VIRTUALIZAÇÃO DE INSTÂNCIAS DE NÉVOA EM MÁQUINAS VIRTUAIS

Higor Sant'Anna[1]; Antonio Coutinho[2];

1. Bolsista PROBIC, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: higorsantanna@ecomp.uefs.br

2. Orientador, Departamento de Tecnologia, Universidade Estadual de Feira de Santana, e-mail: acoutinho@uefs.br

PALAVRAS-CHAVE: Virtualização; Emulação; Computação em Névoa.

INTRODUÇÃO

Um modelo baseado em nuvem tornou-se a abordagem padrão para as plataformas de internet de coisas (*Internet of Things,* IoT). Devido às limitações de mobilidade, localização e baixa latência da arquitetura centralizada em nuvem, as propostas atuais de IoT como Aazam et al. (2014), Satyanarayanan et al. (2009), Dinh et al. (2013) e Hu at al. (2015) estão fomentando uma mudança de paradigma importante em direção a um modelo descentralizado.

Em particular, como definido por Bonomi (2012), a computação em névoa (*fog computing*) é um paradigma que estende os recursos computacionais de forma integrada com elementos de rede como *switches*, roteadores e *gateways*. A abordagem distribuída de névoa reduz a quantidade de dados transferidos para a rede central, capturando e processando os dados necessários localmente em cada dispositivo de borda.

Até onde vai nosso conhecimento, atualmente não existem plataformas de computação em névoa prontamente disponíveis e que podem ajudar os pesquisadores a projetar e verificar algoritmos distribuídos em uma escala IoT real. O objetivo desse trabalho é a implementação de um *framework* que permita a inicialização, a configuração e o controle da execução de instâncias de máquinas virtuais de forma escalável em diferentes máquinas hospedeiras conectadas em rede local.

MATERIAL E MÉTODOS

No decorrer da pesquisa, três ferramentas se mostraram úteis para atingir os objetivos esperados. Foram elas: Docker (Docker, 2017), MaxiNet (Wette, 2014) e Fogbed (Coutinho, 2018). O Docker é uma plataforma de software que usa uma tecnologia baseada em *kernel* Linux chamada conteinerização (Docker, 2017). Um contêiner é um pacote independente que inclui o que é necessário para executar um aplicativo, como código executável, ambientes de tempo de execução, configurações, ferramentas e bibliotecas do sistema. As aplicações em execução dentro de um contêiner Docker utilizando virtualização no nível do sistema operacional são semelhantes às instâncias de Máquinas Virtuais (*Virtual Machines*, VM) tradicionais. No entanto, os contêineres não encapsulam um sistema operacional completo.

MaxiNet é um framework que estende o emulador de redes Mininet para oferecer um ambiente de rede virtual através de um cluster de máquinas virtuais (Lantz, 2010). Mininet é a ferramenta mais comum para emular Redes Definidas por Software (*Software Defined Networking*, SDN). De acordo com o autores, o MaxiNet torna possível a emulação de SDN com milhares de nós virtuais em dezenas de máquinas físicas em rede local (Wette, 2014).

Fogbed é um projeto de software que estende o framework Mininet para criar *testbeds* para computação em névoa em ambientes virtualizados. Usando uma abordagem desktop, o Fogbed permite a implementação de nós de névoa usando contêineres Docker sob diferentes configurações de rede. A interface de programação de aplicativos (*Application Programming Interface*, API) do Fogbed fornece funcionalidades para adicionar, conectar e remover contêineres dinamicamente da topologia da rede. Essas características permitem a emulação da infraestrutura de nuvem e névoa, na qual é possível iniciar e interromper instâncias de computação em qualquer momento durante sua execução. Além disso, é possível alterar as limitações de recursos em tempo de execução para um contêiner, como por exemplo o tempo da Unidade de Processamento Central (*Central Processing Unit*, CPU) e memória disponível.

