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

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

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 [18].

Os desafios referentes à padronização, descoberta e sincronização são os desafios a seres explorados neste trabalho, uma vez que atualmente não existem mecanismos padronizados no qual um membro da rede, seja ele uma *raspberry-pi* gerenciando sensores ou um computador, anuncie seus recursos ou consuma informações de outros membros.

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. Um *middleware* de simulação, que fará o interfaceamento entre um nodo e seus *edges devices*, será o objetivo secundário deste trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresentará uma breve descrição de alguns trabalhos relacionados a área de computação em névoa, bem como protocolos de comunicação que servirão de apoio para este TCC.

2.1 Trabalhos relacionados

O protocolo BGP está situado na quinta camada, a camada de aplicação, do modelo de referência TCP/IP [17]. 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].

Spencer Lewson implementou um protocolo em nível de aplicação [17] 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].

2.1.1 CoAP

Especificado pela RFC-7252[16], 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[16].

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 esta 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 no 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.

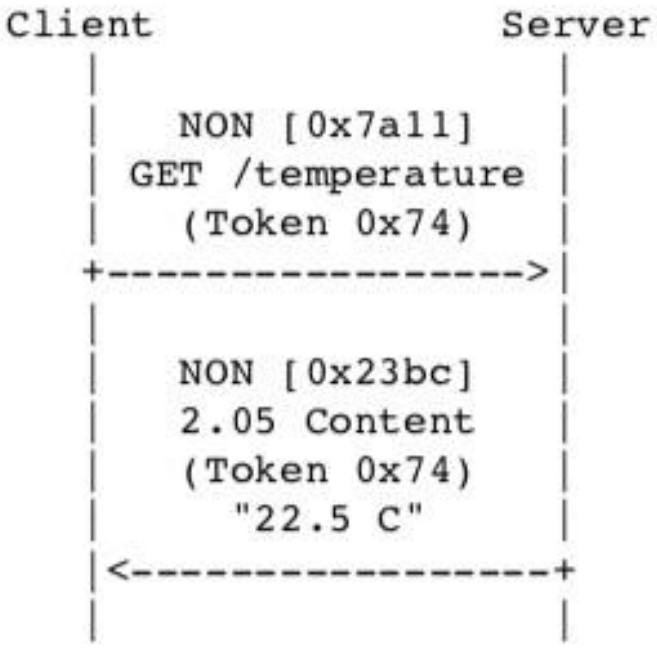

Figura 2.1 – Requisição e resposta utilizando mensagens não confirmáveis.
[16]

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 ma-

peamento basta utilizarmos o cross-proxy, definido na sessão 10 da própria RFC-7252[16], 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. CoAP-URI schema e service discovery, que serão abordados nas sessões a seguir, possibilidade de envio de solicitacoes multicast e unicast, cabeçalho com baixa sobrecarga de dados e segurança na forma de DTLS[12] estão entre as principais caracteristicas do protocolo de aplicação restrita, CoAP.

2.1.2 CoAP - URIs

De forma analoga ao "http"e o "https", o protocolo CoAP provê suporte aos esquemas do tipo "coap"e "coaps", este utilizando segurança DTLS. Estes esquemas são utilizados para identificar e localizar os recursos. Os recursos são organizados hierarquicamente e gerenciados por um servidor que fica escutando por requisições "coap"ou "coaps". A definicao sintatica, segundo Backus-Naur Form (ABNF) [RFC5234], da coap-uri está definida abaixo

coap-URI = "coap://"host [":"port] path-abempty ["?"query]

No coap-URI o campo "host"é obrigatorio e deve ser um endereco ip ou um nome que possa ser resolvido pelo DNS. O campo "path-abempty"identifica o recurso no escopo do host e port que se deseja acessar. Já o campo "query"é utilizado para realizar uma pesquisa no recurso e utiliza o padrao xxxx semelhante ao utilizado em querystrings no protocolo HTTP.

O esquema CoAP-URI suporta o prefixo "/.well-know/", definido da RFC-5785[8]. Este prefixo habilita a descoberta de políticas e outras informações sobre o servidor, como recursos disponíveis.

O exemplo abaixa demostra um fluxo de requisicao e resposta utilizando o prefixo "/.well-know/".

coap://example.net/.well-known/core

resposta

resposta

resposta

Notamos, entao, que o servidor nomeado example.net possui os recursos x,y e z disponíveis para serem utilizados via protocolo coap

2.1.3 CoAP - Service Discovery

3. PROPOSTA DE TRABALHO

Seguindo a organização arquitetural da Figura 3.1, a proposta deste trabalho é fazer com que um *fog node* consiga interagir com os *edges devices* de seus vizinhos. Assim, o nodo que controla a temperatura poderia utilizar recursos do *edge device* que comanda o sensor de chuva, 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.

