas de substituições analisadas quando a política de replicação utilizada é a LCE está na Tabela 1. A política SLRU tem grande vantagem quando a *cache* é 0,1% e sobressai nas métricas *hit ratio*, prep e pnuv, sendo que apenas na métrica de ps0 é que a política Sprot tem vantagem. Quando o tamanho da *cache* é 1% a SProt tem melhores resultados nas métricas *hit ratio*, pnuv e ps0, porém a métrica de prep apresenta melhor resultado na política de substituição SLRU. Ao analisar a política Sprot100, política com 100% de acurácia na predição de mobilidade, pode verificar que em todas as métricas ela apresenta melhoras em relação as outras políticas de substituição. O LCE funcionou bem para as substituições SLRU e SProt. Sendo que SLRU sobressai para o tamanho de *cache* em 0,1% e Sprot para a *cache* em 1%. A métrica de Ps0, fica com melhor resultado para o SProt e isso mostra que a predição de mobilidade teve efeito positivo ao entregar a requisição antes da solicitação saltar para o próximo nó da rede.

A outra política de replicação analisada é a LCD e os resultados estão na Tabela 2. A política LRU se destaca na maioria das métricas quando o tamanho de *cache* é de 0,1%, e quando a rede é alt2_irm8 e na métrica ps0 que o SProt fica a frente. Quando a análise é a do tamanho de *cache* em 1% a política SProt ressalta sobre todas as outras políticas de substituição e nas duas redes para a métrica ps0, já as outras métricas ficam em vantagem, na maioria dos casos, para a política LRU. A política Sprot100 com o espaço de *cache* em 0,1% não apresentou melhores resultados que o LRU. Contudo, quando a *cache* em 1% o SProt100 apresentou melhores resultados na maioria das métricas. No LCD não ocorre tantas substituições e com isso a nova política proposta não tem oportunidade de criar as réplicas extras a partir da predição. Com esses dados pode verificar muitos empates

| Rede | Pol. Substituição | Cache em 0,1% | | | | Cache em 1% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | hit_ratio | prep | pnuv | ps0 | hit_ratio | prep | pnuv | ps0 |
| alt2_irm8 | LRU | 46,582 | 114,677 | 34,716 | 56,679 | 62,598 | 59,750 | 16,129 | 75,520 |
| alt2_irm8 | FIFO | 45,171 | 121,383 | 36,765 | 54,086 | 61,739 | 61,972 | 16,733 | 74,562 |
| alt2_irm8 | RR | 44,829 | 123,069 | 36,403 | 51,788 | 61,254 | 63,254 | 16,807 | 73,275 |
| alt2_irm8 | SLRU_25 | **46,995** | **112,787** | **34,070** | 57,181 | 62,673 | **59,558** | 16,148 | 75,640 |
| alt2_irm8 | SProt_25 | 46,964 | 134,336 | 34,660 | **58,013** | **64,060** | 66,533 | **15,697** | **77,762** |
| alt2_irm8 | SProt100_25 | 53,062 | 106,837 | 28,073 | 68,398 | 69,351 | 52,836 | 12,901 | 83,082 |
| alt6_irm2 | LRU | 32,383 | 208,804 | 31,411 | 56,679 | 51,005 | 96,058 | 11,590 | 75,520 |
| alt6_irm2 | FIFO | 31,115 | 221,389 | 33,405 | 54,086 | 50,126 | 99,498 | 12,019 | 74,562 |
| alt6_irm2 | RR | 31,099 | 221,559 | 32,508 | 51,827 | 49,904 | 100,385 | 11,791 | 73,283 |
| alt6_irm2 | SLRU_25 | 32,768 | **205,180** | **30,721** | 57,181 | 51,163 | **95,453** | 11,508 | 75,640 |
| alt6_irm2 | SProt_25 | **32,321** | 230,805 | 31,917 | **58,013** | **52,062** | 102,507 | **11,481** | **77,762** |
| alt6_irm2 | SProt100_25 | 37,016 | 188,530 | 27,114 | 68,398 | 56,824 | 84,623 | 10,208 | 83,082 |

**Tabela 1. Resultados das Políticas Substituição em Replicação LCE.**

nos resultados, com valores muito próximos do melhor resultado o que mostra que as substituições não realizam mudanças significativas na política de replicação LCD.

