Classificação do tráfego de rede do serviço de mensageria do OpenStack

Adnei W. Donatti[1], **Charles C. Miers**[1], **Guilherme P. Koslovski**[1]

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina

adnei.donatti@edu.udesc.br,

{charles.miers, guilherme.koslovski}@udesc.br

Resumo. *As nuvens computacionais usualmente são estudadas sobre aspectos que afetam diretamente seus usuários, deixando de lado o levantamento de informações sobre a operacionalização e funcionamento interno dos provedores de nuvem. Neste sentido, o presente artigo traz uma proposta de caracterização do tráfego de rede do serviço de mensageria do OpenStack. O conhecimento e entendimento desta tecnologia e de seu impacto em implantações de nuvens OpenStack é parte fundamental no planejamento da infraestrutura da nuvem. Além disso, a caracterização do tráfego produzido pelo serviço de mensageria do OpenStack ajuda a identificar aplicações e serviços em execução no momento em que são realizadas tarefas específicas na nuvem, e.g., criar uma instância de máquina virtual (MV).*

1. Introdução

O paradigma de computação em nuvem proporcionou diversos benefícios aos seus usuários. O acesso a informação é feito de maneira rápida e simplificada, que torna a tecnologia pervasiva e ubíqua na vida das pessoas. Neste sentido, há diversos campos de estudo que visam a *Quality of Service* (QoS) e a experiência dos usuários na nuvem [Chen and Zhao 2012, Chaudhary et al. 2018, Alenezi et al. 2019], deixando de lado, às vezes, operações internas à infraestrutura da nuvem e planejamento dos *data centers*. Assim, nota-se certa carência de informações com relação ao impacto de tarefas internas dos provedores (*e.g.*, disponibilização de MVs) no desempenho da própria nuvem.

No contexto de computação em nuvem, o OpenStack (www.openstack.org) destaca-se como uma solução de código aberto que opera como um sistema operacional (SO) para nuvens, no sentido de gerenciar e coordenar um amplo grupo de recursos computacionais (*e.g.*, recursos de redes, armazenamento e computação) distribuídos pelo *data center* [OpenStack 2019b]. Além disso, o OpenStack conta com uma comunidade expressiva de usuários e desenvolvedores ativos (github.com/openstack), com lançamento de uma nova versão a cada seis meses (atual versão: codinome Ussuri, 21ª versão).

A implantação de nuvens OpenStack pode ser feita de maneira flexível e sob planejamento, visando aspectos de segurança e desempenho da nuvem. Neste sentido, a personalização da nuvem fica a cargo dos administradores responsáveis, que devem estudar desde a topologia/organização até módulos e serviços operantes na nuvem. Um aspecto importante sobre nuvens OpenStack, de modo geral, é a divisão e isolamento das

redes do provedor de acordo com políticas de segurança. Basicamente, a rede do provedor do serviço em nuvem é dividida em três domínios: Domínio de Controle; Domínio de Convidados; e Domínio Público [OpenStack 2019a]. O foco deste trabalho está na caracterização do tráfego produzido pelo *broker* de mensageria RabbitMQ, que é tradicionalmente utilizado em nuvens OpenStack. O tráfego de rede do RabbitMQ é resultante da comunicação entre serviços do OpenStack. Portanto, está na rede administrativa, parte do Domínio de Controle, no qual estão presentes as operações mais internas da nuvem.

Este artigo está organizado de modo que a Seção 2 traz uma visão geral sobre a arquitetura de uma nuvem OpenStack; a Seção 3 fala sobre algumas abordagens para caracterização de tráfego de rede; a Seção 4 explica sobre o uso de *brokers* de mensageria no OpenStack, bem como as complicações envolvidas para caracterizar o tráfego gerado por serviços de mensageria; e a Seção 5 apresenta uma solução para caracterizar o tráfego do RabbitMQ, que é o *broker* de mensageria usado no OpenStack.

