Simulação de Ambientes de Redes Utilizando a Testbed FIBRE - Aplicações na Pesquisa e no Ensino

Diego Pedroso, Bruno Lopes, Cesar Marcondes, Emerson Barea

[1]Universidade Federal de São Carlos (UFSCAR)
Departamento de Computação

{diego.pedroso, bruno.lopes, marcondes, emerson.barea} @ufscar.br

Abstract. *Testbeds are Internet test platforms that allow testing, demonstration and can also be used for teaching purposes. Within the context of the FIBRE project, a prototype was developed that allows experimentation of the data plane of a network, as well as reprogramming of FPGA. Most testbeds do not allow the user to perform experiments that control the data plan and at the same time allow hardware reprogramming. In order to fill this gap and contribute to the advancement of the state of the art, this work aims to propose experiments using the FIBRE testbed to assist in teaching and research, providing researchers and students with a flexible and easy provisioning environment, allowing testing, validation and demonstration.*

Resumo. *As testbeds são plataformas de experimentação para internet que permitem testes, demonstrações e também podem ser utilizadas para fins de ensino. Dentro do contexto do projeto FIBRE foi desenvolvido um protótipo que permite experimentação do plano de dados de uma rede, bem como reprogramação de FPGA. A maioria das testbeds não permitem que o usuário realize experimentos que controlem o plano de dados e ao mesmo tempo permita reprogramação de hardware. Visando preencher essa lacuna, e ao mesmo contribuir com o avanço do estado da arte, este trabalho tem por objetivo propor experimentos utilizando a testbed FIBRE que auxiliem no ensino e em pesquisas, proporcionando que pesquisadores e alunos tenham um ambiente flexível e de fácil provisionamento, permitindo testes, validações e demonstrações.*

1. Introdução e Contexto

A rede mundial de computadores foi projetada em 1969, nos Estados Unidos e tinha como função interligar laboratórios de pesquisa, seguindo modelos e tecnologias que perpetuam até os dias atuais. A internet é baseada num princípio chamado fim a fim [Saltzer et al. 1984], que estabelece que todo processamento complexo deve ser feito nos sistemas finais, e o núcleo da rede apenas executa tarefas simples. O trabalho do núcleo da rede é transmitir pacotes da maneira mais eficiente e flexível possível. Todo o restante deve ser feito nas bordas (sistemas finais) [Kamienski et al. 2005]. A priori, o único tipo de informação a ser preservado pela rede, seriam as tabelas de roteamento.

O projeto inicial da internet suportava um número pequeno de usuários conectados, esse número cresceu e vem crescendo de forma exponencial com o passar dos anos. Uma das realizações mais notáveis da Internet não é necessariamente o que ela é capaz de fazer hoje, mas o fato de ter assumido as dimensões atuais, comparada aos seus propósitos

iniciais. Ela iniciou com objetivos bem modestos, não foi projetada para ser utilizada por milhões de pessoas no mundo inteiro. Com toda certeza, o conjunto de princípios que balizou o seu aparecimento e que hoje suporta a sua evolução é o grande responsável por isso [Kamienski and Sadok 2000]. Para atender problemas como este, novas tecnologias têm sido implementadas, como por exemplo, a tecnologia NAT, além disso, o modelo TCP/IP implica em uma rede sem inteligência e segurança.

Algumas dessas limitações são decorrentes do projeto pioneiro da internet, ao qual não permite modificações em equipamentos e protocolos de rede [Moreira et al. 2009]. Essa ossificação da internet faz com que novas arquiteturas não progridam, em decorrência disso, os problemas citados acima só aumentam. A inovação de tecnologias também é difícil sem experimentação em larga escala, como resultado de todas essas questões elencadas acima, alguns novos paradigmas de arquitetura de internet tem sido construídos e testados pela comunidade acadêmica e de iniciativa privada, com o intento de construir um novo modelo de internet, chamado de Internet do Futuro (IF).

