# Teste de desempenho de Algoritmo de Controle de Congestionamento TCP utilizando testbed FIBRE

**Eduardo Henrique Rocha Zanela**[1], **Cesar Augusto Cavalheiro Marcondes**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de São Carlos (UFSCar)

eduardo.zanela@ufscar.br, marcondes@ufscar.br

***Resumo.*** *O controle de congestionamento do TCP é fundamental para manter um bom funcionamento e desempenho de uma rede de internet, pois controla a taxa de transmissão realizada pelos emissores, evitando ou reduzindo ocasiões de congestionamento. Apesar de diversos estudos realizarem testes entre diferentes TCP's, não existe um protocolo genérico ideal, mas sim protocolos que funcionam melhor em determinadas situações. Winstein e Balakrishan (2013) apresentaram no SIGCOMM*[1] *um programa capaz de gerar algoritmos de controle de congestionamento para serem executados nos pontos finais da rede, o programa Remy. Os resultados apresentados no artigo, através de testes comparativos realizados pelo programa Network Simulation (NS2), demonstraram um funcionamento bastante superior aos protocolos atuais. Porém para a criação de novos algoritmos com diferentes cenários, o Remy requer um alto custo computacional, e ainda sim é necessário uma grande quantidade de dias para sua execução completa. Para a resolução desta problemática, criar uma versão suavizada do Remy permite minimizar o tempo de processamento e o custo computacional. Buscando facilitar o teste de desempenho e visando um teste em rede real, a utilização de uma testbed FIBRE se encaixa perfeitamente no cenário.*

## 1. Introdução

Conforme a evolução tecnologia, o número de situações envolvendo as transferências de dados aumentaram, crescendo também a quantidade de problemas nas redes, um deles, o seu congestionamento. Para que está situação não ocorra, foi desenvolvido o Protocolo de Controle de Transmissão (TCP) [Allman et al. 2009]. O TCP é um protocolo da camada de transporte da arquitetura TCP/IP, e é responsável pelo envio de dados pela rede de forma correta, sem erros, de maneira confiável e na sequência certa para o receptor, possui também um mecanismo de controle de congestionamento que tem como objetivo controlar o tráfego da rede. Existem diversas variações e versões do algoritmo TCP, sendo cada vez mais aperfeiçoado, alvejando o melhor desempenho da rede, com o menor número de perdas possíveis.

Os algoritmos de controle de congestionamento até então eram pré definidos conforme o tráfego da rede e o seu cenário de implementação, porém em 2004, foi proposto [Winstein and Balakrishnan 2013] um protocolo que é criado por um programa chamado de Remy, que com algumas informações da rede como entrada, o próprio computador através de métodos de controle de congestionamento, encontrando assim o melhor algoritmo para uma determinada rede. Sua execução é nos pontos finais da rede, mas com um

[1]http://www.sigcomm.org

comportamento emergente simples, desbancando os protocolos existentes até então. Os testes do resultado da ferramenta Remy foram realizados virtualmente através do simulador Network Simulation[2], e disponibilizado a reprodução dos resultados e todo o código[3] para novas simulações, estudos e testes.

Apesar dos resultados dos testes entre o algoritmo do Remy com outros TCP's existentes serem considerados muito melhores, o Remy apresenta uma problemática para a realização de novos testes com novas configurações de ambientes, pois necessita de máquinas com grande poder computacional e ainda assim acabam levando muito tempo para a conclusão do seu resultado final. Reduzir este poder de processamento facilita na criação de novos algoritmos, possibilitando até implementações em redes reais ou uma generalização de uso como o principal TCP em diversas redes.

A suavização do algoritmo de controle de congestionamento do TCP Remy, requer uma série de passo e métodos como análise de árvores de decisões, combinadas e generalizadas. Para que a avaliação da suavização do protocolo possa ser realizada de maneira realística e sistemática, permitindo incluir variáveis que software como o NS-2 não é capaz de realizar durante a sua simulação, se torna necessária a coleta de amostras temporais de pacotes em redes reais. Tendo esta necessidade em vista, este trabalho objetivou a criação destes traces utilizando-se da infra estrutura de uma testbed do FIBRE (Future Internet Brazilian Environment for Experimentation). Os tracers foram utilizado como entrada para teste de desempenho através do programa TCP Evaluation Suite permitindo assim comparar com outros algoritmos TCP para o mesmo cenário.