| Rede | Pol. Substituição | Cache em 0,1% | | | | Cache em 1% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | hit_ratio | prep | pnuv | ps0 | hit_ratio | prep | pnuv | ps0 |
| alt2_irm8 | LRU | 42,160 | **54,957** | **36,009** | 45,043 | 56,270 | **35,523** | **16,135** | 64,477 |
| alt2_irm8 | FIFO | 41,490 | 56,866 | 36,737 | 43,134 | 55,580 | 36,743 | 16,467 | 63,257 |
| alt2_irm8 | RR | 39,810 | 59,695 | 40,809 | 40,305 | 54,820 | 37,217 | 17,694 | 62,783 |
| alt2_irm8 | SLRU_25 | 41,510 | 55,630 | 37,823 | 44,370 | 56,040 | 35,616 | 16,515 | 64,384 |
| alt2_irm8 | SProt_25 | **41,660** | 58,619 | 37,855 | **45,242** | **56,380** | 38,243 | 16,561 | **65,475** |
| alt2_irm8 | SProt100_25 | 41,560 | 56,682 | 39,491 | 46,500 | 57,660 | 35,507 | 16,293 | 67,886 |
| alt6_irm2 | LRU | **23,630** | **63,590** | **41,099** | **36,410** | **37,540** | **41,759** | 12,576 | 58,242 |
| alt6_irm2 | FIFO | 23,420 | 64,822 | 41,125 | 35,178 | 37,220 | 42,759 | **12,526** | 57,241 |
| alt6_irm2 | RR | 22,440 | 67,038 | 45,331 | 32,962 | 36,190 | 43,333 | 14,207 | 56,667 |
| alt6_irm2 | SLRU_25 | 22,790 | 65,091 | 45,092 | 34,909 | 36,980 | 42,080 | 13,418 | 57,920 |
| alt6_irm2 | SProt_25 | 22,940 | 65,757 | 44,707 | 35,828 | 37,250 | 43,465 | 13,327 | **59,051** |
| alt6_irm2 | SProt100_25 | 22,050 | 66,673 | 49,751 | 34,569 | 37,230 | 41,998 | 14,156 | 60,293 |

**Tabela 2. Resultados das Políticas Substituição em Replicação LCD.**

A última tabela de avaliações é da política de replicação LEAF na Tabela 3. Para todas as redes e para os dois tamanhos de *cache* a política proposta SProt teve melhores resultados em todas as métricas exceto na métrica prep. A política de replicação SLRU apresentou melhor resposta para a métrica prep em todos os casos. A política Sprot100 teve melhores resultados em todas as redes e para os dois tamanhos da *cache* nas métricas *hit ratio*, pnuv e ps0. Contudo, a política SLRU obteve melhores resultados na métrica prep. A replicação LEAF funcionou bem com a SProt mesmo com a quantidade de réplicas maiores. Em média foram criadas 14% mais réplicas com as predições.

A política de replicação LCE difunde rapidamente os objetos na *cache*, fazendo com que a métrica prep fique bastante elevada. A política LCD, sendo mais conservadora tem um número menor de objetos replicados nas simulações. Na política LEAF, apenas os pontos de acesso recebem espaço de armazenamento, portanto é a que tem menores valores prep entre as três políticas de replicação. A métrica pnuv nos informa qual estratégia usada teve maiores requisições no servidor na nuvem, sendo esse o caso mais custos ao

