Criação de fluxos em equipamentos ópticos e *switches* programáveis através de aplicações *northbound* e controlador SDN *open source* utilizando o Laboratório SDN Multicamadas

Luciano Martins[1], Niudomar S. A. Chaves[1], Fernando N. N. Farias[2], Marcos Schwarz[2], Tiago Sutili[1], Rafael C. Figueiredo[1]

[1]Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD) – Campinas, SP – Brasil

[2]Rede Nacional de Pesquisa (RNP) – Campinas, SP – Brasil

{lmartins, nchaves, tsutili, rafaelcf}@cpqd.com.br

{fernando.farias, marcos.schwarz}@rnp.br

Abstract. *This paper describes the tests for creating flows in programmable equipment available at the SDN Multilayer Laboratory – RNP (Rede Nacional de Ensino e Pesquisa), in which they used the open source SDN controller from ONF, the ONOS, to receive calls via open REST protocol for northbound applications and, through these calls, to program white boxes equipments as transponders and switches. This paper also highlights the relevance of the Multilayer SDN laboratory as a Brazilian testbed for experimentation in technologies such as Software Defined Network (SDN), disaggregation, control and orchestration of networks, programmability and standardized open interfaces.*

Resumo. *Este artigo descreve testes de criação de fluxos nos equipamentos programáveis disponibilizados no Laboratório SDN Multicamadas da RNP (Rede Nacional de Ensino e Pesquisa), os quais utilizaram o controlador SDN open source da ONF, o ONOS, para receber chamadas via protocolo aberto REST de aplicações northbound e, através destas chamadas, realizar a programação de equipamentos white boxes, transponders e switches. Este artigo também ressalta a relevância do laboratório SDN Multicamadas como testbed nacional para experimentação de tecnologias de redes definidas por software (Software Defined Network – SDN), desagregação, controle e orquestração de redes, programabilidade e interfaces abertas padronizadas.*

1. Introdução

Nos últimos anos têm ocorrido grandes esforços de instituições, como a ONF[1] (*Open Networking Foundation*) e o TIP[2] (*Telecom Infra Project*) para a desagregação de equipamentos em redes ópticas, que promove a migração de um modelo verticalizado, com soluções proprietárias, para um modelo que utiliza modelos de dados comuns e padrões abertos para as interfaces, facilitando a gerência e manutenção e, consequentemente, diminuindo os custos de OPEX (*Operational Expenditure*) e CAPEX (*Capital Expenditure*).

[1]Organização ONF – https://opennetworking.org/

[2]Organização TIP – https://telecominfraproject.com/

Além da eliminação da dependência de um único fornecedor com soluções e APIs (*Application Programming Interfaces*) proprietárias como ocorre no modelo verticalizado, a desagregação em equipamentos e funções de rede permite a introdução de novos serviços, agilidade na implantação de inovações, além de utilizar um modelo de controle de redes centralizado, conhecido como controle de Redes Definidas por Software (*Software-Defined Network - SDN*) [Kreutz et al. 2015], por meio do qual um único controlador executa funcionalidades de monitoração e configuração de elementos de forma coordenada, uma vez que o mesmo tem a visibilidade de toda a rede.

Dado tal cenário, é de grande importância haver *testbeds* disponíveis para o meio científico e industrial, para que soluções que englobam conceitos de orquestração, desagregação, SDN, programabilidade, controle e padrões abertos (*open source*), sejam experimentadas e desenvolvidas, sem trazer ônus a ambientes de produção. Como exemplo de *testbed* disponível para tais testes, pode-se destacar o projeto COSMOS[3] (*Cloud Enhanced Open Software Defined Mobile Wireless testbed for City-Scale Deployment*) implantado em Nova Iorque - EUA, e também sua evolução, o COSM-IC[4] (*COSMOS Interconnecting Continents*) que tem como objetivo interconectar *testbeds* em diferentes países, incluindo o Brasil, através de parceria com o CPQD. Já nacionalmente, a RNP disponibiliza o *testbed* Laboratório SDN Multicamadas, localizado nas dependências do POP-RJ no Rio de Janeiro, e implantado com tecnologias que permitem experimentos e desenvolvimentos nos conceitos supracitados.

O objetivo deste trabalho é apresentar os testes e os resultados de experimentos que o CPQD realizou no Laboratório SDN Multicamadas da RNP, para a criação de fluxos[5] em equipamentos programáveis, através da utilização de um controlador SDN (em específico, foi adotado o ONOS[6] da ONF). Tal controlador foi implementado e instalado nas dependências do CPQD em Campinas e recebe chamadas e comandos (GET, POST, DELETE) via protocolo aberto REST [Bierman et al. 2017] através de *scripts*, realizando a programação dos equipamentos *transponders* e *switches* via gRPC e protocolo P4[7]. Através destes testes, o CPQD, com apoio da RNP, validou a infraestrutura do Laboratório SDN Multicamadas como *testbed* nacional de alto valor agregado para a academia, a indústria e os provedores de serviços, os quais poderão se beneficiar de seu ambiente de equipamentos *white boxes* e de alta velocidade (100 Gbps).

