Controle de Acesso IoT Escalável e Ciente de Contexto Suportando Múltiplos Usuários

Tiago O. Castro[1], **Daniel F. Macedo**[1], **Aldri Santos**[2]

[1]UFMG, Departamento de Ciência da Computação, Belo Horizonte, Brasil

[2]UFPR, Departamento de Informática, Curitiba, Brasil

{tiago.oliveira,damacedo}@dcc.ufmg.br, aldri@inf.ufpr.br

Resumo. *A Internet das Coisas (IoT) suporta múltiplos usuários e serviços atuando sobre dispositivos heterogêneos, implementando aplicações cientes do contexto. Assim, novos modelos de controle de acesso precisam ser criados e aperfeiçoados com o intuito de prover um gerenciamento ciente de contexto que seja mais responsivo, escalável, seguro e autônomo. Este trabalho apresenta um modelo de controle de acesso baseado em atributos, que fornece a resolução de conflitos e a delegação de acesso em um ambiente multi-usuário. Tendo a escalabilidade em vista, propomos o caching de permissões, bem como um modelo de processamento no qual os dispositivos que possuem poder computacional necessário realizam parte do processamento das políticas. O modelo foi implementado como parte da arquitetura ManIoT e avaliado por experimento num testbed para demonstrar sua eficiência. Resultados mostram que o modelo acelera o gerenciamento de acesso, apoiando ambientes onde diversos usuários e aplicações acessam os dispositivos concorrentemente.*

Abstract. *Internet of Things (IoT) supports many users and services controlling heterogeneous devices, implementing context aware applications. Thus, new access control models must be created in order to support more responsive, scalable, secure and autonomous management. This paper presents an attribute-based access control model, which applies conflict resolution and access delegation in a multi-user environment. With scalability in mind, we propose the caching of access permissions, as well as a policy model in which the devices with enough computational power perform part of the processing. The proposed model was implemented as part of the ManIoT architecture and evaluated by experiment on a testbed to demonstrate its efficiency. Results show that our model accelerates the access management, supporting environments where multiple users and applications seek to access devices concurrently.*

1. Introdução

A "Internet das Coisas" (*Internet of Things* - IoT) engloba objetos do dia a dia que se comunicam visando realizar tarefas em colaboração [Atzori et al. 2010]. A IoT tem estado cada vez mais em evidência devido às inúmeras aplicações e produtos que tem surgido na coleta, monitoramento e comunicação de atividades. Os exemplos de uso são diversos, como pulseiras que controlam a qualidade do sono e calorias queimadas [Sony 2017], relógios que se comunicam com o celular informando mensagens e ligações, dispositivos que permitem identificar intrusos e reagir a essas situações [Denox 2017].

Apesar de diversos dispositivos IoT serem individuais, outros dispositivos geram informações ou realizam tarefas que são do interesse de múltiplas pessoas [Ashton 2011]. Por exemplo, todos os moradores de um prédio irão controlar um portão eletrônico, uma lâmpada em um corredor e um sensor de temperatura. Além disso, os sensores podem ser usados para compor diversas aplicações (ou tarefas), como ativar as luzes no momento que o proprietário chega em casa, ou ligar luzes esporadicamente quando o proprietário está viajando, para dar a impressão que há pessoas em casa [Gubbi et al. 2013].

Diversas pesquisas têm buscado gerenciar de maneira segura tantos os dispositivos quanto os serviços [Hemdi and Deters 2016, Hernández-Ramos et al. 2013, Hussein et al. 2017]. Parte importante desse gerenciamento é o controle de acesso de múltiplos usuários aos diversos recursos de um ambiente [Sandhu and Samarati 1994]. Inúmeros fatores tornam o gerenciamento de acesso em ambientes IoT uma tarefa não-trivial, entre eles: A quantidade e heterogeneidade dos dispositivos, que podem ter variações no poder computacional; a dinamicidade do ambiente, devido à entrada e saída de usuários e dispositivos; as várias aplicações e usuários controlando os mesmos dispositivos, podendo gerar conflitos de acesso; o alto tempo de acesso a informações de contexto em algumas plataformas IoT ou para alguns tipos de grandezas sensoreadas. No entanto, os modelos tradicionais de controle de acesso não consideram todas essas características.

O modelo de controle de acesso baseado em atributos, ABAC (*Attribute Based Access Control*), é um modelo recente que vem ganhando popularidade frente a modelos mais tradicionais. Ele se caracteriza por definir as políticas de acesso com base nos atributos de entidades presentes em um cenário, como usuários, objetos e o ambiente [Hu et al. 2013]. Com isso, este modelo possibilita a incorporação da percepção do contexto, o uso de informações dos dispositivos e sensores (localização, temperatura, luminosidade, etc) ao gerenciamento de acesso, de modo a prover um controle de acesso que se adeque melhor as características da IoT mencionadas anteriormente, seja mais eficiente e seguro. Poucos trabalhos usam o ABAC como ponto de partida para o controle de acesso em IoT, e estes não lidam com problemas de eficiência, heterogeneidade, e escalabilidade.

Este trabalho apresenta um modelo de controle de acesso para gerenciar o acesso de múltiplos usuários aos recursos e dispositivos dos ambientes de IoT. O modelo proposto tem como base o modelo ABAC, e utiliza algoritmos para identificação e solução de conflitos de acesso dos usuários. O modelo também emprega uma sistema hibrido que divide o processamento de acesso entre o *gateway* e os dispositivos para tornar seu gerenciamento mais escalável. Além disso, para reduzir o custo de obtenção de contexto, o modelo usa *caching* de politicas que agilizam o processamento. O modelo foi implementado como parte da arquitetura ManIoT e avaliado num *testbed* para demonstrar sua eficiência. Resultados mostram que o modelo acelera o gerenciamento de acesso, apoiando ambientes onde usuários e aplicações acessam os dispositivos concorrentemente.

O artigo está organizado como segue. A Seção 2 discute os trabalhos relacionados. A Seção 3 descreve o modelo proposto de controle de acesso, a sua resolução de conflitos, bem como a delegação e revogação de permissões. A Seção 4 traz algoritmos para aumentar a escalabilidade e rapidez do processamento das políticas. A Seção 5 traz uma avaliação do sistema. A Seção 6 conclui o artigo e relaciona os trabalhos futuros.