2. OpenStack: Arquitetura

O provisionamento de nuvens OpenStack é feito através de serviços. Estes serviços são divididos em módulos, que ficam espalhados pelos servidores da nuvem (Figura 1). Os principais módulos do OpenStack têm implementação em Python, acessíveis também via *Application Programming Interface*s (APIs), para casos de integração com aplicações externas, customizações de funcionalidades, entre outros. Além disso, a interface *web* e a *Command Line Interface* (CLI) permitem gerenciar as funcionalidades do OpenStack e ter uma visão geral do funcionamento da nuvem, *e.g.*, gerenciar projetos, MVs e topologias de rede [Scharf et al. 2015]. Dentre os módulos comumente abordados ao iniciar o planejamento para uma implantação do OpenStack, estão: Horizon; Keyston; Nova; Neutron; Glance; Swift; e Cinder [OpenStack 2020b].

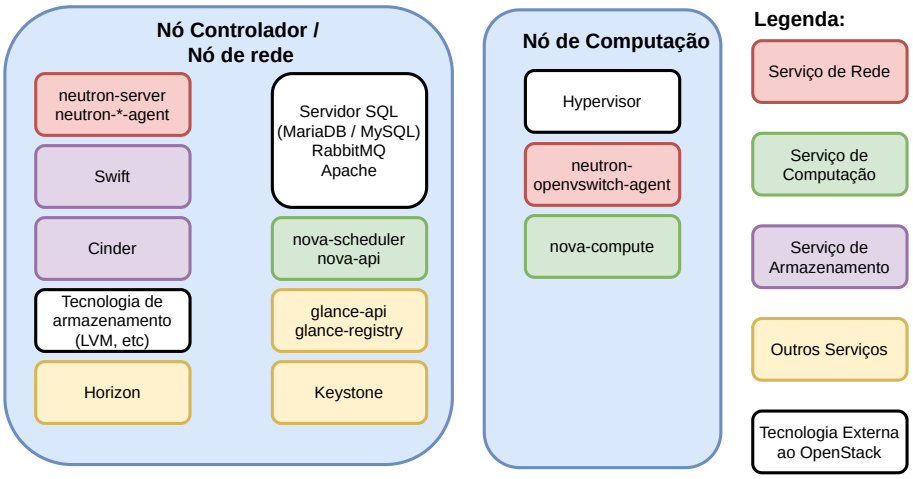

Figura 1. Arquitetura mínima do OpenStack.

A Figura 1 ilustra a distribuição dos módulos/serviços em uma arquitetura simples, com dois tipos de nós. Arquiteturas mais robustas costumam separar alguns serviços em nós e redes dedicadas, *e.g.*, nós de armazenamento em uma rede dedicada a estes. Além disso, um aspecto importante a se notar é a comunicação entre serviços. O uso de *brokers* de mensageria é uma estratégia desejável para ambientes distribuídos, atendendo a questão da fraca acoplagem e heterogeneidade desses sistemas, nos quais os componentes, ou partes, do sistema encontram-se espalhados pelo *data center*. Portanto, os serviços

do OpenStack trocam mensagens via *Advanced Message Queuing Protocol* (AMQP), que é implementado pelo serviço de mensageria de escolha do administrador da nuvem, tipicamente o RabbitMQ (http://rabbitmq.com). Contudo, é possível utilizar o Qpid (http://qpid.apache.org) ou o ZeroMQ (http://zeromq.org) para a mesma finalidade.

Nesse sentido, os serviços OpenStack ainda fazem uso de APIs REST, as quais tornam possível o acesso às funcionalidades dos módulos (*e.g.*, fazer requisição ao Nova para criação de uma MV, fazer requisição ao Keystone para autenticação). Dessa forma, seja via REST ou AMQP, a integração e o trabalho conjunto dos serviços do OpenStack é garantida, estejam esses serviços no mesmo nó ou não. Além disso, toda essa comunicação entre serviços ocorre via rede administrativa, que está contida no domínio de redes mais interno das nuvens OpenStack (Figura 2).