| Rede | Pol. Substituição | Cache em 0,1% | | | | Cache em 1% | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | hit_ratio | prep | pnuv | ps0 | hit_ratio | prep | pnuv | ps0 |
| alt2_irm8 | LRU | 43,485 | 43,321 | 43,321 | 56,679 | 57,657 | 24,480 | 24,480 | 75,520 |
| alt2_irm8 | FIFO | 42,063 | 45,914 | 45,914 | 54,086 | 56,717 | 25,438 | 25,438 | 74,562 |
| alt2_irm8 | RR | 40,921 | 48,124 | 48,124 | 51,876 | 55,494 | 26,734 | 26,734 | 73,266 |
| alt2_irm8 | SLRU_25 | 43,772 | **42,819** | 42,819 | 57,181 | 57,776 | **24,360** | 24,360 | 75,640 |
| alt2_irm8 | SProt_25 | **44,255** | 63,396 | **41,987** | **58,013** | **59,983** | 32,666 | **22,238** | **77,762** |
| alt2_irm8 | SProt100_25 | 51,333 | 49,978 | 31,602 | 68,398 | 66,333 | 25,560 | 16,918 | 83,082 |
| alt6_irm2 | LRU | 27,783 | 43,321 | 43,321 | 56,679 | 40,506 | 24,480 | 24,480 | 75,520 |
| alt6_irm2 | FIFO | 26,632 | 45,914 | 45,914 | 54,086 | 39,584 | 25,438 | 25,438 | 74,562 |
| alt6_irm2 | RR | 25,704 | 48,174 | 48,174 | 51,826 | 38,399 | 26,737 | 26,737 | 73,263 |
| alt6_irm2 | SLRU_25 | 28,018 | **42,819** | 42,819 | 57,181 | 40,624 | **24,360** | 24,360 | 75,640 |
| alt6_irm2 | SProt_25 | **28,415** | 63,396 | **41,987** | **58,013** | **42,840** | 32,666 | **22,238** | **77,762** |
| alt6_irm2 | SProt100_25 | 34,529 | 49,978 | 31,602 | 68,398 | 49,625 | 25,560 | 16,918 | 83,082 |

**Tabela 3. Resultados das Políticas Substituição em Replicação LEAF.**

sistema. Sendo que as políticas de substituição SLRU, LRU e SProt ficam em destaque, e de forma mais objetiva LCD com LRU e alguns casos com FIFO, LEAF com SProt, LCE e *cache* em 0,1% SLRU, LCE e *cache* em 1% com SProt. Ao analisar a métrica ps0, que é a resposta das requisições logo na borda da rede, a política de substituição SProt se mostra eficiente para as políticas de replicação LCE e LEAF, porém na LCD a estratégia do SProt só consegue bons resultados quando o tamanho da *cache* é maior em 1% e quando a *cache* é 0,1% o LRU tem melhor na maioria das redes.

As métricas anteriores estão indiretamente ligadas ao *hit ratio*, pois ele afere de forma geral como foi o desempenho da simulação. Na estratégia LCE com *cache* em 0,1% o SLRU teve resultados superiores, no entanto com a *cache* em 1% o SProt respondeu melhor. Na LCD a política de substituição LRU teve melhor *hit ratio* em todas as execuções, exceto na rede alt2_irm8 com *cache* em 1% que o *hit ratio* foi melhor com Sprot. Já na LEAF o SProt apresentou melhor desempenho em *hit ratio* em todas as execuções.

## 7. Discussão

O problema da ineficiência dos métodos A&A para recuperação das identidades foi apresentado de forma mais profunda por [Hu et al. 2013]. Neste trabalho, os autores discutiram sobre como usualmente as recuperações de identidades necessitam de um envio de uma mensagem através da rede. De forma geral, este trabalho mostra a ineficiência dessa abordagem e como isso afeta o provisionamento de um determinado serviço. Desde então, diversos trabalhos possuem como alvo essa ineficiência na recuperação das identidades [Hu et al. 2015, Bhatt 2018]. Dentre as possíveis formas de se resolver essa ineficiência, um conjunto de trabalhos apresentam as *caches* de identidades e demonstram como podem ser utilizadas pelo sistema para oferecer o desempenho necessário.

O trabalho de [Liu et al. 2018] apresentou um mecanismo de *caching* multinível baseado em análises estatísticas de um *campus* universitário. Em sua proposta, os autores propuseram um sistema de três camadas de A&A similar ao apresentado neste trabalho. 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.

No trabalho de [Siebach and Giboney 2021] é apresentado uma nova arquitetura para A&A com simplificação do processo de autorização, eles protegem os dados dentro do domínio da rede de forma mais eficiente e oferecem mais controle sobre o acesso aos dados. O ganho real da arquitetura é poder mudar as tecnologias dentro do domínio sem precisar reescrever toda a lógica de A&A, e o domínio pode usar atributos que são controlados e mantidos por outros sistemas sem precisar conhecer a implementação dos terceiros. Neste trabalho não foi proposto uma mudança na arquitetura como no [Siebach and Giboney 2021], porém foram utilizados conceitos parecidos ao propor que a entidade PDP do XACML atue dentro dos nós da rede.