2. Controle de Acesso em IoT

Esta seção apresenta os modelos de controle de acesso em IoT existentes na literatura e as limitações quanto ao uso em cenários mais complexos com diversos dispositivos.

O ACL (*Access Control Lists*) é o modelo mais tradicional de controle de acesso. Nele listas guardam tuplas do tipo (usuário, permissões). No entanto, o ACL tem uma limitação de escalabilidade, pois à medida em que o número de objetos e usuário cresce, as listas se tornam muito grandes e de difícil gerenciamento. O RBAC (*Role Based Access Control*) foi proposto para tratar o problema de gerenciamento de várias listas [Sandhu and Samarati 1994]. Nele são criadas diferentes funções (também chamadas de papéis) que os usuários podem assumir em um sistema, associando um conjunto de permissões a essas funções. As funções diminuem o número de permissões armazenadas e facilitam o gerenciamento. Contudo, o RBAC necessita que o administrador estabeleça as funções *a priori*, bem como o mapeamento de funções e usuários. Isso torna-se inapropriado para cenários IoT, que são muito dinâmicos e possuem muitos dispositivos.

Há trabalhos que adaptam o modelo RBAC à IoT. [Zhang and Tian 2010] propuseram o modelo CBAC (*Context Based Access Control*), uma extensão do RBAC, onde restrições de contexto podem ser associadas às funções, assim permissões de uma função somente são ativadas quando as restrições são verificadas. [Liu et al. 2017] foca em um ambiente de IoT industrial, que é caracterizado pela necessidade de compartilhamento de recursos, pela colaboração na realização dos processos e por ser composto de diferentes domínios. Mesmo buscando uma adaptação para a IoT, a limitação decorrente da necessidade de se atribuir funções pré-determinadas aos usuários ainda existe.

Outro conceito utilizado para o controle de acesso em IoT é o da *capability*, que consiste em um *token* entregue a um usuário contendo permissões de acesso sobre um dispositivo. Esse *token* precisa ser apresentado a cada acesso, e pode conter informações como duração das permissões, se elas foram concedidas por uma entidade central ou por algum outro usuário, etc. Trabalhos como [Gusmeroli et al. 2012, Anggorojati et al. 2012, Mahalle et al. 2013] utilizam *capabilities*, fazendo uso de uma entidade intermediária entre dispositivos e usuários que realiza o controle de acesso.

Já em [Hernández-Ramos et al. 2013], os autores propõem um modelo distribuído de controle de acesso, onde os próprios dispositivos controlam o acesso aos seus recursos. Infelizmente, a solução totalmente descentralizada não é prática para dispositivos com capacidades restritas de processamento, como pode ocorrer em alguns dispositivos IoT. Em [Hussein et al. 2017], é proposto um modelo de comunidade para objetos inteligentes heterogêneos, onde os objetos que possuem mais recursos realizam o controle de acesso daqueles que possuem menos recursos. Contudo, o trabalho assume que o usuário ou aplicação será identificado por um dispositivo com capacidade de memória para armazenamento dos *tokens*, sendo que nem sempre isso será verdade.

Em [Hu et al. 2013] foi proposto o modelo ABAC (*Attribute Based Access Control*) que utiliza os atributos dos usuários, dos objetos e do ambiente para criar as políticas de acesso. Essa estratégia diminui o número de políticas quando comparado aos modelos anteriores e dá maior flexibilidade ao sistema, pois permite a modelagem do contexto por meio dos atributos. Por outro lado, o modelo ABAC possui limitações como a maior complexidade para verificar regras baseadas em atributos, a dificuldade

em garantir o *Least Privilege*, e o problema do "Confused Deputy", onde um usuário/aplicação confunde o sistema para que ele realize ações não autorizadas em seu benefício. [Hemdi and Deters 2016] desenvolveram um protótipo empregando o ABAC. No entanto, este trabalho possui limitações em cenários com dispositivos heterogêneos.

3. Modelo de Controle de Acesso Baseado em Percepção ao Contexto

Esta seção descreve o modelo de controle de acesso proposto, que suporta múltiplos usuários a partir da Percepção de Contexto na IoT. Inicialmente, define-se as entidades do modelo, que levam em consideração os usuários/aplicações, dispositivos e principalmente o seu contexto e do ambiente. Em seguida, são apresentadas as políticas para o controle de acesso. Por último, a seção mostra como é feita a delegação e revogação de permissões e como são solucionados os conflitos acesso. As entidades do modelo são descritas abaixo e visualizadas na Figura 1:

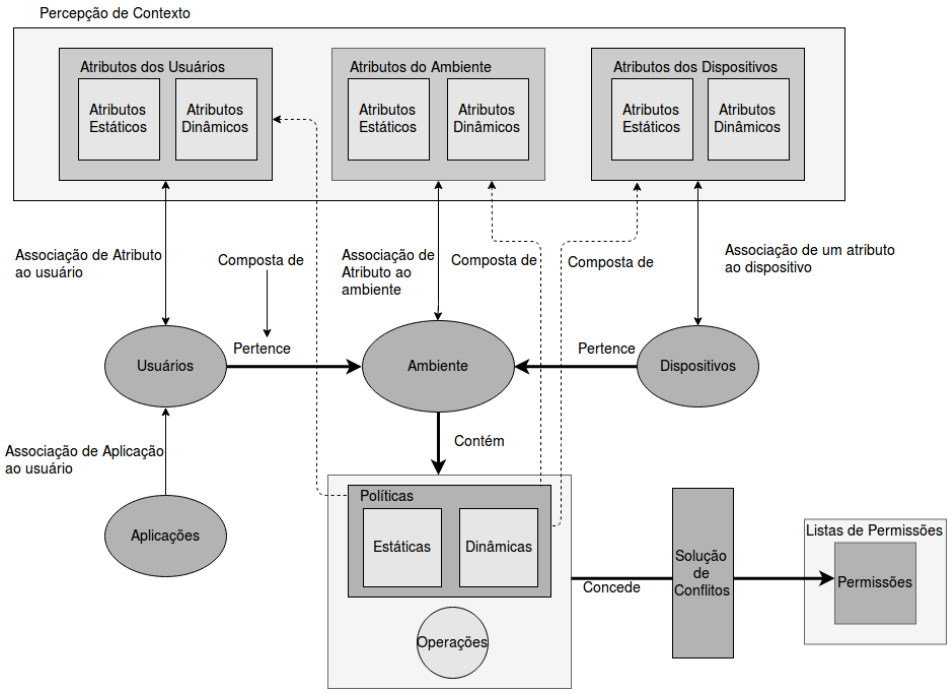

Figura 1. Entidades e relacionamentos do modelo proposto inspirados em [Burmester et al. 2013] e [Servos and Osborn 2017].