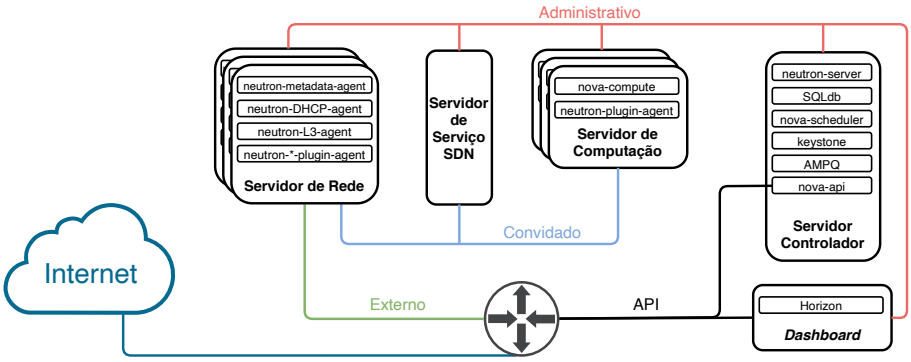

Figura 2. OpenStack: configuração de rede padrão [OpenStack 2019a].

A Figura 2 traz uma configuração de rede padrão recomendada pelo OpenStack. Nesta configuração, as redes do provedor são divididas em três domínios, de acordo com políticas de segurança: **Domínio Público**, formado pelas redes Externa e API, que permite acesso à Internet pelas MVs; **Domínio de Convidados**, formado basicamente pela rede de Convidados, que é responsável pelo tráfego de rede das MVs; e **Domínio de Controle** (ou administrativo), considerado o domínio mais interno da nuvem, já que agrega tráfego operacional, do funcionamento da nuvem, podendo ser composto pela rede administrativa e também uma rede de armazenamento [OpenStack 2020a]. Tipicamente, o gerenciamento dos domínios é feito via *Software Defined Networking* (SDN), *Virtual Local Area Network* (VLAN), ou simplesmente separando as redes fisicamente.

3. Caracterização de tráfego de rede

Dentre técnicas como a correspondência de padrões, estatística e decodificação de protocolo [Dainotti et al. 2012, Finsterbusch et al. 2014], o método de classificação de tráfego melhor aplicável ao contexto do Domínio de Controle do OpenStack é a abordagem baseada em portas. O Domínio de Controle do OpenStack é um ambiente controlado e com aplicações conhecidas, que usam portas TCP bem definidas. Dessa forma, a identificação da porta remete ao serviço/aplicação em execução no momento da captura do tráfego. A Tabela 1 traz um breve comparativo do nível de aplicabilidade considerando alguns desafios das principais técnicas usadas para classificação de tráfego de rede.

As técnicas de **correspondência de padrões** e **decodificação de protocolo** fazem uso da inspeção profunda dos pacotes, *Deep Packet Inspection* (DPI), o que acaba criando restrições maiores. Em relação à **correspondência de padrões**, geralmente são usadas

Tabela 1. Técnicas para classificação de tráfego de rede.

Técnica	Nível de Aplicabilidade	Desafios
Baseada em Portas	Alta	-Requer conhecimento sobre a rede -Conflito de portas -Portas não registradas no IANA
Método estatístico	Alta	-Variabilidade das características dos pacotes de acordo com o cenário -Protocolos com características parecidas
Correspondência de padrões	Média	-Criptografia -Privacidade -Recursos computacionais
Decodificação de protocolo	Baixa	-Alta complexidade -Manter atualizações -Criptografia e protocolos proprietários

expressões regulares (*RegEx*) para encontrar padrões nos conteúdos dos pacotes. Enquanto a **decodificação de protocolo** baseia-se na reconstrução com estado das sessões e informações das aplicações obtidas através do conteúdo dos pacotes. Por outro lado, a técnica **baseada em portas** e o **método estatístico** fazem uso de dados independentes da carga do pacote. A técnica **baseada em portas** usa as portas de comunicação do cabeçalho TCP/UDP para associar os pacotes às aplicações. Já o **método estatístico** faz uso de informações como as portas TCP/UDP, tamanho do pacote, tempo entre chegadas e duração dos fluxos para criar análises e interpretações estatísticas sobre o tráfego [Lin et al. 2006, Dainotti et al. 2012, Finsterbusch et al. 2014, Piskac and Novotny 2011].