O objetivo deste trabalho foi criar um ambiente distribuído escalável para emulação de instâncias de névoa remotamente. Neste sentido, o ambiente de emulação Fogbed foi estendido para permitir a execução de contêineres com imagens do Fogbed em máquinas host de baixo custo, com capacidades limitadas de memória e de processamento, e conectadas em rede local.

RESULTADOS E DISCUSSÃO

Com o intuito de tornar o ambiente de névoa emulado escalável, o mesmo foi particionado através de instâncias do Fogbed executadas em máquinas hospedeiras distribuídas. A API do MaxiNet é utilizada para criar um *cluster* escalável de instâncias do Fogbed para desenvolvimento e teste de componentes de névoa reais. Para esse fim, as instâncias padrão do Mininet foram substituídas por instâncias pré-configuradas do Fogbed. A arquitetura pode ser vista na Figura 1.

Antes de executar um experimento, cada host tem que inicializar um serviço *worker* do MaxiNet que espera por tudo necessário para executar uma emulação local, como *scripts* e configurações. Depois desse passo, a máquina hospedeira está pronta para ser remotamente alocada, configurada e conectada com outras instâncias do Fogbed.

A máquina controladora do emulador assume o papel de *front-end*. O processo do controlador, rodando na máquina controladora, usa a API do MaxiNet para inicializar e controlar a execução de diferentes instâncias do Fogbed distribuídas na rede local.

Dentro de cada instância do Fogbed, nós virtuais e suas infra-estruturas de rede são emuladas. A Figura 1 representa o fluxo de um desenvolvedor utilizando o ambiente. Primeiramente, o desenvolvedor fornece as imagens do contêiner que será utilizado para configurar a emulação (1). Cada contêiner inclui o recursos necessários para executar a aplicação. Usando a API de topologia, o desenvolvedor define um ambiente distribuído no qual ele deseja testar a aplicação (2) e inicia o processo controlador com a definição da topologia (3). O processo controlador do emulador usa a API do MaxiNet para criar as instância necessárias do Fogbed (4). Depois que o ambiente distribuído foi inicializado, o controlador invoca o sistema de gerenciamento em cada instância do Fogbed para fazer o upload dos contêineres e imagens necessárias (5). O sistema de gerenciamento local conecta seu ambiente emulado utilizando o API da instância fornecida. A aplicação é iniciada em cada instância do Fogbed pelo sistema de gerenciamento local, que inicia os processos e serviços necessários em cada nó virtual (6). Então a aplicação pode ser gerenciada da máquina controladora da emulação utilizando um CLI remota para acessar cada instância do Fogbed (7). As estatísticas sobre os fluxos de rede podem ser coletadas em cada instância do Fogbed e armazenadas na máquina que controla a emulação para análises futuras (8). Além disso, o desenvolvedor na máquina que controla têm acesso dados de monitoramento arbitrários gerados pela plataforma (9).

Dentro de uma instância do Fogbed, o monitoramento do tráfego da rede pode ser implementado executando monitores de fluxo para cada interface de rede em nós ou *switches* virtuais. O tempo de execução desses processos são salvos na pasta *VM-to-Host* que é espelhada com o mesmo nome na máquina *host*.