Este artigo está organizado da seguinte forma: a Seção 2 apresenta informações sobre o *testbed* SDN Multicamadas, além da topologia do laboratório e da conexão realizada entre CPQD e o *testbed*. Na Seção 3, é apresentada a arquitetura de camadas utilizada para os testes realizados pelo CPQD na infraestrutura do *testbed*, e a Seção 4 apresenta a metodologia dos testes realizados para a criação de fluxos nos equipamentos do *testbed*. Na Seção 5, são apresentados os resultados da criação de fluxos usando *scripts* para as chamadas REST ao controlador ONOS, bem como das solicitações e programação dos equipamentos *switches* e *transponders* usando P4 e gRPC. Por fim, a Seção 6 apresenta as conclusões e possíveis trabalhos futuros.

---

[3]COSMOS *testbed* – https://www.cosmos-lab.org

[4]Projeto COSM-IC – https://www.ccny.cuny.edu/cint/global-internet-testbed-cosmic

[5]Entradas em tabelas contendo porta de entrada, porta de saída, tipo de *header* e ações de processamento

[6]ONOS - https://opennetworking.org/onos/

[7]P4 - https://p4.org/

2. *Testbed* SDN Multicamadas

Em [Nassif et al. 2021] foi apresentada em detalhes a plataforma SDN Multicamadas, provida e gerenciada pela RNP, que tem como objetivo estudar e desenvolver uma solução de orquestração multicamadas (*i.e.*, utilizando as camadas de pacote e óptica) através do conceito de redes definidas por *software*. Para tanto, é disponibilizado um ambiente laboratorial incluindo as camadas óptica e L2/L3, além de um sistema de emulação de redes com as tecnologias empregadas no *testbed*, de modo a permitir a sua experimentação remota pela comunidade acadêmica e industrial.

Para tanto, o *testbed* emprega soluções de DTN (*Data Transfer Node*) com capacidade de transferência de até 100 Gbps, contando também com um sistema de orquestração de modo a permitir a integração e automação de serviços e ferramentas e, consequentemente, possibilitando a sua interação com a infraestrutura existente e o ambiente de aplicativos. Em específico, a topologia proposta é composta por dois sistemas finais DTNs, dois *switches white box* (L2/L3), dois *transponders* (L1) programáveis e um *switch white box* funcionando como *patch panel* programável, além dos equipamentos de gerenciamento da plataforma. A lista dos principais equipamentos é a seguinte:

1. 1× EdgeCore Wedge 100BF-32X 10G (*white box* com a função de *patch panel*);
2. 2× Edgecore AS7716-24SC WX (*transponder* SDN);
3. 2× Stordis BF2556X-1T (*switches Stratum*);
4. 2× SuperMicro SYS-1029U-E1CTRP2 100G (servidores DTN);
5. 2× SuperMicro SYS-2029GP-TR 100G (servidores DTN).

A Figura 1 apresenta a topologia do laboratório SDN Multicamadas, com seus equipamentos e interligações.

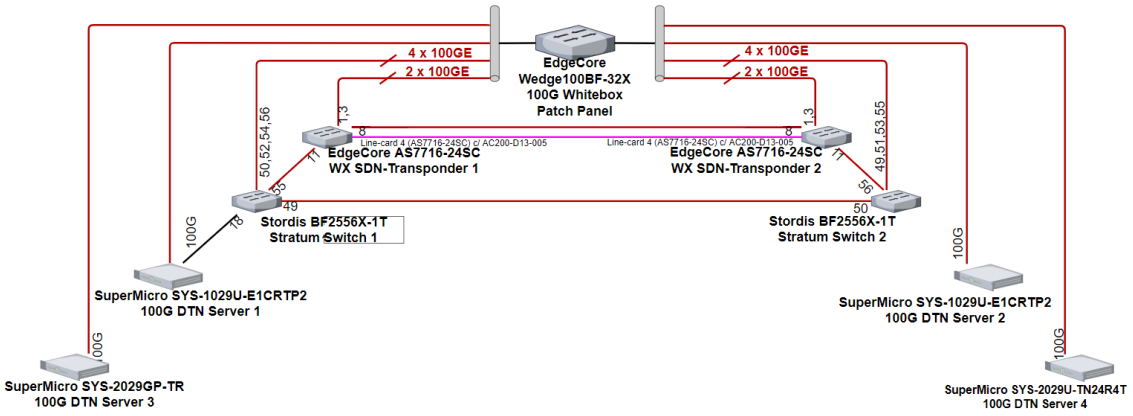


Figura 1. Equipamentos e conexões implementadas no Lab. SDN Multicamadas.

Tando os servidores (SuperMicro DTN Servers 1, 2, 3 e 4), quanto os switches (Stordis Stratum 1 e 2) e *transponders* (Edgecore Cassinis 1 e 2) estão conectados fisicamente a um switch Edgecore Wedge que funciona como um patch panel programável, o que permite conexões diversas através de configuração. O Switch 1 está conectado ao Transponder 1 diretamente via cabeamento DAC 100G, bem como o Transponder 1 se conecta ao Transponder 2, e o Transponder 2 ao Switch 2 com o mesmo tipo de cabeamento.

A visão da topologia lógica com os equipamentos alocados para o CPQD e os identificadores numéricos das interfaces reconhecidas pelo controlador ONOS é apresentada na Figura 2.

DTN3		SW1				TP1				TP2				SW2				DTN4
ens3np0.1925	"----"	68	<->	8	"----"	126	<->	114	"----"	114	<->	126	"----"	16	<->	68	"----"	ens1.2026

Figura 2. Topologia lógica e fluxos criados na topologia.

