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 Este projeto tem o intuito de analisar a escalabilidade e o desempenho do MaxiNet, um sistema real distribuído, que emula uma *Software Defined Network* (SDN) usando o Mininet. Na primeira parte foi analisado o desempenho (latência, débito e CPU) do sistema Mininet, variando a complexidade das topologias. Na segunda parte do projeto foi testado o comportamento do sistema MaxiNet face às variações do número de *workers*, nível de complexidade da topologia e percentagem de CPU utilizada pelos *containers*.

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

Commit : 42b20ab6875055f60ac8bccb293794bb8c854c5b

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, localizada no **Controlador**, sendo que este representa 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 é de 4096) [2]. 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. Na figura 1, encontra-se ilustrada a arquitetura do MaxiNet.

No âmbito da cadeira de Sistemas de Larga Escala, analisámos 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 dos temas abordados na nossa área de especialização (Gestão das redes, da informação e dos serviços).

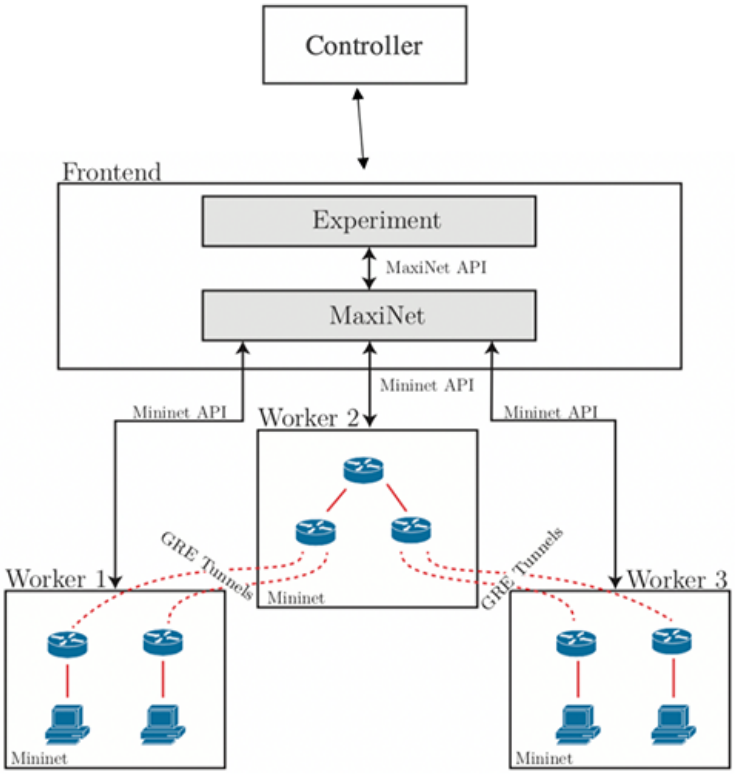

Figura 1. Arquitetura do MaxiNet

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**. Este sistema 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. Além destes componentes, atua um **Controlador** *pox* que tem como finalidade dar resposta aos *switches* referente às solicitações feitas pelos mesmos, e assim permitir que estes possam criar as tabelas de *flows*. Tendo esta função, sempre que houver um pedido na rede, o controlador terá de dar resposta, pelo menos uma vez, o que nos levou a suspeitar que este poderá ser um possível *bottleneck* do MaxiNet. Tal, será aprofundado mais à frente no relatório.

De forma a particionar a rede pelos diversos Workers, o MaxiNet usa a biblioteca **METIS** que tem como objetivo computar n partições de pesos aproximadamente iguais. De forma a otimizar este particionamento, é tido em consideração o peso de cada nó e os limites da largura de banda, especificados na topologia criada.

A nível de configuração, o MaxiNet começa por criar uma lista com os endereços IPs 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, podemos concluir que máquinas mais fracas se tornarão no *bottleneck* da emulação. Outras limitações e possíveis *bottlenecks* foram encontrados aquando a execução dos testes de performance, que serão explorados mais à frente neste trabalho.

