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

“Frase de impacto.”
(Nome de Alguem)

AGRADECIMENTOS

Agradeço a XYZ pessoas por bla bla bla

MAPEAMENTO DE RECURSOS SOB COMPUTAÇÃO EM NÉVOA

RESUMO

Computação em névoa... .Este trabalho propõe desenvolver um protocolo no qual seja possível mapear recursos de dispositivos conectados a rede local.

Palavras-Chave: mapeamento de recursos, computação em névoa, internet das coisas....

MAPPING RESOURCES UNDER THE FOG COMPUTING

ABSTRACT

Traducao do resumo

Keywords: resource mapping, fog computing, internet of things....

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

BGP – Border Gateway Protocol

IAAS – Infrastructure as a Service

IP – Internet Protocol

NIST – National Institute of Standards and Technology

PAAS – Platform as a Service

SAAS – Software as a Service

TCC – Trabalho de Conclusão de Curso

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização do problema

A computação em nuvem tem sido amplamente adotada nos últimos anos por vários tipos de empresas, pois além de disponibilizar diversos tipos de modelos de serviços como SaaS, PaaS e IaaS, provê modelos de implantação de infraestrutura em nuvem privada, comunitária, publica e híbrida.

Ainda temos, segundo definição de computação em nuvem adotada pelo NIST[3]: "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". Essa flexibilidade, tanto de modelo de serviços quando nos modelos de implatações, 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[1] 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 de 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 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[4].

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. Temos, então, a névoa como uma arquitetura de três camadas (clientes <->

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

fog nodes <-> servidores centrais) na qual os servidores centrais podem coexistir com os fog nodes, todavia esses servidores não são essenciais para a execução dos serviços em névoa [4].

1.2 Motivação e justificativa

Descoberta e sincronização, computação e limite de armazenamento, gerenciamento, segurança, padronização, monetização e programabilidade serão os sete desafios que a computação em névoa deverá enfrentar para tornar-se realidade, segundo Vaquero e Rodero-Merino[6].

Padronização, descoberta e sincronização serão os desafios explorados neste trabalho, pois hoje não há 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 nodos.

1.3 Objetivo

Partindo do pressuposto de que cada nodo desta névoa estará executando um middleware que gerencie seus recursos locais, 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.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo contém uma breve descrição de alguns conceitos que serão utilizados durante este trabalho e outras obras relacionadas.

2.1 Trabalho relacionado

Exitem alguns trabalhos que fazem uso de protocolos de comunicação voltados para computação em névoa.

Segundo abc..

3. PROPOSTA DE TRABALHO

Podemos observar na topologia da Figura 3.1, que os nodos não possuem um ponto centralizado de contato. Em razão da topologia distribuida, se for necessário escalarmos a quantadidade 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.

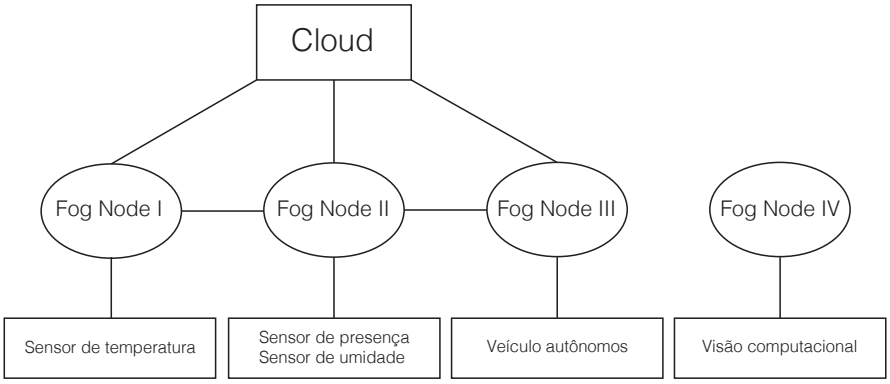

Figura 3.1 – Nodos da névoa.

3.1 Arquitetura

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 utilizada na implementação do projeto.

Dispondo do modelo de referência TCP/IP[5], constituido de cinco camadas: física, enlace, rede, transporte e aplicação, 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.

3.2 Módulos

Nas próximas subseções esclareceremos as funcionalidades que o projeto deverá dispor. Vale enfatizar que a descrição dos módulos 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.

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.

```
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.

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 do método getLocalResources. 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 devará 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.
- Atulização 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[2]. 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] 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.

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

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

[4] 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.

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

[6] 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.