Análise do MaxiNet

Rúben Condesso [81969][1], João Correia [81990][1], and João Costa [82528][1]

Instituto Superior Técnico, Av. Prof. Dr. Cavaco Silva 13, Porto Salvo, Portugal
{ruben.condesso,jpbcorreia,joaocarlos95}@gmail.com

Resumo O objetivo deste projeto é analisar a escalabilidade e desempenho de um sistema real distribuído, no qual foi escolhido o Maxinet. Nesta primeira parte do projeto, analisamos o desempenho (latência, débito e CPU) do sistema Mininet variando a complexidade das topologias utilizadas.Conseguimos executar o Maxinet, mas no entanto não foi possível correr os testes para analisar a escalabilidade deste.

GitHub : https://github.com/joaocarlos95/EngineeringLargeScaleSystems-Project

Commit : 039e21b8713e3b10627135319ccb6de8b6bd498c

Keywords: MaxiNet · Mininet · Software Define Networks.

1 Introdução

Com o surgimento do Software Defined Networking (SDN), os administradores de rede ganharam a capacidade de monitorizar, personalizar e controlar, a computação na rede através de uma interface de programação. O controlador de SDN é a ponte entre as aplicações que controlam a rede e o hardware das mesmas.

Para testar novos algoritmos ou protocolos de uma SDN, são usados frequentemente emuladores de rede, como é o caso do sistema Mininet. Estes têm a vantagem, em relação aos sistemas reais, de criar cenários de testes sem grandes custos. O Mininet emula redes com componentes virtuais entre os quais hosts, switches, controladores e links, contudo não tem a capacidade de o fazer em grande escala (número máximo de máquinas num único OS Kernel foi de 4096) [**?**]. Para ultrapassar estas limitações, foi desenvolvido um sistema distribuído, o MaxiNet, capaz de distribuir a emulação por diversas máquinas físicas sem comprometer o trabalho do sistema.

No âmbito da cadeira de Sistemas de Larga Escala, decidimos analisar a escalabilidade e o desempenho do MaxiNet, uma vez que é um sistema que cumpre os requisitos impostos pelo docente, e principalmente, porque a ideia por detrás deste sistema vai ao encontro nos termas abordados na nossa área de especialização (Gestão das redes, da informação e dos serviços) .

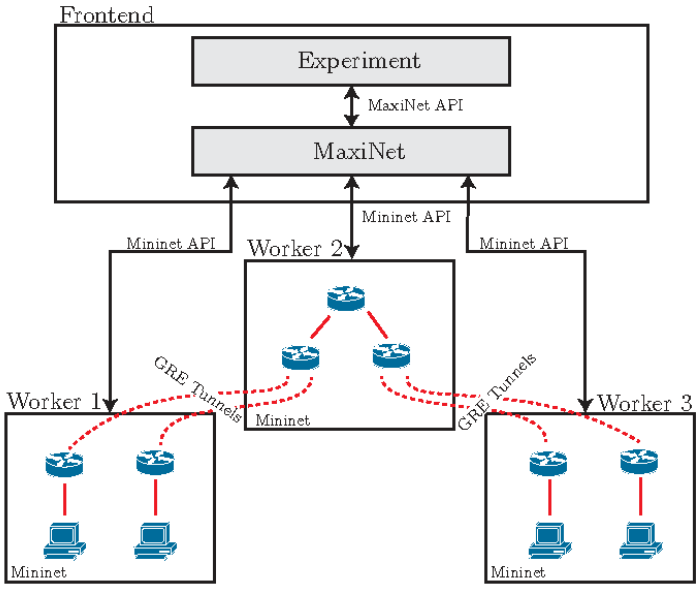

Figura 1. Arquitetura do Maxinet

2 Descrição do Sistema

O MaxiNet é uma camada de abstração que liga múltiplas instâncias de Mininet que correm em diferentes máquinas físicas, designadas por Workers. O sistema MaxiNet atua como um Frontend que liga as várias instâncias do Mininet, configura os diversos nós da rede e e os túneis necessários para que os Workers possam comunicar.

