AVALIAÇÃO DE DESEMPENHO DE REDES SDN UTILIZANDO METODOLOGIA EXPERIMENTAL

Erick Barros Nascimento
Programa de Pós-Graduação em Ciência da Computação (PROCC)
Universidade Federal de Sergipe (UFS), Aracaju – SE, Brasil.
erick.nascimento@fasete.edu.br

Aricio Medeiros
Discente do curso de Bacharelado em Sistemas de informação
Faculdade Sete de Setembro (FASETE) – Paulo Afonso – BA, Brasil.

Methanias Colaço Júnior
Programa de Pós-Graduação em Ciência da Computação (PROCC)
Universidade Federal de Sergipe (UFS), Aracaju – SE, Brasil.

Douglas Dyllon Jeronimo de Macedo
Departamento de Ciência da Informação (CIN)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

RESUMO

O termo *SDN* surge como líder da próxima geração de redes. A vazão e a latência são requisitos mínimos para garantir o bom funcionamento de sistemas interconectados. Algumas ferramentas utilizadas na geração de tráfego, foram idealizadas dos sistemas convencionais de redes para o modelo SDN, permitindo a medição de redes programáveis. Este artigo apresenta um modelo para avaliar o desempenho das redes SDN usando o recurso de geração de tráfego. A avaliação experimental dos ensaios foi realizada por meio de um experimento controlado com simuladores. Os resultados mostram que esta abordagem pode ser utilizada para avaliar o impacto sobre o ambiente de rede, quando o mesmo é escalonado com a adição de novos *hosts* geradores de tráfego.

Palavras-chave: Redes definidas por *software*, redes programáveis, redes.

ABSTRACT

The word SDN emerges as leader of the next network generation. The flow and latency are minimum requirements to ensure a good work of interconnected systems. Some tools used in the generation of traffic were idealized from the conventional network systems to the SDN model, allowing the measure of programmable networks. This paper presents a model that evaluates the performance of SDN networks using a resource of traffic generation. The experimental evaluation of the tests was conducted through a controlled

experiment with simulators. The results show that this approach may be used to evaluate the impact over the network environment when it is scaled with the addition of new traffic generator hosts.

Keywords: Software Networks, Programmable Networks, Networks.

1 INTRODUÇÃO

O controle distribuído e os protocolos que são responsáveis pelo encaminhamento de pacotes na rede, são embarcados em equipamentos de arquitetura proprietária. O fato é que, independentemente da verticalização imposta pelos fabricantes desses equipamentos, os problemas das redes de computadores ainda são os mesmos desde o início dos sistemas computacionais interconectados.

A chegada do paradigma *Software Defined Network (SDN)*, impulsionou à arquitetura convencional para uma mudança, devido a flexibilização da configuração complexa dos ambientes de rede atuais, mitigando à alta expertise exigida dos administradores por conta dos diferentes fabricantes (KREUTZ et al. 2015). Redes programáveis estão sendo vistas como a tecnologia mais promissora para o futuro das redes e da própria internet (ROWSHANRAD et al. 2014).

Foi realizado uma revisão sistemática, sobre as características que compõem as redes definidas por software, com ênfase em medição e testes de desempenho. A partir desse estudo, uma rede foi projetada em diferentes cenários para inferir estatisticamente os dados capturados.

O restante do trabalho foi estruturado de acordo como segue. A Seção 2, apresenta uma breve descrição sobre o que é uma rede SDN e sua composição, a definição dos metadados que farão parte da experimentação, e um exemplo de simulador de redes com suporte a redes programáveis. Na Seção 3, encontram-se a definição e o planejamento do experimento. Na Seção 4, é apresentado a operação do experimento. Na seção 5, contém os resultados do experimento. Na seção 6, são apresentados os trabalhos relacionados. Por fim, a seção 7 apresenta a conclusão e os trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 Redes Definidas Por Software (SDN)

O termo *Software Defined Network (SDN)*, foi originalmente concebido com a finalidade de representar o trabalho do desenvolvimento em torno do protocolo *OpenFlow* na Universidade de Stanford (SHENKER et al., 2011). Refere-se a uma arquitetura de rede que desacopla o controle dos nós da rede de cada equipamento, e entrega o gerenciamento para um controlador remoto denominado *Controller*. De acordo com KREUTZ et al. (2015), a arquitetura SDN pode ser contextualizada em quatro abordagens:

1. O controle e os *data planes (switches)* são desacoplados, isto significa que seu gerenciamento, não se aplica por concentrador separadamente, e sim, por fluxos definidos no controlador.
2. Os enlaces são chamados de fluxos. Um fluxo no contexto *SDN*, é uma sequência de pacotes entre uma fonte e um destino, para controlar os dispositivos encaminhadores de pacotes (NEWMAN et al. 1998), (GUDE et al. 2008).
3. O controle lógico é separado, atuando de forma externa. É análogo a um sistema operacional tradicional, fornecendo os recursos essenciais para facilitar a programabilidade dos recursos da rede (KREUTZ et al. 2015).
4. A rede de computadores é programável através de uma linguagem de programação de alto nível, por exemplo, C/C++ e *Python*. Essas aplicações são implantadas na camada superior, e o sistema operacional da rede, do inglês *Network Operation System (NOS)* é responsável por fazer a ligação entre as aplicações e os dispositivos que estão na parte inferior das camadas do sistema.