As próximas seções estão divididas nas seguintes maneiras: a Seção 1.1 apresenta um breve conceito sobre o protocolo de controle de congestionamento; a Seção 1.2 explica o que é o programa RemyCC (Remy Control Congestion); a Seção 1.3 apresenta a funcionalidade do programa TCP Evaluation Suite; a Seção 2 apresenta a infraestrutura do testbed FIBRE escolhido para a realização dos testes; a Seção 3 descreve os experimentos e os resultados obtidos; a Seção 4 relata a experiência de uso da testbed FIBRE; e finalizando com a Seção 5 onde é apresentado a conclusão do trabalho.

### 1.1. Protocolo de Controle de Congestionamento

O congestionamento de uma rede se dá a uma ocasião onde a quantidade do tráfego de entrada se torna maior que a capacidade do tráfego de saída, gerando assim um gargalo, e piorando ao longo do tempo caso o tráfego de entrada não seja reduzido. Segundo [Tanenbaum 2003] existem dois problemas potenciais na rede, a capacidade da rede e a capacidade do receptor.

A Figura 1(a) demonstra situações de tráfego de dados: uma situação perfeita onde o crescimento é linearmente e mantêm a quantidade de tráfego no limite máximo da rede, sem que exista perda de dados; a segunda situação assimila o que seria uma resposta desejada para um bom tráfego, e, por fim, um exemplo situacional de um tráfego que entrou em colapso devido a crescente emissão de pacotes na rede. A Figura 1(b) apresenta um crescente atraso dos pacotes devido ao congestionamento de rede.

Para solucionar o problema do congestionamento em redes, propôs-se um con-

[2]www.isi.edu/nsnam/ns/

[3]http://web.mit.edu/remy/

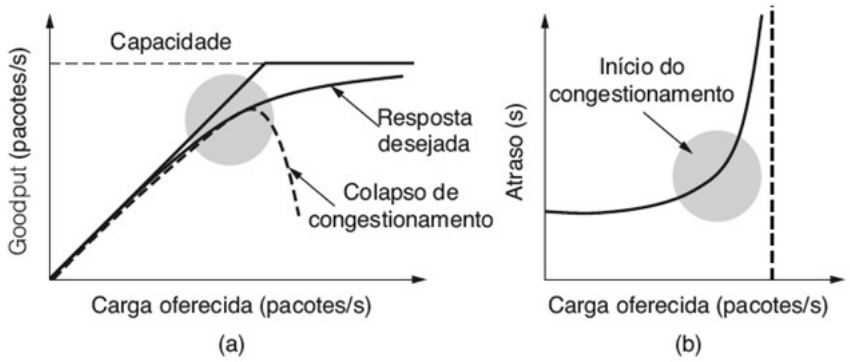


**Figura 1. Congestionamento de rede [Tanenbaum 2003].**

trolador de congestionamento. O termo controle de congestionamento [Jacobson 1988], é usado para descrever os esforços realizados pelos nós da rede para impedir ou responder a condições de sobrecarga da rede. O controle de congestionamento diz respeito à resposta de fluxos a limitações de largura de banda ou à presença de outros fluxos [Andrew et al. 2008]. Para [Chaudhary and Kumar 2017] o controle de congestionamento é um processo para evitar o comprometimento da rede devido ao congestionamento de rede e perda de pacotes.

### 1.2. RemyCC

Publicado por [Winstein and Balakrishnan 2013], o Remy é um procedimento de pesquisa heurística que gera protocolos de controle de congestionamento como um proxy para a solução ideal.

Para [Sivaraman et al. 2014] o Remy é um protocolo ótimo para um modelo imperfeito de uma rede real. Se até então os algoritmos sempre foram criados a partir de ação humana, o Remy quebra este conceito, e permite utilizar uma nova abordagem de controle de congestionamento com um novo estilo de designer de protocolo de camada de transporte dando mais liberdade aos arquitetos de redes e designers da camada de link.