O [Cremonezi et al. 2019] propõe o uso de armazenamento nos nós da rede usando as identidades como serviços e os objetos são persistidos por meio de políticas de *cache* conhecidas. Ele avalia a distribuição em diferentes capacidades de armazenamento e comprovam que os algoritmos probabilísticos possuem um desempenho melhor que os demais. Este trabalho surgiu da necessidade de verificar melhor os algoritmos de replicações e substituições sugerido em [Cremonezi et al. 2019]. A pesquisa foi evoluída usando uma entrada de dados real, antes era uma entrada simulada pela distribuição Zipf. E neste trabalho é proposto uma política de substituição com predição de mobilidade.

## 8. Conclusões

Nesse artigo foi avaliado uma nova política de substituição para auxiliar na distribuição de identidades em névoas computacionais quando os usuários têm alta mobilidade e precisam de múltiplos recursos. Para distribuir as identidades foi necessário descentralizar o PIP e o PDP movendo esses componentes para cada nó da rede. A política proposta faz uma segmentação do espaço em *cache* em dois espaços, protegido e oportunidade, e utiliza a predição de mobilidade para os objetos colocados na área protegida. A avaliação foi feita com um simulador de eventos discreto e utilizou os dados de acesso de uma universidade federal. Os resultados mostram 64% de acerto na recuperação de identidades na névoa, redução de 6,1% em requisições ao servidor na nuvem e 6,2% mais entregas de identidades nos pontos de acesso do que as outras políticas. A política proposta como uma solução ótima através de um oráculo tem o potencial de atingir até 69,3% de acerto e fica 5,2% melhor que o alcançado com a predição atual quando no LCE. A solução ótima alcançou 66,3% de acerto e fica 6,35% melhor que a predição atual quando no LEAF, e de forma geral foi alcançado uma redução de 16,5% das requisições feitas ao servidor na nuvem e 16,6% mais entregas de identidades nos pontos de acesso comparado com as outras políticas propostas. Portanto, a proposta apresentada melhora a eficiência do uso da rede e reduz consideravelmente as requisições feitas nos servidores na nuvem além de entregar mais identidade nos pontos de acessos. Como trabalhos futuros, fica a implementação de uma predição que alcance melhores resultados de predição de mobilidade, verificar o comportamento em redes com profundidades diferentes e avaliar outros algoritmos de replicação e substituição.

## Agradecimentos

Os autores agradecem à UFJF, CAPES, FAPEMIG (APQ-00999-18), FAPESP (2018/23062-5) e ao CNPq (processos número 426701/2018-6, 313844/2020-8 e 432204/2018-0) pelo apoio financeiro para o desenvolvimento desta pesquisa.

## Referências


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*.

Bhatt, S. (2018). *Attribute-Based Access and Communication Control Models for Cloud and Cloud-Enabled Internet of Things*. PhD thesis, Univ. of Texas.

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 *Simp. Brasileiro de Redes de Computadores e Sistemas Distribuídos*. SBC.

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

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

Hassan, K., Javaid, N., Zafar, F., Rehman, S., Zahid, M., and Rasheed, S. (2018). A cloud fog based framework for efficient resource allocation using firefly algorithm. In *Int. Conf. on Broadband and Wireless Computing, Communication and Applications*.

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

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 *Int. Conf. on Information Reuse and Integration*. IEEE.

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*.

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

Laoutaris, N., Che, H., and Stavrakakis, I. (2006). The lcd interconnection of lru caches and its analysis. *Performance Evaluation*.

Liu, B., Yang, Y., and Zhou, Z. (2018). Research on hybrid access control strategy for smart campus platform. In *Advanced Information Technology, Electronic and Automation Control Conf.* IEEE.

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

Podlipnig, S. and Böszörmenyi, L. (2003). A survey of web cache replacement strategies. *Computing Surveys*.

Ranjith, D. and Srinivasan, J. (2013). Identity security using authentication and authorization in cloud computing. *Int. Journal of Computer & Organization Trends*.

Siebach, J. and Giboney, J. (2021). The abacus: A new architecture for policy-based authorization. In *Hawaii Int. Conf. on System Sciences*.

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*.

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

Weiser, M. (1999). The computer for the 21st century. *Mobile Computing and Communications Review*.