2.1. Conexão do CPQD ao *testbed* SDN Multicamadas

Para viabilizar os testes de validação entre o ambiente emulado e o ambiente real, foi empregada a topologia apresentada na Figura 3 para interconexão entre o ambiente de desenvolvimento no CPQD e o *testbed* SDN multicamadas, processo que também demandou os seguintes passos de implementação:

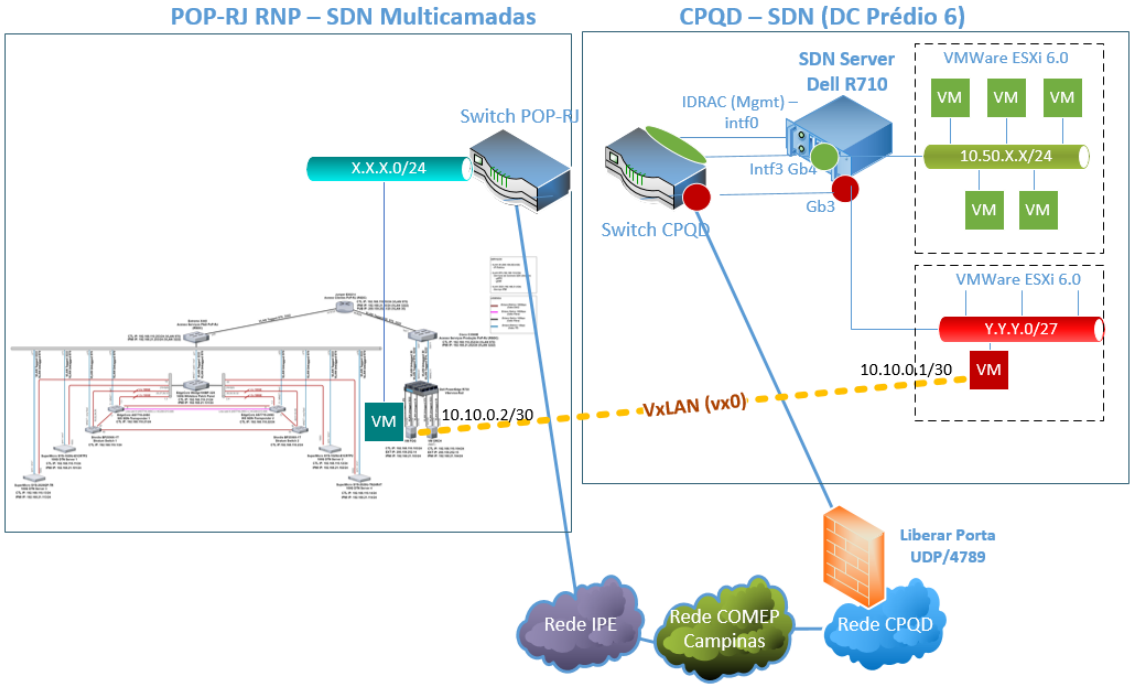


Figura 3. Topologia de interconexão entre o CPQD e o *testbed* SDN Multicamadas.

1. Criação de um túnel VXLAN[8] entre um servidor da RNP e um servidor do CPQD, permitindo conectividade do CPQD aos equipamentos do laboratório SDN Multicamadas;
2. Testes de conectividade entre os 2 pontos;
3. Testes de acesso SSH com usuário e senha fornecidos pela RNP aos equipamentos do laboratório SDN Multicamadas.

[8]VXLAN - protocolo de encapsulamento que fornece a conectividade de data centers usando o tunelamento para estender as conexões de Camada 2 em uma rede de Camada 3 subjacente. O uso da VXLAN foi uma opção para esta conexão em específico entre CPQD e o *testbed*, mas outras soluções podem ser possíveis, de acordo com a análise dos profissionais da RNP e dos recursos existentes em cada instituição que deseja se conectar.

2.2. Vantagens e desvantagens do *testbed* SDN Multicamadas

A seguir são descritas as principais vantagens do ambiente físico disponibilizado pelo laboratório SDN Multicamadas:

- Ambiente com *hardware* real e sistema operacional *Stratum*, onde se pode realizar testes com protocolos de padrão aberto como Netconf, P4, gNMI, gRPC e gNOI;
- Possibilidade de testes de desempenho entre equipamentos, podendo-se emular diferentes distâncias de propagação através da adição de carretéis de fibra óptica;
- Uso das funcionalidades reais dos equipamentos, em que pode-se averiguar a maturidade do desenvolvimento pelos fabricantes nas soluções abertas para programabilidade.

Quanto às desvantagens do uso do ambiente físico, pode-se pontuar a necessidade de interação com técnicos do POP-RJ e RNP caso haja algum problema, como, por exemplo, falta de energia, indisponibilidade dos equipamentos ou outro tipo de problema de acesso aos elementos de forma remota.

3. Arquitetura em camadas para os testes e meios de acesso às camadas

A arquitetura de camadas usada nos testes é mostrada na Figura 4, que apresenta:

1. **[1] Equipamentos programáveis**: nesta camada estão os *transponders* (Cassinis) e *switches* programáveis (Stordis) que são acessados via SSH, programados via P4 e gerenciados via gRPC/gNOI;
2. **[2] Controlador SDN**: nesta camada está o controlador *open source* da ONF (ONOS), juntamente com os modelos YANG do grupo OpenConfig que detalham as variáveis a serem gerenciadas nos elementos programáveis da camada [1];
3. **[3] Aplicações/*Scripts***: nesta camada estão implementados alguns *scripts* programados em Shell Script e em Python para automatização dos testes propostos, na qual são realizadas chamadas via RESTCONF para o ONOS para o levantamento da topologia e das portas e para realizar a solicitação de criação de fluxos (entradas nas tabelas) dos equipamentos.