Apesar de existir uma grande quantidade de TCP, não existe um genérico que sirva para diversas situações, mas sim a situação contrária, existem TCP's situacionais que são incapazes de adaptar os seus algoritmos de controle de congestionamento aos novos cenários, gerando assim uma limitação no TCP. Os autores de [Winstein and Balakrishnan 2013] afirma que o Remy pode gerar algoritmos tanto para redes onde alguns parâmetros são facilmente reconhecidos, por exemplo Data centers, quanto para redes onde conhecimento prévio é menos preciso, como redes celulares.

A Figura 2 permite entender melhor as etapas da execução do programa Remy. Após realizar todos os testes possíveis (n-testes) e gerar uma árvore de possibilidades, o Remy verifica o melhor algoritmo e o apresenta como o resultado final. O algoritmo otimizado é então executado nos pontos finais reais da rede, não existindo mais nem um aprendizado adicional após a otimização.

Algoritmos produzidos por Remy, podem substituir outros algoritmos de controle de congestionamento de uma implementação de TCP, pois o mesmo permite ser alterado como uma biblioteca de rede ou módulo de kernel que implementa controle de congestionamento, possibilitando alteração de seu código em ocasiões que a rede sofre alterações.

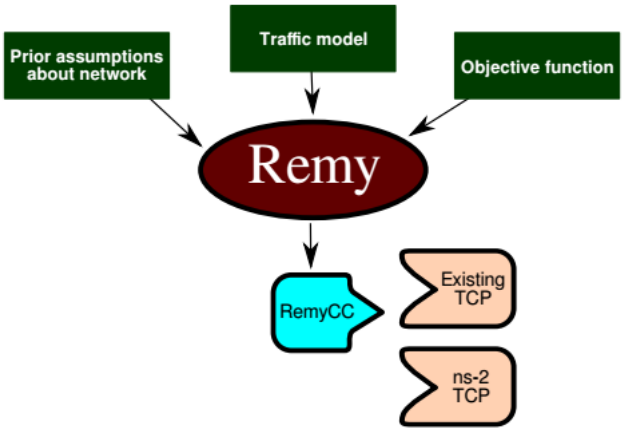


**Figura 2. Etapas do programa Remy [Winstein and Balakrishnan 2013].**

”É um desafio garantir que o desempenho de um sistema distribuído em redes de teste bem caracterizadas se estenderá a redes reais, que inevitavelmente diferem das que se prevê no desenvolvimento e continuarão a evoluir ao longo do tempo”[Sivaraman et al. 2014].

Os autores de [Sivaraman et al. 2014] utilizaram em seus testes uma máquina de 80 núcleos que demorava por cinco dias para concluírem suas simulações. Em [Winstein and Balakrishnan 2013] foi utilizado um servidor de 48 núcleos no MIT, e seu tempo de demora foi entre uma a duas semanas. E segundo Sivaraman é necessário um grande poder de processamento, e ainda sim demora um longo tempo para finalizar a sua execução.

### 1.3. TCP Evaluation Suite

O TCP Evaluation Suite [Andrew et al. 2008] é um conjunto de ferramentas de testes que avaliam relações entre diferentes TCP’s, através de experimentos com redes idênticas, permitindo experimentar novas implementações nestes algoritmos e suas sensibilidades de diferenças das características entre os protocolos.

Seu objetivo principal é colaborar para que pesquisadores possam avaliar de maneira rápida e fácil suas propostas de extensões do TCP utilizando um conjunto padrão de casos de testes em comum, permitindo assim relacionar e constatar possíveis melhorias quanto ao TCP padrão. Segundo [Chrost and Chydzinski 2009] o Evaluation Suite é um conjunto de avaliações de TCP’s que se concentra na avaliação de desempenho específica do TCP, com o objetivo de colaborar com pesquisadores que criam novas extensões do TCP.

O TCP Evaluation Suite utiliza um conjunto pré-gerado de configurações de rede, fluxos e carga de trabalho para ser transferidos pelos TCP’s, possibilitando o uso de cenários idênticos para diferentes algoritmos. Em sequência é avaliado o comportamento estatístico de uma grande quantidade de fluxos que coexistem em uma topologia de rede complexa, visto que as experiências podem ser refeitas diversas vezes usando sementes aleatórias e com as mesmas configurações de carga de trabalho.