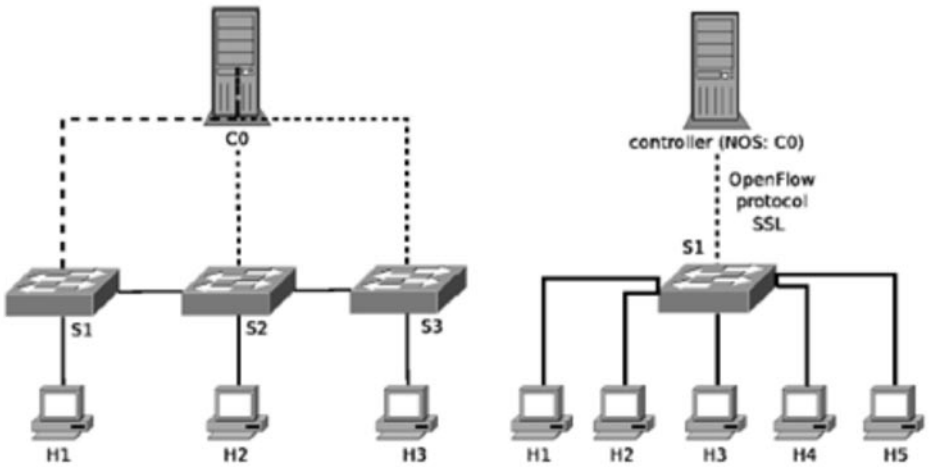
2.2 Definição de Metadados

Foi realizado um levantamento sobre os metadados necessários para concepção da semântica necessária na criação de procedimentos de geração de tráfego numa rede *SDN*, ou seja, quais os parâmetros que descrevem esse tipo de procedimento que podem permitir o escalonamento da rede. A identificação desses metadados foi realizada por meio de revisão sistemática e, principalmente, analisando outros procedimentos já realizados por outros pesquisadores das áreas de Sistemas Distribuídos. Após a identificação dos metadados – descritos nas seções 2.1.1 e 2.1.2, foram contextualizados os modelos conceitual e lógico de uma rede *SDN*.

2.2.1 Modelo Conceitual (Tipo 1)

Além do controlador central, entre o controlador e o *switch*, haverá uma conexão segura protegendo o controlador, permitindo acesso através de uma ferramenta no espaço do usuário. Sendo assim, de acordo com o contexto mínimo, os metadados fundamentais são: o controlador, os concentradores *(switches)*, os *hosts* e o *link* entre o controlador e o *switch*. O modelo conceitual descrito na Figura 2 possibilita à inferência, já na Figura 1, as topologias que serão analisadas são dispostas compondo o quadro desejável para concepção dos metadados.

Figura 1: Topologia Linear e *Single*. Arquitetura mínima do sistema

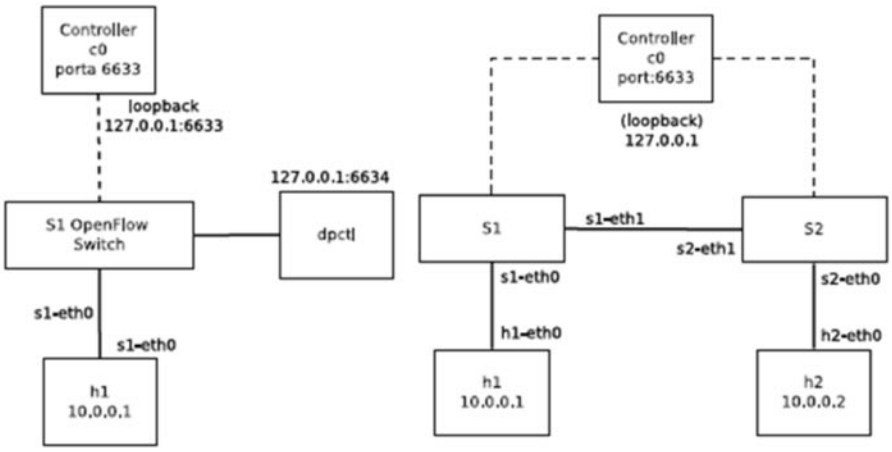


Fonte: Júnior; Nascimento; Silva, 2017.

2.2.2 Modelo Lógico (Tipo 2)

Diante da composição mínima descrita no Tipo 1, existe potencialidade para inferência dessa topologia comparando-a com outros tipos já utilizados como arquitetura conceitual. Os valores da rede poderão sofrer alteração à medida que a quantidade de hosts aumenta, elevando o tráfego por conta da requisição de serviços. Os metadados que contemplam o Tipo 2 – são além dos já identificados no Tipo 1 – os seguintes: vazão, latência e atraso do ambiente de rede.

Figura 2: Esquema conceitual dos metadados identificados para procedimento em arquitetura mínima (*Single-esq*) e (Linear-dir) definidas na seção 2.1.1

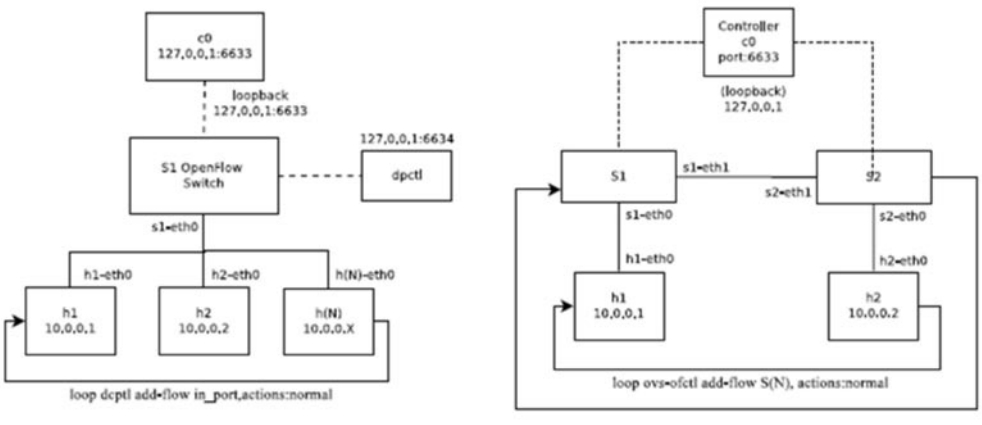


Fonte: Júnior; Nascimento; Silva, 2017.

