Internet do Futuro e Programabilidade da Rede - uma visão prática de SDN/OpenFlow e P4

Italo Valcy S. Brito (UFBA e PoP-BA/RNP)
Adriana Viriato Ribeiro (UFBA e PoP-BA/RNP)

{italovalcy, adrianavr}@ufba.br

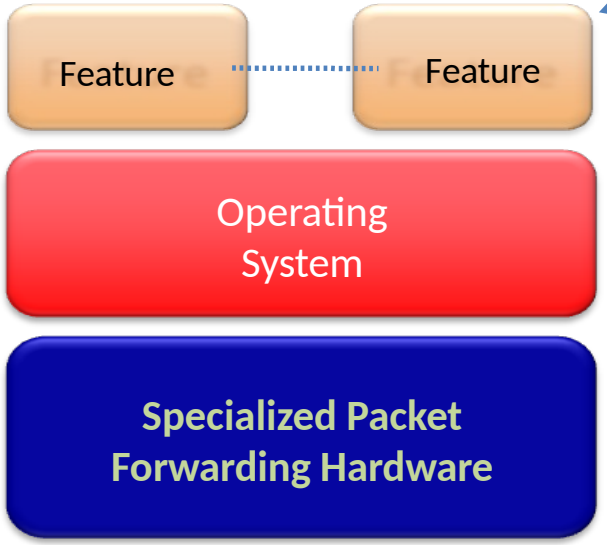
24 a 28 de Setembro de 2018

Agenda

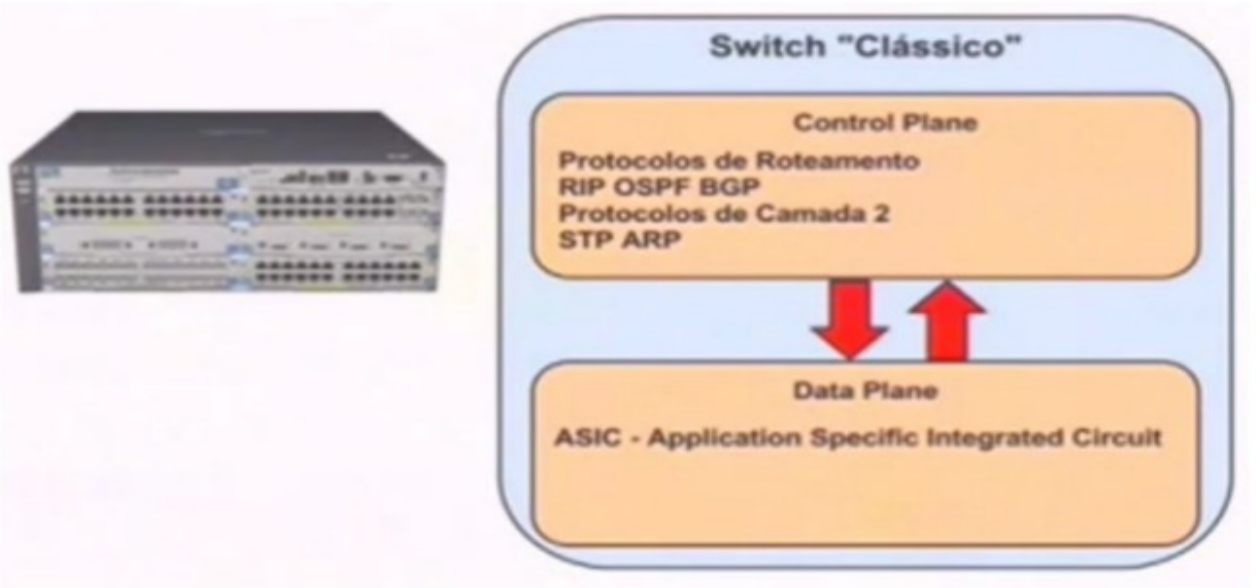
- Desafios para Internet do Futuro, programabilidade da rede
- Conceitos e arquitetura de SDN/OpenFlow, aplicações e APIs
- Ambiente de experimentação Mininet
- Desagregação de rede e *whiteboxes*
- Programabilidade do *Dataplane* com P4

A rede atual

Roteamento, gerencia, mobilidade, controle de acesso, VPN, ...



