8-10-2020

Uso da Computação em Névoa para Coleta e Análise de Dados em Cidades Inteligentes

Angelo Borsoi Ross
Aluno de Graduação, Universidade Tecnológica Federal do Paraná – Câmpus Curitiba, angeloborsoiross@hotmail.com

Daniel Fernando Pigatto
Universidade Tecnológica Federal do Paraná – Câmpus Curitiba, pigatto@utfpr.edu.br

Ana Cristina B. Kochem Vendramin
Universidade Tecnológica Federal do Paraná – Câmpus Curitiba, criskochem@utfpr.edu.br

Uso da Computação em Névoa para Coleta e Análise de Dados em Cidades Inteligentes

Artigo Completo

Angelo Borsoi Ross
Aluno de Graduação, Universidade Tecnológica Federal do Paraná – Câmpus Curitiba
angeloborsoiross@hotmail.com

Daniel Fernando Pigatto
Professor, Universidade Tecnológica Federal do Paraná – Câmpus Curitiba
pigatto@utfpr.edu.br

Ana Cristina B. Kochem Vendramin
Professora, Universidade Tecnológica Federal do Paraná – Câmpus Curitiba
criskochem@utfpr.edu.br

Abstract

This work proposes a two-layer communication architecture based on fog computing for smart cities. The architecture aims to store, process, and aggregate sensor data. To choose the most suitable communication protocol for the proposed architecture, we analyze the performance of the two most popular protocols for Internet of Things (IoT), the Message Queuing Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP). We considered latency, bandwidth consumption, efficiency in case of data loss, and energy efficiency as performance metrics. After the protocol is chosen, we employ a topic-based publish/subscribe system for sending and receiving messages, as well as issuing alerts based on received information. Two case studies on monitoring are shown: water quality in reservoirs and city air conditions. Results show that the application is able to receive data and issue alerts for both case studies.

Key-words

Fog Computing; Internet of Things; Smart Cities.

Resumo

Este trabalho propõe uma arquitetura de comunicação de dois níveis baseada em computação em névoa para cidades inteligentes com o objetivo de armazenar, tratar e agregar dados de sensores. Para escolha do protocolo de comunicação mais adequado para a arquitetura proposta, é analisado o desempenho dos dois protocolos mais populares para Internet das Coisas (IoT), MQTT (*Message Queuing Telemetry Transport*) e CoAP (*Constrained Application Protocol*). As métricas de desempenho quanto à latência, consumo de largura de banda, eficiência perante a perda de dados e eficiência energética são consideradas. Após a escolha do protocolo, emprega-se um sistema publicação/subscrição baseado em tópicos para envio e recebimento de mensagens, além da emissão de alertas com base nas informações recebidas. Dois estudos de caso de monitoramento são analisados: da qualidade da água em reservatórios e das condições do ar em uma cidade. Diante dos resultados, pode-se constatar que a aplicação é capaz de receber os dados e gerar alertas para ambos os estudos de caso.

Palavras-chave

Computação em Névoa; Internet das Coisas; Cidades Inteligentes.

Introdução

A internet das coisas, do inglês *Internet of Things* (IoT), é um paradigma de comunicação que visa interligar objetos da vida cotidiana. Xia et al. (2012) afirmam que esta tecnologia levará a uma rede altamente distribuída de dispositivos que se comunicará com seres humanos e outros dispositivos, tornando a internet ainda mais onipresente. Um dos principais exemplos de aplicação de IoT no contexto urbano são as cidades inteligentes e suas especializações, como automação residencial, transporte público, monitoramento da rede elétrica, reservatórios de água, qualidade do ar, entre outros (Zanella et al. 2014). Um setor que pode ser beneficiado pelo conceito de cidades inteligentes é o de saneamento. Companhias de saneamento devem tratar, cuidar e monitorar barragens e reservatórios a fim de manter a qualidade adequada da água. Esse monitoramento deve considerar coletas de amostras da água visando atender a requisitos que incluem aspectos microbiais, químicos e radiológicos (OMS 2011). A solução mais empregada para o monitoramento dos reservatórios de água é a coleta manual de amostragens para análise. Algumas soluções já exploram o uso de tecnologias IoT, como é o caso do uso de estruturas flutuantes, como as boias, equipadas com sensores e alimentadas através de baterias. Essa solução pode representar uma alternativa de custo mais acessível e que viabiliza a automatização do processo de análise da água. Porém, os equipamentos e sensores comumente precisam ser colocados em situações pouco ideais: expostos a elementos da natureza como chuva, vento, poeira e sol, operando com baterias e conectividade sem fio limitadas, poder de processamento baixo, entre outros fatores. Para auxiliar com o processamento e armazenamento desses dados, pode-se utilizar tecnologias em nuvem, as quais não sofrem as mesmas limitações que os dispositivos IoT. Associando-se esses conceitos, tem-se a computação de borda e a computação em névoa, que, apesar de diferentes entre si (a primeira faz interface com dispositivos finais e a segunda é uma camada intermediária entre a borda e a nuvem), visam melhorar serviços e tarefas de processamento de dados provenientes de dispositivos IoT localizados nas bordas da rede, antes de enviá-los a serviços de nuvem (Shi et al. 2016) (Butler 2018).