- **Usuários (U)** - Pessoas que podem enviar solicitações de sensoreamento ou atuação, sendo identificadas por algum dispositivo, desde celulares até *wearables*.
- **Aplicações (AP)** - São programas que controlam automaticamente todos ou parte dos dispositivos IoT. Aplicações precisam estar associadas a algum usuário para que consigam permissões de acesso aos dispositivos.
- **Dispositivos (D)** - Objetos inteligentes, sensores que se encontram em um determinado domínio/ambiente. Cada dispositivo possui um índice que especifica-o no caso de existirem diversas unidades de um mesmo objeto inteligente. Possui também um conjunto de Operações (OP), que indica as ações que o usuário pode realizar sobre ele. Existem dois tipos de dispositivos:
  - *Tipo 1* - Dispositivos que possuem poder computacional necessário para processar políticas de acesso.

  - *Tipo 2* - Dispositivos que não possuem poder computacional necessário para processar políticas de acesso.
- **Ambiente (AM)** - Consiste em um espaço geográfico, que possui diversos usuários, aplicações e dispositivos associados.
- **Atributos (AT)** - Atributos caracterizam uma determinada entidade, desde usuários, aplicações até o ambiente. Assim como em [Rajpoot et al. 2015], os atributos são divididos em estáticos e dinâmicos:
  - *Atributos Dinâmicos* - Atributos cujo valor está associado ao tempo. O tempo pode ser modelado como uma função monótona crescente, T = $t_1, t_2, ...$ onde $t_i - t_{i-1}$ é uma unidade de tempo ou latência. Essa definição é semelhante a de atributos de tempo real apresentada no trabalho [Burmester et al. 2013].
  - *Atributos Estáticos* - São atributos cujo valor demora um grande número de unidades de tempo para ser modificado (e.g. a idade de um usuário, o tipo de um dispositivo).

  Os atributos são tuplas $(entidade, tipo, nome, valor)$ para atributos estáticos e $(entidade, tipo, nome, valor, tempo)$ para atributos dinâmicos: Entidade pode ser algum usuário, aplicação, dispositivo ou ambiente; O tipo indica se o atributo é estático ou dinâmico; Nome é um identificador único do atributo; Valor é a medida instantânea do atributo.

3.1. Políticas

As políticas usam atributos das entidades e informações de contexto para determinar as permissões de um usuário ou aplicação sobre um dispositivo ou grupo de dispositivos. As políticas são avaliadas em dois momentos. O primeiro momento ocorre quando o usuário envia uma solicitação para utilizar um sensor, sendo avaliadas políticas que possuem somente atributos estáticos. O segundo momento ocorre quando o usuário, depois de ter recebido a permissão, envia um comando para o sensor, onde as políticas com atributos dinâmicas são processadas. Isto garante que o processamento considere eventuais mudanças nos atributos dinâmicos. Abaixo está um exemplo de política:

Se (Usuário.idade < 18), Permitir em (Objeto1[1]), Operações (Leitura)

Essa primeira política é um exemplo de política baseada em atributos estáticos. A seguir temos uma política que utiliza atributos dinâmicos:

Se (Usuário.localização == UFMG), Permitir em (Objeto2[2]), Operações (Leitura)

Além disso, as políticas podem combinar atributos estáticos (consumo) e dinâmicos (volume):

Se (Lâmpada.consumo < 30 AND Som.volume >= 25), Permitir em (Objeto1[1]), Operações (Leitura)

3.2. Listas de Permissões

As listas de permissões armazenam permissões de acesso concedidas aos usuários. Para executar uma operação sobre um dispositivo, primeiramente o usuário deve fazer uma solicitação e, caso essa solicitação seja atendida, a permissão será armazenada em uma lista. É importante notar que as listas não armazenam todas as permissões do usuário, mas

sim aquelas que ele requisitou em um dado período, podendo sofrer alterações de acordo com o a percepção do contexto.

As listas permitem a solução de possíveis conflitos de acesso. Por meio delas é possível verificar se algum usuário ou aplicação está utilizando algum dispositivo que outra entidade deseja operar sobre (o gerenciamento de conflitos é descrito na Seção 3.4). Além disso, as listas também possibilitam que os usuários usem *wearables* e *beacons* para interagir com o sistema. Como as listas são armazenadas em *gateways*, os usuários não precisam carregar consigo *tokens* com as permissões de acesso. As listas ainda garantem uma maior privacidade na delegação de permissões, pois os *wearables* não armazenam dados do usuário que está delegando a permissão.

3.3. Delegação e Revogação

A delegação de privilégios pode ocorrer de duas maneiras:

Permissões: O modelo proposto contém listas para armazenamento das permissões concedidas aos usuários. A delegação de permissões pode ocorrer quando a permissão que o usuário deseja delegar está nessa lista. É usada uma estrutura de dados em árvore para armazenar a delegação de permissões entre os usuários, facilitando o seu gerenciamento.

Atributos: O usuário pode delegar algum atributo não dinâmico que possua, de modo a conceder novas permissões de acesso ao usuário que está recebendo o atributo. Por exemplo, um usuário A pode conceder o atributo "grupo=maniot" para o usuário B, de modo que o usuário A compartilhará todas as permissões de acesso ligadas ao grupo maniot com o usuário B.

