PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MAPEAMENTO DE RECURSOS SOB COMPUTAÇÃO EM NÉVOA

FELIPE BRIZOLA BERGUES DURO

Proposta de Trabalho de Conclusão apresentada como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Prof. Dr. Sérgio Johann Filho

Porto Alegre
2018

LISTA DE FIGURAS

Figura 2.1 – Requisição e resposta utilizando mensagens não confirmáveis. 9
Figura 3.1 – Organização arquitetural. 12
Figura 3.2 – Pilha de protocolos. 13
Figura 4.1 – Cronograma de atividades. 16

LISTA DE ALGORITMOS

LISTA DE SIGLAS

BGP – Border Gateway Protocol

COAP – Constrained Application Protocol

HTTP – Hypertext Transfer Protocol

IP – Internet Protocol

LE – Low Energy

NIST – National Institute of Standards and Technology

TCC – Trabalho de Conclusão de Curso

TCP – Transmission Control Protocol

RFC – Request for Comments

UDP – User Datagram Protocol

SUMÁRIO

1 INTRODUÇÃO **6**
1.1 CONTEXTUALIZAÇÃO 6
1.2 MOTIVAÇÃO E JUSTIFICATIVA 7
1.3 OBJETIVO 7

2 FUNDAMENTAÇÃO TEÓRICA **8**
2.1 BGP 8
2.2 COAP 8
2.3 CONSTRAINED RESTFUL ENVIRONMENTS (CORE) LINK FORMAT 10
2.4 TRABALHOS RELACIONADOS 11

3 PROPOSTA DE TRABALHO **12**
3.1 ARQUITETURA 12
3.2 MÓDULOS 13
3.2.1 DESCOBERTA E SINCRONIZAÇÃO DE RECURSOS 13
3.2.2 MIDDLEWARE 14
3.3 RESULTADOS ESPERADOS 14
3.4 VALIDAÇÃO 14
3.5 CENÁRIOS DE TESTE 14

4 METODOLOGIA CRONOGRAMA DE DESENVOLVIMENTO **16**

REFERÊNCIAS **17**

1. INTRODUÇÃO

1.1 Contextualização

A computação em nuvem tem sido amplamente adotada nos últimos anos por usuários finais e empresas de vários segmentos e portes. As facilidades proporcionadas por este modelo de computação faz com que exista tal preferência, pois segundo a definição adotada pelo NIST [10]: "computação em nuvem é um modelo que permite acesso a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser provisionados e liberados com um pequeno esforço de gerenciamento ou interação com provedor de acesso". Toda essa flexibilidade pode justificar o aumento no emprego desse modelo de computação.

Como a computação em nuvem não é Panacéia[1], podemos utilizar a definição de Bonomi, Milito, Zhu e Addepalli [4] que diz: "a computação em nuvem libera as empresas e os usuários finais de muitos detalhes de especificações. Essa facilidade torna-se um problema para aplicações sensíveis à latência, que requerem que nós próximos atendam suas necessidades de forma eficiente". Como a interação entre os nós e os servidores na nuvem ocorrem através da internet, a baixa latência torna-se indispensável para aplicações que requerem eficiência na comunicação entre nós (por exemplo robôs, drones, e carros autônomos). Portanto há uma lacuna entre aplicações que já utilizam modelos de computação em nuvem e aplicações que necessitam de baixa latência de rede e comunicação entre nós próximos, e é nesse hiato que a computação em névoa surge.

A computação em névoa é um assunto relativamente novo e teve sua primeira definição, dada pela Cisco Systems[2] em 2012, como uma extensão do paradigma de computação em nuvem provendo armazenamento, computação e serviços de rede entre dispositivos finais e os servidores na nuvem [13].

Atualmente a computação em névoa tornou-se um paradigma próprio e não mais uma mera extensão da computação em nuvem. Esse paradigma criou o conceito de *fog nodes*, que abrangem desde dispositivos finais com baixa capacidade computacional até servidores poderosos na nuvem. Assim, os *fog nodes* passam a fazer parte da implementação dos serviços em nuvem. O que torna a computação em névoa interessante é a capacidade dessa variedade de dispositivos cooperarem uns com os outros de forma distribuída.