Trabalhos realizados na área de IoT incluem novos protocolos de comunicação específicos à área, integração de IoT e nuvem e gerenciamento de serviços em ambientes com restrições de recursos. Com foco em cidades inteligentes, Zanella et al. (2014) utilizam o CoAP como protocolo de comunicação e discutem a implantação de uma “ilha de IoT”, a qual se refere a uma rede de sensores com mais de 300 nós, implementada na Universidade de Pádua, para realizar demonstrações sobre redes inteligentes e serviços de saúde. Janet et al. (2019) propõem um sistema para monitorar o nível da água em lagos, reservatórios e represas, a fim de montar um sistema automático de proteção contra alagamentos e outros problemas que possam surgir. O projeto apresenta um sistema simples que utiliza um microcontrolador, um sensor de umidade, painéis solares e motores para demonstrar um sistema autônomo capaz de fechar e abrir comportas baseado no nível da água. Mitton et al. (2012) apresentam um modelo onde os sistemas IoT seriam tratados de maneira similar aos sistemas em nuvem: abstraídos, virtualizados e agrupados em sistemas "nuvem" por provedores especializados. A aplicação desta ideia requer uma infraestrutura capaz de gerenciar sensores heterogêneos e que consiga tratar dos mais variados problemas que possam surgir desde a interação com o usuário até a comunicação com a internet. Sarkar e Misra (2016) propõem uma arquitetura de computação em névoa e a comparação de seu desempenho com um modelo tradicional de computação em nuvem. O trabalho aponta que o uso da computação em névoa resulta em mais de 40% na redução do consumo energético em situações em que 25% das aplicações IoT demandam serviços de tempo real com baixa latência. Truong et al. (2015) propõem uma rede veicular baseada na computação em névoa. Os autores mencionam que este tipo de rede deve suportar planejamento de rotas, disseminação de alertas e serviços de nuvem para veículos.

Este artigo propõe uma arquitetura de comunicação de dois níveis baseada em computação em névoa para monitoramento de reservatórios de água e monitoramento das condições do ar em cidades inteligentes.

Computação em Névoa

De acordo com Butler (2018), a computação em névoa é o conceito de uma malha de rede que se estende das bordas onde dados são criados até onde esses serão eventualmente armazenados, seja na nuvem ou no centro de dados de um cliente. A computação em névoa pode ser entendida como uma rede distribuída que conecta dois ambientes: IoT e computação em nuvem. Ela estende a nuvem para ficar mais próxima dos dispositivos que produzem e agem sobre dados de IoT. Esses dispositivos, chamados nós de névoa, podem

ser instalados em qualquer lugar com uma conexão de rede. Segundo a Cisco (2015), qualquer dispositivo com poder de computação, armazenamento e conectividade à rede pode ser um nó da névoa.

Dispositivos IoT geralmente possuem poucos recursos, necessitando de protocolos de comunicação com baixo requisito computacional e energético, e que visem eficácia de transmissão de dados. Os dois protocolos de comunicação mais utilizados para dispositivos com perfil restrito, situados em ambientes com acesso limitado aos recursos, são o *Message Queuing Telemetry Transpor*t (MQTT) e o *Constrained Application Protocol* (CoAP) (Tinoco et al. 2016).