2.2.3. Modelo Lógico Experimental (Tipo 3)

A inferência do modelo conceitual descrito nesta seção, aponta para a necessidade de *Benchmarking* de redes definidas por *software*. Como o *Benchmarking* existente é uma combinação de tecnologias de TI existentes, as métricas de avaliação de desempenho ainda não puderam ser consolidados na sua totalidade nas redes SDN (KIM et al. 2015). Portanto, para obter os metadados para concepção do Tipo 3, os modelos conceituais do experimento foram projetados para avaliação de desempenho do ambiente. Esses modelos foram dispostos como especificação do Tipo 3 e descritos na figura 3.

Figura 3: Esquema conceitual dos metadados identificados para inferência de arquitetura estruturada (*Single-esq*) e (Linear-dir) definidas na seção 2.1.2

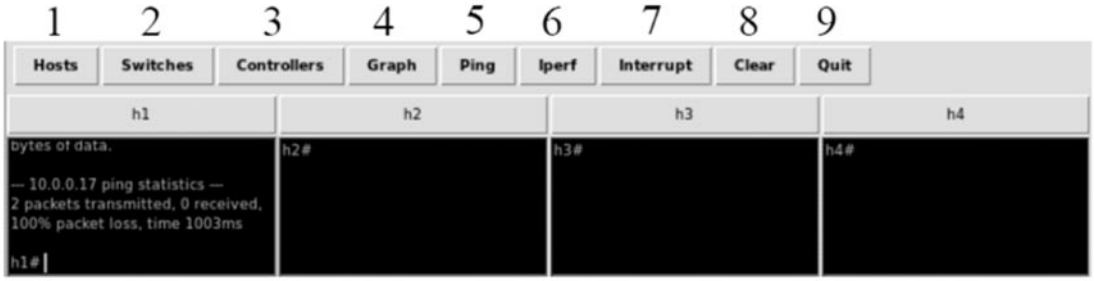


Fonte: Júnior; Nascimento; Silva, 2017.

2.3 Simulador de Redes

A figura 4, ilustra a tela inicial de uma GUI para redes *SDN*, que usa o *Mininet Emulator*. Em (1), temos o botão de configuração de *hosts* virtuais. Em (2), um gerenciador de *switches* para controlar o *Data Plane*. Em (3), o gerenciamento dos controladores. Em (4), temos o monitoramento do ambiente. Gráficos gerados a partir de uma *Command Line Interface* (CLI) fornecem dados da rede. Em (5), a ferramenta Ping verifica se há comunicação entre dispositivos. Em (6), a ferramenta de *Benchmarking Iperf* já vem instalada no sistema operacional utilizado. Em (7), o botão para interrupção do sistema. Em (8), o botão para realizar a parada e "destruição" da rede. E por fim, em (8), o botão sair do sistema.

Figura 4: Tela principal da GUI *Mininet Console*



Fonte: Júnior; Nascimento; Silva, 2017

3 DEFINIÇÃO E PLANEJAMENTO DO EXPERIMENTO

Nesta seção e na próxima, o trabalho será apresentado como um processo experimental. O mesmo seguirá as diretrizes definidas em (WOHLIN et al. 2012). Portanto a meta do trabalho é avaliar, através de um experimento controlado, o comportamento de uma rede SDN emulada sobre um *hypervisor* local, verificando a relação existente entre suas conexões e a conexão com o controlador da rede. O experimento terá como público-alvo, Pesquisadores da área de Redes e Sistemas Distribuídos com ênfase em Redes Programáveis, Computação Aplicada e Distribuída. O objetivo foi formalizado utilizando o modelo GQM proposto por Basili em [BASILI and WEISS 1983]: analisar uma Rede de Computadores Definidas por meio de *Software*, com a finalidade de avaliar (contra o atraso da rede) o impacto causado pelo aumento da geração de tráfego, com relação ao desempenho da rede, do ponto de vista de gestores e administradores de ambientes de Data Center programáveis, no contexto de pesquisadores de Redes Definidas por *Software (SDN)*.

3.1 Planejamento

As questões de pesquisa do experimento que precisam ser respondidas são as seguintes:

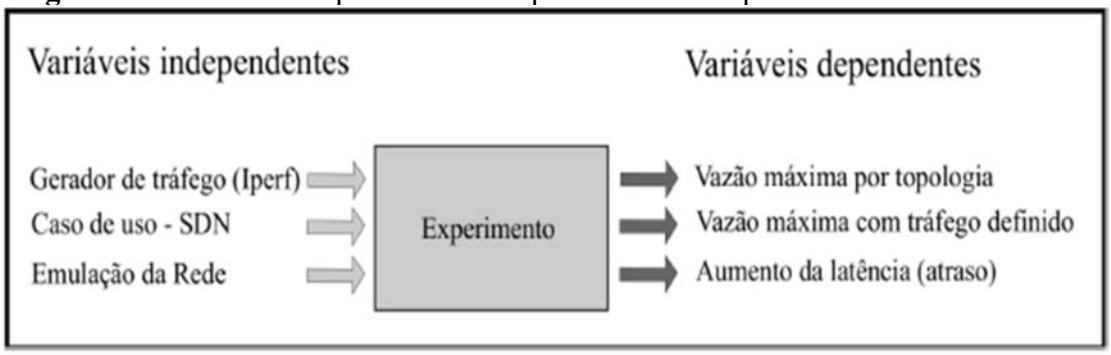
1º) O aumento de hosts na rede ocasiona uma diminuição significativa da vazão da rede?
2º) O aumento de hosts na rede ocasiona um aumento significativo da latência da rede?