Para que o usuário revogue a delegação de uma permissão ou de pertencimento a um grupo, ele deve enviar um comando para a arquitetura. Assim, a arquitetura atualiza a árvore de delegações de permissões, realizando a revogação.

3.4. Gerenciamento de Conflitos

Conflitos de acesso ocorrem quando dois ou mais usuários desejam alterar o estado de um atuador ou uma configuração de um dispositivo IoT. Repare que leituras de dados não geram conflitos. Um exemplo é uma casa inteligente na qual existe uma aplicação que gerencia o consumo de energia e outra a segurança. Quando o dono sai de casa, a aplicação de segurança tenta ligar uma lâmpada, enquanto a de gerência de energia quer desligar a mesma lâmpada para evitar gastos desnecessários. Grupos de operações e prioridades são utilizados para resolver conflitos como esse.

Grupos de Operações: Três grupos de operações são definidos de acordo com os tipos das operações:

- **Grupo 1** - Operações que não visam alterar o estado do sensor/dispositivo, tais como a consulta a um sensor ou a leitura de um parâmetro de configuração do dispositivo. Estas operações não geram conflitos.
- **Grupo 2** - Operações que alteram o estado do sensor/dispositivo, como ligar e desligar uma luz, um eletrodoméstico, abrir ou fechar uma porta. Estas ações geram conflitos na atuação.
- **Grupo 3** - Operações que alteram o modo como o dispositivo/sensor irá operar, como a frequência com a qual os dados são coletados, outras configurações de protocolos, etc. Estas ações podem gerar conflitos de configuração.

Prioridades de aplicativos e usuários: O gerenciamento de conflitos suporta o "acesso exclusivo", em que um usuário ou aplicação solicita o uso exclusivo do dispositivo baseado na sua prioridade. Suponha que um usuário A possui prioridade 10, um usuário B possui prioridade 20, e uma aplicação C possui prioridade 5. O usuário A solicitou e obteve o acesso exclusivo a um dispositivo. Caso os usuários B e C façam requisições que não gerem conflito para este mesmo dispositivo, as ações serão autorizadas. Entretanto, caso C solicite uma operação que gere conflito, ela será negada por ele possuir uma prioridade menor. O usuário B, então, poderia retirar o acesso exclusivo do usuário A.

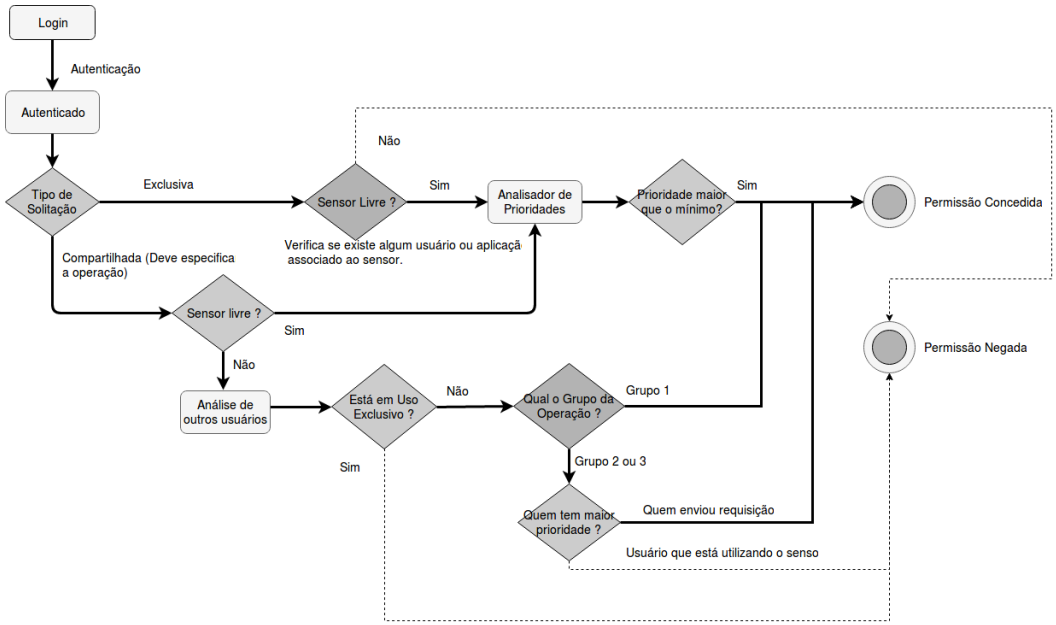

Figura 2. Processo de autorização para solucionar conflitos.

O procedimento completo para solucionar os conflitos é ilustrado pelo fluxograma da Figura 2. Após o usuário ser autenticado, ele pode solicitar uso exclusivo ou compartilhado do sensor. Caso escolha o acesso exclusivo, será necessário verificar se ninguém está utilizando o sensor e também se o usuário tem prioridade maior do que a mínima necessária para utilizá-lo de maneira exclusiva. No caso de uso compartilhado, é verificado se o sensor está livre e se a permissão do usuário é maior do que a mínima necessária. Caso a solicitação seja por uso compartilhado e o sensor não esteja livre, será verificado se ele não está em uso exclusivo e se existe algum usuário com prioridade maior realizando uma operação do mesmo grupo. Para operações do grupo 2 ou 3 (que alteram o estado do sensor) a solicitação será concedida se não existir nenhum usuário com maior prioridade que aquele que fez a requisição já autorizado a realizar a operação. Caso a operação seja do grupo 1, a solicitação sempre é concedida, pois ela não altera o estado do sensor.

4. Reduzindo o Tempo de Acesso

Esta seção apresenta as modificações feitas na arquitetura IoT para permitir que o modelo de controle de acesso seja executado mais rapidamente. Para tanto, é proposto um sistema de controle de acesso híbrido, que executa parcialmente no *gateway* IoT quanto nos dispositivos, além de um mecanismo de *caching* de políticas.