Indo agora mais concretamente ao encontro do nosso trabalho e da disciplina em causa, foi utilizado o **Containernet** para executar as várias componentes do sistema MaxiNet. Este permite utilizar *docker containers* como Mininet *hosts*, o que permite a criação de testes *benchmark* interessantes. Entre as principais características do *Containernet*, estão a possibilidade de adicionar, remover e associar *containers* (o que não é possível no Mininet) às topologias Mininet. Essa característica permite usar o *Containernet* como uma infraestrutura de *Cloud* onde podemos começar e parar instâncias computacionais, a qualquer altura. Além disso, é permitida alteração da limitação dos recursos usados, como por exemplo, a utilização de CPU de cada *Worker*.

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 *script* referido, que irá lançar sob forma de *containers* (usando o Containernet) o FrontendServer, o Controlador e os respetivos *Workers*, que se encontram na mesma rede local. Podemos verificar que existe uma associação de cada *Worker* ao FrontendServer (verificado pelo comando MaxiNetStatus), e que existe total conectividade entre todos os intervenientes da rede. De notar que o número de *Workers* é definido durante a execução do *script*.

3 Testes

Nesta secção irão ser apresentados os testes de performance e de escalabilidade implementados. Começámos por testar o sistema Mininet individualmente, correndo num único container (Worker) ligado ao FrontendServer. Os testes executados encontram-se apresentados em baixo:

- Variação do débito conforme o aumento da complexidade da rede, onde foi aumentado o número de *hosts* e *switches*, bem como o número de *hops* efetuados
- Variação do débito com a limitação do CPU
- Variação da latência conforme o aumento da complexidade da rede, onde foi aumentado o número de *hosts* e *switches*, bem como o número de *hops* efetuados

Relativamente ao primeiro teste, começámos por testar topologias simples (*hosts* ligados através de um *switch*, onde fomos aumentado o número de *hosts* ligados a cada *switch*. De seguida, aumentámos o número de *hops* utilizando uma topologia em árvore. Para este teste, utilizámos a ferramenta *iperf*, e limitámos a banda utilizada a um valor específico, de forma a que esta funcionasse como um valor teórico. No segundo teste, utilizámos sempre a mesma topologia, e fomos variando a utilização de CPU. De notar que voltámos a usar o *iperf*. O último teste feito ao sistema Mininet, foi idêntico ao primeiro, sendo que foi utilizada a ferramenta *ping* (em vez do *iperf*), e a limitação foi assente na latência (em vez da largura de banda). A discussão dos resultados obtidos serão explorados na secção seguinte.

De seguida efetuámos testes referentes ao MaxiNet, de maneira a averiguar o comportamento do sistema face às diferentes configurações que uma emulação poderia tomar. As métricas consideradas foram a latência entre *hosts*, a latência entre o *host* e o Controlador (RTT), por este ser um ponto fulcral em toda a implementação do MaxiNet, foi também considerado a utilização de memória do sistema, o débito binário calculado, e as diferenças encontradas entre a utilização de UDP e TCP. Tais métricas permitiram avaliar os seguintes fatores:

1. Análise da latência e da memória utilizada pelo sistema de acordo com:
   (a) complexidade da topologia de rede
   (b) número de *Workers*
   (c) limitação de CPU utilizado em cada *Workers*
2. Análise ao RTT do Controlador de acordo com:
   (a) complexidade da topologia de rede
   (b) número de *Workers*
   (c) a limitação de CPU utilizado em cada Workers
3. Análise da Capacidade de resposta do Controlador, *Workers* e *Frontend*, face à a limitação de CPU
4. Análise do débito binário, limitando a largura de banda dos *links* da topologia, de acordo com:
   (a) complexidade da topologia de rede
   (b) número de *Workers*
   (c) limitação de CPU utilizado em cada *Workers*
5. Análise do débito, face à utilização dos protocolos UDP ou TPC , de acordo com:
   (a) complexidade da topologia de rede

(b) número de *Workers*
(c) limitação de CPU utilizado em cada *Workers*

6. Análise da latência, limitando o *delay* dos *links* da topologia, de acordo com:
   (a) complexidade da topologia de rede
   (b) número de *Workers*
   (c) limitação de CPU utilizado em cada *Workers*

4 Discussão de Resultados

Nesta secção, serão apresentados e discutidos os resultados obtidos, referentes aos testes indicados na secção anterior. Dividimos a análise de resultados em duas partes: uma parte que diz respeito ao sistema Mininet, e a outra parte referente ao MaxiNet.

Mininet

Como foi referido no anteriormente, executamos o Mininet num único *container* (*Worker*), e testámos o comportamento deste sistema com a variação de 3 métricas: latência, débito e uso de CPU.