De forma a particionar a rede pelos diversos Workers, o MaxiNet usa uma biblioteca METIS que tem como objetivo computar n partições de igual peso. A nível de configuração, o MaxiNet começa por criar uma lista com os endereços IP's de cada Worker, onde cada qual vai correr uma instância do Mininet. De seguida, é associado a este cluster, uma topologia de rede, criando deste modo a interface à API do MaxiNet. Por fim, é adicionado um controlador encarregue de manipular e gerir toda a rede emulada. Com a emulação configurada, será possível invocar diferentes comandos, nos nós da topologia criada, durante a execução da emulação.

Há que ter em conta as limitações existentes no MaxiNet. O sistema não tem noção da performance de cada Worker assim como também não tem noção da rede física onde está a ser executado, o que remete o seu uso a sistemas heterogéneos. A carga do sistema é distribuída uniformemente por todos os Workers, logo facilmente, podemos concluir que máquinas mais fracas se tornarão no bottleneck da emulação.

Indo agora mais concretamente ao encontro do nosso trabalho e da disciplina em causa, utilizámos o Containernet para executar as várias componentes do sistema MaxiNet. O Containernet permite utilizar docker containers como Mininet hosts, o que permite criar testes de benchmark interessantes. Entre as principais características do Containernet, estão a possibilidade de adicionar, remover e associar containers às topologias Mininet, e executar comandos dentro dos containers usando o Mininet CLI. Além do Containernet, utilizamos o pox para lançar o controller utilizado na gestão das topologias criadas.

3 Resultados

Nesta secção, irão ser apresentados e discutidos os resultados dos testes de performance e de escalabilidade impletamentados. Começámos por testar o sistema Mininet individualmente, correndo num único container (Worker) ligado ao FrontEndServer. Isto deveu-se ao facto de termos tido dificuldades, primeiro na instalação, e posteriormente na execução e no teste de desempenho do MaxiNet, dificuldades essas que serão expostas na conclusão. Referente ao resultados obtidos do Maxinet, iremos expôr o que conseguimos fazer corretamente, usando este sistema, no que toca à conectividade, testes bem sucedidos, e finalmente, iremos referir os parâmetros que queríamos ter abordado, mas que não conseguimos devido aos obstáculos encontrados.

Mininet

Como foi referido no parágrafo anterior, executamos o mininet num único container (Worker), e testámos o comportamento deste sistema com a variação de 3 parâmetros: latência, débito e uso de CPU.

Relativamente à latência e ao débito obtido, fizemos dois tipos de testes: no primeiro teste verificamos como varia estes parâmetros com o aumento de hosts na rede, usando topologias com apenas 1 hop, e comparamos com o valor teórico; e no segundo teste verficar como variam estes parâmetros com um maior número de hops (de 1 para 2), e com o aumento da complexidade da topologia, e comparámos igualmente com o valor teórico.

Para testar a latência utilizámos o ping, criámos uma topologia onde cada link tinha um certo delay, logo o valor teórico é a soma dos delays de cada link por onde o ping passa. Comparámos esse valor com a média obtida das simulações efetuadas. Para o teste do débito, usámos como ferramente de teste o iperf, e usámos a mesma topologia utilizada do teste da latênica, mas no invés de limitarmos o delay, limitámos a bandwidth, que por sua vez, seria sempre o valor teórico. Os resultados obtidos estão ilustrados na figura 2:

Topologia	ping(ms)	Valor teórico(ms)	Bandwidth Max (Mbits/sec)	Bandwidth Max teórico (Mbits/sec)
1Sw-2PCs	43,115	40	10,1	10
1Sw-60Pcs	43,254	40	9,85	10
1Sw-120PCs	43,591	40	9,77	10
1Sw-200PCs	46,411	40	9,63	10
Tree(depth=2,fanout=2)	86,455	80	10,1	10
Tree(depth=2,fanout=6)	86,768	80	9,85	10
Tree(depth=2,fanout=10)	86,896	80	9,79	10
Tree(depth=2,fanout=20)	87,118	80	9,71	10

Figura 2. Resultados experimentais da latência e do débito

Com os resultados presentes na tabela 1 acima, podemos concluir que à medida que a complexidade da topologia aumenta, a latência do ping aumenta e o débito diminui. Nas simulações efetuadas, verificámos que este aumento de latência e diminuição do débito ocorre de forma constante com o aumento da complexidade da rede. Para redes muito complexas, não conseguimos obter quaisquer resultados válidos, pois como neste caso só temos 1 worker, não há uma divisão de carga de trabalho por mais supostos workers, o que acaba por acontecer é que o PC deixa de ter capacidade para esse aumento de complexidade. Logo, não há um ponto de estagnação do aumento e diminuição dos valores em causa.