Os *gateways* estão presentes na maioria das soluções IoT para intermediar a comunicação do usuário e das aplicações com os diversos objetos inteligentes. Eles servem como ponte entre os protocolos de rede e de aplicação utilizados pelos usuários (normalmente protocolos REST ou JSON sobre IP), traduzindo-os para comandos inteligíveis para os dispositivos (por exemplo, mensagens seguindo formatos proprietários, transmitidos sobre ZigBee, Z-Wave ou SigFox). Neste trabalho apresenta-se um sistema no qual o *gateway* também armazena os *tokens* de acesso, devido ao seu maior poder computacional e de memória. Assim, sempre que uma solicitação ou comando é enviada para o *gateway*, o mesmo irá determinar se a requisição será atendida ou não, armazenando as permissões concedidas. A Figura 3 mostra o sistema de controle de acesso, indicando quais componentes estão instalados no *gateway* e quais rodam nos dispositivos IoT. Abaixo são descritas as funções de cada um dos componentes:

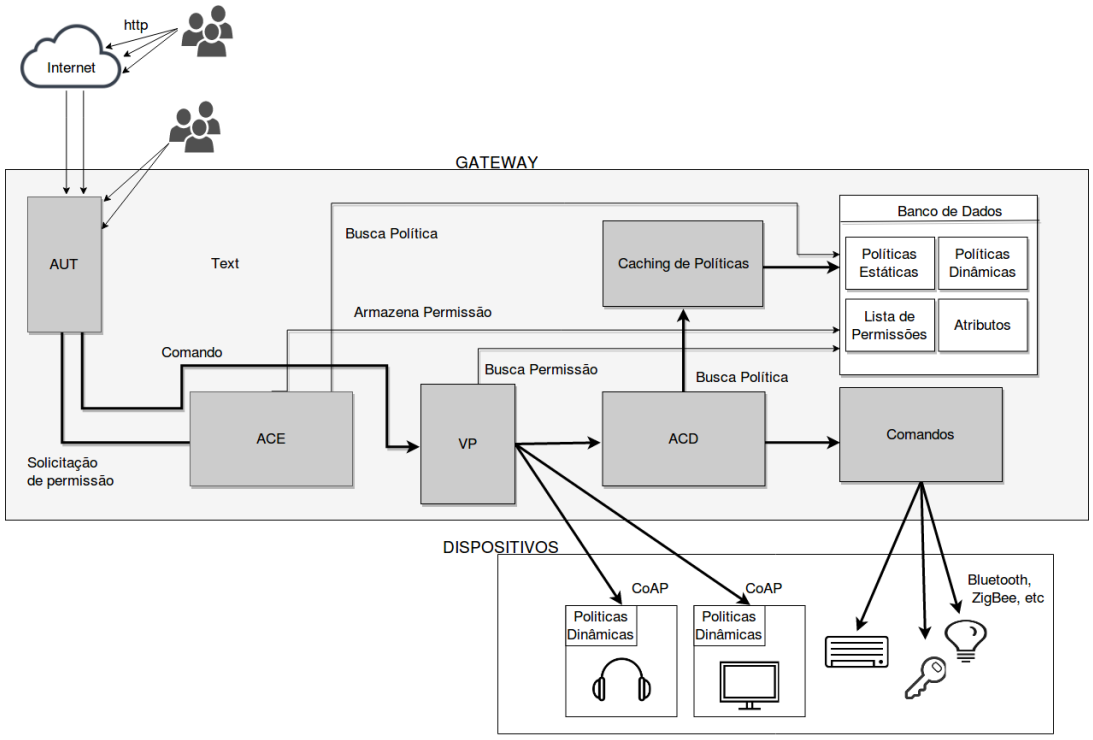

Figura 3. Visão geral dos componentes do sistema de controle de acesso.

Componente de Autenticação (AUT): Recebe *login* e senha do usuário, retornando um ID aleatório (*token* de acesso) utilizado para enviar comandos e requisições para a arquitetura enquanto o usuário estiver autenticado. Além disso, o ID pode ser utilizado pelas aplicações associadas ao usuário para solicitar permissões de acesso aos dispositivos, de modo que a aplicação não armazena *login* ou senha do usuário.

Análise de Contexto Estática (ACE): Realiza a análise da parte das políticas contendo atributos estáticos, concedendo ou não permissão para que o usuário opere sobre algum sensor. Caso o componente conceda a permissão, ela será armazenada em uma lista de permissões. Além disso, aqui são solucionados os conflitos de acesso conforme descrito na Seção 3. É importante enfatizar que o ACE trata a solicitação de permissão. Comandos são enviados diretamente para o VP, pois é necessário já ter uma permissão concedida pelo ACE para enviar comandos.

Verificador de Permissões (VP): Verifica se o usuário tem permissão para realizar a operação especificada no comando recebido. Feito isso, o VP também analisa o tipo do dispositivo. Caso ele seja destinado a um dispositivo do Tipo 1, o componente repassa o comando para o dispositivo, que será responsável por processar a política dinâmica e, se necessário, executar o comando e retornar a resposta. Caso ele seja do Tipo 2, o comando será repassado ao ACD, realizando a análise dinâmica no próprio *gateway*.

Análise de Contexto Dinâmica (ACD): Recebe um comando e realiza a análise da parte dinâmica das políticas. Caso a análise autorize a operação, o comando será repassado até chegar ao dispositivo.

Cache de Políticas (CP): Como as regras do componente ACD trabalham com atributos variáveis, existe um atraso para se obter as informações necessárias dos sensores. Este componente determina se os atributos dinâmicos da política devem ser reavaliados. O raciocínio é que, se uma regra foi verificada há um dado instante de tempo, ela pode não precisar ser verificada novamente. A Subseção 4.3 descreve com mais detalhes o *caching*.

4.1. Quebra das Políticas em Estáticas e Dinâmicas

A quebra das políticas em parte estática e dinâmica ocorre para que se consiga dividir a análise de contexto em dois momentos. Como dito na Seção 3, as políticas estáticas são compostas apenas de atributos estáticos e as politicas dinâmicas por atributos dinâmicos. Logo, dada uma política criada pelo usuário *Se ((Usuário.idade<30 AND Objeto2.luminosidade<20))*, esta é dividida em duas políticas, uma estática e outra dinâmica: *Se (Usuário.idade<30)* (parte estática) e *Se (Objeto2.luminosidade<20)* (parte dinâmica).

