DESENVOLVIMENTO DE MODELO HIERÁRQUICO DE MIDDLEWARES COM APLICAÇÃO DE FOG COMPUTING PARA REDES IOT

João T. M. de Menezes, Pedro H. L. da Costa, Dayanne F. da Cunha,
Francisco Lopes de Caldas Filho, Lucas M. C. e Martins e Fábio Lúcio Lopes de Mendonça
Laboratório LATITUDE, Departamento de Engenharia Elétrica, Universidade de Brasília (UnB), Brasília-DF, Brasil

RESUMO

O crescimento em larga escala da geração e do tráfego de dados da Internet permitiu o desenvolvimento de análises complexas em diversos contextos (social, transporte e saúde). Esse progresso gerou desafios na coleta, processamento, persistência e apresentação de dados para gerar informações, inclusive para a Internet das Coisas. Nessas circunstâncias, este artigo propõe um Middleware IoT genérico que pode ser implantado em uma estrutura descentralizada, permitindo a redução do tráfego e armazenamento de dados na Central Cloud através da aplicação de regras definíveis que sumarizam os dados coletados e transferem os resultados em uma rede IoT. A proposta foi testada implantando os Middlewares de maneira hierárquica no contexto de uma cidade inteligente e comparando-os com o mesmo Middleware em uma rede com uma Nuvem Centralizada. Os resultados foram positivos, o Middleware auxiliou na aplicação dos dois casos e, no caso do middleware hierárquico, foi possível reduzir o tráfego de dados para armazená-los de forma distribuída.

PALAVRAS-CHAVE

IoT, Middleware, Data, Traffic, Fog

1. INTRODUÇÃO

O crescente desenvolvimento de tecnologias permite a obtenção e controle de dados antes inexplorados, proporcionando um melhor controle de variados sistemas da sociedade, tais como transporte, cuidados de saúde, monitoramento de ambientes e de pessoas (Reinsel *et al.*, 2018; Atzori *et al.*, 2010; Elijah *et al.*, 2018). Esse avanço também resultou em um aumento do volume de dados já coletados, seja pelo aumento da capacidade de persistência de dados, da velocidade de coleta ou da distribuição de geradores de dados.

Esse crescimento do volume digital possibilita uma melhor gestão de aspectos humanos, sociais ou tecnológicos. As informações presentes em dados extraídos de fontes variadas são utilizadas em aplicações que, muitas vezes, necessitam de um pré-processamento para a efetiva obtenção da informação útil para sua atuação. Há diversas estimativas relacionadas ao crescimento do volume de dados e informações trafegando na rede, em especial com a expansão da IoT (Gubbi *et al.*, 2013).

O relatório Cisco *Visual Networking Index* (Cisco, 2019) apresenta previsões para o tráfego mundial total da Internet e questões relacionadas como previsões sobre o número de dispositivos conectados à Internet de forma quantitativa e sempre crescentes. O estudo expõe pontos de análise como mudanças relativas ao aumento da variedade de dispositivos e conexões, que mostra uma estimativa de mais de 25 bilhões de dispositivos no mundo e destes, mais da metade correspondem a conexões M2M (*Machine to Machine*, representando conexões diretas entre dois dispositivos). Isso reflete no crescimento da quantidade de dispositivos IoT presentes na rede em âmbito mundial e no crescimento do tráfego de aplicações presentes na Internet. A estimativa é que em 2022 o tráfego global da Internet seja de 150.000 GB por segundo. Diante de tal cenário, estudos relativos a mudanças para suportar o tráfego previsto são indispensáveis.

O aumento da complexidade dos sistemas deixou mais clara a necessidade de processamento automático dos dados. Esse processamento pode gerar informação e ser atrelado a uma atuação também automática sobre o sistema. Portanto, a coleta, o controle e o processamento dessa imensa quantidade e variedade de dados se tornaram imprescindíveis para o gerenciamento dos sistemas e atuadores no mundo. O conceito de IoT permite,

com facilidade, a interoperabilidade de sistemas ou subsistemas, sendo um ótimo candidato para tal tarefa.