O Evaluation Suite consiste em um conjunto de scripts TCL[4]:

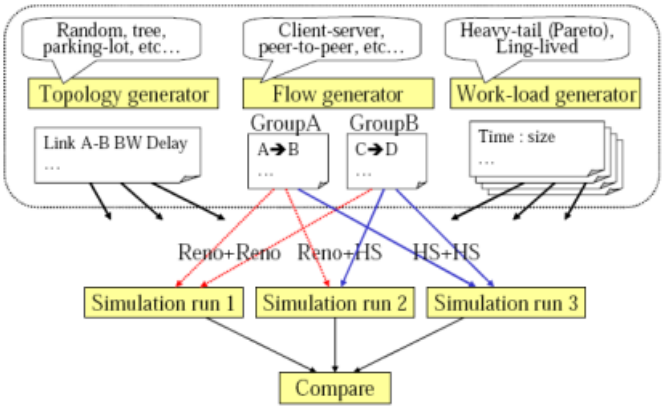


**Figura 3. Comparativo TCP Evaluation Suite [Andrew et al. 2008].**

Para [Andrew et al. 2008] uma simulação deve ser o mais realista possível e bem definidos, pois assim permite que os testes possam prever o comportamento dos protocolos de controle de congestionamento em redes reais, e faça que os ruídos estatísticos não mascare efeitos importantes.

## 2. Infraestrutura utilizada para a experimentação

Dentre as diversas ilhas do FIBRE[5], esse trabalho utilizou a testbed FIBRE disponibilizada pela ilha que se encontra na UFSCar[6], considerada uma ilha de netfpga, permitindo realizar os testes de desempenho e a captação dos tracers.

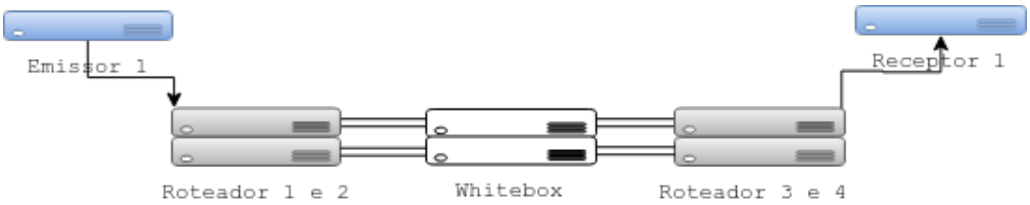


**Figura 4. Máquinas utilizadas na Testbed.**

A infra-estrutura funcional durante os testes é composta basicamente por 6 Netfpga modelo Supermicro X9SCM-F e 5 Whitebox modelo 1st generation – SuperServer 5018A-TN7B (Figura 4).

As netfpga foram divididas em uma topologia Dumbbell (Figura 5), sendo: uma netfpga atuando como emissor, inserindo dados de quatro emissores de dados diferentes; uma netfpga atuando como receptor, captando dados para quatro receptores diferentes; e as quatro restantes programadas como roteadores e por fim entre as máquinas configuradas como roteadores se encontra os dois whitebox. A topologia usada para o experimento é util para representar uma rede com canal de transmissão saturada.

[4]http://nrlweb.cs.ucla.edu/tcpsuite/
[5]https://fibre.org.br/start-using-fibre/local-portals/
[6]https://fibre.org.br/start-using-fibre/register/ufscar-island/

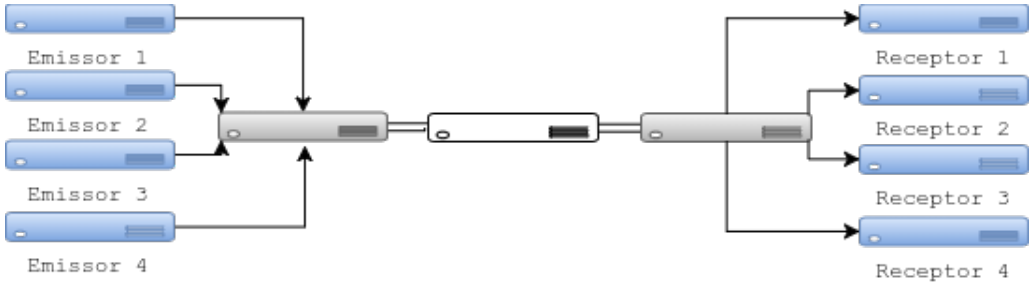


**Figura 5. Topologia da Testbed**

Dentre as características da netfpga se encontra a facilidade de programação do hardware, que permite transformá los em diversos dispositivos de redes de computadores, uma delas a possibilidade de operar de modo emissor/receptor. A infraestrutura é composta também por whitebox, que são caixas que permite realizar testes nas estruturas internas ou funcionamento de uma aplicação, como por exemplo a captação de informação da rede.