O MQTT é um protocolo de mensagens simples e leve que utiliza o *Transmission Control Protocol* (TCP) como protocolo de transporte (ISO 2016). O MQTT trabalha com três níveis de qualidade de serviço (*Quality of Service* - QoS) para transmissão de mensagens (IBM 2018): 1) QoS 0: mensagem é entregue no máximo uma vez ou não é entregue; 2) QoS 1: mensagem é sempre entregue pelo menos uma vez. Se a origem da mensagem não receber uma confirmação do destino, a mensagem será enviada novamente com o sinalizador "DUP" (indicando um pacote duplicado) até que uma confirmação seja recebida. 3) QoS 2: mensagem é entregue exatamente uma vez e deve ser armazenada na origem e destino até ser processada. É o modo de transmissão de mensagens mais confiável, porém mais lento. Esse modo exigirá pelo menos dois pares de transmissões de mensagens entre a origem e o destino antes de a mensagem principal ser excluída da origem. No primeiro par de transmissões, a origem transmite a mensagem e recebe a confirmação do destino de que ele armazenou a mensagem. Se a origem não receber uma confirmação, a mensagem será enviada novamente com o sinalizador "DUP" configurado até que uma confirmação seja recebida. No segundo par de transmissões, a origem informa ao destino que ele pode concluir o processamento da mensagem, através da mensagem de controle "PUBREL". Se o emissor não receber uma confirmação, a mensagem "PUBREL" é reenviada até que uma confirmação seja recebida.

O CoAP é um protocolo voltado a redes compostas por nós com recursos limitados, pouca disponibilidade de energia, e que podem sofrer elevada perda de dados (IETF 2014). Ele é baseado na troca de mensagens compactas transportadas utilizando o protocolo de transporte *User Datagram Protocol* (UDP).

Análise de Desempenho dos Protocolos MQTT e CoAP

Com foco nos protocolos de comunicação para computação em névoa, alguns autores voltam seus estudos para a análise de desempenho do MQTT e CoAP. A Tabela 1 apresenta um resumo dos resultados obtidos pelos artigos encontrados na literatura, onde destaca-se o protocolo que obteve o melhor desempenho considerando as seguintes métricas: latência, consumo de largura de banda, eficiência perante perda de dados (isto é, como os protocolos se comportam quando há perda de pacotes) e eficiência energética.

Artigo	Latência	Consumo de Largura de Banda	Eficiência Perante Perda de Dados	Eficiência Energética
Thangavel et al. 2014	MQTT	CoAP	MQTT	--
Chen and Kunz 2016	CoAP	CoAP	CoAP	--
Naik 2017	CoAP	CoAP	--	CoAP
Silva 2016	--	CoAP	CoAP	--
Bandyopadhyay and Bhattacharyya 2013	--	CoAP	CoAP	CoAP

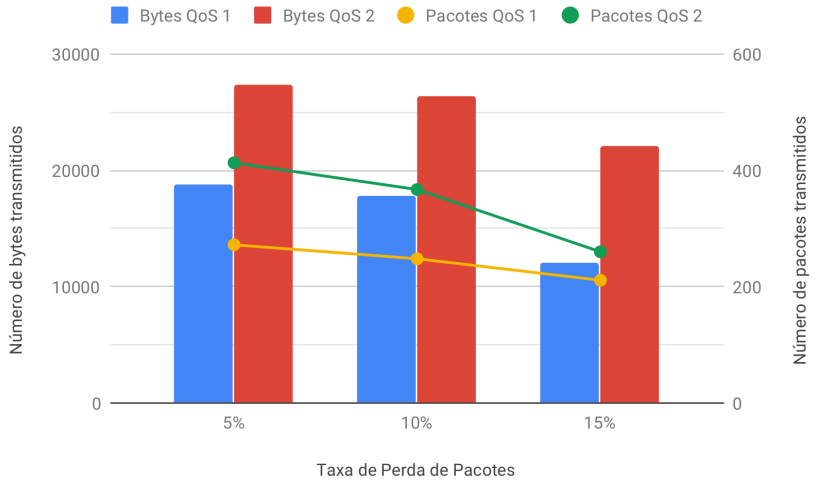
Tabela 1. Melhor protocolo de comunicação por métrica de desempenho