O processamento de dados é importante para obter informações úteis sobre as métricas cruas coletadas, que muitas vezes é feito a partir de uma base de dados extensa na *Cloud* ou por outras entidades que possuem acesso aos dados que foram persistidos nela, sendo que neste caso, há um gasto de banda para a obtenção dos dados necessários para o processamento. Este trabalho tem como interesse buscar uma solução mais otimizada que a arquitetura de *Cloud*, visto que para a persistência, processamento e resposta/atuação não se mostra uma boa alternativa para todas as aplicações. A aglomeração de dados em um único ponto torna o processamento mais custoso e lento, além de adicionar uma latência para respostas às requisições de aplicações ou clientes (Lee & Lee, 2018; Tong *et al.*, 2016). Nesse cenário, armazenar dados de maneira distribuída, permitindo acesso a informações em toda a rede, pode ser uma boa alternativa (Huacarpuma *et al.* 2017) e os conceitos de *Fog Computing* se encaixam nesse problema (Atlam *et al.*, 2018).

O trabalho possui esta Introdução (apresentação da problemática), Trabalhos Relacionados, (traz pesquisas relacionadas ao presente trabalho), Proposta (descrição da proposta), Testes e Resultados (testes para validar a proposta e resultados obtidos) e finaliza com a Conclusão (conclusão do trabalho a respeito dos resultados obtidos e trabalhos futuros que podem ser desenvolvidos a partir da presente proposta).

2. TRABALHOS RELACIONADOS

O uso de uma descentralização do processamento para atender requisições de forma mais eficaz e sem uso abusivo de banda de certas aplicações já foi o centro de outras pesquisas. Tong *et al.* (2016) argumenta que uma arquitetura hierárquica de *Cloud* com processamento na borda e não na internet, atende requisições com maior eficiência e rapidez, pois permite a agregação das cargas de pico dos usuários móveis em diferentes servidores em nuvem. Tal arquitetura busca a utilização eficiente dos recursos de nuvem, por meio de um algoritmo de posicionamento de carga de trabalho. Esse algoritmo decide em que programas móveis de servidores de nuvem na borda deverão ser colocados e a quantidade de capacidade computacional, que deve ser provisionada para executar cada programa.

O uso de processamento próximo aos clientes com armazenamento distribuído já tinha sido proposto por Yu *et al.* (2017), alegando que haveria uma diminuição considerável da latência entre os servidores e os usuários finais, evitaria picos de uso de banda pela Internet e estenderia a vida útil de dispositivos que funcionam a baterias.

A aplicação do paradigma de *fog computing* para *smart-cities* já foi proposto por Mohamed *et al.* (2017), em que se expõe uma visão de Arquitetura Orientada a Serviço (SOA) para as entidades existentes em uma Rede IoT, e Silva *et al.* (2016), em que se identifica e separa diferentes abstrações e aplica uma hierarquia multi-level para a coleta, processamento, integração e alertas.

O laboratório de Internet das Coisas em que esse trabalho foi desenvolvido (UIoT), já produziu trabalhos de desenvolvimento de um *gateway* semântico IoT (Caldas Filho *et al.*, 2017). Esse *gateway* era capaz de compreender mensagens de comunicação através de *Sockets* TCP e UDP, MQTT e ZigBee, e assim fazer todo o gerenciamento de serviços oriundos de dispositivos e conectá-los aos *Middlewares* que os interligam com a Internet. Além desse *gateway*, já existia, também, um *Middleware* que continha a capacidade de identificação das entidades presentes na rede e um controle de mensageria, como exposto nos estudos de Silva *et al.* (2016), Ferreira and de Sousa Jr. (2017) e Martins *et al.* (2017).