As *testbeds*, são plataformas fundamentais para a criação e validação de novas tecnologias e arquiteturas de Internet do futuro. Visto que, oferecem um ambiente totalmente controlável, escalável e próximo das características reais de uma rede como a internet. Elas permitem a validação de arquiteturas *from-scratch*, com novos conceitos, baseada em novas tecnologias e principalmente, visando solucionar todos as implicações de escalabilidade, segurança e desempenho que atravancam este modelo atual de internet. Com elas pode-se realizar experimentos controlados e assim comprovar a viabilidade técnica e econômica de soluções de internet, [Khan et al. 2013] demostra importância das ferramentas de simulação na internet. A quantidade de trabalhos suportados por simuladores relacionados à redes de computadores é significativa. A maioria das *testbeds* não permitem que o usuário realize experimentos que controlem o plano de dados e ao mesmo tempo reprogramem um hardware FPGA. Visando preencher essa lacuna, e ao mesmo tempo contribuir com o avanço do estado da arte, este trabalho tem por objetivo propor experimentos utilizando a *testbed* FIBRE que auxiliem no ensino e em pesquisas, proporcionando que pesquisadores e alunos tenham um ambiente flexível e de fácil provisionamento, permitindo testes, validações e demonstrações.

Este trabalho esta organizado da seguinte forma, a seção 2 faz uma abordagem geral as *testbeds* e trabalhos relacionados com o tema, a seção 3 demonstra o ciclo de vida do experimento no ambiente, a seção 4 apresenta a arquitetura da *testbed* e seus componentes, a seção 5 mostra a proposta de experimento, por fim a seção 6 conclui e apresenta os trabalhos futuros.

2. Visão Geral das Testbeds e Trabalhos Relacionados

As pesquisas recentes em ambientes que utilizam *testbeds* para gerar resultados podem ser classificadas em diversos tipos, abaixo descreveremos alguns ambientes de experimentação mais utilizados pela academia.

A *testbed* GENI *(Global Environment for Network Innovations)*, é uma rede de computadores de propósito geral e compartilhado, conectados em topologias especificadas pelo experimentador através de uma rede de camada dois programável, permitindo até a troca de sistema operacional para experimentos [Berman et al. 2014]. Esse ambiente é utilizado para experimentação em larga escala.

A testbed Panlab (*Pan-European Laboratory*) é um dos projetos do European Union 6th Framework Programme, e tem como finalidade prover uma plataforma de redes que interconecta diversos laboratórios [Marcondes et al. 2012], oferecendo um serviço fim-a-fim, coordenado e centralizado.

O ORCA (*Orchestration and Reconfiguration Control Architecture*) é um ambiente *open source* para experimentação em redes 5G [Kazaz et al. 2017] que utiliza gerenciamento distribuído e seguro de recursos em domínios federados. É um projeto desenvolvido pelo NICL Lab da Duke University, está integrado com o Openstack[1] e com o xCAT[2], para suportar o provisionamento de nó *baremetal*, basicamente atua como um serviço de provisionamento de redes dinâmicas multicamadas.

A *testbed* ORBIT (*Open-Access Research Testbed for Next-Generation Wireless Networks*) permite emulações de rede sem fio de duas camadas, foi projetada para obter uma experimentação reprodutível, utilizando o *framework* de controle OMF [3] para orquestração do experimento, também conta com uma rede de máquinas SandBox[4], utilizadas para depuração dos experimentos.

O projeto FIRE (*Future Internet Research and Experimentation*) é focado em explorar potenciais novos recursos [Gavras et al. 2007], bem como explorar novas arquiteturas e protocolos de rede,com o objetivo de prover soluções para o futuro da internet. A iniciativa FORGE (Forging Online Education through FIRE) é uma plataforma que permitem que alunos do ensino superior ao redor do mundo realizem experimentos em IF, [Mikroyannidis et al. 2016] demostrou a que as *testbed* podem ter um impacto valioso na experiência de aprendizagem para educadores e estudantes.

3. Ciclo de Vida do Experimento

Nesta seção são descritas as fases que compõe o ciclo de vida do experimento. Basicamente o ciclo de vida do experimento consiste em 3 etapas, configuração, execução e resultados, a seguir serão descritos cada um dos ciclos em detalhes.

Na primeira fase, a **Descoberta de Recursos** é responsável pelas permissões que o usuário possui para gerir os recursos da *testbed*, quanto mais alto é o nível de privilégio, maior é o número de recursos que o usuário tem disponível para operar. Não é possível multiplexar as placas NETFPGAs visto que o *hardware* é reconfigurado, portanto dois usuários não podem fazer uso do mesmo recurso ao mesmo tempo.