Empregando a estrutura acima detalhada, foram feitos testes para exercitar a resposta dos elementos de cada camada nos principais protocolos indicados na Figura 4, através das seguintes ferramentas de acesso:

1. **Acesso via SSH em terminal do Linux:** através de SSH, é possível acessar o sistema Stratum dos equipamentos para verificar o sistema de arquivos e comandos específicos dos *transponders* e *switches*;
2. **Acesso usando p4rt-cli**[9] **e gnmi-cli**[10] **em terminal do Linux:** estas ferramentas permitem que chamadas diretas de criação e verificação dos equipamentos sejam realizadas empregando protocolo P4 e gNMI sem demandar um controlador SDN para interação com os equipamentos;
3. **Acesso ao controlador ONOS usando curl**[11] **e ONOS GUI:** o acesso às interfaces *northbound* providas pelo controlador ONOS é feito pelo protocolo RESTCONF, no qual operações "GET", "POST" e "DELETE" possibilitam a obtenção

[9] p4runtime-shell - https://github.com/p4lang/p4runtime-shell
[10] gNMI-CLI - https://github.com/openconfig/gnmi
[11] curl - https://curl.se/

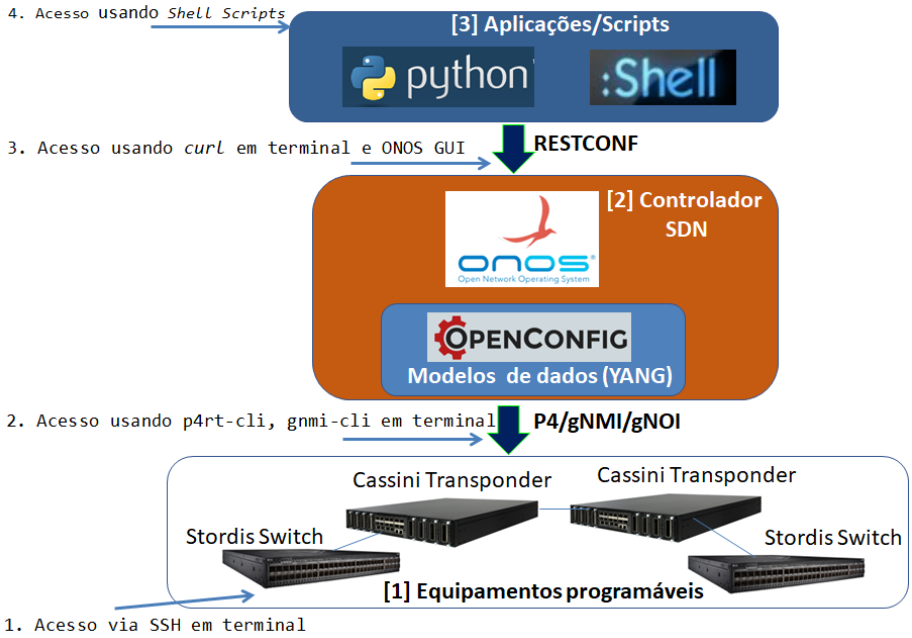


Figura 4. Arquitetura de camadas empregada nos testes propostos.

de informações, a configuração dos elementos e, até mesmo, a exclusão de parâmetros da base de dados do ONOS e, consequentemente, do equipamento programável. Em específico, arquivos em formato JSON são utilizados para que as operações sejam executadas e ferramentas como curl e, até mesmo, a GUI do controlador ONOS permitam realizar tais operações;

4. **Acesso usando *scripts*:** para que as operações indicadas anteriormente sejam automatizadas e decisões de configuração sejam tomadas, o CPQD desenvolveu *scripts* em Shell Script e em Python. Ressalta-se que tais chamadas podem também ser implementadas utilizando outras aplicações ou outra linguagem de programação, ou, até mesmo, sistemas como OSS/BSS e orquestradores.

O fluxograma do *script* principal desenvolvido pode ser visualizado na Figura 5:

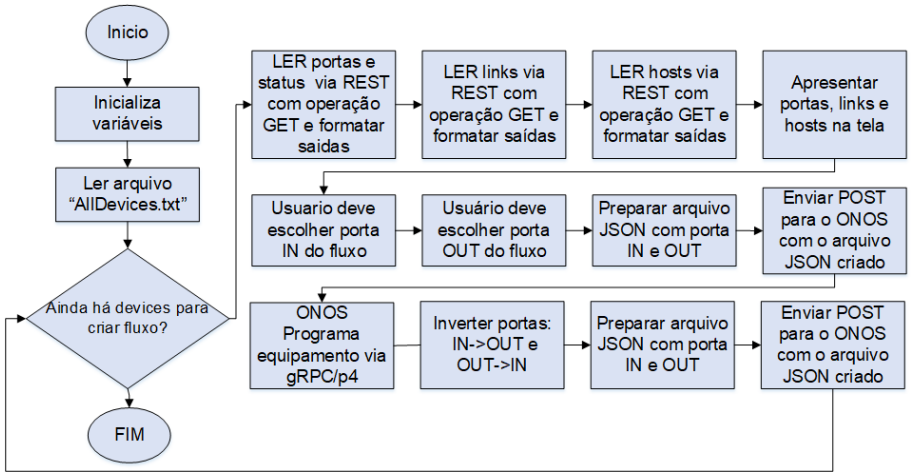


Figura 5. Fluxograma do *script* desenvolvido pelo CPQD para gerar fluxos

4. Metodologia para os testes de criação de fluxos nos equipamentos

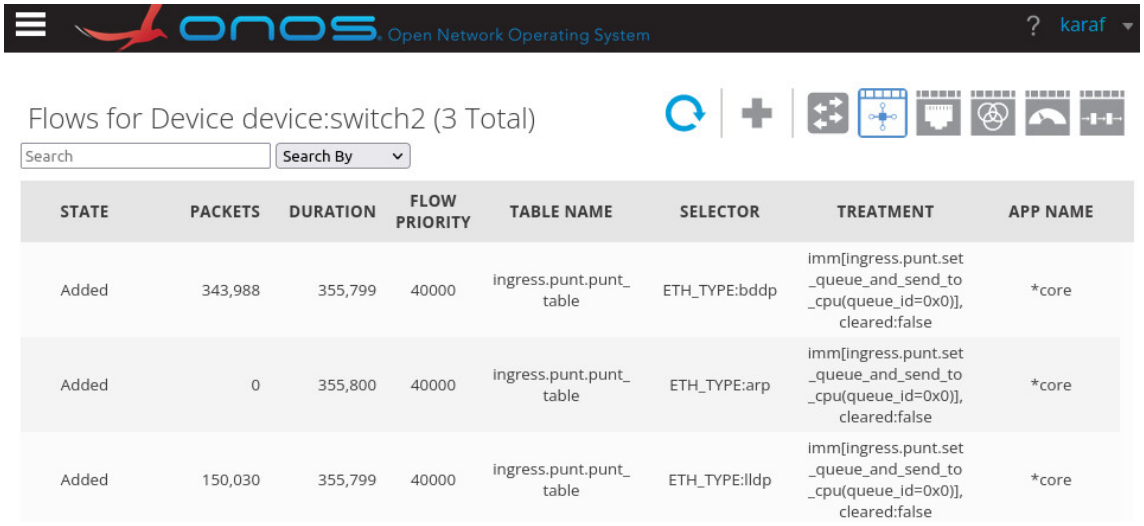
A metodologia utilizada para os testes de criação de fluxos na infraestrutura do Laboratório SDN Multicamadas é descrita a seguir:

1. Acesso à CLI do P4 Runtime (p4RT) dos equipamentos Cassini e Stordis;
2. Verificação dos fluxos pré-existentes nas tabelas dos equipamentos, executando comandos dentro da CLI do p4RT;
3. Acesso à ONOS Core REST API pela GUI do ONOS na URL `http://[IPServerONOS]:8181/onos/v1/docs/#/`;
4. Verificação dos fluxos pré-existentes nas tabelas dos equipamentos através da GUI do ONOS;
5. Criação de fluxos no ONOS através da GUI usando REST;
6. Criação de fluxos no ONOS através de chamadas REST em Shell Script;
7. Verificação da criação de fluxos através da GUI do ONOS e via operação GET através da ONOS Core REST API;
8. Checagem da criação de fluxos nos equipamentos através da CLI do P4 Runtime;
9. Teste de conectividade entre os servidores DTN após a criação de fluxos.

5. Resultados

5.1. Acesso à CLI do P4 Runtime dos equipamentos e verificação dos fluxos pré-existentes nas tabelas usando o p4rt-cli e a GUI do ONOS

Tanto os *switches* quanto os *transponders* do laboratório SDN Multicamadas possuem uma tabela ternária nomeada *ingress.punt.punt_table*, na qual são listadas as entradas para determinados tipo de quadros, como, por exemplo, Ethernet (*hdr.ethernet.ether_type*), incluindo também ações para processamento dos dados, como, por exemplo, configurar a fila e enviar o quadro para a CPU (*ingress.punt.set_queue_and_send_to_cpu*). As entradas (fluxos) iniciais dos equipamentos podem ser vistas na GUI do ONOS, através da seleção de um dos dispositivos, no caso o *device:switch2*, e clicando no símbolo referente aos fluxos, conforme apresentado na Figura 6.



STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	343,988	355,799	40000	ingress.punt.punt_table	ETH_TYPE:bddp	imm[ingress.punt.set_queue_and_send_to_cpu(queue_id=0x0)], cleared:false	*core
Added	0	355,800	40000	ingress.punt.punt_table	ETH_TYPE:arp	imm[ingress.punt.set_queue_and_send_to_cpu(queue_id=0x0)], cleared:false	*core
Added	150,030	355,799	40000	ingress.punt.punt_table	ETH_TYPE:lldp	imm[ingress.punt.set_queue_and_send_to_cpu(queue_id=0x0)], cleared:false	*core