A proposta do trabalho é a construção de um *Middleware* IoT genérico que pode ser hierarquizado permitindo redução da quantidade de dados na rede e o acesso a informação gerada por dispositivos, através de aplicação de conceitos relacionados a *Fog computing* (Atlam *et al.*, 2018; Bonomi *et al.*, 2012).

3. PROPOSTA

A hierarquização de *Middlewares* consiste na distribuição destes no contexto de uma cidade inteligente visando a redução da quantidade de dados trafegada na rede. A ideia é que os *Middlewares* estejam localizados geograficamente próximos dos dispositivos que possuem serviços das entidades de abstrações de uma cidade: casa, rua, bairro e cidade. Dessa forma, existiria um *Middleware* para cada abstração citada de forma hierárquica como pode ser visto na Figura 1. Assim, os dados oriundos de dispositivos são enviados para o

Middleware do seu nível hierárquico e as informações presente nesses dados podem ser passadas para *Middlewares* em nível hierárquico maior conforme configuração de regra. Esse modelo permite que apenas dados de interesse sejam trafegados ao longo da rede.

Figura 1. *Middlewares* hierárquicos em uma cidade

Para exemplificar, supõe-se que se deseja medir a temperatura média de um bairro a cada 20 minutos. Existem casas nesses bairros com sensores de temperatura que produzem dados a cada 30 segundos. Se existir apenas um *Middleware* centralizado na rede, esses dados serão enviados pela Internet e serão armazenados juntos com os dados de todos os dispositivos da rede. Para saber a temperatura do bairro, será necessário solicitar todos esses dados presentes no *Middleware* depois processá-los e extrair a informação da média do bairro a cada 20 minutos. Em um cenário hierárquico, os dados dos sensores das casas serão enviados para *Middlewares* de casas e, já que estão geograficamente próximos, os dados não precisam trafegar pelo *backbone* da rede. Esses dados serão armazenados apenas com dados oriundos da casa em questão. Para saber a temperatura média do bairro, não é necessário a leitura desses dados a cada 10 segundos. Aqui utiliza-se conceitos de *fog computing*, onde o bairro pode cadastrar regras para os *Middlewares* das casas resumizarem os dados de temperatura das casas e enviarem a média para *Middlewares* das ruas a cada 5 minutos, e também cadastrar regras para os *Middlewares* das ruas, resumizarem essa informação e enviarem as médias de temperaturas das ruas a cada 20 minutos. Dessa forma, será possível calcular a temperatura média do bairro a cada 20 minutos com uma redução de tráfego de dados no *backbone* da rede. Este cenário pode ser expandido de forma que possam existir vários *Middlewares* de bairros conectados ao *Middleware* de uma cidade.

3.1 UIoTBox

Para permitir a aplicação do modelo hierárquico proposto, propõe-se um *Middleware* IoT, denominado UIoTBox. Ele é composto pelos seguintes módulos: Unidade de Persistência, Interface, Listener, Collector, Dispatcher, Rule, Monitor e API Rest. Com exceção da Unidade de Persistência onde utilizou-se a tecnologia MongoDB, o restante dos módulos foram desenvolvidos com a linguagem de programação Python. Na Figura 2 é demonstrado um diagrama do UIoTBox com os módulos.

Figura 2. Módulos que o compõe e um fluxo da informação do UIoTBox

Estes módulos possibilitam ao *Middleware*: a persistência de dados recebidos; o cadastro de fórmulas de sumarização de dados; o gerenciamento de eventos para fórmulas de sumarização; e envio de resultados de sumarização para fontes externas. Sua implementação foi feita pensando em uma adequação a uma forma genérica, permitindo ser utilizado em outras aplicações e arquiteturas IoT.