Em uma abordagem *bottom-up*, podemos descrever a arquiterura da computação em névoa como um conjunto de *edge devices*[3] que se comunicam com os *fog nodes*, e es-

[1] Mecanismos ou práticas que, hipoteticamente, são capazes de solucionar os problemas e/ou dificuldades [2].

[2] Empresa estadunidense líder na fabricação de equipamenteos de rede [14].

[3] Dispositivo que controla o fluxo de dados no limite entre duas redes [15].

ses com servidores centrais. Entretanto, a comunicação entre os *fog nodes* e os servidores centrais nao é essencial para a execução dos serviços em névoa [13].

1.2 Motivação e justificativa

Serão sete os desafios que a computação em névoa deverá enfrentar para tornar-se realidade, segundo Vaquero e Rodero-Merino [19].

Os problemas referentes à padronização, descoberta e sincronização são os desafios a seres explorados neste trabalho, uma vez que atualmente não existem mecanismos no qual um membro da rede, seja ele uma *raspberry-pi* gerenciando sensores ou um computador, mapeie os recursos disponíveis e divulgue os seus na rede.

1.3 Objetivo

O objetivo principal deste trabalho de conclusão é construir um protocolo de rede que seja capaz de: descobrir, sincronizar e utilizar recursos de nodos em uma rede sob computação em névoa.

O interfaceamento entre os recursos vinculados aos *fog nodes* serão apresentados em forma de simulação, ou seja, os *edge devices* não estarão atrelados aos *fog nodes* de fato.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresentará alguns protocolos de comunicação e técnicas que servirão de apoio para o desenvolvimento deste TCC. Por fim, alguns trabalhos relacionados a área serão expostos, bem como o posicionamento deste trabalho perante aos demais.

2.1 BGP

O protocolo BGP está situado na quinta camada, a camada de aplicação, do modelo de referência TCP/IP [18]. Abaixo, de forma sucinta, elencaremos algumas funcionalidades básicas do protocolo que embasarão o restante deste trabalho.

- A responsabilidade deste protocolo é manter a troca de informações sobre roteamentos entre sistemas autônomos [11].
- O roteador ao entrar na rede pela primeira vez deve-se conectar ao seu vizinho. Após a conexão estabelecida, os roteadores compartilham entre sí suas tabelas de roteamento [11].
- Posteriormente, as atualizações nas tabelas dos roteadores dão-se de forma incremental à medida que as mudanças na rotas ocorrem [11].
- Mensagens de *keep alive* são trocadas periodicamente a fim de garantir conectividade entre os roteadores [11].

2.2 CoAP

Especificado pela RFC-7252, o CoAP, protocolo de aplicação restrita, foi projetado para aplicações máquina-a-máquina e tem como foco a transferência de documentos web entre nodos com recursos limitados em redes de baixa qualidade[17].

O modelo de interação cliente/servidor é o padrão adotado pelo CoAP, entretanto, o fato do protocolo ter sido projetado para aplicações máquina-a-máquina faz com que os dispositivos comumente desempenhem o papel de cliente e servidor simultaneamente.

Quando mensagens CoAP de requisição e resposta são trocadas, estas devem conter o código do método ou código da resposta, respectivamente. Além dos códigos, as mensagens podem conter outras informações, como o recurso que se deseja acessar e o tipo de mídia que se está transportando. Por fim, um token é utilizado para que haja a correspondência entre requisição e resposta.

A Figura 2.1 ilustra uma solicitação entre cliente e servidor, na qual o cliente deseja obter a temperatura. Analisando a troca de mensagens, notamos que o cliente envia uma solicitação não confirmável com token 0x74 e código GET para acessar o recurso temperatura no servidor. O servidor por sua vez retorna o código de resposta 2.05, que indica sucesso, o mesmo token que recebeu na solicitação do cliente e o valor 22.5 C.