Para avaliar essas questões, serão utilizadas duas métricas: 1ª) Média das vazões obtidas com o aumento de hosts na rede; 2°) Média dos atrasos obtidos com o aumento de *hosts* na rede. Tendo os objetivos e as métricas definidos, serão considerados as seguintes hipóteses:

• H0 vazão: O aumento do tráfego da rede não afetarão desempenho do sistema.
• H1 vazão: O aumento do tráfego da rede afetarão desempenho do sistema.
• H0 latência: O aumento do tráfego da rede não afetarão atraso do sistema.
• H1 latência: O aumento do tráfego da rede afetarão atraso do sistema.

Para ambas as hipóteses, a H0 e que se deseja rejeitar. Para averiguar quais as hipóteses serão rejeitadas, haverá uma análise das variáveis dependentes e independentes localizadas na figura 5.

Figura 5: Variáveis Independentes e Dependentes do Experimento



Fonte: Júnior; Nascimento; Silva, 2017.

A seguir, serão descritas as variáveis independentes do experimento. Para os cenários, consideraremos as situações decorrentes do uso de redes de computadores com vários hosts sendo ingressados na rede. Os Tipos 1 e 2 descritos na seção 3, são abordados abaixo, e serão utilizadas duas métricas como variáveis dependentes:

- Média do *Troughput* (Vazão máxima) alcançado durante a transferência de 1 *Gigabyte* de dados através do enlace, utilizando os protocolos TCP e UDP.
- Média do *Jitter* (média dos atrasos) da rede em decorrência da transferência de 1 *Gigabyte* de dados, com geração de carga a cada 10 segundos.

3.1.1 Descrição dos cenários utilizados no experimento

Projetou-se o experimento utilizando o contexto de amostras pareadas, com os resultados inferidos por cada topologia (*Single* e Linear) para avaliação do desempenho. Cada cenário, será executado um por vez, e a saída dessas informações serão armazenadas em arquivo de texto. As opções de execução, ocorrerão da seguinte forma: Configuração feita com base na descrição dos objetos do Tipo 1 e do Tipo 2. Após a criação da rede e conexão dos *hosts* virtuais, um dos *hosts* ficará responsável pela aplicação servidora. A forma como o ambiente executará o experimento, será de forma automática, favorecendo o não aparecimento de vieses que poderão aparecer ao acaso. Esses cenários que foram utilizados no experimento são enumerados abaixo:

1. Cenário (01) Topologia *Single*: O objetivo deste ensaio, será gerar procedimento de conexão e geração de tráfego na rede, considerando as entidades contidas no modelo conceitual do Tipo 1, presentes na tabela 1.
2. Cenário (02) Topologia Linear: O objetivo deste ensaio, será gerar procedimento para conexão e geração de tráfego na rede, considerando as entidades contidas no modelo conceitual do Tipo 2, presentes na tabela 1.

Tabela 1: Características do cenário e seus atributos quanto a topologia

Características do ambiente de rede		
Descrição	C01	C02
Topologia	*Single*	Linear
Modelo Conceitual	Tipo 1	Tipo 2
Controller	POX	POX
Ferramenta no espaço do usuário	DPCTL	OVS-OFCTL
Switch	*OpenFlowVSwitch*	*OpenFlowVSwitch*
Enlace	Wired	Wired
Link	*Gigabit Ethernet*	*Gigabit Ethernet*
Quantidade de *hosts*	1 – 60	1 – 60

Fonte: Júnior; Nascimento; Silva, 2017.

3.1.2 Instrumentação

O processo de instrumentação iniciará com a configuração do ambiente mediante planejamento da coleta de dados. A coleta dos dados será realizada no laboratório de mestrado do Programa de Pós – Graduação em Ciência da Computação – PROCC da Universidade Federal de Sergipe – UFS. Como os objetos do experimento são virtualizados, os mesmos possuirão as mesmas configurações. As tecnologias utilizadas estão enumeradas na sequência:

1. ***Oracle VM VirtualBox***: *VirtualBox* é um aplicativo de virtualização completa multiplataforma. Sua versão 5.0.26 é a versão estável para realização do experimento. Este hospedeiro terá uma máquina virtual convidada baseada em Linux com o *Mininet Emulator* instalado.
2. ***Mininet Emulator:*** É um emulador de redes que cria ambientes virtuais de *hosts*, *switches*, *controllers* e *links*. É executado por padrão em plataformas redes *Linux*, e as *switches* suportam uma API de alta flexibilidade por meio de redes programáveis *(MininetTeam)*. Esse emulador foi utilizado para concepção dos cenários 1 e 2 descritos na seção 2.
3. **Sistema gerador de tráfego IPERF:** *Iperf* é uma ferramenta *OpenSource* usada para medição de velocidade, vazão, qualidade de link e latência. Utiliza os protocolos da camada de transporte TCP e UDP. [JHA et al. 2015]. Essa ferramenta foi utilizada em todos os procedimentos realizados no experimento.
4. **Laboratório de Teste e Execução**: Serão criados dois ambientes de rede separados, sem canal de comunicação entre os mesmos, de acordo com o modelo conceitual definido no tipo 3 da seção 2.1.3.