As placas de netfpga possui quatro portas de saída de rede, para as situações de emissor e receptor foram utilizadas todas portas, sendo cada porta tratada como um IP. Para as configurações dos roteadores foi utilizado duas portas como entrada e duas como saída, permitindo assim o tráfego de apenas duas redes por vez, necessitando por sua vez dois roteadores. As Whitebox possuem seis portas de redes, sendo utilizada em um três portas como entrada e três como saída, e na outra apenas uma porta como entrada e outra como saída, deixando outras quatro portas disponíveis.

Para a instalação do software TCP Evaluation Suíte foi instanciado uma máquina virtual com o sistema Debian 8, com 20GB de HDD e 8GB de memória na máquina HP Server ProLiant DL380p Gen8, para o gerenciamento do servidor utilizou-se o framework OCF (Ofelia Control Framework) o qual possibilita realizar o gerenciamento dos recursos utilizados Server, enquanto para o gerenciamento das netfpga e do whitebox utilizou-se o framework OMF (Control Monitoring Framework).

Utilizou-se o SSH via VPN para acesso a testbed. Para isto foi necessário o download da VPN[7] do FIBRE, e a execução do mesmo para a conexão.

## 3. Descrição do experimento e resultados

Inicialmente para cada máquina, emissor/receptor, foi dado um endereço estático (IP) com todas na mesma faixa de comunicação (192.168.88.xxx). Para a verificação da comunicação, emissão de dados e comunicação entre si, foi utilizado o software Iperf[8]. Para cada receptor foi criado uma porta de acesso, entre 2000 a 2004. Neste momento inicial foram feitos testes verificando o tempo em segundos de transmissão, numero de bytes para transmissão do teste e entrada de dados para transmissão por arquivo. A Figura 6 apresenta a comunicação simultânea entre dois emissores com o mesmo servidor com uma faixa de tempo de 20 segundos, na primeira situação (Figura 6(a)) a largura de banda possui uma variação entre 930-947 mbits, com um total de transmissão de 2.18 gb en-

[7]https://wiki.rnp.br/download/attachments/87101626/fibre-vpn.zip
[8]https://iperf.fr/

quanto a segunda situação (Figura 6(b)) apresenta um melhor desempenho, com um total de transmissão de 2.19 gb, e uma faixa de largura de banda de 936-953 mbits.

Após os primeiros testes, foi criado um arquivo de regras para execução via shell para o FIBRE contendo 128 regras, sendo 32 para cada emissor, emitindo de maneira aleatória para os receptores.

```
Client connecting to 192.168.88.68, TCP port 2000
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.88.254 port 57692 connected with 192.168.88.68 port 2000
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   113 MBytes   947 Mbits/sec
[  3]  1.0- 2.0 sec   111 MBytes   933 Mbits/sec
[  3]  2.0- 3.0 sec   111 MBytes   931 Mbits/sec
[  3]  3.0- 4.0 sec   112 MBytes   938 Mbits/sec
[  3]  4.0- 5.0 sec   111 MBytes   933 Mbits/sec
[  3]  5.0- 6.0 sec   111 MBytes   932 Mbits/sec
[  3]  6.0- 7.0 sec   112 MBytes   936 Mbits/sec
[  3]  7.0- 8.0 sec   111 MBytes   929 Mbits/sec
[  3]  8.0- 9.0 sec   112 MBytes   938 Mbits/sec
[  3]  9.0-10.0 sec   111 MBytes   931 Mbits/sec
[  3] 10.0-11.0 sec   112 MBytes   936 Mbits/sec
[  3] 11.0-12.0 sec   111 MBytes   934 Mbits/sec
[  3] 12.0-13.0 sec   111 MBytes   934 Mbits/sec
[  3] 13.0-14.0 sec   111 MBytes   930 Mbits/sec
[  3] 14.0-15.0 sec   111 MBytes   934 Mbits/sec
[  3] 15.0-16.0 sec   113 MBytes   945 Mbits/sec
[  3] 16.0-17.0 sec   111 MBytes   928 Mbits/sec
[  3] 17.0-18.0 sec   112 MBytes   941 Mbits/sec
[  3] 18.0-19.0 sec   111 MBytes   932 Mbits/sec
[  3] 19.0-20.0 sec   111 MBytes   930 Mbits/sec
[  3]  0.0-20.0 sec  2.18 GBytes   935 Mbits/sec
```