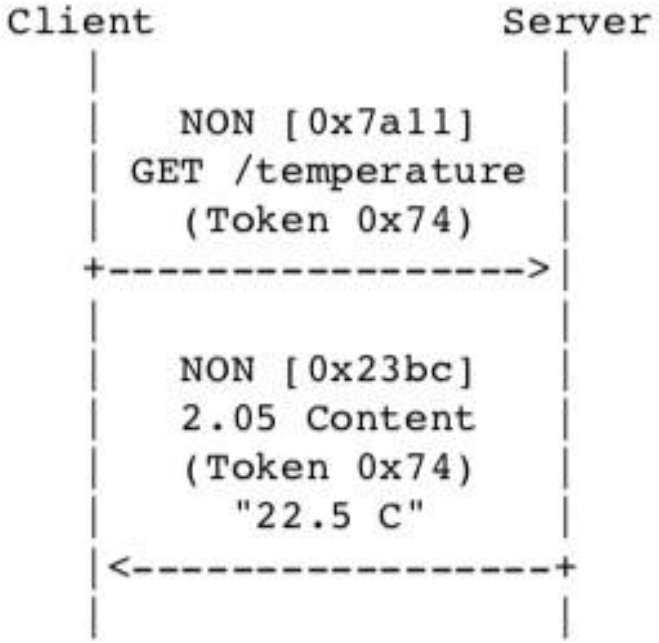

Figura 2.1 – Requisição e resposta utilizando mensagens não confirmáveis. [17]

A arquitetura REST, assim como no protocolo HTTP[5], foi utilizada na projetização do protocolo CoAP. Como ambos compartilham da mesma arquitetura, realizar o mapeamento de HTTP para CoAP e vice-versa é bastante simples. Para realizar tal mapeamento basta utilizarmos o *cross-proxy*, definido na sessão 10 da própria RFC-7252[17], que converte o método ou tipo de resposta, tipo de mídia e opções para os valores HTTP correspondentes.

Além de modelo de comunicação cliente/servidor e arquitetura REST, o CoAP também dispõem de outros princípios comuns ao HTTP e que são conceitos padrão na web como suporte a URIs[3] e tipos de mídia da Internet(MIME)[7].

Entre o HTTP e CoaP nem tudo são semelhanças, pois, obviamente, este possui particularidades para que consiga atender aos requisitos no qual se propõe a resolver. Entre as diferenças está a troca de mensagens assíncronas utilizando transporte orientado a datagramas, como UDP. Apesar de, naturalmente, o protocolo UDP não prover confiabilidade, no CoAP é possível definir que as mensagens possuam tal aspecto.

Constrained RESTful Environments (CoRE) Link Format, que será abordado na sessão a seguir, possibilidade de envio de solicitações multicast e unicast, *service disco-*

very, cabeçalho com baixa sobrecarga de dados e segurança na forma de DTLS[12] estão entre as principais caracteristicas do protocolo de aplicação restrita, CoAP.

2.3 Constrained RESTful Environments (CoRE) Link Format

Segundo a RFC-6690, a finalidade deste especificação é realizar REST em nodos com recursos limitados sendo importante em aplicações maquina-a-maquina[16]. Em aplicação deste tipo é primordial que as configurações não dependam de interação de humanos, e portanto, que mantenham seu funcionamento sem configurações estáticas.

A principal função desta especificação é fornecer identificadores, denominados URIs, para os recursos hospedados em servidores. Para além, é possível que essas URIs possuam atributos relacionados aos recursos. Estes atributos dividem-se basicamente em dois sendo eles rt, *resource type* e if *interface description*.

O primeiro é utilizado para para atribuir um nome que descreva de forma clara e enxuta um recurso, já o segundo tem por objetivo indicar uma interface especifica que interagirá com o recurso destino.

Um aspecto relevante ao CoRE, e de suma importancia para aplicações maquina-a-maquina, é o fato de possuir como URI padrão o prefixo */.well-known/core* , definido na RFC-5785[8]. Este prefixo é utilizado para que o servidor exponha suas políticas e recursos disponíveis.

O protocolo CoAP preve suporte à CoRE e ao prefixo */.well-known/core*. Contanto com esses recursos, é possível traçarmos uma pequena analogia entre esquemas HTTP/HTTPS e *coap*/*coaps*, este utilizando segurança DTLS. Estes esquemas são utilizados para identificar e localizar os recursos coap na rede. Sendo assim, servidores coap ficam aguardando por requisições que utilizem tal esquema.