4 OPERAÇÃO DO EXPERIMENTO

A seguir, são enumeradas as etapas para execução do processo de experimentação.

4.1 Preparação

1. Alocação de recursos para implantação do *hypervisor*: foi instalado um *hypervisor* do tipo 2, ou seja, virtualização completa de *hardware*. Nesse formato o hospedeiro vai dar suporte completo ao sistema operacional que trará no *kernel* as ferramentas necessárias para compor a rede.

2. Estudo sobre alocação de banda em *switches OpenFlow*: configuração do sistema operacional e dos *softwares* que contemplam o desenvolvimento de *switches OpenFlow.* O script que realizará a conexão dos hosts virtuais através de um *link Gigabit Ethernet*.

Com o sistema todo preparado, e com o cliente *Security Shell (SSH[1]*) instalado, o administrador da rede ficar apto para execução dos testes. Além disso, outros pesquisadores também participar˜ao dos procedimentos de execução do experimento como observadores. Os mesmos, possuirão autonomia para criticarem caso haja discrepância com o modo de condução do experimento.

4.2 Execução

Ao final das etapas anteriores, deu-se início ao experimento, seguindo de acordo com o planejamento descrito na seção 3.1. Durante o experimento, o mesmo teve de ser reiniciado para intervenção nos *scripts* que automatizaram o experimento, não contendo nenhum viés que comprometesse a execução do sistema.

4.2.1. Coleta de Dados

Nos parâmetros utilizados para validação dos dados, os dados de tempo são irrelevantes, porque o tempo sofrerá variação à medida que a carga da rede demande alto tráfego. O primeiro *host* que se conecta à rede, denominado h1, conterá à aplicação servidora para geração de tráfego, e os demais *hosts* (h2 variando até h60) se associarão ao ambiente automaticamente. H2 então começarão a transferir 1 GB de dados por esse segmento.

Após a transferência, cada *host* que finalizou, ficará gerando 100 Mbps de tráfego a cada 20 segundos, e assim sucessivamente, finalizando no host 60. Na segunda fase do experimento (Topologia Linear), a mesma metodologia será utilizada. Em ambas as etapas os *scripts* foram executados para o protocolo TCP e UDP, neste último (UDP), os resultados das médias dos atrasos serão exibidos.

4.3 Validação dos Dados

Para realização do experimento foi considerado um fator, Desempenho de Redes, e dois tratamentos, Geração de tráfego TCP e UDP para topologias TOPO e LINEAR. Diante desse con-

texto, foram computadas as médias de vazão máxima TCP e as médias de vazão máxima UDP. Através do protocolo UDP foi possível coletar o atraso da rede.

Como auxílio para a análise, interpretação e validação, foram utilizados três tipos de testes estatísticos, Teste de *Komolgorov-Smirnov*, ANOVA e Teste F. O teste *Komolgorov-Smirnov* foi usado para verificar a normalidade das amostras. O teste ANOVA foi utilizado para verificar se todas as médias (tratamentos) são iguais. E por último, o teste F para verificar se há relação entre as variáveis dependentes e independentes. Todos os testes estatísticos foram feitos utilizando a ferramenta *RStudio – Rstudio* inc (Walter et al. 2012).

5 RESULTADOS

5.1 Análise e Interpretação dos Dados

Para responder às questões, foram analisados as seguintes variáveis dependentes: a) Vazão máxima (*Throughput*) das topologias *SINGLE* e LINEAR; b) Aumento da latência da rede.

5.1.1. Vazão máxima (*Throughput)* das topologias SINGLE e LINEAR

Para responder à Questão de pesquisa 01, os resultados relacionados a Vazões máxima TCP e UDP são apresentados na tabela 03.

Os resultados do Cenário 01 (C01), mostram que a vazão média de tráfego TCP ficou em 596,80 MBPS, apontando para uma perda de vazão de 59,68%. Iniciado o tráfego com o protocolo UDP, foi obtido uma vazão média de 130,88 MBPS. Por fim, a média de atraso utilizando a topologia do cenário C01, ficou em 2.582ms.

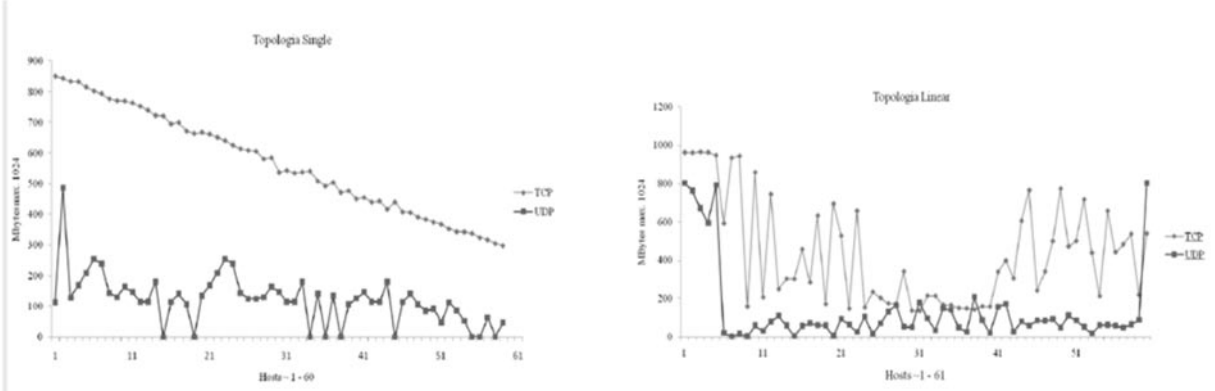
No Cenário 02 (C02), os resultados mostram que a vazão média de tráfego TCP ficou em 436,92 MBPS, apontando para uma perda de vazão de 49,7%. Iniciado o tráfego com o protocolo UDP, foi obtido uma vazão média de 130,88 MBPS. Por fim, a média de atraso utilizando a topologia descrita em C02, ficou em 4.267ms. Uma observação importante acerca do Cenário 02 (C02), diz respeito a limitação de 100 MB imposta pela própria ferramenta, na geração de tráfego UDP nos segmentos.