(a) Pacote Cliente X para o Servidor Z

```
Client connecting to 192.168.88.68, TCP port 2000
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 192.168.88.83 port 49892 connected with 192.168.88.68 port 2000
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec   114 MBytes   953 Mbits/sec
[  3]  1.0- 2.0 sec   112 MBytes   942 Mbits/sec
[  3]  2.0- 3.0 sec   112 MBytes   940 Mbits/sec
[  3]  3.0- 4.0 sec   112 MBytes   943 Mbits/sec
[  3]  4.0- 5.0 sec   112 MBytes   942 Mbits/sec
[  3]  5.0- 6.0 sec   112 MBytes   941 Mbits/sec
[  3]  6.0- 7.0 sec   112 MBytes   937 Mbits/sec
[  3]  7.0- 8.0 sec   112 MBytes   943 Mbits/sec
[  3]  8.0- 9.0 sec   112 MBytes   944 Mbits/sec
[  3]  9.0-10.0 sec   112 MBytes   938 Mbits/sec
[  3] 10.0-11.0 sec   112 MBytes   936 Mbits/sec
[  3] 11.0-12.0 sec   112 MBytes   943 Mbits/sec
[  3] 12.0-13.0 sec   112 MBytes   944 Mbits/sec
[  3] 13.0-14.0 sec   112 MBytes   940 Mbits/sec
[  3] 14.0-15.0 sec   112 MBytes   942 Mbits/sec
[  3] 15.0-16.0 sec   112 MBytes   940 Mbits/sec
[  3] 16.0-17.0 sec   112 MBytes   940 Mbits/sec
[  3] 17.0-18.0 sec   112 MBytes   938 Mbits/sec
[  3] 18.0-19.0 sec   112 MBytes   940 Mbits/sec
[  3] 19.0-20.0 sec   112 MBytes   941 Mbits/sec
[  3]  0.0-20.0 sec  2.19 GBytes   941 Mbits/sec
```

(b) Pacote Cliente Y para o Servidor Z

**Figura 6. Envio de pacotes entre Cliente/Servidor**

A Figura 8 apresenta uma transmissão entre 4 emissores para o mesmo receptor, nela pode-se analisar que cada emissor envia um tamanho diferente de pacote, assim como uma faixa maior no tempo de transmissão, reduzindo sua largura de banda.

```
Server listening on TCP port 2000
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.88.68 port 2000 connected with 192.168.88.254 port 57684
[  5] local 192.168.88.68 port 2000 connected with 192.168.88.254 port 57686
[  6] local 192.168.88.68 port 2000 connected with 192.168.88.83 port 49854
[  7] local 192.168.88.68 port 2000 connected with 192.168.88.254 port 57688
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-20.1 sec  1022 MBytes   426 Mbits/sec
[  5]  0.0-22.9 sec   605 MBytes   221 Mbits/sec
[  6]  0.0-20.0 sec  1.11 GBytes   476 Mbits/sec
[  7]  0.0-32.8 sec   905 MBytes   232 Mbits/sec
[SUM]  0.0-32.8 sec  2.47 GBytes   647 Mbits/sec
```