Considerando a métrica latência, o trabalho de Thangavel et al. (2014) afirma que o MQTT é o melhor protocolo, com uma exceção no caso de alta perda de pacotes durante a transmissão. Já os trabalhos de Chen and Kunz (2016) e Naik (2017) apontam o CoAP como superior na métrica latência. Em relação ao consumo de largura de banda, todos os trabalhos concordam que o CoAP consome menos largura de banda. Naik (2017) não analisa a eficiência dos protocolos perante perda de dados. Os demais artigos, com exceção do escrito por Thangavel et al. (2014), o qual define o MQTT como superior em relação à eficiência perante perda de dados, concordam que o CoAP é o protocolo mais aconselhável em situações com maiores taxas

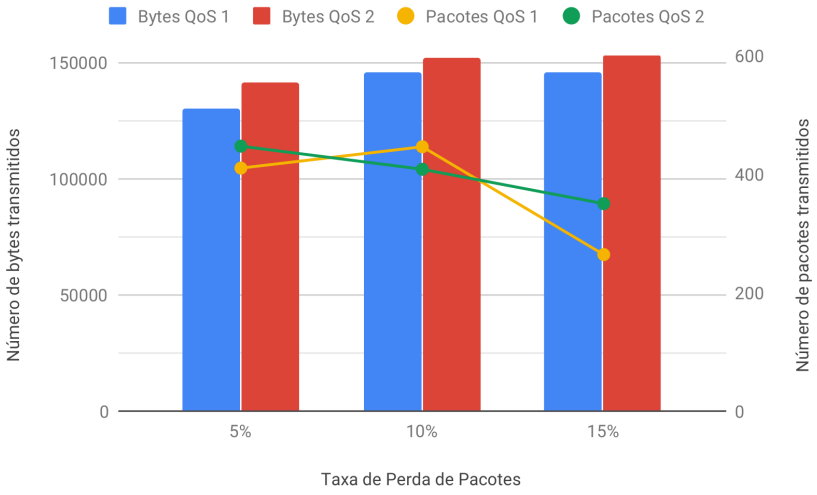
de perda de pacotes. Apenas Bandyopadhyay and Bhattacharyya (2013) e Naik (2017) analisam a métrica eficiência energética, sendo que ambos apontam o CoAP como o melhor protocolo a ser utilizado quando o consumo energético é uma preocupação.

O presente trabalho também analisou o comportamento dos protocolos MQTT e CoAP variando a taxa de perda de pacotes. Para isso, um *firewall* foi instalado no dispositivo (um Raspberry Pi 3B+) que recebe os pacotes dos sensores e este foi configurado para aleatoriamente descartar esses pacotes seguindo uma probabilidade de descarte de 5%, 10% e 15%. Dois tamanhos de pacotes foram escolhidos: 10 e 1000 *bytes*. Para cada protocolo, um total de 100 pacotes foram transmitidos, em intervalos de 0,1 segundo, variando a taxa de perda e o tamanho dos pacotes. O nível QoS do MQTT e as mensagens não confirmáveis do CoAP não foram considerados devido à característica não confiável.

A Figura 1 e a Tabela 2 mostram a quantidade de bytes e o número de pacotes transmitidos pelo protocolo MQTT, variando a taxa de perda de pacotes e o tamanho das mensagens.



(a) 10 *bytes*



(b) 1000 *bytes*

Figura 1. Transmissões do protocolo MQTT por taxa de perda de pacotes

Taxa de Perda	Mensagens 10 bytes		Mensagens 1000 bytes	
	QoS 1 bytes / pacotes	QoS 2 bytes / pacotes	QoS 1 bytes / pacotes	QoS 2 bytes / pacotes
5%	18789 / 272	27389 / 413	130052 / 410	141294 / 447
10%	17767 / 248	26371 / 367	145550 / 446	151910 / 408
15%	12034 / 211	22131 / 260	146054 / 264	153051 / 308

Tabela 2. Quantidade de bytes e pacotes transmitidos pelo protocolo MQTT

Devido à garantia de entrega de mensagem uma única vez, o QoS2 pode ser considerado mais confiável e, consequentemente, a versão com o modo de transferência mais lento, pois gera mais bytes e pacotes para realizar confirmações de recebimento. Com o aumento do tamanho da mensagem, a diferença no número de bytes enviados pelo Qos1 e QoS2 tende a diminuir. Esta característica torna o QoS 2 favorável em momentos nos quais a razão entre os dados úteis transmitidos e *overhead* adicionado é positiva e quando o impacto do *overhead* pode ser considerado pouco relevante no desempenho geral da aplicação.