Na fase de **Descrição do Experimento** o usuário configura e orquestra como será o seu experimento e pode fazer isso de 2 formas, na primeira delas, de forma mais simples, o usuário pode utilizar a nossa plataforma gráfica web, e arquitetar seu experimento. Essa plataforma permite que o usuário configure o seu experimento de uma forma simples e fácil, quais máquinas deseja utilizar e qual tipo de topologia. Assim como descarregar seus arquivos de configuração padrão (*bitstream*) e (P4) e receber os resultados pela mesma interface.

A opção é utilizar a biblioteca NETFPGA-OMF6, que possibilita que o experi-

[1]https://www.openstack.org
[2]https://xcat.org
[3]http://www.fibre-ict.eu/index.php/cmf/omf
[4]https://tools.ietf.org/html/rfc3165#page-55

mentador tenha ao mesmo tempo detalhes nas configurações, em um curto período de tempo para configuração, visto que as funções prontas só precisam de parâmetros para serem configuradas.

Figura 1. Ciclo de Vida do Experimento, Adaptado de [Berman et al. 2014].

A **Reserva de Recursos** permite que o usuário reserve os recursos necessários para o seu experimento, lembrando que ele não é capaz de reservar recursos aos quais não tem permissão, ou que já foram agendados para outro experimento que ocorrerá simultaneamente ao seu.

Na fase de execução, é onde todo ambiente entra da fase de configuração e produção. A fase **Configuração do Ambiente** consiste no usuário especificar (através da biblioteca ou website) a ordem cronológica do seu experimento, bem como suas configurações, tais como definir quais interfaces físicas vai utilizar, definir os endereços IP de cada interface, configurar uma ou mais VLANs, etc. Como mencionado anteriormente, esse *setup* pode ser feito pela interface *web*, ou usando a biblioteca NETFPGA-OMF. De qualquer forma será gerado o mesmo tipo de *script* ED, ao qual o OMF é capaz de interpretar e executar.

Durante a **Execução do Experimento**, vários processos e programas são executadas em diversos nós da *testbed* ao mesmo tempo. Fica a critério do usuário escolher quais indicadores quer observar ou coletar resultados, isso acontece na fase de **Coleta e Filtro de dados**, um exemplo desta etapa, o experimentador deseja ver um registrador de determinada máquina NetFPGA ou o fluxo de dados em determinada interface de uma máquina Whitebox. Por fim, na experimento os resultados especificados na fase de coleta e filtro retornam para o usuário na forma de metadados.

O **Modo Padrão de Operação** ao fim do experimento permite que uma rotina desfaça todas as configurações feitas pela usuário, deixando a *testbed* em modo *clean state*,e já prepara o ambiente para experimentos futuros.

4. Arquitetura da Testbed

A seguir serão apresentados todos os componentes da testbed, desde especificações dos equipamentos até os programas utilizados para gestão e operação do ambiente de experimentação. Dentro do contexto do projeto FIBRE (*Future Internet BRazilian environment for Experimentation*) foi desenvolvido um protótipo de *testbed* que permite

experimentação do plano de dados de uma rede, bem como reprogramação de FPGA (*Field Programmable Gate Array*).

A testbed utiliza arcabouço de controle OMF (Control Monitoring Framework), na versão 6, para gerência e orquestração dos experimentos. O OMF permite que o usuário escreva um ED (Experiment Description), que, por sua vez, é submetido ao EC (Experiment Controlle), que é responsável pelo controle do projeto em nome do usuário. O EC emite solicitações no plano de gerenciamento para configurar os recursos conforme especificado no ED. Uma vez que os pré-requisitos do experimento são atendidos, o EC envia diretivas aos RCs (Resource Controller) associados a cada recurso. Nesse cenário os RCs são todos os recursos disponíveis na testbed. Esses recursos vão desde máquinas NetFPGAs até o *switchs openflow*. A configuração dos recursos também é feita pelo RC, seguindo as diretrizes que foram criadas pelo experimentor no ED, os RCs correspondem a comandos enviados pelos ECs.