Para exemplificarmos, abaixo esta um ciclo de requisição e resposta entre um cliente e um servidor com nomenclatura *example.net*. Este cliente deseja saber as políticas e recurso disponíveis pelo servidor e para tal utiliza o prefixo padrao */.well-known/core*.

```
REQ: GET coap://example.net/.well-known/core

RES: 2.05 Content
     </sensors/temp>;if="sensor",
     </sensors/light>;if="sensor"
```

Analisando a resposta é fácil notar que este servidor possui dois recursos disponíveis, sendo ambos com interface do tipo sensor, porem um utilizado para medir temperatura e outro para medir intensidade de luz. Juntamente aos dados, o codigo 2.05 é retornado indicando que a operação transcorreu com sucesso.

2.4 Trabalhos Relacionados

Spencer Lewson implementou um protocolo em nível de aplicação [18] capaz de realizar a comunicação entre nodos sob computação em névoa. A especificação do protocolo e um *middleware*, capaz de realizar o gerenciamento dos recursos dos dispositivos, são os principais componentes deste trabalho [9].

Sua implementação requer que haja um ponto central de comunicação entre os nodos, uma vez que a conectividade entre eles ocorre via *Bluetooth LE*. A existência desse ponto justifica-se pelas regras de implementação do *Bluetooth LE*, na qual descreve dispositivos de duas naturezas: centrais e periféricos. Dispositivos centrais são responsáveis por descobrir dispositivos periféricos que estão interessados em criar conexão. Portanto, a característica do *Bluetooth LE* faz com que a topologia de rede e a arquitetura do projeto não seja distribuída [9].

3. PROPOSTA DE TRABALHO

Seguindo a organização arquitetural da Figura 3.1, a proposta deste trabalho é fazer com que um *fog node* conheça, de forma autônoma, os recursos disponibilizados por seus vizinhos. Assim, cada *fog node* saberá quais são os *edge devices* disponíveis na rede, portanto, o nodo que possui o sensor de chuva saberia que existe um outro nodo na rede capaz de medir a temperatura, por exemplo.

Podemos observar na topologia da Figura 3.1, que os nodos não possuem um ponto centralizado de contato. Em razão da topologia distribuída, se for necessário escalarmos a quantidade de nodos na rede, a mesma não deverá sofrer impactos significativos de performance.

Nas seções a seguir abordaremos a arquitetura, módulos e submódulos que compõem o projeto, bem como resultados esperados, validações e cenários de teste.

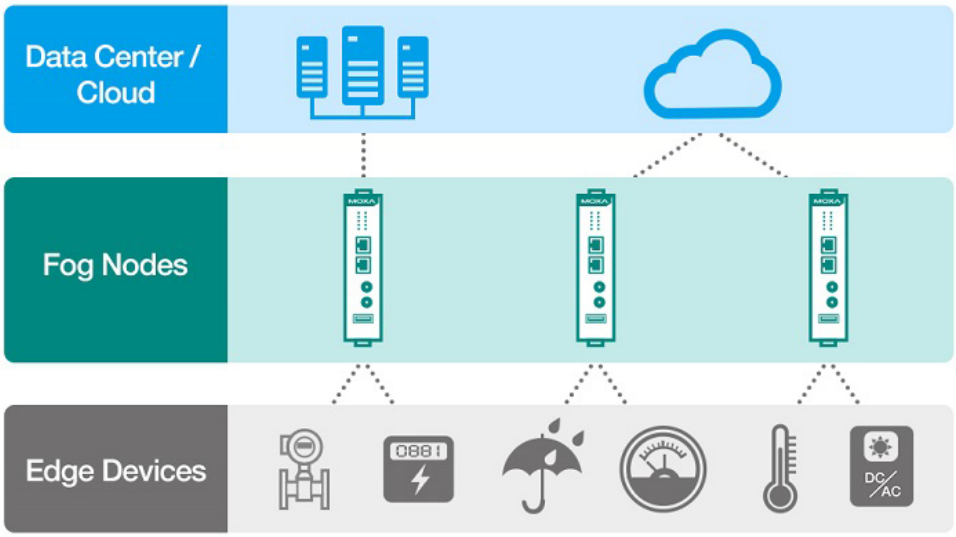

Figura 3.1 – Organização arquitetural.
[1]

3.1 Arquitetura

Esta seção define a pilha de protocolos a serem utilizados neste projeto, bem como a justifica pela escolha dos mesmos. A pilha de protocolos atuará em conjunto com a organização arquitetural previamente definida na Figura 3.1.