A Tabela 3 e a Figura 2 mostram a quantidade de bytes e o número de pacotes transmitidos pelo CoAP, variando a taxa de perda de pacotes e o tamanho das mensagens. Ao utilizar o CoAP, percebe-se um aumento na quantidade de *bytes* e pacotes transmitidos proporcional ao aumento da taxa de perda de pacotes.

O protocolo CoAP gerou um tráfego menor se comparado ao MQTT QoS2. Este comportamento se deve à garantia de entrega (confiabilidade) provida pelo MQTT com QoS2. Considerando as métricas analisadas (latência, consumo de largura de banda, eficiência em situações com perda de dados e eficiência energética), escolheu-se o CoAP como protocolo de comunicação para a arquitetura de coleta e análise de dados em cidades inteligentes.

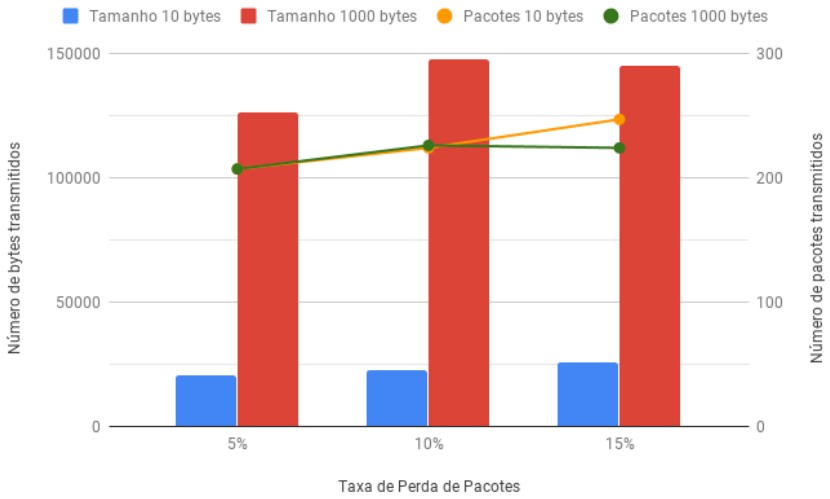


Figura 2. Transmissões do protocolo CoAP por taxa de perda de pacotes

Taxa de Perda	Mensagens 10 bytes bytes / pacotes	Mensagens 1000 bytes bytes / pacotes
5%	20375 / 207	126305 / 207
10%	22500 / 224	147490 / 226
15%	25375 /247	145260 / 224

Tabela 3. Quantidade de bytes e pacotes transmitidos pelo protocolo CoAP

Arquitetura

A arquitetura proposta é composta por dois níveis, como pode ser visto na Figura 3, além de uma interface que possibilite a comunicação com a Internet.