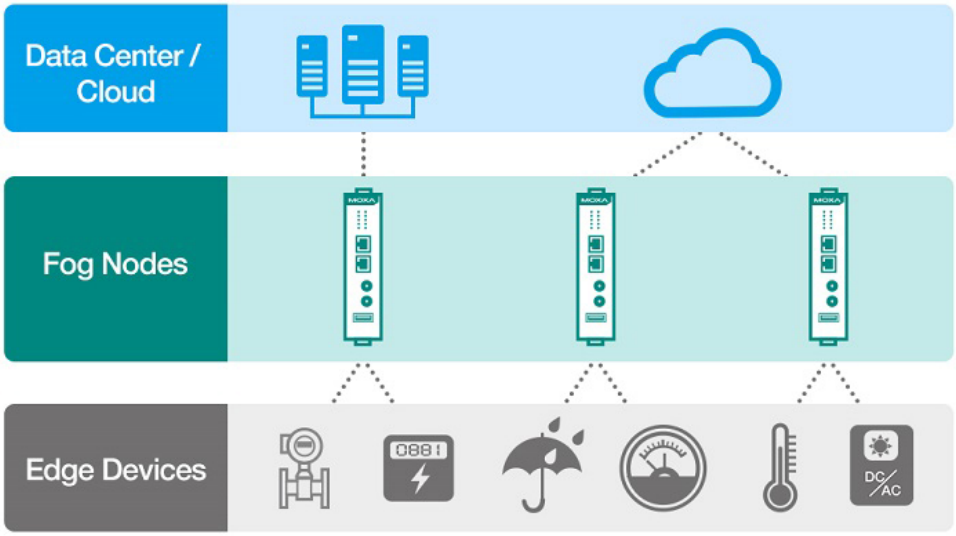

Figura 3.1 – Organização arquitetural.
[1]

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 [17]. Este projeto utilizará IPv6 no nível de rede e UDP no nível de transporte.

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.

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 [17] 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

Nas próximas subseções esclareceremos as funcionalidades que o projeto deverá dispor. Vale enfatizar que a descrição dos submódulos apresentados referem-se a proposta de TCC I. Consequentemente, o detalhamento de alguns módulos não serão abordados de forma aprofundada neste momento.

3.2.1 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.

3.2.2 Descobrimento e sincronização

Cada nodo da rede deve manter uma lista com os endereços IP's que fazem parte do mapeamento. Juntamente à cada endereço IP, deverá haver uma lista com os recursos providos por ele. Em vista disso, cada nodo da névoa possui um mapeamento global de recursos disponíveis na rede.

Quando um nodo entrar na rede pela primeira vez, deverá enviar um pacote para os endereços *unicasts* indicando os recursos que disponibilizará. O Pseudocódigo 3.1 demonstra, de forma simplificada, a política de atualização que cada nodo deverá implementar.

```
1: function Police(ip, resources)
2:     if exists(ip)
3:         update(ip, resources)
4:     else
5:         insert(ip, resources)
```

Algoritmo 3.1 – Política de atualização de recursos.

A manutenibilidade da lista de recursos globais é relevante para que a implementação do protocolo tenha sucesso, pois a névoa deverá saber quando um nodo, ou recurso dele, deixou de fazer parte da rede. Para tal, faz-se necessário a utilização de alguns mecanismos de controle. Esses controles são realizados em duas esferas, uma trata do desvinculamento do nodo da rede e outra quando um recurso deixa de fazer parte dela.

O desvinculamento pode ser abordado de forma similar as mensagens de *keep alive* utilizadas no protocolo BGP, por exemplo. Mensagens de keep alive são adotadas para que os nodos da rede avisem seus vizinhos que ainda estão operação, pois sem esse procedimento seria difícil saber quando remover um IP da lista de recursos globais. Portanto, para manter a lista o mais atualizada possível, este protocolo implementará mensagens deste tipo.

No momento em que um nodo requisitar um recurso global da névoa, e este estiver fora de operação, o nodo que foi solicitado deverá enviar um novo pacote para os endereços *unicasts* indicando quais são os seus recursos existentes no momento. O nodo solicitado percebe que este recurso não esta mais disponível porque recebeu na requisição o id do recurso, e este não está em sua lista local. Lembrando que o nodo requisitado está consciente desta divergência em virtude da execução continua dos métodos providos pelo *middleware*. Portanto, o gatilho para a atualização dos recursos na névoa é disparado pelo primeiro nodo que requisitar um recurso inválido.

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

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

[18] 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.