O UIoTBox é, portanto, a consolidação de alguns módulos em Python que, juntos, formam um *Middleware* genérico capaz de suportar regras para análise e envio de dados ou alertas. Ele é capaz de receber, persistir, processar dados de entrada e enviar informações para diversas fontes. A função de cada módulo construído pode ser explicada pelo fluxo do recebimento de um dado apresentado Na Figura 2.

O passo número 1 da Figura 2 representa o cadastro de um novo cliente no UIoTBox, que, no caso, é um sensor de temperatura. Para tal, o dispositivo deve enviar uma mensagem seguindo o fluxo de cadastro para a correta identificação no *Middleware*. Isso é realizado na entidade que atua como a porta de entrada do UIoTBox. O Listener. Este tem a responsabilidade do cadastramento de dispositivos, serviços de envio de dados e então de receber os dados de diferentes sensores e tecnologias IoT. Os dispositivos podem ser tanto sensores quanto outros *Middlewares*. Atualmente, o UIoTBox possui *Sockets* TCP e UDP, MQTT e ZigBee. O cadastro de clientes é realizado no *endpoint /client*. Agora o dispositivo deve cadastrar o(s) serviço(s) que ele provê e que geram dados que serão enviados. O serviço é cadastrado com uma mensagem para o *endpoint /service*. Com o(s) serviço(s) cadastrado(s), o sensor pode iniciar a transmissão de dados. Os dados são enviados com uma mensagem seguindo um padrão definido para o *endpoint /d/data*.

Os dados enviados de um dispositivo para o UIoTBox permanecem em memória no Listener até que o Collector retire esses dados para então realizar a persistência. Isso é representado pelo passo número 2 da Figura 2, em que ocorre a retirada de dados mantidos em *buffer*. Assim, o papel do Collector é permitir a coleta de dados de diferentes fontes de interesse e os persistir na unidade de persistência. No presente estudo, o Collector atua como a ponte entre os dados recebidos pelo Listener e a Unidade de Persistência. Para tal atividade, o Collector, utiliza uma *Interface* relacionada ao Listener, e realiza um GET no *endpoint /getall* periodicamente. A entidade *Interface* atua como uma classe abstrata de conexão que possui os métodos *send* e *get* para mandar e solicitar dados com seus respectivos parâmetros atrelados a interface.

O passo 3 da Figura 2 ilustra o último passo do recebimento dos dados pelo UIoTBox, a persistência dos registros recebidos. O Collector simplesmente passa os dados obtidos no passo anterior para a Unidade de Persistência, a partir do uso da *Interface* que segue os parâmetros da Unidade de Persistência.

Quando há uma Regra registrada, o fluxo do UIoTBox segue as etapas 4, 5, 6 e 7 da Figura 2. A *Rule* é a entidade que define as regras que o UIoTBox deverá seguir sobre os serviços que estão enviando dados a ele. As regras têm como objetivo definir o mecanismo de monitoramento dos serviços e, então, realizar operações e envio dos dados processados. Ela possui 4 objetos, *Filter*, *Event*, *Fórmula*, *Interface*.

O *Filter* é o elemento da *Rule* que contém os parâmetros da *query* para filtragem de dados na Unidade de Persistência. Ele é, portanto, o identificador dos serviços de interesse para a regra em questão. O objeto *Event* define o gatilho para aplicação da regra cadastrada. Ele contém campos para definir a quantidade de ocorrências

e o tipo do evento. *Fórmula* é a entidade que representa a operação que será executada com os dados obtidos a partir de *Filter* na ocorrência de um *Event*. Esta operação é identificada como uma *string*. A *Interface* contém as informações necessárias para indicar qual interface utilizar. Esta é especificada, pelo campo **type**, e seus parâmetros são passados a partir de um objeto de nome *params*.