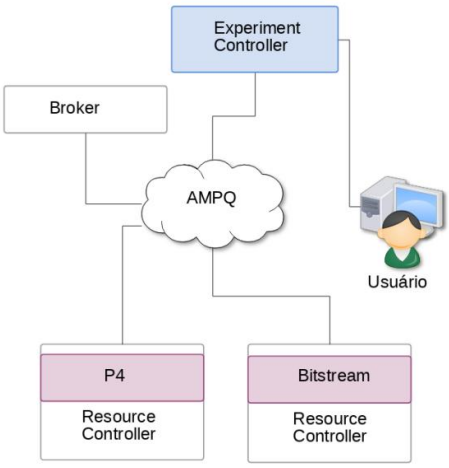

Figura 2. Arquitetura do Arcabouço de Controle OMF6.

O *software* de mensageria RabbitMQ[5] é utilizado para criar canais e filas de comunicação entre os RCs, ele se comunica através do protocolo AMQP (*Advanced Message Queuing Protocol*), e conecta todos os nós da *testbed*, basicamente ele atua como um *middleware* orientado a mensagens.

4.1. Interfaces NetFPGA

A arquitetura NetFPGA, (*Field-Programmable Gate Array*) ou arranjo de portas programável, é uma união de memória e processadores de sinais com quatro portas Ethernet de 1Gbps. A testbed conta com 9 servidores com placas FPGA integradas. A máquinas de cada servidor contém 2 discos de 500GB cada, um processador Intel Core 2 Quad 2.66GHz, com 4 núcleos e 8GB de memória RAM DDR3 com 1334 MHz. A documentação oficial do conjunto de *drivers* e programas da NetFPGA 1G utiliza o sistema operacional Fedora, na versão 13, para sua operação. Pelo fato de ser uma distribuição antiga, e já descontinuada, todos os *drivers* e *sofwares* foram adaptados para uma distribuição mais recente, do CentOs 7, na arquitetura 64 bits.

[5]https://www.rabbitmq.com

A portabilidade do conjunto de *softwares* foi possível visto que ambos os sistemas pertencem a distribuição Red Hat Linux[6]. O primeiro modelo de placas programáveis é chamado de NetFPGA 1G, com quatro portas *ethernet* de 1G cada, e ainda possuiu um processador FPGA Xilinx Virtex-II Pro 50 com dois núcleos PowerPC[7], para processamento dos pacotes, 53.136 elementos lógicos e ciclo de relógio de 8ns (125 MHz). A NetFPGA evolui sua tecnologia para modelos mais potentes, mais velozes e com maior capacidade de memória, porém com uma arquitetura de hardware similar, que obedece os mesmos conceitos da placa NetFPGA 1G.

A placa utiliza um barramento PCI ou PCI *express* para comunicação e programação da FPGA. A placa NetFPGA dispõe de dois bancos de memórias SRAM *Cypress* (*Static Random Access Memory)* ou memória estática de acesso aleatório, possuindo um espaço total de 4608KiB. A memória SRAM pode realizar operações de I/O (*Input/Output*) por ciclo de *clock*, e retorna os dados em três ciclos de *clock*. A placa NetFPGA dispôe ainda de dois *slots* de memória DRAM Micron (*Dynamic Random Access Memory*) ou memória de acesso aleatório dinâmico, totalizando 64MiB de memória. Esse tipo de memória trabalha de forma assíncrona, portanto necessita de uma atualização contínua dos dados.

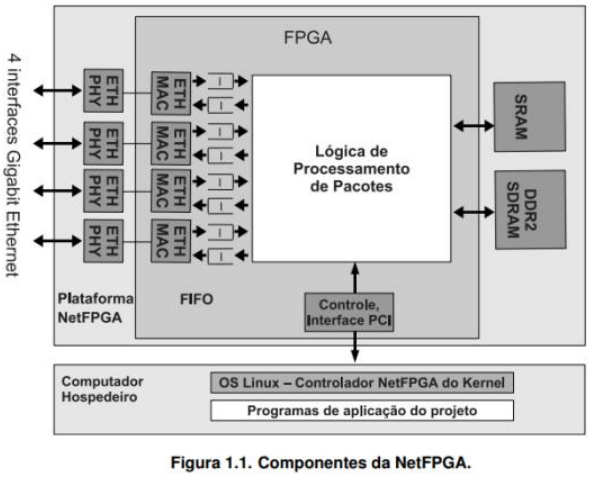

Figura 3. Componentes da Placa NetFPGA. Fonte: [Goulart et al. 2015].