Esses resultados, sugerem que o uso de topologias Lineares possuem, em média, menor perda de vazão em decorrência do aumento do número de hosts. Com isso, a partir da análise prévia de dos dados, supõem-se que à resposta para a Questão 01 de Pesquisa seria "sim", que o aumento de hosts não afeta o desempenho do sistema. Mas, não é possível fazer tal afirmação sem evidências estatísticas suficientemente conclusivas. Diante das suposições, foi definido um nível de significância de 0.05 em todo o sistema, e foi aplicado o Teste de Kolmorogov-*Smirnov* (KS) para análise da distribuição normal. Em C01 e C02, com protocolo TCP, ambas as amostras apresentaram valores, leia-se p-*value*, associados ao KS maiores que o nível de significância adotado.

Tento obtido valores maiores para os dois ambientes, assume-se então que a distribuição é normal para ambos os casos. Sendo assim, O Teste de Hipótese aplicado nesse contexto será o Teste F, pelo fato de todas as amostras serem maiores que 30. No C01 (TCP), verificou-se que o p-value de 0.728 é maior que o nível de significância adotado. Confirmando a diferença entre as médias de 159.8866 Mbps. No C02 (TCP), verificou-se um p-value de 0.1196 maior que o nível de significância de 0.05. Confirmando a diferença entre médias TCP para os dois cenários. A vazão média nos dois cenários é significativamente diferente, ou seja, a hipótese (H0) de que o aumento de hosts não afeta o desempenho do sistema é rejeitada, pois o teste de significância é maior que 0.05 em ambos os casos, C01 e C02 com protocolo TCP.

A Figura 6 contém o gráfico que representa o escalonamento da rede de 1 a 60 hosts das topologias *Single* e Linear. Vemos que na topologia *Single*, a cada novo *host* participando do ambiente, houve uma perda significativa, apontando para um possível colapso do sistema caso o mesmo continue a crescer. Já na topologia Linear, o sistema se manteve oscilante, variando de acordo com a carga aplicada pelos hosts, conforme configuração do script, com geração de tráfego por host a cada 20 segundos, até finalização do experimento.

Figura 6: Vazões Máxima TCP e UDP de 1 a 60 hosts com 1 GBPS de Tráfego



Fonte: Júnior; Nascimento; Silva, 2017.

5.1.2 Atraso máximo da Rede (*Jitter*)

Para responder à questão de pesquisa 02, os resultados relacionados a vazões máxima UDP e o Jitter são apresentados na tabela 03. Nesta segunda fase de ensaios, foi utilizado uma Análise de Variâncias (ANOVA) para verificar a relação entre essas variáveis, já que as vazões e o atraso se distribuem normalmente. Essa análise caracterizou-se como teste paramétrico, levando em consideração o comportamento das amostras, o teste comparou a diferença das amostras e a magnitude dessa diferença.

Com a aplicação do teste ANOVA, para C02(UDP), verificou-se um p-value de 1.124e-05 que é menor que o nível de significância de 0.05, isto é 95% de certeza. Esse p-value menor que 0.05 também foi encontrado no teste TCP de C01. Com relação ao *Jitter*, verificou-se um p-value de 9.094e-13, com nível de significância de 95%, que é menor que o nível de significância adotado, caracterizando-se a diferença entre as médias.

Diante os resultados, chega-se à conclusão da rejeição da H0 para tráfego TCP nas duas topologias, aceitando H1, e confirmando que o aumento da quantidade de hosts com tráfego TCP tem impacto significativo no desempenho do sistema nas topologias *SINGLE* e LINEAR, e que a diminuição de vazão da rede não caracteriza o aumento da latência, ou seja, os atrasos se mantiveram os mesmos, independentes da vazão.