Continua-se o fluxo da situação exposta na Figura 2, onde existe um serviço chamado de "Temperature". Para ilustrar este novo fluxo, suponhamos que a regra cadastrada consiste no envio da média dos serviços de nome igual a "Temperature" a cada 10 amostras recebidas para o UIoTBox superior. Em suma, *Rule* possui *Filter* identificando o nome do serviço igual a "Temperature", um *Quantity Event* com **value** igual a 10 e ocorrência ilimitada, ou seja, igual a -1, *Fórmula* igual a "Average" e *uiot_service* como *Interface*.

Nesse momento aparece a entidade *Monitor* que consiste no módulo responsável por monitorar constantemente a Unidade de Persistência para verificar a ocorrência de eventos e faz execução da *Rule* conforme configuração desta, representado pelos passos 4 a 6 da Figura 2. O *Monitor* consulta as regras cadastradas, e, para cada regra, verifica os dados a partir dos filtros presentes para aferir a ocorrência do evento. Se o evento ocorrer, o *Monitor* executa a fórmula e instancia o *Dispatcher* passando a *Interface* presente na regra para enviar o dado processado para seu devido destino. Como o evento possui uma quantidade de ocorrências, sempre que este ocorrer, o número de ocorrências é decrementado e quando tem valor igual a zero, a regra é excluída. Usando a *interface* *uiot_service*, os dados são enviados para o UIoTBox superior como um registro de dados para o *endpoint* */g/data*, representado pela etapa 7 na Figura 2, na qual reiniciará o fluxo apresentado neste *Middleware*.

4. TESTES E RESULTADOS

Para testar o *Middleware* e a aplicação da arquitetura hierarquizada, realizou-se uma comparação da arquitetura proposta com uma arquitetura de *cloud* centralizada. A comparação aconteceu por meio de três cenários de testes nas duas arquiteturas no contexto de uma cidade. Cenário 1: uma rua deseja obter a média de umidade dos últimos 30 minutos de todas as suas casas. Cenário 2: um bairro deseja identificar momentos em que a média de temperatura nas casas (calculada a cada 5 minutos) de uma rua estiver maior que 25°C. Cenário 3: um bairro deseja receber o valor máximo de um serviço de luminosidade a cada 1.000 amostras. Em tais cenários, existem dois sensores de temperatura e umidade (DHT11) e um sensor de luminosidade que enviam dados a cada 5 segundos. Para envio dos dados, os sensores são conectados em módulos de hardware ESP8266. Os *Middlewares* foram implantados em Raspberry Pi 3B.

No cenário da *cloud* centralizada, todos os sensores enviam para o mesmo lugar, como mostra a Figura 3. Observa-se também que não há a separação de casas, ruas e bairros, apenas uma abstração disso. Para atender os cenários, é necessário registrar todas as regras no mesmo *Middleware* para realização de sumarização.

Figura 3. Caso *cloud* centralizada

O cenário da rede hierárquica utilizado para teste está representado na Figura 1. Os sensores de temperatura e umidade são conectados a *Middlewares* de casas. Tais *Middlewares* estão conectados a um *Middleware* de rua. Neste último também é conectado o sensor de luminosidade. O *Middleware* da rua está conectado ao *Middleware* do bairro.

Aplicando os conceitos de *Fog Computing*, o armazenamento dos dados é realizado nos *Middlewares* espalhados geograficamente pela cidade e regras são cadastradas nos *Middlewares* de nível inferior para enviar os dados sumarizados para níveis superiores. Portanto, para atender aos cenários, uma regra que tenha um Filtro igual ao serviço de “Temperature”, Fórmula igual a “Average”, Evento igual a 5 minutos e Interface igual a um serviço UIoTService, e outra regra que altera apenas o Filtro do serviço para ser igual ao serviço de umidade foi registrada nos *Middlewares* casas. No *Middleware* da rua, foi registrada a regra que tem Filtro igual a Serviço “Temperature”, Evento igual a “Maior que 25° C”, Interface igual a “UIoTService” e nada em Fórmula e outra regra que tem Filtro igual a Serviço de Luminosidade, Fórmula é igual a “Average”, Evento igual a 100 amostras e Interface igual a UIoTService.