**Figura 7. Pacotes de roteador 2**

Após os testes iniciais de configuração de largura de banda, executou-se o script contendo as regras, dando um intervalo de 10 segundos entre execução de regras. Os pacotes emitidos pelos emissores variavam entre os tamanhos de 512mb a 2.560mb (2.5gb). Os pacotes enviados são compostos pelo valor do emissor, valor do receptor, um date time + o tempo em micro segundo de retardo do pacote. A Figura 8 apresenta uma situação de traceroute de envio de 10 pacotes.

```
1 3 1021Mb 2018-03-26T015:18:31.647698e+01ms
2 4 638Mb 2018-03-26T015:18:31.727112e+01ms
3 4 1110Mb 2018-03-26T015:18:32.496502e+00ms
2 4 989Mb 2018-03-26T015:18:32.711906e+01ms
1 3 1013Mb 2018-03-26T015:18:33.964094e-01ms
3 2 846Mb 2018-03-26T015:18:35.951195e+00ms
1 1 2100Mb 2018-03-26T015:18:36.118790e+00ms
4 1 905Mb 2018-03-26T015:18:36.367796e+00ms
4 2 580Mb 2018-03-26T015:18:37.122077e+01ms
2 3 1310Mb 2018-03-26T015:18:38.125073e+00ms
```

**Figura 8. Pacotes de roteador 2**

Após a finalização do experimento, foi captado as informações dos tráfegos na whitebox, e em cada receptor, viu-se que em mesmo em situações de transmisão de carga máxima (2.5gb) o tráfego de dados se manteve, e não existiu nem uma situação de perda de pacote que causa-se algum dano na rede.

Os resultados esperados com os testes na Testbed FIBRE foram: (1) melhor entendimento no funcionamento, e utilização da estrutura FIBRE, (2) geração de traces em cenário real com a topologia (Dumbbell), (3) comparação de desempenho entre algoritmos de controles de congestionamento TCP e (4) utilização dos resultados para publicações cientificas.

## 4. Experiência de utilização do Testbed FIBRE da UFSCar

Apesar de ser a primeira experiência de uso de uma Testbed do FIBRE, os testes ocorreram com sucesso, a infraestrutura atendeu todas as necessidades, o suporte dado pela equipe técnica responsável foi ótimo, e quaisquer dúvidas ou dificuldades foram sanadas.

Apesar da ilha estar passando por melhorias, instalando novas máquinas e incluindo novas implementações, estas não se tornaram empecilho durante os testes.

Toda programação de hardware e coleta de dados foram realizadas manualmente através da execução de scripts via SSH (Secure Shell). Durante o período de utilização

do testbed a equipe responsável pela ilha realizou atualizações e novas configurações das máquinas, porém nada que comprometesse nos testes realizados.

## 5. Conclusão

Este trabalho apresentou dois experimentos realizados no testbed FIBRE localizado na UFSCar para a coleta de tracers das topologias ponto-a-ponto e barramento. No total foram utilizados 238 tracers, sendo 112 da topologia Dumbbell e 126 da topologia de barramento. Todo acesso foi via SSH. Os resultados obtidos nos testes serão apresentados primeiramente durante a apresentação final da tese de mestrado em janeiro de 2019. Os dados obtidos até o presente momento, assim como os próximos testes, deverão ser apresentados em formas de artigos e papers, e poderão ser também utilizados por toda comunidade acadêmica visando novas propostas de melhorias. A estrutura apresentado no trabalho também pode ser escrita em um playbook e deixar disponível em um github em forma de playbook para a comunidade acadêmica. Ampliar a infraestrutura de teste, fazendo testes com duas ou mais ilhas podem melhorar os resultados de testes.

## Referências


Allman, M., Paxson, V., and Blanton, E. (2009). Tcp congestion control. Technical report.

Andrew, L., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., and Rhee, I. (2008). Towards a common tcp evaluation suite. In *Proc. PFLDnet*.

Chaudhary, P. and Kumar, S. (2017). A review of comparative analysis of tcp variants for congestion control in network. *International Journal of Computer Applications*, 160(8).

Chrost, L. and Chydzinski, A. (2009). On the evaluation of the active queue management mechanisms. In *Evolving Internet, 2009. INTERNET'09. First International Conference on*, pages 113–118. IEEE.

Jacobson, V. (1988). Congestion avoidance and control. In *ACM SIGCOMM computer communication review*, volume 18, pages 314–329. ACM.

Sivaraman, A., Winstein, K., Thaker, P., and Balakrishnan, H. (2014). An experimental study of the learnability of congestion control. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 479–490. ACM.

Tanenbaum, A. S. (2003). 4ª edição. *Rio de Janeiro: Editora Campus*.

Winstein, K. and Balakrishnan, H. (2013). Tcp ex machina: Computer-generated congestion control. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 123–134. ACM.