A fim de facilitar a compreensão da arquitetura deste projeto, a Figura 3.2 explicita a pilha de protocolos que o projeto fará uso para implementar as funcionalidades propostas.

Figura 3.2 – Pilha de protocolos.

Dispondo do modelo de referência TCP/IP, constituido de cinco camadas: física, enlace, rede, transporte e aplicação [18]. Este projeto utilizará IPv6 no nível de rede e UDP no nível de transporte.

[REVER] A utilização de IPv6 na camada de rede justifica-se pela grande quantidade de endereços válidos na internet que o protocolo provê, pois após o firmamento da internet das coisas é possível que cada dispositivo tenha um endereço IP único. Sendo assim, a escalabilidade, no que diz respeito a quantidade de endereços, está garantida.[REVER]

Manter todos os nodos conectados, utilizando TCP por exemplo, despenderia uma quantidade de trafego significativo na rede. Além disso, o intuito desta implementação é transitar uma pequena quantidade de dados a cada requisição. Portanto a utilização de datagramas UDP faz sentido neste projeto.

O protocolo proposto, entitulado *Resource Mapping* na Figura 3.2, atuará na camada de aplicação do modelo de referência TCP/IP [18] e será responsável por padronizar, descobrir e sincronizar os nodos da névoa. Os maiores desafios neste modelo proposto são manter o estado global dos recursos acessível a todos os nodos, e garantir que o desempenho seja satisfatório com o objetivo permitir a escalabilidade da solução.

3.2 Módulos

De forma geral, cada nodo da rede deverá manter uma lista com os endereços IP‘s que fazem parte do mapeamento. Atrelado à cada endereço IP, deverá haver uma lista com os recursos providos por este. Em vista disso, cada nodo portará um mapeamento global de recursos disponíveis na névoa.

O detalhamento das funcionalides que o projeto deverá dispor, bem como ilustrações relacionadas aos fluxos, serão abordadas nas proximas subseções.

3.2.1 Descoberta e sincronização de recursos

Como primeiro passo do mapeamento, devemos considerar a entrada de um novo nodo, com seus recursos, na rede. No momento em que o nodo dispor de um endereço IP

válido, este deverá enviar um pacote para o endereço de broadcast indicando que possui recursos a serem disponibilizados.

Ao receberem o pacote enviado por broadcast, os nodos que desejarem saber quais recursos estão sendo providos por este novo membro, deverão realizar uma query unicast para o endereço de origem indicando tal intenção.

Sendo assim, quem estiver interessado em obter os recursos do novo nodo, deverá realizar a query utilizando a URI */.well-known/core* do protocolo CoAP. Lembrando que este fluxo de requisição e resposta, utilizando a URI */.well-known/core*, faz com que o nodo requisitado retorne todos seus recurso ao solicitante.

3.2.2 Middleware

O *middleware* utilizado neste trabalho realizará o mapeamento local nos nodos e descobrirá quais são os recursos disponíveis a serem compartilhados na névoa. Utilizaremos uma simulação para realizar tal mapeamento local, portanto o mapeamento de fato não pertence ao escopo deste projeto.

3.3 Resultados esperados

Espera-se que este trabalho resulte em um protocolo funcional a nível de prova de conceito e que seja capaz de descobrir, sincronizar e consumir recursos de dispositivos sob computação em névoa. Além disso, temos como objetivo fazer com que a névoa configure-se de forma autônoma, ou seja, quando um recurso ou nodo entrar ou sair da rede, a mesma deverá manter-se coerente.

3.4 Validação

Para validarmos o funcionamento do protocolo, utilizaremos um simulador de dispositivos a ser definido no decorrer deste TCC. Estes dispositivos executarão o *middleware* citado no item 3.2.1 deste trabalho.

3.5 Cenários de teste

- Entrada de algum equipamento na rede e este anunciando seus recursos.

- Atualização das listas globais quando algum equipamento deixar de responder as mensagens de *keep alive*.
- Utilização de recursos de nodos da rede.
- Desligamento de recursos de algum dispositivo da rede, e posterior a isso, a tentativa de acesso a esse recurso por algum nodo. O equipamento que perdeu o recurso deverá anunciar seus novos recursos atualizando a lista global dos outros nodos da névoa.