Note que caso as duas políticas, estática e dinâmica, determinem que o usuário tem permissão para realizar uma operação, o resultado será equivalente ao da primeira política para um mesmo instante no tempo. Essa divisão evita que a parte dinâmica seja analisada sem necessidade caso a parte estática não seja atendida. A política estática é verificada no componente ACE quando uma solicitação de permissão é realizada. Ela retornará PERMITIR, armazenando a permissão na lista de permissões, caso o usuário tenha o atributo idade com valor menor que 30. Quando o usuário enviar o comando à arquitetura, a lista de permissões será consultada para determinar se ele está autorizado a realizar a operação desejada. Após encontrar a permissão na lista, será analisada a política dinâmica associada à política estática que concedeu a permissão. Assim, a parte dinâmica da política somente é verifica quando o usuário envia o comando para a arquitetura.

4.2. Controle de Acesso Híbrido

Este trabalho traz uma abordagem que diferencia dispositivos que conseguem realizar parte do controle de acesso daqueles que não conseguem. Assim, a análise das políticas com base em atributos dinâmicos pode ser feita no *gateway* ou no próprio dispositivo que o usuário deseja acessar, dependendo da capacidade computacional desse dispositivo.

A abordagem híbrida mantém os benefícios da descentralização, permitindo ao sistema ser mais escalável, ao mesmo tempo que suporta dispositivos com processamento limitado, se adaptando à heterogeneidade dos dispositivos IoT. A Figura 4 mostra o processamento feito exclusivamente no *gateway*, e a Figura 5 conta com o auxílio dos dispositivos. Em ambas as figuras, para que o comando do usuário seja executado no dispositivo 1, é preciso que um valor coletado no dispositivo 2 seja maior que 50.

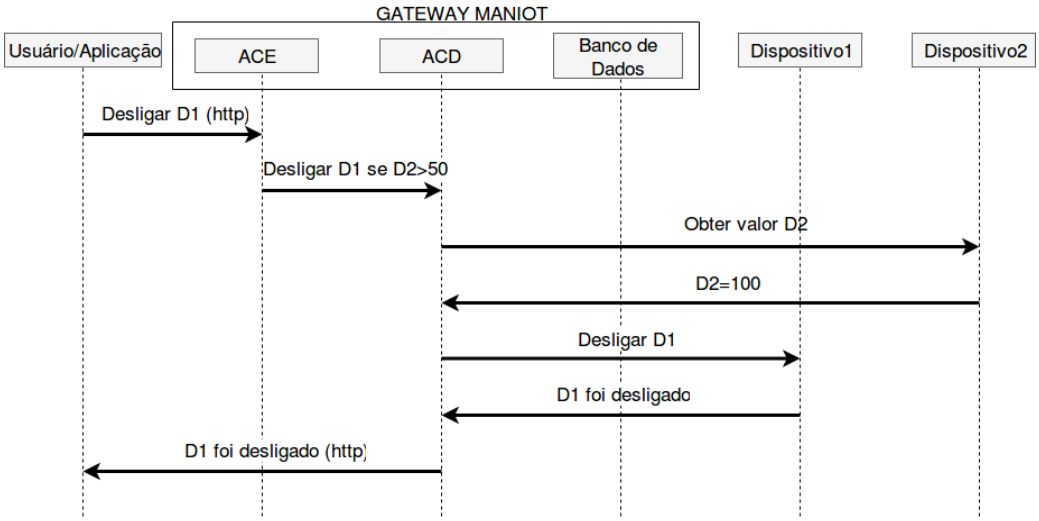

Figura 4. Atividades do controle de acesso ao executar no gateway somente.

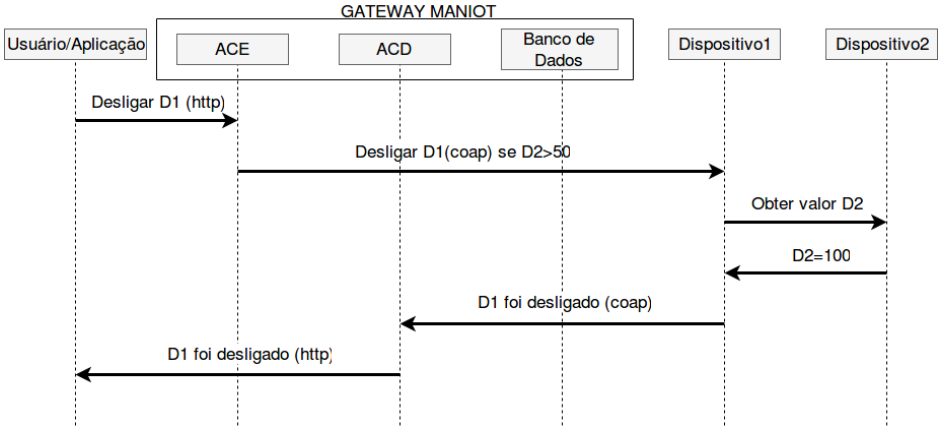

Figura 5. Controle de acesso ao executar no gateway e nos dispositivos IoT.

4.3. Caching de Políticas

O sistema de *caching* foi desenvolvido porque alguns sensores podem precisar de um grande intervalo de tempo para obtenção das suas leituras. Isso ocorre, por exemplo, em sensores de concentração de gases (ciclo de leitura da ordem de minutos) [Hanwei Electronics 2018], ou plataformas comerciais como IFTT [IFTTT 2018] e Smartthings [Samsung 2018], onde a comunicação com o dispositivo deve passar por um servidor em nuvem. O estado da arte, até onde temos conhecimento, assume que as informações de contexto são obtidas com baixo custo e atraso.