Figura 6. Fluxos iniciais verificados no ONOS - exemplo Stordis 2

Adicionalmente, tais fluxos podem também ser visualizado através da execução do *script* “p4rt-cli”, presente no tutorial avançado do ODTN[12], conforme apresentado na Figura 7.

```
#source start-p4rt.sh 192.168.110.2 28000
*** Connecting to P4Runtime server at 192.168.110.2:28000 ...
*** Welcome to the IPython shell for P4Runtime ***
P4Runtime sh >>> te = table_entry["ingress.punt.punt_table"]
           ...: te.read(lambda x: print(x))
%%%%%%%%%%%%%%%%%%%%
table_id: 33598026 ("ingress.punt.punt_table")
match {  field_id: 3 ("hdr.ethernet.ether_type")
        ternary {    value: "\x08\\x06"    mask: "\\xff\\xff"  }}
action {  action {
            action_id: 16804007 ("ingress.punt.set_queue_and_send_to_cpu")
            params { param_id: 1 ("None")      value: "\\x00" }}}
priority: 40001
%%%%%%%%%%%%%%%%%%%%
table_id: 33598026 ("ingress.punt.punt_table")
match {  field_id: 3 ("hdr.ethernet.ether_type")
        ternary {    value: "\x88\\xcc"    mask: "\\xff\\xff"  }}
action {  action {
            action_id: 16804007 ("ingress.punt.set_queue_and_send_to_cpu")
            params { param_id: 1 ("None")      value: "\\x00" }}}
priority: 40001
%%%%%%%%%%%%%%%%%%%%
table_id: 33598026 ("ingress.punt.punt_table")
match {  field_id: 3 ("hdr.ethernet.ether_type")
        ternary {    value: "\\x89\\x42"    mask: "\\xff\\xff"  }}
action {  action {
            action_id: 16804007 ("ingress.punt.set_queue_and_send_to_cpu")
            params { param_id: 1 ("None")      value: "\\x00" }}}
priority: 40001
%%%%%%%%%%%%%%%%%%%%
```

Figura 7. Fluxos iniciais verificados com p4rt-cli - exemplo Stordis 2

5.2. Adição de fluxo via REST no ONOS e no switch P4

Dando prosseguimento aos testes propostos, foi desenvolvido um *script* em Shell que, através da operação REST “Get”, recupera informações dos equipamentos como componentes e interfaces. Depois disto, através da visualização das informações no *prompt* do Linux, o usuário pode escolher as portas de entrada dos equipamentos e as portas de saída desejadas, permitindo que o *script* crie os arquivos JSON para que a operação REST “Post” envie as configurações ao controlador ONOS para a criação dos fluxos. No caso específico da implementação aqui descrita, na Figura 2 foi apresentada a topologia com os IDs das portas reconhecidas pelo ONOS, na qual o “device:switch2” (SW2) é conectado nas portas 16 e 68 em modo bidirecional (*i.e.*, “in:16/out:68” e “in:68/out:16”), conforme exemplificado na tela do *prompt* do Linux apresentada na Figura 8.

[12]NG-SDN Tutorial Advanced - https://github.com/opennetworkinglab/ngsdn-tutorial

```
1) Entre com a(s) porta(s) de ENTRADA do device:switch2 (separar por ESPAÇO):
68
Voce digitou 1 portas IN: [68]

2) Entre com a(s) porta(s) de SAIDA do device:switch2 (separar por ESPAÇO):
16
Voce digitou 1 portas OUT: [16]

CRIACAO DE FLUXOS PARA in=[68] out=[16 0x10]
==============
{"flows":[{"deviceId":"device:switch2","flowId":"54324671746344273"}]}
CRIACAO DE FLUXOS PARA out=[68 0x44] in=[16]
%%%%%%%%%%%
{"flows":[{"deviceId":"device:switch2","flowId":"54324672512030549"}]}
```

Figura 8. Entrada de portas de entrada e saída em *script* e criação de fluxo bidirecional no SW2.

A Figura 9 apresenta um exemplo de arquivo JSON onde a porta de entrada é a 16 e a porta de saída é a porta 68 (44 em hexadecimal). Este arquivo é utilizado na chamada REST que é feita com o comando "curl" no Linux.

```
{"flows":
[  {
    "tableId": "ingress.punt.punt_table",
    "appId": "org.stratumproject.bcm-pipeconf",
    "groupId": 0,
    "priority": 0,
    "timeout": 0,
    "isPermanent": true,
    "deviceId": "device:switch2",
    "treatment": {
     "instructions": [
      { "type": "PROTOCOL_INDEPENDENT",
       "subtype": "ACTION",
       "actionId": "ingress.punt.set_egress_port",
       "actionParams": {
         "port": "44"     }
      }
     ],
     "deferred": []
    },
    "selector": {
     "criteria": [
      { "type": "IN_PORT",
        "port": "16"
      }
     ]
    }}]}
```

Figura 9. Arquivo JSON para adição de fluxo no *switch* 2 (SW2).

Com base em tal topologia e no arquivo gerado pelo *script*, o comando utilizado para a criação de cada fluxo emprega a sintaxe "*curl -sSL –user user:pass -X POST -H'Content-Type:application/json' http://ipServer:8181/onos/v1/flows -d@arquivo.json*". Por fim, é importante salientar que, para cada porta de entrada e de saída, o *script* cria um "arquivo.json", e são enviados 2 chamadas usando a sintaxe do comando acima, garantindo assim a criação do fluxo bidirecional.

O processo de criação de fluxos e de seleção das portas, no *script* que o CPQD desenvolveu, ocorre para cada um dos equipamentos mostrados na Figura 2 (*i.e.*, SW1, TP1, TP2 e SW2). Este *loop* pode ser observado no fluxograma da Figura 5.