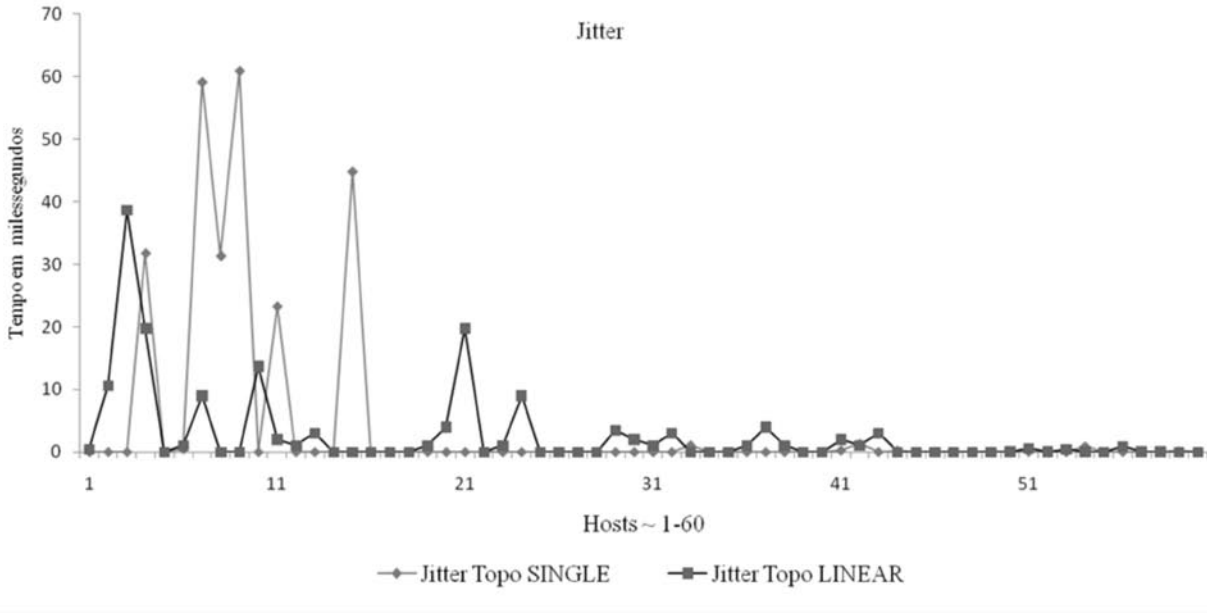
Por fim, a Figura 7 contém o gráfico relacionado ao *Jitter*, confirmando as evidências dos testes estatísticos, mostrando a estabilização dos atrasos após o vigésimo host associar-se ao sistema.

Tabela 2: Geração de Tráfego e Vazões obtidas na topologia SINGLE e LINEAR

Inferência (Mbps) / Protocolo	Cenário 01			Cenário 02		
	TCP	UDP	*Jitter* (ms)	TCP	UDP	*Jitter* (ms)
Média	596.80	130.88	2.582	436.92	137.56	4.267
Desvio Padrão	173.41	60.68	6.352	274.66	208.23	13.44
P-*value*	0.728	0.033	1.25e-06	0.1196	1.124e-05	9.094e-13

Fonte: Júnior; Nascimento; Silva, 2017.

Figura 7: Média dos Atrasos de 1 a 60 hosts com 1 GBPS de Tráfego.



Fonte: Júnior; Nascimento; Silva, 2017.

5.2 Ameaças à Validade

Ameaças à validade interna: O método aplicado para geração de tráfego e escalonamento, foi adotado nas duas topologias. Entretanto, a ferramenta utilizada nos cenários C01 e C02, não informa o *Jitter* em tráfego TCP, o que poderia ser analisado por testes estatísticos. Por este motivo, os resultados dos testes com UDP, foram obtidos apenas no ensaio C02.

O experimento foi realizado com sistema virtualizado, sendo que, apenas um administrador poderá operá-lo, contando com alguns participantes como ouvintes, logo, as sugestões e implementações de outros administradores de redes, não puderam beneficiar a realização dos ensaios. O aprendizado do sistema SDN se deu por forma autodidata, caracterizando o estudo da psicologia denominado *Demand Characterization,* que compreende o aprendizado de um experimento no ato que ele aconteça [TURVEY 1973].

Ameaças a validades externas: Não houve uma padronização da concepção dos scripts que escalonaram o sistema SDN, subentendendo, que uma má configuração pode ter influenciado negativamente os resultados do experimento.

Ameaças à validade de construção: Os modelos conceituais foram elaborados seguindo a documentação da API *OpenFlow* e da documentação do *Mininet Emulator* (HUANG et al. 2014), entretanto, a documentação orienta sobre o modelo para topologias, e não para modelos de topologias para inferências experimentais.

6 TRABALHOS RELACIONADOS

As redes programáveis, demonstram ser o novo formato dos ambientes interconectados. A abordagem utilizada neste trabalho, foi executada com abordagens guidas, ou seja, conexões *ethernet.* Sendo que, a variante Mininet WiFi, é um emulador voltado para implementação de Redes *Wireless* Definidas por *Sofware* (SDWN) [COSTANZO et al. 2012]. Da mesma forma do *Software Defined Network* (SDN), as redes SDWN suportam programação centralizada no controlador da rede, por meio de *Wireless Boxes* (APs). Com relação a testes e verificação, *debugging* e ajuda na solução de problemas, o *OpenFlow Operation per Second* (OFLOPS) (SHERWOOD and YAP 2011), permite simular operações em *OpenFlow Switches,* por meio de um *framework* modular, gerando transferência do controlador para o *switch* e vice-versa, quantificando a performance dos ativos no *Data Plane*. [JARSCHEL et al. 2012].

No lado do controlador, existem ferramentas que implementam testes específicos para esse elemento da rede. Em (CANINI et al. 2012), o *Cbench* foi desenvolvido por *Robert Sherwood*, e sua principal aplicação são ferramentas específicas para *Benchmarking* do controlador da rede SDN. O Cbench monitora por meio de processo único as várias instâncias aplicadas ao controlador, gerando dados estatísticos obtidos de múltiplas *switchs* interconectadas.

Condições de corrida também são observadas em ferramentas como o NICE [CANINI et al. 2012], que gera *streams* de pacotes de teste de eventos. Por fim, com relação a segurança de sistemas computacionais temos o *FlowChecker* (AL-SHAER and AL-HAJ 2010), *OFTEN* [KUZNIAR et al. 2012] e o *Veriflow* [KHURSHID et al. 2013]. Essas ferramentas verificam irregularidades no sistema, tal como violação. Enquanto as duas primeiras são baseadas em análises *offline*, a última tem capacidade de análise online. Verificações incluindo segurança, problemas de acessibilidade também estão incluídas neste tipo de ferramenta.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A baixa performance dos sistemas em rede, podem levar a intervenientes quando existem sistemas dependentes de qualidade de tráfego. As redes *SDN* vem para flexibilizar o controle desses sistemas, entretanto, e também pelo fato da padronização ainda estar em andamento, pesquisadores implementam testes experimentais, com o objetivo de garantir que esses sistemas suportem os desafios das redes programáveis.