Todas essas regras também foram registradas no *Middleware* do cenário de *cloud* centralizada, alterando a Interface para uma interface HTTP com um cliente que apenas recebe os dados. Nesse caso, os sensores de temperatura enviam dados para o *Middleware* a cada 5 segundos e o de luminosidade a cada 5 segundos.

No cenário hierárquico, com as regras cadastradas, os dados que irão trafegar pelo *backbone* da rede, ou seja, entre os *Middlewares*, são os das casas para rua e da rua para o bairro no momento de ocorrência dos eventos. Sendo $\frac{1}{T_1}$ a frequência de envio de cada amostra de dado pelo sensor e P o período total de execução do experimento, a quantidade de dados enviada por um sensor (Q1) pode ser definida pela equação:

$$Q1 = \frac{1}{T_1} \times P \Rightarrow P = Q1 \times T1 \quad (1)$$

A comunicação entre *Middlewares* ocorre conforme os eventos das regras, assim o intervalo de envio de cada dado é regido pelo evento, no caso chama-se de T2. O período de execução corresponde ao período do experimento P. Assim, a quantidade total de dados (Q2) enviada por regra cadastrada é:

$$Q2 = \frac{1}{T_2} \times P \Rightarrow P = Q2 \times T2 \quad (2)$$

Igualando as equações 1 e 2 para descobrir a relação entre a quantidade de dados enviadas, obtém-se:

$$Q1 \times T1 = Q2 \times T2 \Rightarrow \frac{Q2}{Q1} = \frac{T1}{T2} \quad (3)$$

Portanto, a relação do total de dados enviados entre o cenário com regras (Q2) e o cenário sem regras (Q1) é $\frac{T1}{T2}$. Salienta-se que no cenário da *cloud centralizada*, os sensores enviam direto para o *Middleware* seguindo a Equação 1. Percebe-se pela Equação 3 que a redução na quantidade de dados transmitidos depende do tempo de ocorrência dos eventos, quanto maior ele for, menos dados são enviados.

As Tabelas a seguir mostram os dados transmitidos em ambos os cenários.

Tabela 1. Fonte e Quantidade de Dados Enviados a *Cloud*

Fonte de dados	Quantidade de dados
Estação 1 (Temperature)	232,374 kB
Estação 1 (Humidity)	230,944 kB
Estação 2 (Temperature)	232,286 kB
Estação 2 (Humidity)	231,216 kB
Sensor de Luminosidade	232,082 kB
Total	1.158,902 kB

Tabela 2. Fonte e Quantidade de Dados Enviados à Rua 1 - Arquitetura Hierárquica

Fonte de dados	Quantidade de dados
Casa 1 (Temperatura)	3,862 kB
Casa 1 (Umidade)	3,862 kB
Casa 2 (Temperature)	3,885 kB
Casa 2 (Umidade)	3,862 kB
Total	15,471 kB

Tabela 3. Fonte e Quantidade de Dados Enviados ao Bairro - Arquitetura Hierárquica

Fonte de dados	Quantidade de dados
Rua 1 (Temperatura Maior que 25 °C)	2,248 kB
Rua 1 (Luminosidade)	2,576 kB
Total	4,824 kB

O total de dados gerados que é trafegado pela Internet no caso da rede Hierárquica é a soma dos dados que foram enviados para os *Middlewares* da rua e do bairro, totalizando 20,295 kB. Esse valor é 98,25% menor do que o valor enviado pela Internet no cenário da *cloud* centralizada que foi de 1158,902 kB. Como mostrado na equação (3), a redução de dados depende de como a regra foi configurada.