5.3. Verificação dos fluxos adicionados

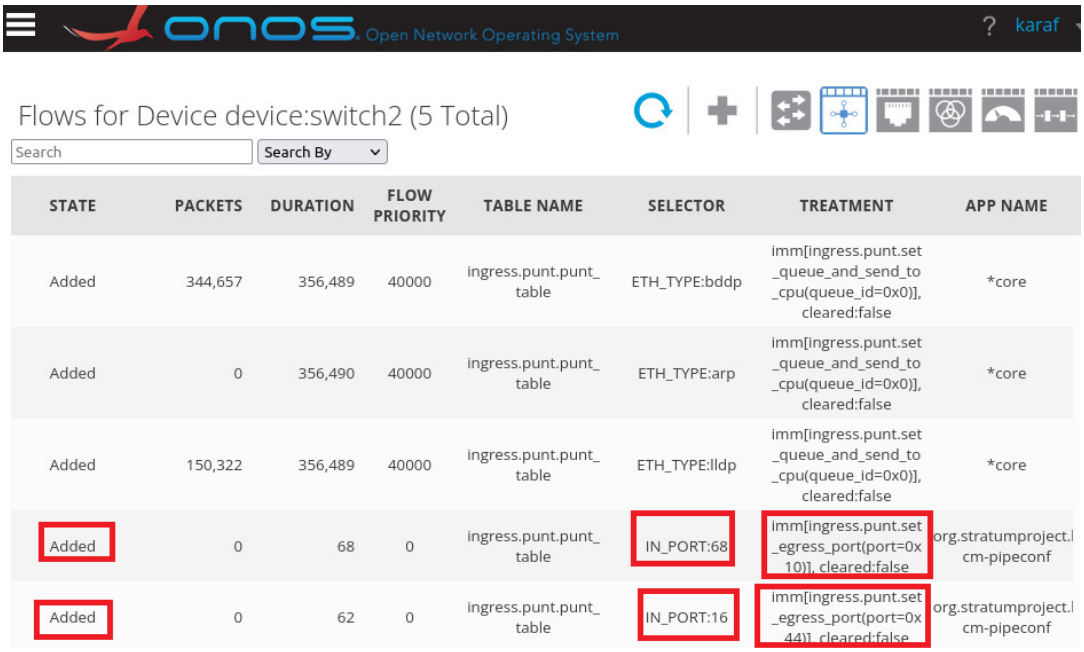
Por fim, os fluxos adicionados após o comando "Post" descrito na seção anterior podem ser visualizados usando o *script* "p4rt-cli", ou alternativamente via GUI do ONOS, conforme mostram as Figuras 10 e 11 respectivamente. Conforme pode ser verificado nas

figuras, os fluxos criados em todos os equipamentos e a conectividade entre os servidores DTN3 e DTN4 foi bem sucedida.

```
table_id: 33598026 ("ingress.punt.punt_table")
match { field_id: 1 ("standard_metadata.ingress_port")
        ternary { value: "\\x00\\x44" mask: "\\x01\\xff" }}
action { action {
           action_id: 16820507 ("ingress.punt.set_egress_port")
           params { param_id: 1 ("None") value: "\\x00\\x10" }}}
priority: 1
```

```
table_id: 33598026 ("ingress.punt.punt_table")
match { field_id: 1 ("standard_metadata.ingress_port")
        ternary { value: "\\x00\\x10" mask: "\\x01\\xff" }}
action { action {
           action_id: 16820507 ("ingress.punt.set_egress_port")
           params { param_id: 1 ("None") value: "\\x00\\x44" }}}
priority: 1
```

Figura 10. Visualização no p4rt-cli dos fluxos adicionados pelo ONOS depois das chamadas REST realizadas via *script*.



STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	344,657	356,489	40000	ingress.punt.punt_table	ETH_TYPE:bddp	imm[ingress.punt.set_queue_and_send_to_cpu(queue_id=0x0)], cleared:false	*core
Added	0	356,490	40000	ingress.punt.punt_table	ETH_TYPE:arp	imm[ingress.punt.set_queue_and_send_to_cpu(queue_id=0x0)], cleared:false	*core
Added	150,322	356,489	40000	ingress.punt.punt_table	ETH_TYPE:lldp	imm[ingress.punt.set_queue_and_send_to_cpu(queue_id=0x0)], cleared:false	*core
Added	0	68	0	ingress.punt.punt_table	IN_PORT:68	imm[ingress.punt.set_egress_port(port=0x10)], cleared:false	org.stratumproject.cm-pipeconf
Added	0	62	0	ingress.punt.punt_table	IN_PORT:16	imm[ingress.punt.set_egress_port(port=0x44)], cleared:false	org.stratumproject.cm-pipeconf

Figura 11. Visualização no GUI do ONOS dos fluxos adicionados pelo ONOS depois das chamadas REST realizadas via *script*.

A Figura 12 apresenta uma tentativa de teste de conectividade usando ”ping”entre DTN3 (10.0.0.3) e o DTN4 (10.0.0.4), antes de serem criados os fluxos nos switches e transponders. Pode-se observar a falha através da mensagem “Destination Host Unreachable”.

Após a criação de todos os fluxos, o teste de conectividade teve sucesso, o que pode ser observado na Figura 13.