O *caching* proposto reduz o tempo de processamento dos comandos evitando a avaliação sem necessidade de políticas de acesso frequentemente consultadas que precisam de dados de sensores (presença, luminosidade, temperatura, etc). O sistema permite que o usuário determine qual será o valor do intervalo T no qual a política dinâmica, não precisará ser analisada novamente após ser verificada uma primeira vez. Um cenário onde esse sistema seria útil é o de controle de um moinho, no qual o usuário só pode abrir a porta caso a concentração de Metano seja inferior a 300ppm. Se o valor retornado em um dado momento for 2000 ppm, e sabe-se que existe um tempo mínimo, determinado por

administradores com conhecimento da aplicação, para que o valor atual decaia de uma certa quantidade (por exemplo, dez minutos), pode-se determinar que o sensor não precisa ser lido novamente durante este intervalo. Assim, a política dinâmica que autoriza a execução não precisa ser reavaliada durante esse intervalo de tempo, o que diminui o tempo para processar o comando.

5. Avaliação e Demonstração de Funcionamento

Esta seção apresenta uma avaliação do sistema de controle de acesso à IoT com objetivo de avaliar o seu funcionamento e a sua eficiência. Testes foram realizados considerando a divisão de políticas em estáticas e dinâmicas, a solução de conflitos, o sistema híbrido, o *caching* de políticas, a escalabilidade quanto ao número de requisições enviadas.

O sistema de controle de acesso foi implementado como um componente da arquitetura ManIoT [Antunes et al. 2016]. Ela busca prover um gerenciamento sobre as coisas de um determinado cenário, visando um controle genérico e escalável. Além disso, por meio de módulos centralizados e distribuídos, a arquitetura promove um controle que não só toma decisões de caráter local, mas também em um âmbito global com base nas informações adquiridas nos módulos de gerenciamento local. O controle de acesso proposto foi implementado como um serviço interno da arquitetura ManIoT. Nossa implementação emprega REST para a comunicação entre os usuários, aplicações e o MANIOT, e o protocolo CoAP para a comunicação com as máquinas virtuais que simulam os dispositivos.

Os testes foram realizados utilizando o *testbed* FUTEBOL UFMG (`http://futebol.dcc.ufmg.br`). Foram alocados MiniPCs para simular dispositivos IoT, um *gateway* e um cliente. Um MiniPC simula o *gateway* e outro simula um cliente, enquanto 4 MiniPCs simulam dois dispositivos IoT cada, empregando máquinas virtuais. Metade dos dispositivos IoT (um por MiniPC) tratam políticas dinâmicas, enquanto a outra metade delega o tratamento das políticas para o *gateway*.

As seguintes métricas foram aferidas para demonstrar a eficiência do modelo: tempo de resposta, tempo necessário para processar uma determinada quantidade de comandos da solução híbrida, centraliza e suas respectivas versões com caching. Os resultados apresentados abaixo mostram a média de 4 execuções independentes de cada teste, bem como o intervalo de confiança com uma confiança de 95 porcento.

5.1. Tempo de Resposta

O tempo de resposta do sistema é avalido medindo-se o tempo total gasto para o usuário enviar uma solicitação de uso de um dispositivo, enviar um comando e receber uma resposta. Sempre que ocorre o envio da solicitação, é processada um política estática e concedida a permissão ao usuário. Em seguida o comando é enviado e três cenários podem ocorrer: No primeiro, nenhuma política dinâmica precisa ser processada para que a operação seja realizada. No segundo caso, uma política dinâmica é processada, comunicando-se com algum sensor por meio do protocolo CoAP para obter o valor de alguma leitura. No terceiro caso também é processada uma política dinâmica, mas é necessário obter a leitura de um sensor SmartThings, sendo necessário que a requisição passe pela nuvem.

A Tabela 1 contém os resultados obtidos. Como esperado, a política dinâmica que utiliza as informações do sensor SmartThings leva o maior tempo médio para executar, porque como mencionado, a requisição precisa passar pela servidor da empresa fabricante

Tabela 1. Tempo de resposta de envio de comando com as diferentes políticas.

	Média (s)
Nenhuma Política Dinâmica	0.33350 $\pm$0.0063
Política Dinâmica (Objeto Local)	0.44300 $\pm$0.01
Política Dinâmica (Objeto SmartThings)	0.66275 $\pm$0.027

dos dispositivos. O segundo maior tempo médio é o da política que busca informações em um sensor local. O menor tempo médio ocorre quando não é necessário processar nenhuma política dinâmica para executar o comando.

5.2. Sistema Híbrido

O sistema híbrido é avaliado ao comparar o tempo necessário para processar um certo número de comandos da solução híbrida com uma solução totalmente centralizada. A Figura 6 mostra o resultado obtido. Como se imaginava, a solução híbrida leva menos tempo para executar os comandos do que a solução centralizada. Isso ocorre porque na solução híbrida, para 4 dos 8 dispositivos (aqueles do Tipo 1), o *gateway* repassa os respectivos comandos para os próprios dispositivos realizarem o processamento das políticas dinâmicas. Já na solução centralizada, o *gateway* processa as políticas de todos os 8 dispositivos. Esse grau de paralelismo permite o sistema ser mais escalável, e lidar com um número maior de requisições em menos tempo, o que é essencial para a IoT.

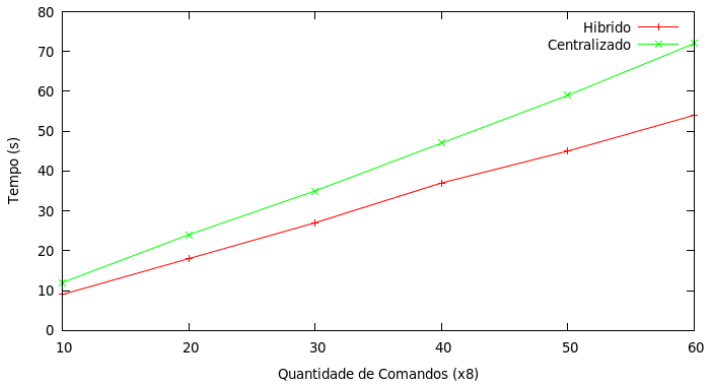

Figura 6. Sist. centralizado vs. híbrido - tempo de processamento de comandos.