No cenário da *cloud centralizada* observa-se que, a partir da Tabela 1, as estações meteorológicas enviaram juntas 464,66 kB de dados de temperatura e 462,16 kB de dados de umidade para o *Middleware*. No caso dos *Middlewares* hierarquizados, os dados de temperatura e umidade foram enviados a cada 5 minutos pelas casas, somando 7,747 kB e 7,724 kB, respectivamente. A redução de dados de umidade trafegados pela Internet foi de aproximadamente 98.33%. Para calcular a redução de dados de temperatura, deve-se considerar o tráfego do *Middleware* da rua para o *Middleware* do bairro, que ocorre quando a média da temperatura ultrapassa 25 °C. Assim, o total de dados de temperatura no cenário hierárquico foi de 9,995 kB, que é um valor 97.85% menor que no caso da *cloud centralizada*.

Para a redução no tráfego de dados de umidade, pode-se substituir os valores de 5 segundos em T1 (intervalo entre envio de dados) e 300 segundos em T2 (intervalo entre ocorrência dos eventos) na Equação (3) e observar que a redução no cenário hierárquico foi de 98,33% (1 - 0,017). O valor experimental de 98,33% obtido condiz com o valor esperado.

$$\frac{Q_2}{Q_1} = \frac{5}{300} = 0,017 \quad (4)$$

Não é possível utilizar o modelo matemático para prever a redução do tráfego dos dados de temperatura já que a regra do *Middleware* da rua não tem periodicidade determinável. Ainda assim, é possível aplicar a Equação (3) nos dados trocados entre os *Middlewares* das casas e da rua. Aqui, a frequência do envio dos sensores e dos *Middlewares* são as mesmas dos valores de umidade, então vale o valor exposto na equação (5). Nesse caso, os *Middlewares* das casas enviaram juntos uma quantidade de dados 98,33% menor.

Observa-se na Tabela 1 que no caso da *cloud centralizada* foram enviados 232,082 kB de dados de luminosidade. No caso da rede hierarquizada, apenas a média dos valores a cada 100 amostras foi enviada do *Middleware* da rua para o *Middleware* do bairro, totalizando 2,256 kB, conforme Tabela 3, valor 99.02% menor. Como a média dos dados é realizada a cada 100 amostras e o sensor envia dados a cada 5 segundos, a média é executada a cada 500 segundos. Substituindo tais valores na Equação (3), obtém-se:

$$\frac{Q_2}{Q_1} = \frac{5}{500} = 0,01 \quad (5)$$

Matematicamente, a redução no cenário hierárquico foi de 99% (1 - 0,01). O valor experimental de 99,03% obtido, condiz com o valor esperado. As reduções obtidas são exorbitantes, no entanto, deve-se observar que essa redução depende das regras configuradas, pois essa redução pode ser menor se a frequência do evento também for menor. Tudo depende de como os *Middlewares* estão configurados, sendo que o ponto principal é que, a partir de todos os módulos implementados, esses oferecem essa possibilidade.

5. CONCLUSÃO

Neste trabalho foi desenvolvido um *Middleware* IoT genérico capaz de receber regras de sumarização e envio de dados, permitindo a disposição destes de forma hierárquica a fim de reduzir o tráfego de dados na rede e permitir armazenamento distribuído. Para validar a solução proposta, comparou-se duas arquiteturas: uma *cloud centralizada* e uma rede com os *Middlewares* hierarquizados no contexto de uma cidade. O mesmo *Middleware* foi utilizado nos dois casos.

Com os resultados obtidos, foi possível testar o funcionamento do *Middleware* e mostrar experimentalmente a capacidade de redução no tráfego de dados aplicando um modelo de descentralização de dados e cadastro de regras para a obtenção de informações. As reduções obtidas são exorbitantes, entretanto deve-se sinalizar que esta redução depende da configuração das regras, então quando se leva em conta a precisão das informações, esta redução pode ser menor, dependendo da configuração do *Middleware*.