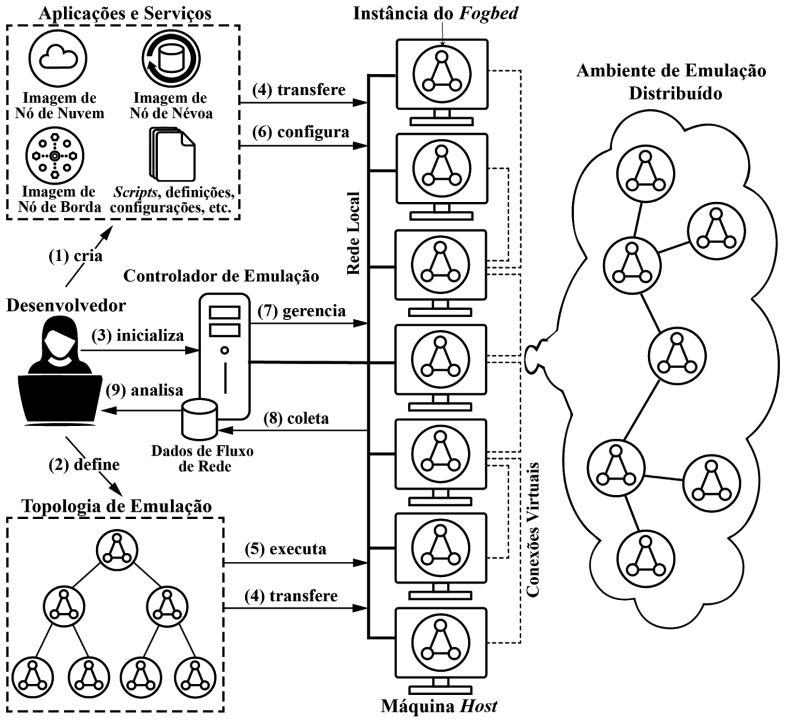

Figura 1: Fluxo de trabalho de uma emulação distribuída usando instâncias do Fogbed.

CONSIDERAÇÕES FINAIS

Neste trabalho, foi obtido êxito na criação de um sistema de emulação de névoa distribuído estendendo o *framework* Fogbed. Utilizando este *framework* de emulação distribuído, pesquisadores poderão emular sistemas reais baseados em névoa.

Ainda pretende-se dar seguimento ao trabalho de duas formas: criando estudos de caso para validar o *framework* desenvolvido e as aplicações; e usar a tecnologia Blockchain (Greve et al., 2018), que com sua arquitetura descentralizada e propriedades como imutabilidade e privacidade tornaria a comunicação entre os nós de névoa mais segura.

REFERÊNCIAS

M. AAZAM, I. KHAN, A. A. ALSAFFAR, e E.-N. HUH, "Cloud of things: Integrating internet of things and cloud computing and the issues involved," in Applied Sciences and Technology (IBCAST), 2014 11th International Bhurban Conference on. IEEE, 2014, pp. 414–419.

M. SATYANARAYANAN, P. BAHL, R. CACERES, e N. DAVIES, "The case for vm-based cloudlets in mobile computing," IEEE pervasive Computing, vol. 8, no. 4, 2009.

H. T. DINH, C. LEE, D. NIYATO, e P. WANG, "A survey of mobile cloud computing: architecture, applications, and approaches," Wireless communications and mobile computing, vol. 13, no. 18, pp. 1587– 1611, 2013.

Y. C. HU, M. PATEL, D. SABELLA, N. SPRECHER, e V. YOUNG, "Mobile edge computing a key technology towards 5g," ETSI White Paper, vol. 11, 2015.

F. BONOMI, R. MILITO, J. ZHU, e S. ADDEPALLI, "Fog computing and its role in the internet of things," in Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012, pp. 13–16.

DOCKER (2017). project home page. Dispon´ıvel em: https://www.docker.com. Acesso em: 20 set. 2017.

P. WETTE, M. DRAXLER, A. SCHWABE, F. WALLASCHEK, M. H. ZAHRAEE, e H. KARL, "Maxinet: Distributed emulation of software-defined networks," in Networking Conference, 2014 IFIP. IEEE, 2014, pp. 1–9.

B. LANTZ, B. HELLER, e N. MCKEOWN, "A network in a laptop: rapid prototyping for software-defined networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010, p. 19.

COUTINHO, A., GREVE, F., PRAZERES, C., e CARDOSO, J. (2018). Fogbed: A rapid prototyping emulation environment for fog computing. In Communications Workshops (ICC Workshops), 2018 IEEE International Conference on. IEEE. pp. 1–7.

GREVE, F., SAMPAIO, L., ABIJAUDE, J., COUTINHO, A., VALCY, I., e QUEIROZ, S., "Blockchain e a revolução do consenso sob demanda" in WBLOCKCHAIN, Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. SBC, 2018, pp. 209–260.