O *software* ISE da Xilinx[8] permite sintetizar a placa FPGA, e tem como objetivo para lidar com as limitações de hardware da plataforma [Bilar 2016]. Um conjunto de interfaces e abstrações contidos no software gerenciam a manipulação de recursos e possibilitam a comunicação entre diferentes módulos. O sistema é compatível com as distribuições Windows e Linux, porém é recomendado o uso da segunda arquitetura. A placa Netfpga pode ser configurada de acordo com os projetos de referência, esses projetos fazem com que a placa funcione como um equipamento de rede, tais como roteador, *switch* ou uma placa de rede *ethernet*. Esses projetos dão suporte para o desenvolvimento de outros projetos mais complexos.

[6]https://www.redhat.com/pt-br
[7]https://www.cebix.net/downloads/bebox/PRG.pdf
[8]https://www.xilinx.com

A base dos projetos de referência é o chamado reference_nic, que nada mais é do que uma placa de rede *ethernet*. As 4 interfaces físicas da placa NetFPGA são mapeadas pelas distribuições Linux como **nf2cx**, onde a variável x varia de 0 até 3, e tem a mesma função de outras interfaces de rede comuns do sistema, tais como eth0, wlan0, etc. Os pacotes que chegam da rede podem ser recebidos pela interface *ethernet* da placa NetFPGA, ou ainda, pelo barramento PCI, que conecta a placa NetFPGA na placa mãe. Então, o **reference_nic** instancia oito módulos **rx_queue** para gerenciar o recebimento dos pacotes nas interfaces. Os módulos **rx_queue** e **tx_queue** fazem a interface entre a placa FPGA, o controlador MAC e o controlador PCI [Goulart et al. 2015]. Dentro de cada máquina NetFPGA existe um módulo controlador do OMF chamado RC, esse módulo permite operar ações oriundas do ED, e que são comandadas pelo EC. O módulo presente na NetFPGA permite que o experimentador faça *upload* de um arquivo do tipo *bitstream*, que é capaz de reprogramar a placa FPGA a cargo do usuário. Essa funcionalidade dá total flexibilidade, uma vez que pode ser utilizada para desenvolvimento de protótipos de pesquisa, implementação e testes de novos protocolos ou sistemas de rede.

4.1.1. Máquinas Whitebox

A Whitebox serve como um componente para regular o fluxo de dados de acordo com o controlador SDN (*Software Defined Networks*). Ao contrário do OVS (*Open Virtual Switch*) convencional, a Whitebox é apenas um *switch* cru e vazio que não tem inteligência. Para poder funcionar como deve ser, as Whitebox requerem um software de comutação virtual que possa ser incorporado diretamente. Além disso, as Whitebox também devem receber ordens do controlador SDN [Manggala et al. 2015] para poderem funcionar. Essa é a diferença mais fundamental entre as Whitebox e os *switchs* convencionais. A *testbed* conta com 5 máquinas Whitebox, que possuem 8 placas de rede *ethernet* Gigabit, 1 disco SSD *(Solid State Drive)* 256GB, 8GB de memória RAM DDR3 HyperX 1866Mhow toHz, com processador Intel Atom CPU C2758 2.40GHz com 6 núcleos e placa mãe A1SRM-LN7F-2758. O sistema operacional utilizado foi o Ubuntu 17.10, na arquitetura 64 bits.

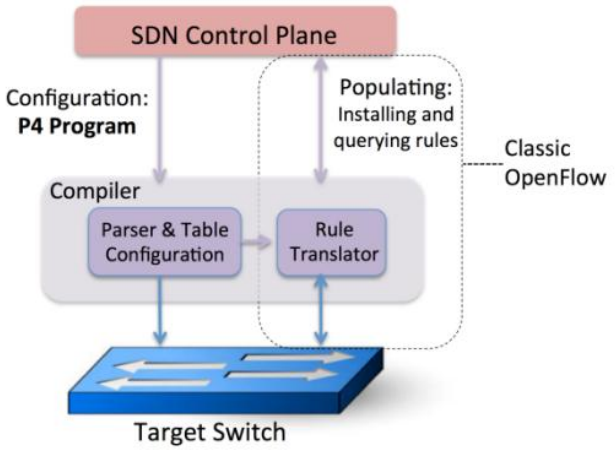

Figura 4. Atuação da Linguagem P4 nas Máquinas Whitebox. Fonte:[Bosshart et al. 2014]

Todas as máquinas Whitebox contém um controlador RC, que tem função semelhante ao controlador as máquinas NetFPGAs, ele permite que o experimentador faça *upload* de um arquivo com extensão P4[9]. A linguagem de programação P4 permite controlar o *dataplane* do *switch* virtual contido na máquina Whitebox. Desta forma, o experimentador pode programar e controlar o *pipeline* do OVS.