4. METODOLOGIA CRONOGRAMA DE DESENVOLVIMENTO

Para construção do projeto utilizaremos Python, em sua versão 3.x, como linguagem de programação. Mais precisamente, utilizaremos a interface de baixo nível de rede da biblioteca padrão da linguagem [6]. Algumas ferramentas para auxiliar o desenvolvimento serão utilizadas, como Visual Studio Code para edição de código e GitHub para o versionamento.

Atividades	Março				Abril				Maio				Junho				Julho			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Definição do tema em específico		X	X																	
Elaboração da introdução, motivação e justificativa			X	X																
Elaboração da proposta de trabalho				X	X															
Elaboração da fundamentação teórica					X															
Revisão e refinamento do documento					X															
Entrega do documento						X														
Definição dos capítulos posteriores							X	X												
Aprofundamento teórico							X	X	X	X	X	X	X	X						
Elaboração dos próximos capítulos							X	X	X	X	X	X	X	X	X					
Revisão do documento final															X	X				
Entrega do documento final																	X			

Figura 4.1 – Cronograma de atividades.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] “Should you consider fog computing for your iiot?.” Capturado em: https://www.moxa.com/newsletter/connection/2017/11/feat_02.htm, 2018.

[2] “Significado de panaceia.” Capturado em: https://www.dicio.com.br/panaceia/, 2018.

[3] Berners-Lee, T.; Fielding, R.; Masinter, L. “Rfc 3986, uniform resource identifier (uri): Generic syntax”. Capturado em: http://rfc.net/rfc3986.html, 2005.

[4] Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. “Fog computing and its role in the internet of things”. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13–16.

[5] Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. “Rfc 2616, hypertext transfer protocol – http/1.1”. Capturado em: http://www.rfc.net/rfc2616.html, 1999.

[6] Foundation, P. S. “socket — low-level networking interface”. Capturado em: https://docs.python.org/3/library/socket.html, 2018.

[7] Freed, N.; Borenstein, D. N. S. “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types”. Capturado em: https://rfc-editor.org/rfc/rfc2046.txt, Nov 1996.

[8] Hammer-Lahav, E.; Nottingham, M. “Defining Well-Known Uniform Resource Identifiers (URIs)”. Capturado em: https://rfc-editor.org/rfc/rfc5785.txt, Abr 2010.

[9] Lewson, S. “Fog protocol and fogkit: A json-based protocol and framework for communication between bluetooth-enabled wearable internet of things devices”, 06 2015.

[10] Mell, P. M.; Grance, T. “Sp 800-145. the nist definition of cloud computing”, Relatório Técnico, Gaithersburg, MD, United States, 2011.

[11] Rekhter, Y.; Li, T. “A border gateway protocol 4 (bgp-4)”, 1995.

[12] Rescorla, E.; Modadugu, N. “Datagram Transport Layer Security Version 1.2”. Capturado em: https://rfc-editor.org/rfc/rfc6347.txt, Jan 2012.

[13] Roman, R.; Lopez, J.; Mambo, M. “Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges”, *CoRR*, vol. abs/1602.00484, 2016, 1602.00484.

[14] Rouse, M. “Cisco systems, inc.” Capturado em: https://whatis.techtarget.com/definition/Cisco-Systems-Inc, 2018.

[15] Rouse, M. “Edge device.” Capturado em: https://whatis.techtarget.com/definition/edge-device, 2018.

[16] Shelby, Z. “Constrained RESTful Environments (CoRE) Link Format”. Capturado em: https://rfc-editor.org/rfc/rfc6690.txt, Ago 2012.

[17] Shelby, Z.; Hartke, K.; Bormann, C. “The Constrained Application Protocol (CoAP)”. Capturado em: https://rfc-editor.org/rfc/rfc7252.txt, Jun 2014.

[18] Tanenbaum, A.; Wetherall, D.; Translations, O. “Redes de computadores”. PRENTICE HALL BRASIL, 2011.

[19] Vaquero, L. M.; Rodero-Merino, L. “Finding your way in the fog: Towards a comprehensive definition of fog computing”, *SIGCOMM Comput. Commun. Rev.*, vol. 44–5, Out 2014, pp. 27–32.