Diante desse contexto, o presente trabalho apresenta uma contribuição acerca do *Benchmarking* das redes *SDN*, bem como, fomenta a experimentação no ambiente industrial. Assim como no desenvolvimento de software, em Redes *SDN*, várias ferramentas *OpenSource* estão disponíveis para este fim. O *Mininet Emulator* é a ferramenta de prototipação, implementação e testes mais flexível na concepção de ambientes *SDN*, e possibilita o uso do paradigma experimentalista e análise de dados.

Na análise dos dados, ficou bem claro que o uso do protocolo *TCP* afeta consideravelmente o sistema, e que a diminuição da vazão do sistema não caracteriza o aumento do atraso da rede. Por fim, como trabalhos futuros, novos experimentos com uma ferramenta específica para controladores *SDN*, poderão ser utilizados para encontrar evidências, de que sistemas de redes programáveis substituir˜ao os sistemas convencionais no futuro próximo.

REFERÊNCIAS

AL-SHAER, E. and AL-HAJ, S. (2010). **Flowchecker:** Configuration analysis and verification of federated openflow infrastructures. In Proceedings of the 3rd ACM workshop on Assurable and usable security configuration, pages 37–44. ACM.

BASILI, V. R. and WEISS, D. M. (1983). **A methodology for collecting valid software engineering data**. Technical report, DTIC Document.

CANINI, M., VENZANO, D., PERESÍNI, P., KOSTIC, D., and REXFORD, J. (2012). **A nice way to test openflow applications**. In Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), pages 127–140.

COSTANZO, S., GALLUCCIO, L., MORABITO, G., and PALAZZO, S. (2012). **Software defined wireless networks**: Unbridling sdns. In 2012 European Workshop on Software Defined Networking, pages 1–6. IEEE.

GONDIM, E. B. (2015). **Monitoramento de Desempenho com Middleboxes em Redes Definidas por Software**. PhD thesis, Universidade de Brasília.

GUDE, N., KOPONEN, T., PETTIT, J., PFAFF, B., CASADO, M., MCKEOWN, N., and SHENKER, S. (2008). **Nox**: towards an operating system for networks. ACM SIGCOMM Computer Communication Review, 38(3):105–110.

HUANG, T.-Y., JEYAKUMAR, V., LANTZ, B. B., FEAMSTER, N., WINSTEIN, K., and SIVARAMAN, A. (2014). **Teaching computer networking with mininet**. In ACM SIGCOMM.

JARSCHEL, M., LEHRIEDER, F., MAGYARI, Z., and PRIES, R. (2012). **A flexible openflow-controller benchmark**. In 2012 European Workshop on Software Defined Networking, pages 48–53. IEEE.

JHA, N. K., AGARWAL, N., and SINGH, P. (2015). **Realization of congestion in software defined networks**. In Computing, Communication & Automation (ICCCA), 2015 International Conference on, pages 535–539. IEEE.

KHURSHID, A., ZOU, X., ZHOU, W., CAESAR, M., and GODFREY, P. B. (2013). **Veriflow**: verifyingnetwork-wide invariants in real time. In Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), pages 15–27.

KIM, T., KOO, T., and PAIK, E. (2015). **Sdn and nfv benchmarking for performance and reliability**. In Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific, pages 600–603. IEEE.

KREUTZ, D., RAMOS, F. M., VERISSIMO, P. E., ROTHENBERG, C. E., AZODOLMOLKY, S., and UHLIG, S. (2015). **Software-defined networking**: A comprehensive survey. Proceedings of the IEEE, 103(1):14–76.

KUZNIAR, M., CANINI, M., and KOSTIC, D. (2012). **Often testing openflow networks**. In 2012 European Workshop on Software Defined Networking, pages 54–60. IEEE.

MininetTeam. **Mininet overview**, year = 2016 (accessed August 16, 2016), howpublished = ”http://mininet.org/overview/”.

NEWMAN, P., MINSHALL, G., and LYON, T. L. (1998). **Ip switching-atm under ip**. IEEE/ACM Transactions on networking, 6(2):117–129.

ROWSHANRAD, S., NAMVARASL, S., ABDI, V., HAJIZADEH, M., and KESHTGARY, M. (2014). **A survey on sdn, the future of networking**. Journal of Advanced Computer Science & Technology, 3(2):232.

SHENKER, S., CASADO, M., KOPONEN, T., MCKEOWN, N., et al. (2011). **The future of networking**, and the past of protocols. Open Networking Summit, 20.

SHERWOOD, R. and YAP, K. (2011). Cbench controller benchmarker. Last accessed, Nov. Turvey, M. T. (1973). **On peripheral and central processes in vision**: Inferences from an information-processing analysis of masking with patterned stimuli. Psychological review, 80(1):1.

WALTER, O. M. F. C., HENNING, E., KONRATH, A. C., da Cunha Alves, C., and SAMOHYL, R.W. (2012). **Uma visão geral do rstudio aplicado ao ensino de controle estatístico do processo**. In XL CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA.

WOHLIN, C., RUNESON, P., H¨OST, M., OHLSSON, M. C., REGNELL, B., and WESSLÉN, A. (2012). **Experimentation in software engineering**. Springer Science & Business Media.