5. Propostas de Experimentos para Ensino

A abordagem teórica em disciplinas relacionadas a hardware [Dias 2013] e redes de computadores possibilita aos estudantes uma noção geral e superficial dos conceitos apresentados. A *testbed* FIBRE proporciona para educadores a possibilidade de realizar demonstrações práticas de como os sistemas de rede operam. O código abaixo é um exemplo de um experimento, onde o usuário realiza um experimento de roteamento utilizando placas NetFPGAs. Com poucas linhas de código é possível instanciar um experimento que pode ser facilmente replicado e demonstrado em ambientes acadêmicos.

Algoritmo 1: Trecho de Código para Experimento com Roteamento Usando Máquina NetFPGA.

```
1  início
2      init (netfpga3 - netfpga9)
3      set vlan 43 [netfpga3 - netfpga9];
4      connect [netfpga3 on netfpga4 (nf2c0)] and [netfpga9 (nf2c1) on netfpga8];
5      (...)
6      reprogram [netfpga4 - netfpga8] ← router.bit;
7      reprogram [netfpga3, netfpga9] ← traffic_gen.bit;
8      configure router [netfpga4 - netfpga8] ← config.c;
9      (...)
10     configure interface from netfpga3 [nf2c0 10.0.0.1/24];
11     configure interface from netfpga9 [nf2c0 200.0.0.2/24];
12     (...)
13     start iperf-server in [netfpga9] parameters –server -U
14     start iperf-client in [netfpga3] parameters -u -m 1000 to 200.0.0.2
15     (...)
16     result = (log_iperf_.txt + dump_reg_netfpga.txt);
17 fim
18 retorna result;
```

O processo experimental começa iniciando as máquinas *netfpga3* até *netfpga9*, depois essas máquinas são configuradas na mesma VLAN *(*Virtual Local Area Network), a topologia lógica pode ser observada na Figura 5. O próximo passo é a reprogramação das NetFPGAs, quando o usuário faz o *upload* o arquivo (.bit), que neste caso fará com que a placa opere como um roteador. O arquivo **config.c** tem as configurações de roteamento que serão enviados para todos os roteados da topologia.

A seguir são configuradas as interfaces de rede das máquinas (cliente e servidor de tráfego), após essas configurações um servidor gerador e consumidor de tráfego são instanciados nas máquinas que ficam nos extremos da topologia.

[9]https://p4.org

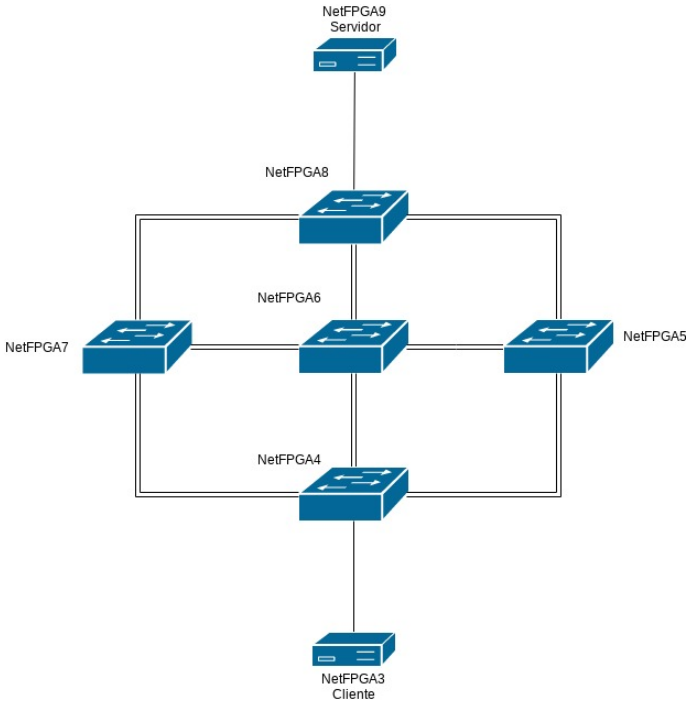

Figura 5. Topologia Lógica do Experimento com Máquinas FPGAs.