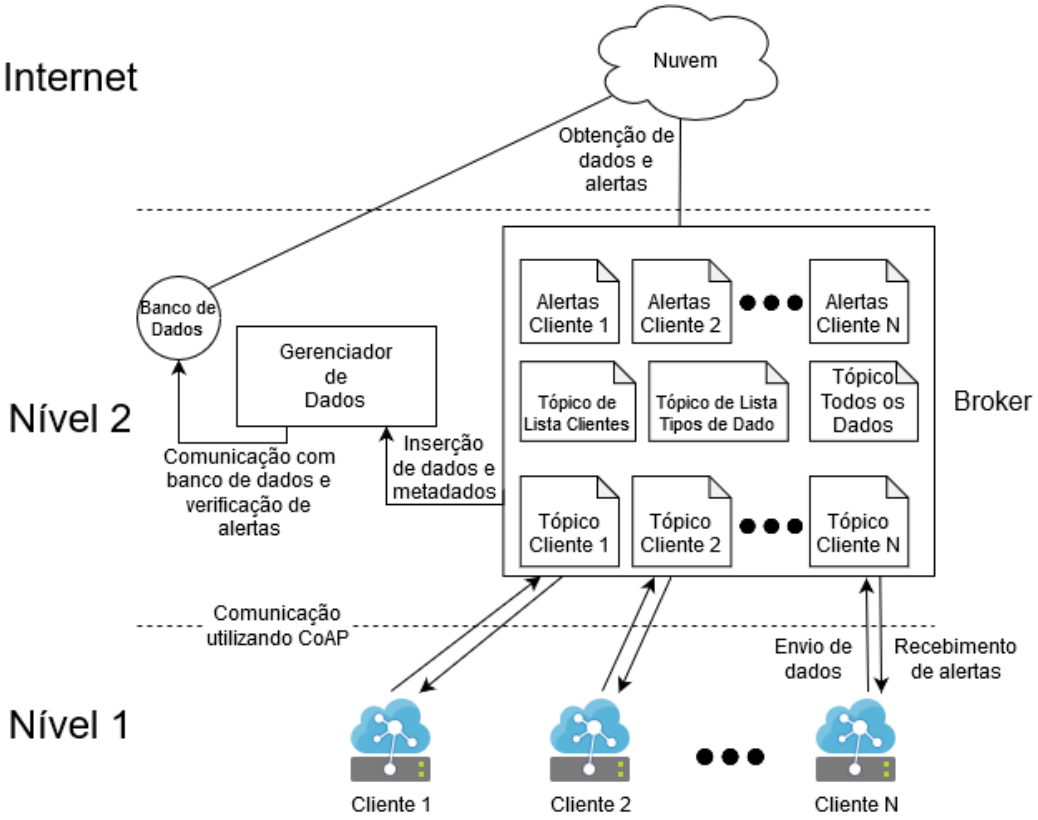


Figura 3. Arquitetura proposta

Os clientes encontram-se no nível 1 da arquitetura. Estes se comunicam com um nó central chamado de "*broker* de eventos", situado no nível 2, através de um paradigma chamado "publicação/subscrição". O *broker* é um nó intermediário que utiliza o protocolo CoAP para a comunicação entre publicadores e assinantes. Publicadores enviam mensagens para estes *brokers* enquanto assinantes enviam requisições de subscrição (registro de interesse em eventos) e recebem notificações de mensagens informando a ocorrência de eventos de seu interesse (Coulouris et al. 2012). No presente trabalho, o paradigma publicação/subscrição adota o modelo de subscrição baseado em tópicos (ou assuntos) (Coulouris et al. 2012), organizados de forma hierárquica. Cada notificação de mensagem é expressa em termos do número

de campos, onde cada campo denota um tópico. Subscrições são, então, definidas em termos do tópico de interesse.

Dois tópicos foram criados para cada cliente para que este possa enviar e receber dados ("`/client/CLIENTE`") e para o tratamento de alertas de eventos ocorridos no cliente ("`/alert/CLIENTE`"), onde CLIENTE representa o nome do cliente registrado. Além desses tópicos do cliente, outros tópicos foram criados para obter informações sobre: tipo específico de dado ("`/datatype/TIPO`"), lista de clientes registrados ("`/list/clients`"), lista de tipos de dados cadastrados ("`/list/datatypes`") e todos os dados cadastrados ("`/alldata`").

No nível 2 encontra-se também o gerenciador de dados, o qual cria uma ponte de interação entre o banco de dados e o *broker*. Ambos os tópicos do *broker* e o banco de dados podem ser acessados por aplicações na nuvem para obtenção dos dados coletados e alertas.

Estudos de Caso

A arquitetura proposta foi testada em dois estudos de caso de monitoramento: (A) qualidade da água em reservatórios; (B) qualidade do ar de uma cidade.

Estudo de Caso A: Qualidade da Água

Em um grande reservatório de água há a necessidade de realizar um monitoramento de diversas características, tais como nível da reserva, temperatura da água, quantidade de particulado, turbidez, fluxo de água etc. Seguindo estas características, pode-se especular sobre alguns tipos de alertas que podem ser gerados, tais como: temperatura abaixo ou acima de um parâmetro pré-definido, o nível da água abaixo ou acima do valor normal, baixa ou alta corrente/circulação de água em pontos determinados e, finalmente, baixa tensão na bateria dos clientes (isto é, sensores de monitoramento).