Nesse sentido, a técnica de classificação de tráfego baseada em portas TCP mostra-se simples e bastante eficaz em cenários como a rede administrativa do OpenStack, no qual se tem conhecimento prévio da rede e de possíveis aplicações em execução. Além disso, também não há problemas com criptografia e privacidade de dados. Os principais desafios desta técnica encontram-se no uso de portas TCP não reconhecidas pelo experimentador (*e.g.*, não registradas no IANA) e conflitos de portas entre diferentes aplicações.

4. Mensageria no OpenStack

Os serviços de mensageria, de modo geral, são responsáveis pela troca de mensagens assíncronas entre dois sistemas distintos que não precisam conhecer um ao outro. Nuvens OpenStack fazem uso por padrão do *broker* de mensageria RabbitMQ. A comunicação interna, entre módulos e serviços OpenStack, é feita via AMQP através da rede administrativa. Além disso, a troca de mensagens pode ocorrer entre serviços distintos dentro de um mesmo módulo (*e.g.*, *nova-scheduler*, serviço escalonador do Nova, comunicando-se com o *nova-compute*, serviço de computação do Nova, para que a instância seja inicializada no servidor escolhido) ou até mesmo entre serviços de módulos diferentes (*e.g.*, *neutron-server* passando configurações de rede da MV ao *nova-compute*).

O formato de comunicação indireta entre dois sistemas é o que viabiliza e torna importante o uso de *brokers* de mensageria em sistemas distribuídos. A troca de mensagens entre os dois sistemas, no caso do OpenStack, entre dois serviços, é feita com intermédio do *broker*, que recebe e encaminha as mensagens para filas de acordo com certas características e condições preestabelecidas. Na sequência, o sistema destinatário terá acesso à mensagem na fila. Basicamente, tem-se um mecanismo orientado a eventos para troca de mensagens assincronas entre sistemas distintos.

Se por um lado, o uso de aplicações com APIs REST viabiliza bastante a aplicação de abordagens de classificação de tráfego baseadas em portas TCP, o AMQP, implementado pelo RabbitMQ, torna o cenário da rede administrativa do OpenStack mais desafiador. Enquanto que as API REST fazem uso de portas TCP bem definidas (*e.g.*, *nova-api*, com portas TCP/8773 e TCP/8775; e *glance-api*, com porta TCP/9292), as trocas de mensagens feitas com o RabbitMQ usam portas efêmeras, as quais não seguem um padrão por serviço OpenStack. Portanto, pela falta de um conjunto de portas predefinidas, há um certo aumento na complexidade da caracterização dos serviços que estão trocando mensagens via RabbitMQ.

5. Proposta & Resultados

A fim de seguir com o uso de uma abordagem de classificação de tráfego baseada nas portas de comunicação TCP dos pacotes, já que é uma técnica que mostra-se viável e eficaz no contexto do Domínio de Controle do OpenStack, é necessário lidar com a caracterização do tráfego de rede produto da troca de mensagens com o *broker* de mensageria RabbitMQ. Neste sentido, é necessário identificar os serviços que usam o *broker* de mensageria, bem como as portas TCP usadas para conexão com o RabbitMQ. Dessa forma, o marco inicial é uma lógica de criação de um mapa de serviços e suas respectivas portas TCP que têm conexão com o RabbitMQ durante o uso da nuvem (a Figura 3 mostra alguns processos que tem conexão TCP estabecida com o serviço AMQP, do RabbitMQ).

```
1  COMMAND     PID       USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
2  ceilomete  6225 ceilometer    7u  IPv4  194228      0t0  TCP ctl:52264->ctl:amqp (ESTABLISHED)
3  /usr/bin/  6442    neutron    5u  IPv4  149342      0t0  TCP ctl:50828->ctl:amqp (ESTABLISHED)
4  /usr/bin/  6442    neutron    6u  IPv4  149343      0t0  TCP ctl:50830->ctl:amqp (ESTABLISHED)
5  /usr/bin/  6442    neutron    7u  IPv4  149344      0t0  TCP ctl:50832->ctl:amqp (ESTABLISHED)
6  glance-ap  7613     glance   10u  IPv4  287851      0t0  TCP ctl:56828->ctl:amqp (ESTABLISHED)
7  glance-ap  7613     glance   11u  IPv4  287460      0t0  TCP ctl:56872->ctl:amqp (ESTABLISHED)
8  glance-ap  7615     glance   10u  IPv4  302164      0t0  TCP ctl:54956->ctl:amqp (ESTABLISHED)
9  glance-ap  7615     glance   11u  IPv4  302165      0t0  TCP ctl:54958->ctl:amqp (ESTABLISHED)
```