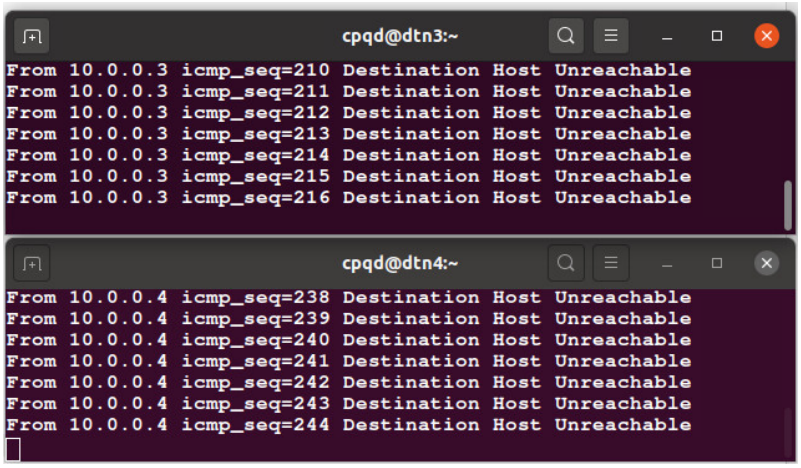


Figura 12. Falha na conectividade ANTES da criação dos fluxos

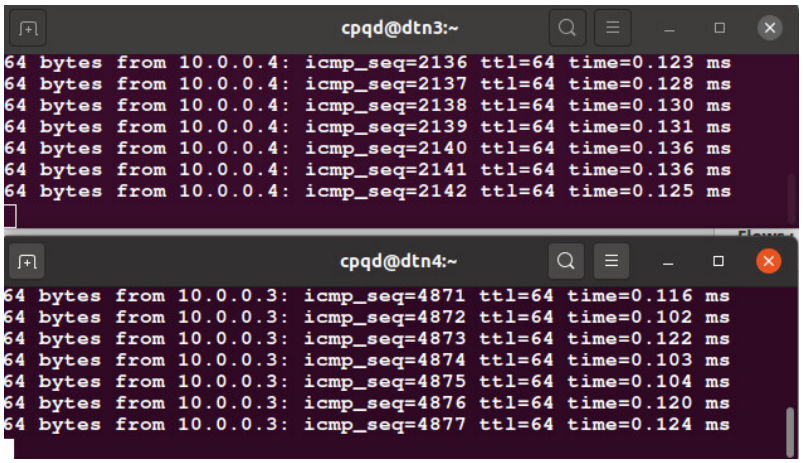


Figura 13. Sucesso na conectividade APÓS a criação dos fluxos

6. Conclusões e trabalhos futuros

Este artigo apresentou uma visão geral da topologia do laboratório SDN Multicamadas, bem como da interconexão realizada entre as instalações do CPQD e o *testbed*. Com base em tal estrutura, foi proposta uma metodologia de testes para a criação de fluxos entre os equipamentos, culminando nos resultados demonstrando o controle da infraestrutura do *testbed*, instalada no Rio de Janeiro, por um controlador ONOS implementado no CPQD em Campinas, São Paulo. Adicionalmente, através dos testes, o CPQD validou o ambiente do *testbed* para a programabilidade dos equipamentos com P4 e gRPC na interface *southbound* do controlador ONOS e o uso de APIs *northbound* abertas usando o REST em *scripts* na camada de aplicação.

Todas as ações, testes e desafios durante o tempo de execução das atividades permitiram ao CPQD:

1. Conhecer e aplicar a metodologia de acesso ao ambiente do *testbed* que a RNP proporciona para as instituições parceiras;
2. Configurar um sistema Linux para estabelecer uma VPN L2 usando VxLAN e superar obstáculos para a conectividade, tais como *firewall* e rotas faltantes;
3. Instalar e configurar o controlador ONOS em um sistema com *docker containers*;

4. Conhecer a estrutura de arquivos JSON para configurar o ONOS para aprender a topologia com dispositivos, *links* e fluxos;
5. Interagir com o sistema Stratum da ONF e com a CLI do P4 Runtime para configurar e verificar as tabelas e fluxos configurados;
6. Programar *scripts* na camada de aplicação para executar operações de "Get" e "Post" para interagir com a API REST *northbound* do ONOS e solicitar a criação de fluxos para o controlador.

Como propostas de melhorias para o *testbed*, para comportar experimentos mais complexos, podem ser enumeradas:

1. Instalação de equipamentos ópticos mais complexos em linha entre os *transponders* Cassini, tais como ROADM (*Reconfigurable Optical Add/Drop Multiplexer*) para testes de escolha do melhor caminho por exemplo;
2. Instalação de elementos de rede programáveis (*white boxes*) como ONU (*Optical Network Unit*) e ONT (*Optical Network Terminal*) para testes com Open FTTX (Fiber To The "X");
3. Instalação de *white boxes* para testes de OpenRAN (*Open Radio Access Networks*).

Para trabalhos futuros, o CPQD vislumbra como grande potencial do uso do *testbed* disponibilizado pela RNP, incluindo as seguintes atividades:

1. Desenvolvimento e testes de controle e orquestração em vários domínios tecnológicos, como 5G, FTTx, WDM e redes de dados;
2. Interação com outros *testbeds* internacionais;
3. Desenvolvimentos em sistemas OSS/BSS para o uso de padrões abertos e o conceito de desagregação.

Referências

Bierman, A., Björklund, M., and Watsen, K. (2017). RESTCONF Protocol. RFC 8040.

Kreutz, D., Ramos, F. M. V., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey - proceedings of the ieee. 103:14 – 76.

Nassif, N. A., Martins, L., Domingos, L. A., Farias, F. N. N., Lopes, J. L. S., Borges, L., Chaves, N. S. A., Schwarz, M., and Figueiredo, R. C. (2021). Aplicação Northbound baseada em Transport API (TAPI) usando Controlador ONOS e transponder configurável de padrão aberto para a criação de circuito óptico fim a fim.