Um sistema foi desenvolvido para o estudo de caso A a fim de auxiliar na geração de dados e possibilitar a geração de alertas e validação da arquitetura proposta. Os dados foram inicializados com um valor padrão e passaram por um processo iterativo, sendo modificados e enviados a cada 15 segundos. No início da execução do sistema, notou-se o primeiro alerta (ver Figura 4), o qual indica que foi inserido um tipo de dado "*pressure_2*" e uma mensagem de alerta "`753,8 > 750`". Essa mensagem indica que um dado referente à pressão da água com valor "753,8" foi recebido e está acima do valor de limite absoluto superior especificado, que é de "750". Alguns minutos depois deste primeiro evento, mais alertas foram recebidos. A Figura 5 mostra uma segunda captura de tela na qual, além da mensagem de alerta para pressão da água mencionada anteriormente, há também duas outras mensagens de alerta, uma para o nível de tensão (tipo de dado "*volts*") na bateria do sensor de monitoramento e outra para a temperatura da água. O primeiro alerta indica que a pressão da água está acima do limite estipulado ("`821,09 > 750`"). A segunda mensagem é um alerta apontando que o nível de tensão da bateria do cliente "4,18" está dentro do intervalo estipulado. A terceira mensagem indica que a temperatura da água está acima do valor limite especificado ("`48,7 > 45`").

```
(venv) pi@myrkheim:~/tcc/tcc-fog-architecture/tests (dev) $ python alert_client.py localhost alert/water
_client0
Sending request to coap://localhost/alert/water_client0 with method GET and payload ""
First response
Response code: 2.05 Content
Payload: b'[{"n":"pressure_2","t":1573338307,"p":false,"a":"761.8000000000001 > 750"},{"n":"water_level_
1","t":1573338306,"p":false,"a":"7.400000000000001 < 10"}]'

===== ALERT =====
* Datatype: pressure_2
* Alert message: 753.8 > 750
===== ALERT =====
```

Figura 4. Primeira mensagem de alerta gerada durante o estudo de caso A

```
===== ALERT =====
* Datatype: pressure_2
* Alert message: 821.0999999999999 > 750
-----------------
* Datatype: volts
* Alert message: ['-4.2 < 4.181000000000002 < 4.2', None, None]
-----------------
* Datatype: temp
* Alert message: 48.7 > 45
===== ALERT =====
```

Figura 5. Demais mensagens de alerta geradas durante o estudo de caso A

Estudo de Caso B: Qualidade do Ar em uma Cidade

É comum a necessidade de monitoramento da qualidade do ar em cidades grandes. O estudo de caso B é baseado em um conjunto de dados real, disponibilizado publicamente pela *startup* Airly (2017), que visa utilizar redes de sensores para prover alertas sobre a qualidade do ar na cidade de Cracóvia, na Polônia. Segundo informações disponíveis junto à publicação dos dados, a Cracóvia é conhecida como uma das cidades mais poluídas da Europa. Os dados possuem a concentração de material particulado PM1, PM2,5 e PM10, onde "PM" é uma abreviação para "*Particulate Matter*" e o número representa o diâmetro do material em micrômetros, além da temperatura, umidade e pressão do ar coletados por 56 sensores, em diversos pontos da cidade, durante o ano de 2017. Várias mensagens de alerta podem ser geradas no estudo de caso B para alertar a população sobre possíveis riscos à saúde, tais como temperatura elevada, baixa umidade relativa do ar e alta concentração de poluentes.

Após iniciado o estudo no caso B, os primeiros alertas foram recebidos (ver Figura 6a). As primeiras mensagens de alerta indicavam que o valor do dado "*pm1*" estava acima do valor limite especificado (`"63 > 60"`). Na sequência foram recebidas notificações sobre uma variação na média dos 5 valores anteriores de temperatura, tipo de dados "*temp*", indicando que esta variação estava acima do valor especificado (`"9 > 6,25"`). O cliente foi parado e então reiniciado, escolhendo aleatoriamente um novo ponto nos dados para o início do envio. Imediatamente foram recebidos alertas sobre a umidade "*humidity*" e a temperatura "*temp*" (ver Figura 6b). A mensagem de alerta indicou que a umidade estava acima do valor limite cadastrado (`"96 > 95"`) e a temperatura estava muito abaixo da média especificada (`"3 < 12"`).

```
===== ALERT =====
* Datatype: pm1
* Alert message: 63 > 60
===== ALERT =====
```

```
===== ALERT =====
* Datatype: temp
* Alert message: 9 > 1.25*5.0
===== ALERT =====
```