Durante o desenvolvimento da solução, não foram levadas em conta questões de segurança, logo é necessário aprofundamento no tema. Algoritmos de *Machine Learning* e teorias estatísticas para geração de sumarização automatizada e obtenção de informações precisas, com base nos dados persistidos e transitados, também são pontos que podem ser explorados, já que a solução proposta oferece meios para tal mesmo que, no cenário atual, as tarefas são executadas manualmente.

AGRADECIMENTO

Os autores agradecem o apoio das Agências brasileiras de pesquisa, desenvolvimento e inovação CNPq (Projeto INCT SegCiber 465741/2014-2), CAPES (Projetos FORTE 23038.007604/2014-69 e PROBRAL 88887.144009/2017-00) e FAPDF (Projetos UIoT 0193.001366/2016 e SSDDC 0193. 001365/2016), bem como o suporte do Laboratório LATITUDE/UnB (Projeto SDN 23106. 099441/2016-43), e as cooperações com o Ministério da Economia (TEDs DIPLA 005/2016 e ENAP 083/2016) e o Gabinete de Segurança Institucional da Presidência da República (TED 002/2017).

REFERÊNCIAS

- Atlam, H. F., Walters, R. J., & Wills, G. B. (2018). Fog Computing and the Internet of Things: A Review. *big data and cognitive computing*, Vol. 2(2), 10.
- Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787-2805.
- Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012, Agosto). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile Cloud Computing* (pp. 13-16). ACM.
- Caldas Filho, F. L. de; Martins, L. M. C. e.; Araújo, I. P.; Mendonça, F. L. L. de; Costa, J. P. C. L. da; de Sousa Júnior, R. T. (2017, Dezembro). Design and Evaluation of a Semantic Gateway Prototype for IoT Networks. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC '17 Companion* (pp. 195-201), ACM.
- Cisco (2019). *Cisco Visual Networking Index: Forecast and Trends, 2017–2022* (White Paper C11-741490-00). [s.l.]: Cisco Systems, Inc.
- Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., & Hindia, M. N. (2018). An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet of Things Journal*, 5(5), 3758-3773
- Ferreira, H. G. C. & de Sousa Jr., R. T. (2017). Security analysis of a proposed internet of things middleware. *Cluster Computing*, 20(1), 651-660.
- Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements and future directions. *Future Generation Computer Systems*, 29(7), 1645-1660.
- Huacarpuma, R. C.; de Sousa Júnior, R. T.; Holanda, M. T. de; Albuquerque, R. de O., García Villalba, L., & Kim, T. H. (2017). Distributed Data Service for Data Management in Internet of THings Middleware. *Sensors*, 17(5).
- Lee, J. & Lee, J. (2018). Hierarchical Mobile Edge Computing Architecture Based on Context Awareness. *Applied Sciences*, 8(7), 1160.
- Martins, L. M. C. e; Caldas Filho, F. L. de; de Sousa Júnior, R. T.; Giozza, W. F.; Costa, J. P. C. L. da (2017, Dezembro). Increasing the Dependability of IoT Middleware with Cloud Computing and Microservices. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC '17 Companion* (pp. 203-208), ACM.
- Mohamed, N., Al-Jaroodi, J., Jawhar, I., Lazarova-Molnar, S., & Mahmoud, S. (2017). SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services. *IEEE Access*, 5, 17576-17588.
- Reinsel, D.; Gantz, J. & Rydning, J. (2018). *The Digitization of the World* (White paper US44413318). [s.l.]: IDC.
- Silva, C. C. de M., Ferreira, H. G. C., de Sousa Júnior, R. T., Buiati, F., & Villalba, L. J. G. (2016). Design and Evaluation of a Services Interface for the Internet of Things. *Wireless Personal Communications*, 91(4), 1711-1748.
- Tong, L., Li, Y., & Gao, W. (2016, Abril). A Hierarchical Edge Cloud Architecture for Mobile Computing. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. IEEE.
- Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., & Yang, X. (2017). A Survey on the Edge Computing for the Internet of Things. *IEEE access*, 6, 6900-6919.