No caso do teste da limitação do uso de CPU, fizemos apenas um tipo de teste, usando sempre a mesma topologia, onde fomos limitando gradualmente a percentagem de CPU a ser utilizada em cada simulação (usando o iperf). Os resultados obtidos estão ilustrados na figura 3:

CPU	Bandwidth Max(Gbits/sec)
100%	9,84
50%	8,15
25%	4,16
10%	1,67
5%	0,822
1%	0,151

Figura 3. Resultados experimentais do debito e relação ao CPU utilizado

Com os resultados presentes na tabela 2, podemos concluir que dos 100% aos 50 % de utilização de CPU, a diminuição do débito é escassa, o que nos leva a concluir que apenas é necessário apenas 50% de CPU para a execução do teste em causa. A partir dos 50% de utilização, verificámos uma queda abrupta dos valores de débito, onde podemos concluir que o PC vai tendo cada vez mais capacidade para executar o teste.

MaxiNet

Para a utilização do sistema MaxiNet, criámos o script StartUp.sh que nos permite uma instalação automática do sistema e das respetivas dependências. Para execução do MaxiNet, corremos o scipt StartUp.sh, que irá lançar em forma de containers (usando o containernet) o FrontEndServer, o controller e os respetivos Workers, onde todos se encontram na mesma rede local. Podemos verificar que existe uma associação de cada Worker ao FrontEndServer (verificamos pelo comando MaxiNetStatus), e existe total conectividade entre todos os intervenientes da rede. O número de Workers é definido durante a execução do script.

Dentro do container referente ao FrontEndServer, poderão ser corridos os testes simplePing.py e dockerSimpleping.py, que conseguimos correr com sucesso garantem uma correta construção da rede. Na tentativa de testar a escalabilidade do sistema, executámos o simplePing.py com vários workers, no entanto os resultados obtidos foram sempre muito semelhantes, quer com 1 Worker quem com N Workers, o que nos leva a concluir que o sistema não estava a fazer uma divisão da carga de trabalho pelos vários Workers. Criámos diferentes topologias mas os resultados foram idênticos, e desta forma, não fomos capazes de testar a escalabilidade do MaxiNet, para esta entrega.

4 Conclusão

Relativamente aos resultados encontrados no teste do sistema Mininet, foram ao encontro das nossas expetativas e do que aprendemos nas aulas da disciplina em causa. Houve uma expectável relação entre o aumento da topologia criada, com o aumento da latência nos pings feitos, e a diminuição do débito imposto entre dois hosts da rede.

A instalação e execução do MaxiNet foi uma séria dificuldade neste projeto, principalmente por ser um sistema que não tem tido atualizações nos últimos anos, a única versão do sistema operativo em foi testado foi Ubuntu 14.04, e não existe muita informação de como testar este sistema da melhor maneira. Começamos por utilizar o nosso sistema operativo preferencial, Ubuntu 18.04, onde tivemos alguns problemas na instalação, que posteriormente foram ultrapassados. Conseguimos estabelecer uma relação de conectividade entre os N Workers e o FrontEndServer, mas não conseguimos corre com sucesso o teste do simplePing.py. A única solução encontrada foi mudar de versão do sistema operativo, onde passado mais dificuldades de instalação, conseguimos correr com sucesso o teste do simplePing.py. Estas dificuldades encontradas foram de facto um entrave no nosso projeto. Apesar de termos conseguido correr corretamente o sistema e de estabelecer as respetivas relações entre todos os intervenientes do sistema MaxiNet, não fomos capazes de testar a sua escalabilidade, sendo esse um dos objetivos do trabalho.

Referências

1. Philip Wette, Martin Dr¨axler, Arne Schwabe, Felix Wallaschek, Mohammad Hassan Zahraee, Holger Karl: MaxiNet: Distributed Emulation of Software-Defined Networks**,** (2014)
2. MaxiNet, http://github.com/MaxiNet/MaxiNet.
3. Containernet, http://github.com/containernet/containernet.
4. Mininet, http://mininet.org/overview/.