5.3. Caching de Políticas

O *caching* de políticas também é avaliado comparando-se o tempo gasto para processar uma determinada quantidade de comandos. São comparadas as soluções centralizada e híbrida com as suas respectivas versões utilizando o *caching* de políticas. Para a solução com *caching*, após o primeiro envio de um comando inicia-se a contagem de um tempo T, que para o experimento foi 60s, no qual a política dinâmica responsável por conceder permissão de acesso não precisará ser avaliada novamente.

As Figuras 7 e 8 mostram os resultados para comparação dos sistemas sem e com *caching*. O sistema de *caching* diminui o tempo necessário para a avaliação das políticas, uma vez que as políticas dinâmicas não precisam ser verificadas. Mas ainda um tempo considerável é necessário para enviar o comando, identificar se as políticas estão em *cache*, executar o comando e retornar a resposta para o usuário.

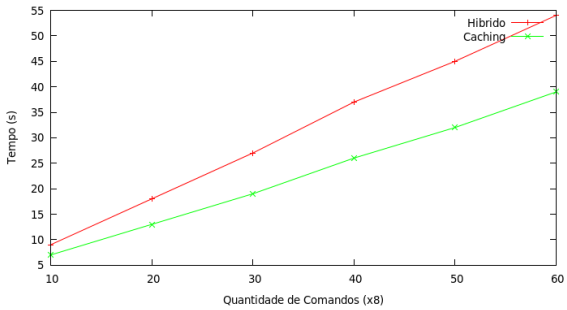

Figura 7. Sistema híbrido Versus híbrido com caching.

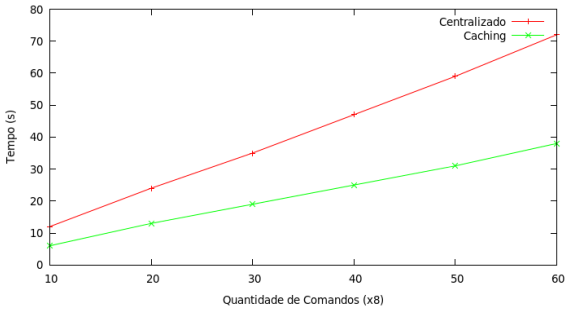

Figura 8. Sistema centralizado e centralizado com caching.

6. Conclusão e Trabalhos Futuros

Este artigo apresentou um sistema de controle de acesso para ambientes IoT que permite a delegação de acesso entre aplicativos e usuários. Além disso, o modelo acelera a execução das políticas de acesso a partir de modelos de execução híbridos, bem como a análise das políticas em fases, dividindo a análise em regras estáticas e dinâmicas. A avaliação em um ambiente experimental indicou que o modelo proposto acelera a avaliação de políticas, e permite ambientes onde diversos usuários e aplicações acessam os dispositivos concorrentemente. Como trabalhos futuros, pretende-se avaliar a arquitetura em cenários com diversas aplicações e usuários acessando os dispositivos ao mesmo tempo, bem como o efeito do tempo de *caching* das regras no desempenho do sistema.

Referências

Anggorojati, B., Mahalle, P. N., Prasad, N. R., and Prasad, R. (2012). Capability-based access control delegation model on the federated iot network. In *15th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 604–608. IEEE.

Antunes, J. B., Denés, I. L., Santos, M., Castro, T. O., Macedo, D. F., and dos Santos, A. (2016). Maniot: Uma plataforma para gerenciamento de dispositivos da internet das coisas. In *Workshop de Gerenciamento de Redes e Serviços (WGRS)*.

Ashton, K. (2011). That 'internet of things' thing. *RFiD Journal*, 22(7).

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Comput. Netw.*, pages 2787–2805.

Burmester, M., Magkos, E., and Chrissikopoulos, V. (2013). T-abac: An attribute-based access control model for real-time availability in highly dynamic systems. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 143–148. IEEE.

Denox (2017). Denox. [Disponível em: `https://denox.com.br/`; acessado em 29-Maio-2017].

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660.

Gusmeroli, S., Piccione, S., and Rotondi, D. (2012). Iot access control issues: a capability based approach. In *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 787–792. IEEE.

Hanwei Electronics, c. (2018). MQ-7 gas sensor. `https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf`.

Hemdi, M. and Deters, R. (2016). Using rest based protocol to enable abac within iot systems. In *IEEE 7th Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–7. IEEE.

Hernández-Ramos, J. L., Jara, A. J., Marin, L., and Skarmeta, A. F. (2013). Distributed capability-based access control for the internet of things. *Journal of Internet Services and Information Security (JISIS)*, 3(3/4):1–16.

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). *NIST special publication*, 800(162).

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.

IFTTT (2018). Ifttt. [Disponível em: `https://ifttt.com/`; acessado em 18-Março-2018].

Liu, Q., Zhang, H., Wan, J., and Chen, X. (2017). An access control model for resource sharing based on the role-based access control intended for multi-domain manufacturing internet of things. *IEEE Access*.

Mahalle, P. N., Anggorojati, B., Prasad, N. R., and Prasad, R. (2013). Identity authentication and capability based access control (iacac) for the internet of things. *Journal of Cyber Security and Mobility*, 1(4):309–348.

Rajpoot, Q. M., Jensen, C. D., and Krishnan, R. (2015). Attributes enhanced role-based access control model. In *International Conference on Trust and Privacy in Digital Business*, pages 3–17. Springer.

Samsung (2018). Smartthings. [Disponível em: `https://www.samsung.com/us/smart-home/smartthings/`; acessado em 18-Março-2018].

Sandhu, R. S. and Samarati, P. (1994). Access control: principle and practice. *IEEE communications magazine*, 32(9):40–48.

Servos, D. and Osborn, S. L. (2017). Current research and open problems in attribute-based access control. *ACM Computing Surveys (CSUR)*, 49(4):65.

Sony (2017). Sony smartband. [Disponível em: `http://support.sonymobile.com/br/swr10/`; acessado em 29-Maio-2017].

Zhang, G. and Tian, J. (2010). An extended role based access control model for the internet of things. In *International Conference on Information Networking and Automation (ICINA)*, volume 1, pages V1–319. IEEE.