(a) PM1 e temperatura

```
===== ALERT =====
* Datatype: humidity
* Alert message: 96 > 95
-----------------
* Datatype: temp
* Alert message: 3 < 0.75*16.0
===== ALERT =====
```

(b) Umidade e temperatura

Figura 6. Mensagens de alerta geradas durante o estudo de caso B

Conclusão

O objetivo deste trabalho foi criar uma arquitetura de dois níveis com base na computação em névoa para coleta e análise de dados provenientes de sensores em cidades inteligentes, principalmente para efetuar o monitoramento de reservatórios de água e o monitoramento de condições do ar. Dado que esses sensores possuem restrições quanto ao seu consumo energético e a rede pode dispor de baixa disponibilidade de largura de banda e elevadas taxas de perdas de pacotes, busca-se um protocolo de comunicação que otimize o uso de recursos. Analisando o desempenho dos protocolos de comunicação MQTT e CoAP, determinou-

se o CoAP como o protocolo mais adequado aos objetivos propostos considerando as seguintes métricas de desempenho: latência no envio dos dados, largura de banda consumida na transmissão dos dados, eficiência em situações de perda de dados e consumo energético.

Dois estudos de casos de monitoramento foram considerados: qualidade da água em reservatórios e condições do ar de uma cidade. A arquitetura mostrou-se eficiente para armazenar, tratar e agregar dados provenientes de sensores e gerar alertas durante o monitoramento. Como trabalhos futuros, dada a flexibilidade da arquitetura, espera-se aplicá-la em outros setores de uma cidade inteligente.

Referências

Airly. 2017. Air Quality Data from Extensive Network of Sensors.

Bandyopadhyay, S., and Bhattacharyya, A. 2013. Lightweight Internet Protocols for Web Enablement of Sensors Using Constrained Gateway Devices.

Butler, B. 2018. What Is Fog Computing? Connecting the Cloud to Things.

Chen, Y., and Kunz, T. 2016. Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network.

Cisco. 2015. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are.

Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. 2012. Distributed Systems: Concepts and Design, (5th ed.), Pearson.

IBM. 2018. Qualidades de Serviço Fornecidas Por Um Cliente MQTT.

IETF. 2014. The Constrained Application Protocol (CoAP).

ISO. 2016. ISO/IEC 20922:2016.

Janet, J., Balakrishnan, S., and Rani, S. S. 2019. “IoT Based Lake and Reservoir Management System,” International Journal of Lakes and Rivers (12:1), pp. 27–32.

Mitton, N., Papavassiliou, S., Puliafito, A., and Trivedi, K. S. 2012. Combining Cloud and Sensors in a Smart City Environment, SpringerOpen.

Naik, N. 2017. Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP.

OMS. 2011. Guidelines for Drinking-Water Quality, World Health Organization.

Sarkar, S., and Misra, S. 2016. “Theoretical Modelling of Fog Computing: A Green Computing Paradigm to Support IoT Applications,” Iet Networks (5:2), IET, pp. 23–29.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. 2016. “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal (3:5), IEEE, pp. 637–646.

Silva, D. S. da. 2016. Comparação de Protocolos de Comunicação CoAP, MQTT e HTTP Utilizando Eclipse Ponte e Amazon Web Services.

Thangavel, D., Ma, X., Valera, A., Tan, H.-X., and Tan, C. K.-Y. 2014. Performance Evaluation of MQTT and CoAP via a Common Middleware.

Tinoco, C., Maestrelli, M., and Maia, T. 2016. A Internet Das Coisas.

Truong, N. B., Lee, G. M., and Ghamri-Doudane, Y. 2015. “Software Defined Networking-Based Vehicular Adhoc Network with Fog Computing,” 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, pp. 1202–1207.

Xia, F., Yang, L. T., Wang, L., and Vinel, A. 2012. "Internet of Things." International Journal of Communications Systems (25:9), pp. 1101-1102.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. 2014. “Internet of Things for Smart Cities,” IEEE Internet of Things Journal (1:1), pp. 22–32. (https://doi.org/10.1109/JIOT.2014.2306328).