Ao final do experimento o usuário recebe um arquivo com os *logs* do servidor e do cliente, juntamente com os dados dos registradores dos máquinas NetFPGAs que operaram como roteadores.

6. Conclusões e Trabalhos Futuros

Este artigo propôs um experimento para o ensino prático de arquitetura e redes de computadores. Os conceitos apresentados nesse trabalho podem permitir que pesquisadores e alunos tenham um ambiente flexível e de fácil provisionamento, permitindo testes, validações e demonstrações. O uso de *testbeds* para educação pode contribuir para a capacitação e especialização de educadores e alunos, visto que promove o desenvolvimento prático de projetos relacionados a diversas áreas da computação. O diferencial da *testbed* FIBRE é o ganho de tempo construção do experimento, e o aumento expressividade, por proporcionar aos usuários experimentos com reconfiguração de hardware e controle do plano de dados. Como trabalho futuro, pretende-se propor experimentos que controlem o plano de dados, em conjunto com os demais projetos de referência existentes para placas NetFPGA.

Referências

Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I. (2014). Geni: A federated testbed for innovative network experiments. *Computer Networks*, 61:5 – 23. Special issue on Future Internet Testbeds – Part I.

Bilar, Guilherme, M. R. M. C. (2016). Linguagem de domínio especifico para geração de firewalls modulares em hardware. *1º Workshop de Teses e Dissertações - Departamento de Computação UFSCAR*, pages 1–2.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., et al. (2014). P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95.

Dias, M. A. (2013). Microcomputador re-configurável em fpga para ensino de arquitetura de computadores na ciência da computação. *RENOTE*, 11(3).

Gavras, A., Karila, A., Fdida, S., May, M., and Potts, M. (2007). Future internet research and experimentation: the fire initiative. *ACM SIGCOMM Computer Communication Review*, 37(3):89–92.

Goulart, P., Cunha, Í., Vieira, M., Marcondes, C., and Menotti, R. (2015). Netfpga: Processamento de pacotes em hardware. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2015*.

Kamienski, C., Mariz, D., Sadok, D., and Fernandes, S. (2005). Arquiteturas de rede para a próxima geraçao da internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2005*, 9(3):1–50.

Kamienski, C. A. and Sadok, D. (2000). Qualidade de serviço na internet. *Minicursos do Simpósio Brasileiro de Redes de Computadores, SBRC, Belo Horizonte/MG*, pages 1–5.

Kazaz, T., Liu, W., Jiao, X., Moerman, I., Seskar, I., Paisana, F., Vermeulen, T., Pollin, S., Felber, C., Kotzsch, V., et al. (2017). Orchestration and reconfiguration control architecture. In *EuCNC 2017*, pages 9–11.

Khan, S., Aziz, B., Najeeb, S., Aziz, A., Muhammad, U., and Ullah, S. (2013). Reliability of network simulators and simulation based research. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, pages 180–185.

Manggala, A. W., Tanwidjaja, A., et al. (2015). Performance analysis of white box switch on software defined networking using open vswitch. In *Wireless and Telematics (ICWT), 2015 1st International Conference on*, pages 1–7. IEEE.

Marcondes, C. A. C., Martins, J., Monteiro, J. A. S., Abelém, A. J. G., Nascimento, V., Machado, I., Salvador, M., and Rothenberg, C. E. (2012). Estado da arte de sistemas de controle e monito-ramento de infraestruturas para experimentação de redes de comunicação. *Minicursos do SBRC. SBC, Ouro Preto, MG*, pages 3–7.

Mikroyannidis, A., Collins, D., Tranoris, C., Denazis, S., Pareit, D., Vanhie, J., Moerman, I., Jourjon, G., Fourmaux, O., Dominigue, J., and Marquez-Barja, J. (2016). Forge : an elearning framework for remote laboratory experimentation on fire testbed infrastructure. In *Building the future internet through FIRE*, pages 521–559.

Moreira, M. D., Fernandes, N. C., Costa, L., and Duarte, O. (2009). Internet do futuro: Um novo horizonte. *Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC*, 2009:1–59.

Saltzer, J. H., Reed, D. P., and Clark, D. D. (1984). End-to-end arguments in system design. *ACM Transactions on Computer Systems (TOCS)*, 2(4):277–288.