Figura 3. Exemplo de saída padrão do comando *lsof* à porta TCP/5672 em um dos experimentos.

Portanto, implementa-se uma ferramenta utilitária[1] que interpreta a saída do *List of Files (lsof)*, no sentido de gerar um mapa de serviços e portas que têm conexão com o RabbitMQ no momento em que o tráfego da rede está sendo medido. O *lsof*, é um comando dos sistemas Unix que permite listar descritores de arquivos abertos pelos processos em execução no SO. Neste caso, um *socket* também pode ser interpretado como um arquivo do sistema. Assim, usa-se o parâmetro *-i* para identificar arquivos de rede acessados pela porta 5672, que é a porta usada pelo RabbitMQ. Então, tem-se *lsof -i :5672*. O comando retorna uma lista de todos os processos utilizando a porta 5672 (Figura 3).

6. Considerações

A análise e caracterização do tráfego de rede em ambientes controlados, nos quais as aplicações em execução são conhecidas, como é o caso da rede administrativa do OpenStack, pode ser feita de maneira simples e eficaz através da abordagem baseada em portas. O *broker* de mensageria, RabbitMQ, trouxe um aumento na complexidade da caracterização

[1] github.com/Adnei/service_identifier

do tráfego gerado por ele. Uma vez que são usadas portas efêmeras nas trocas de mensagens assíncronas entre os serviços. Por fim, uma solução encontrada foi usar informações do SO para identificar e mapear os serviços e suas conexões com o AMQP, tornando viável o emprego de uma abordagem de classificação de tráfego baseada em portas.

Agradecimentos: Os autores agradecem o apoio do LabP2D/UDESC e da FAPESC.

Referências

Alenezi, M., Almustafa, K., and Meerja, K. A. (2019). Cloud based sdn and nfv architectures for iot infrastructure. *Egyptian Informatics Journal*, 20(1):1 – 10.

Chaudhary, R., Aujla, G. S., Kumar, N., and Rodrigues, J. J. P. C. (2018). Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis. *IEEE Communications Magazine*, 56(2):118–126.

Chen, D. and Zhao, H. (2012). Data security and privacy protection issues in cloud computing. In *2012 International Conference on Computer Science and Electronics Engineering*, volume 1, pages 647–651.

Dainotti, A., Pescape, A., and Claffy, K. C. (2012). Issues and future directions in traffic classification. *IEEE Network*, 26(1):35–40.

Finsterbusch, M., Richter, C., Rocha, E., Muller, J., and Hanssgen, K. (2014). A survey of payload-based traffic classification approaches. *IEEE Communications Surveys Tutorials*, 16(2):1135–1156.

Lin, P., Li, Z., Lin, Y., Lai, Y., and Lin, F. C. (2006). Profiling and accelerating string matching algorithms in three network content security applications. *IEEE Communications Surveys Tutorials*, 8(2):24–37.

OpenStack (2019a). Openstack documentation.

OpenStack (2019b). What is openstack?

OpenStack (2020a). Networking architecture. https://docs.openstack.org/security-guide/networking/architecture.html. Accessed: 2020-09-28.

OpenStack (2020b). Openstack services. https://www.openstack.org/software/project-navigator/openstack-components#openstack-services. Accessed: 2020-09-28.

Piskac, P. and Novotny, J. (2011). Using of time characteristics in data flow for traffic classification. In *AIMS*.

Scharf, M., Stein, M., Voith, T., and Hilt, V. (2015). Network-aware instance scheduling in openstack. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6.