Os resultados obtidos dos 3 testes referidos na secção 3, 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 ilustrados na figura 2, 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 estes resultados 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 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 menos capacidade para executar o teste.

MaxiNet

Referente à avaliação de performance do sistema MaxiNet, e tendo em conta os testes referidos na secção 3, passaremos à ilustração dos resultados obtidos. Após uma exploração profunda do sistema e das suas limitações, conseguimos entender o seu funcionamento, e logo o seu pipeline. Este encontra-se ilustrado na figura 4:

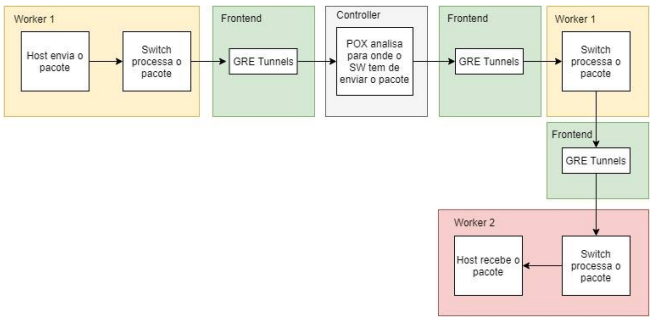

Figura 4. Pipeline com 2 *Workers*

Está especificado o caminho que um pacote leva, desde que sai do *host* de origem, referente ao *worker* 1, até que chega ao *host* de destino, no *worker* 2. No caso de haver um *Worker* apenas, o papel do Frontend deixa de ter efeito, sendo que deixa de ser necessário. Neste caso, é equivalente a ter um sistema Minenet. Fazendo um teste de *ping*, numa *Tree 2,3*, no MaxiNet, com 1 *worker*, e fazendo o mesmo teste no Mininet, obtemos resultados muito semelhantes: 0.378 ms contra 2.0256 ms. De notar que o valor do MaxiNet revela ser um pouco mais alto, mas temos de ter em consideração que 30 em 30 segundos, o *switch* tem de comunicar com o Controlador, se tirássemos isso, os valores seriam praticamente iguais.

Para analisar o comportamento da latência com o aumento do número dos w*workers*, fixamos uma topologia, à qual foram incrementando o número de *workers*, como mostram as figuras 5, 6, 7 e 8.

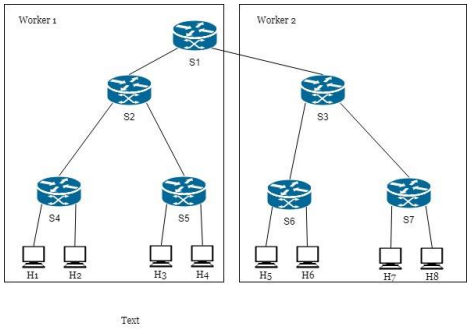

Figura 5. *Tree23* com 2 *Workers*

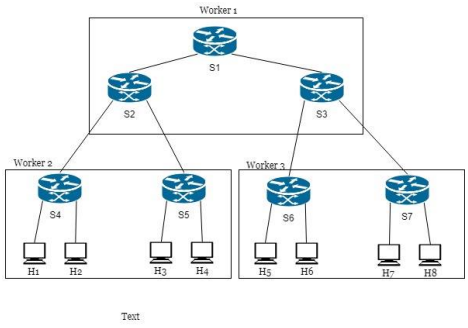

Figura 6. *Tree23* com 3 *Workers*

Os resultados obtidos para este teste encontram-se ilustrados na figura 9:

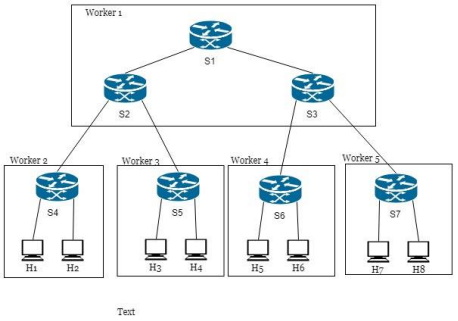

Figura 7. *Tree23* com 5 *Workers*

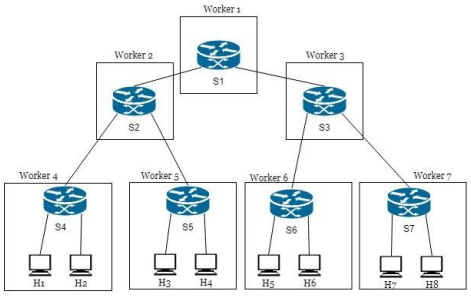

Figura 8. *Tree23* com 7 *Workers*

Podemos concluir que, para a topologia usada, o número de *workers* compensa até 3, mas a partir desse número deixámos de ter resultados favoráveis, em relação à latência calculada. Isto significa que a partir desse número, o peso da emulação e gestão de cada *worker*, não compensa que haja uma divisão dos mesmos pela topologia em causa, sendo que não é complexa o suficiente para ter um nível tão elevando de *workers*.

No seguimento do teste anterior, desta vez fixámos uma topologia mais complexa, e fomos aumento o número de *workers*. Fomos verificando uma melhoria de resultados na latência, como se pode ver na figura 10, o que pode ser explicado pelo elevado nível de complexidade da topologia em causa. Logo, foi feita uma boa divisão de trabalho por cada *worker* adicionado à rede.

O teste seguinte passou por analisar a latência, com a variação do CPU no Frontend, Controlador e *Workers*, cujos resultados obtidos estão ilustrados na figura 11. Podemos verificar que até a uma limitação de 20%, nenhuma das 3 componentes apresenta quaisquer variações consideráveis, no entanto, a partir deste valor a situação inverte-se. Continuando a variar a limitação do CPU, concluímos que o Frontend continua a apresentar resultados estáveis, em termos de latência, ao contrário do Controller e *workers*, cujos valores subiram bastante

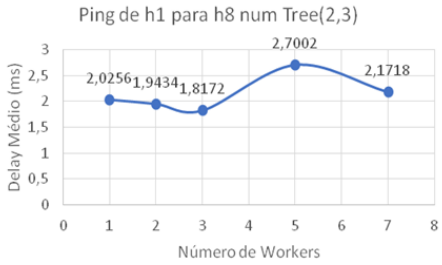

Figura 9. Variação do *Delay* médio em função do número *Workers*

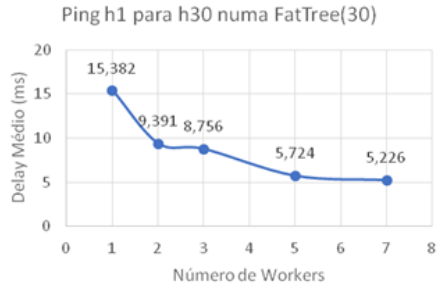

Figura 10. Variação do *Delay* médio em função no número de *Workers*

a partir do 10%. Tais resultados, levam-nos a suspeitar (mais uma vez) do facto do controlador e os *workers* puderem ser um *bottleneck* do sistema, ao contrário do Frontend, como ficou demonstrado.

Na figura 12, podemos verificar como as latência e a memória usada pelo sistema, variaram com o aumento do complexidade da topologia.

Como era esperado, todos os fatores existentes na figura, vão aumentando (ou seja piorando) proporcionalmente ao crescimento da complexidade da topologia. Estes resultados vão ao encontro do que já tinha sido concluído em relação ao sistema Mininet anteriormente.

Relativamente à métricas 5 e 6 referidas na secção 3, não conseguimos obter resultados válidos devido às limitações existentes no MaxiNet. Para a métrica 5, tentámos avaliar o comportamento do sistema face à limitação da largura de banda, com o aumento dos *workers*, complexidade da topologia, e variação do CPU utilizado, usando o *iperf*. Por mais que variássemos a largura de banda de cada *link*, com mais ou menos *workers*, e com uma topologia muito ou pouco complexa, os valores devolvidos pelo *iperf* eram sempre os mesmos. Acabámos por não perceber a origem deste problema, e assim, não concluímos o teste ao sistema com esta métrica.

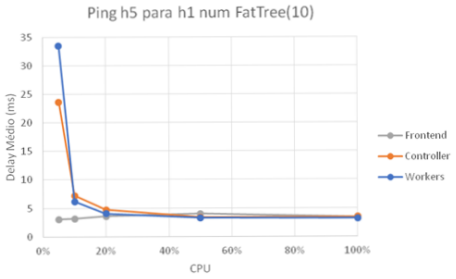

Figura 11. Variação da latência em função do CPU, com 2 *workers*

Complexidade	RTT min (ms)	RTT medio (ms)	Memoria (MB)	RTT Controller (ms)
FatTree5	0,284	3,236	0,049	59,2
FatTree10	0,396	4,127	0,05	67,1
FatTree20	0,313	5,297	0,052	115,23
FatTree40	0,33	12,339	0,055	290,5
FatTree80	0,321	45,93	0,061	710,5
FatTree120	0,356	106,38	0,066	1748,5

Figura 12. Variação da latência e memória de acordo com o aumento da complexidade

Na métrica 6, tentámos descortinar as diferenças entre o uso do UDP ou TCP, no entanto, o MaxiNet não suporta *iperf* com TCP e o *ping* pertence à camada de rede (e não à de transporte). Uma vez mais, as limitações existentes do sistema, não nos permitiram executar todos os testes que achámos adequados ao nosso contexto.

5 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.

Ao longo desta segunda parte do projeto, encontraram-se sempre presentes diversas dificuldade em executar o sistema MaxiNet, tal como já se tinha sucedido na primeira entrega. Se nessa fase, tivemos imensos problemas em instalar e correr o sistema, na segunda fase, deparámo-nos com vários barreiras referentes aos testes que idealizámos fazer ao início. As barreiras mais relevantes foram levantadas no fim da secção 4, contudo, como não conseguimos efetuar todos os testes, não conseguimos obter resultados mais objetivos e precisos, relativamente ao comportamento do sistema.

Em relação aos resultados que conseguimos, efetivamente, obter com sucesso, podemos concluir alguma correlação com resultados obtidos no Mininet. Em ambos os sistemas, verificámos uma degradação de resultados em termos de latência, com o aumento da complexidade da topologia em causa. Mais concretamente no MaxiNet, a memória usada, foi sempre aumentando de forma gradual, neste caso concreto. O aumento do número de *Workers* também influenciou o valor da latência obtida, concluindo-se que é preciso haver um bom balanceamento entre a complexidade da topologia e o número de *Workers*. Ou seja, com o auxílio dos resultados obtidos para estas fatores, verificámos que caso a topologia fosse pouco complexa e o houvessem muitos *Workers*, os resultados se degradariam. No fundo, o número de *Workers* existentes na topologia teria se compensar o nível de complexidade da mesma. As figuras 9 e 10, espelham perfeitamente esta conclusão. Nessas figuras, também avaliámos a capacidade de partição do METIS. Na figura 9, quando fomos nós a dividir os *workers* pela (nossa) topologia, os resultados obtidos foram piores e mais inconstantes, na figura 10, foi o METIS a ter essa responsabilidade, os resultados apresentaram valores mais positivos e constantes.

Em termos de limitações encontradas aquando da execução dos testes, verificámos que cada *switch* não conseguia ligar-se a mais do que 8 *hosts*. Esta limitação poderia ser ultrapassada caso fosse possível dividir um *switch* por mais do que um *Worker*, sendo esta medida também outra limitação do sistema. Sem a possibilidade de divisão de um *switch* por mais do que um *worker*, estes podem-se tornar, facilmente, num *bottleneck* do sistema.

A última limitação encontrada no sistema, prendeu-se com o facto de apenas termos conseguido utilizar um tipo de *switch* nas topologias criadas. Tentámos usar o *default switch* do MaxiNet, mas as topologias não eram criadas de maneira nenhuma. Apenas conseguimos criar as topologias, usando um *switch* OVS, do tipo *learning*.

No que diz respeito ao degradamento do sistema, concluímos que o Controlador e os *Workers* facilmente se podem tornar num *bottleneck*, caso a percentagem de utilização de CPU se aproxime do limite. Já o FrontendServer comportou-se bem face às limitações do CPU. Este facto pode ser justificado pelo seu papel, ou seja, não desempenha um papel fulcral em *runtime*, uma vez que a sua função é essencial apenas no lançamento da topologia.

Uma maneira de contornar a má *performance* do sistema face à limitação do CPU do Controlador, seria dividi-lo por diversas instâncias, separando deste modo o poder computacional como também fornecendo respostas aos *switches* de uma maneira mais eficiente para topologias complexas (última coluna da figura 12).

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/.
5. SDN controller, https://searchnetworking.techtarget.com/definition/SDN-controller-software-defined-networking-controller