Verticalização
Fechado, proprietário
Inovação lenta

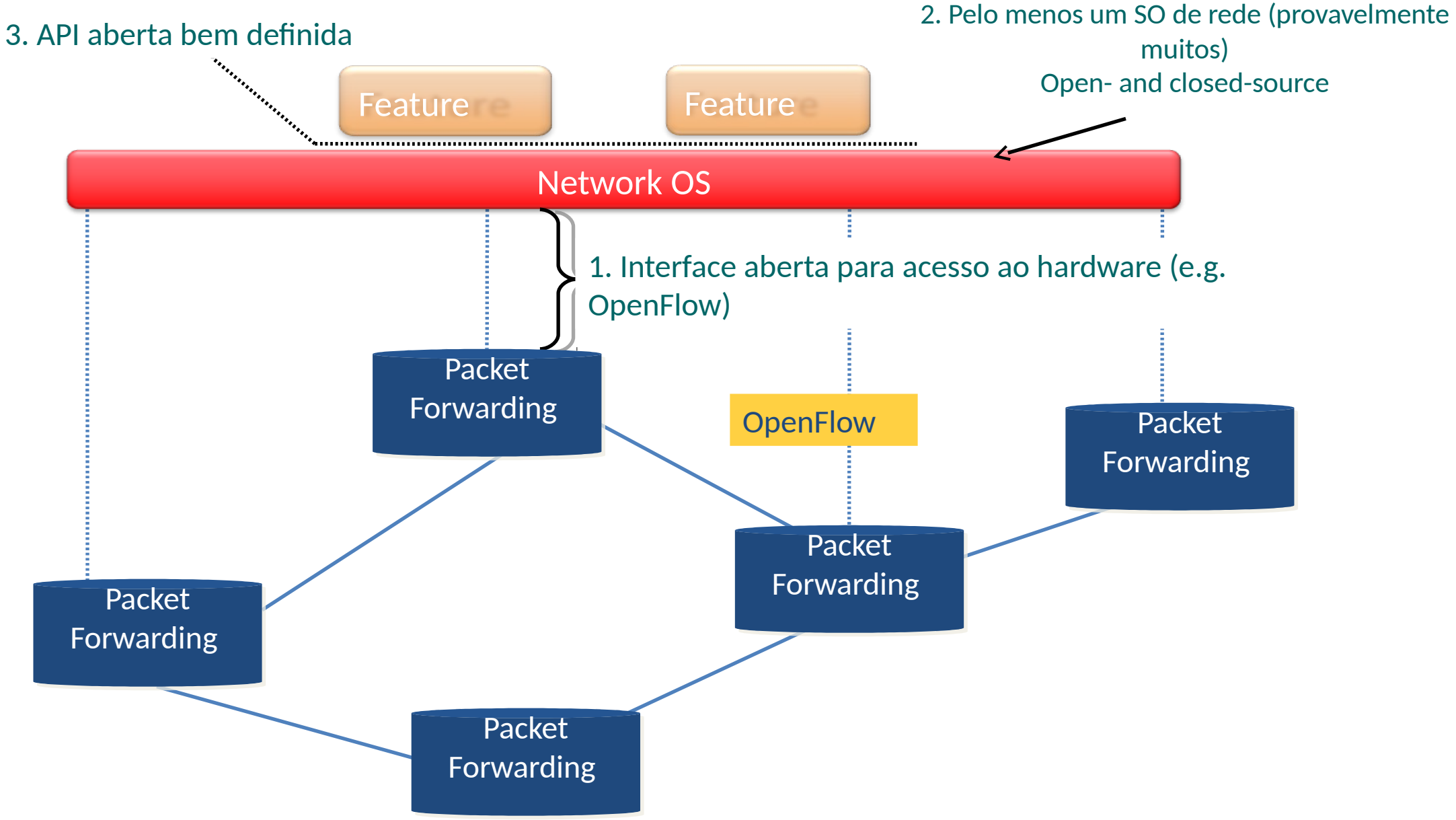


Programabilidade da Rede

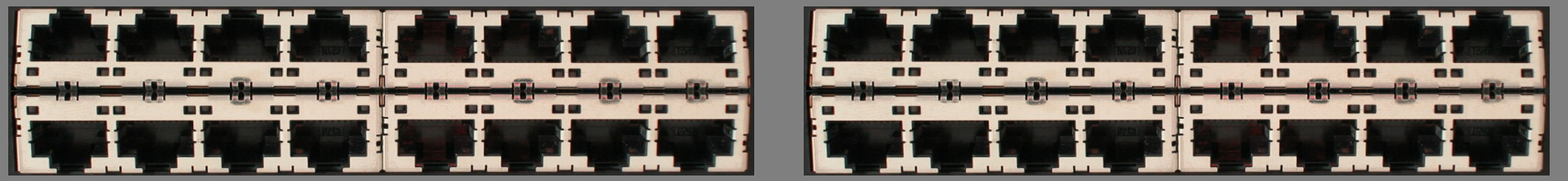
- Customização do funcionamento da rede em função das demandas das aplicações
  - Conteúdos nomeados, Eng. Tráfego, Segurança
- Gerenciamento logicamente centralizado
- Integração entre os dispositivos de rede e a infraestrutura de TI
  - DevOps, Integração Continua, Chef, Ansible
- Novas interfaces de interação com dispositivos de rede (ex: REST APIs, Asics, APIs, etc)

SDN – Rede Definida por Software

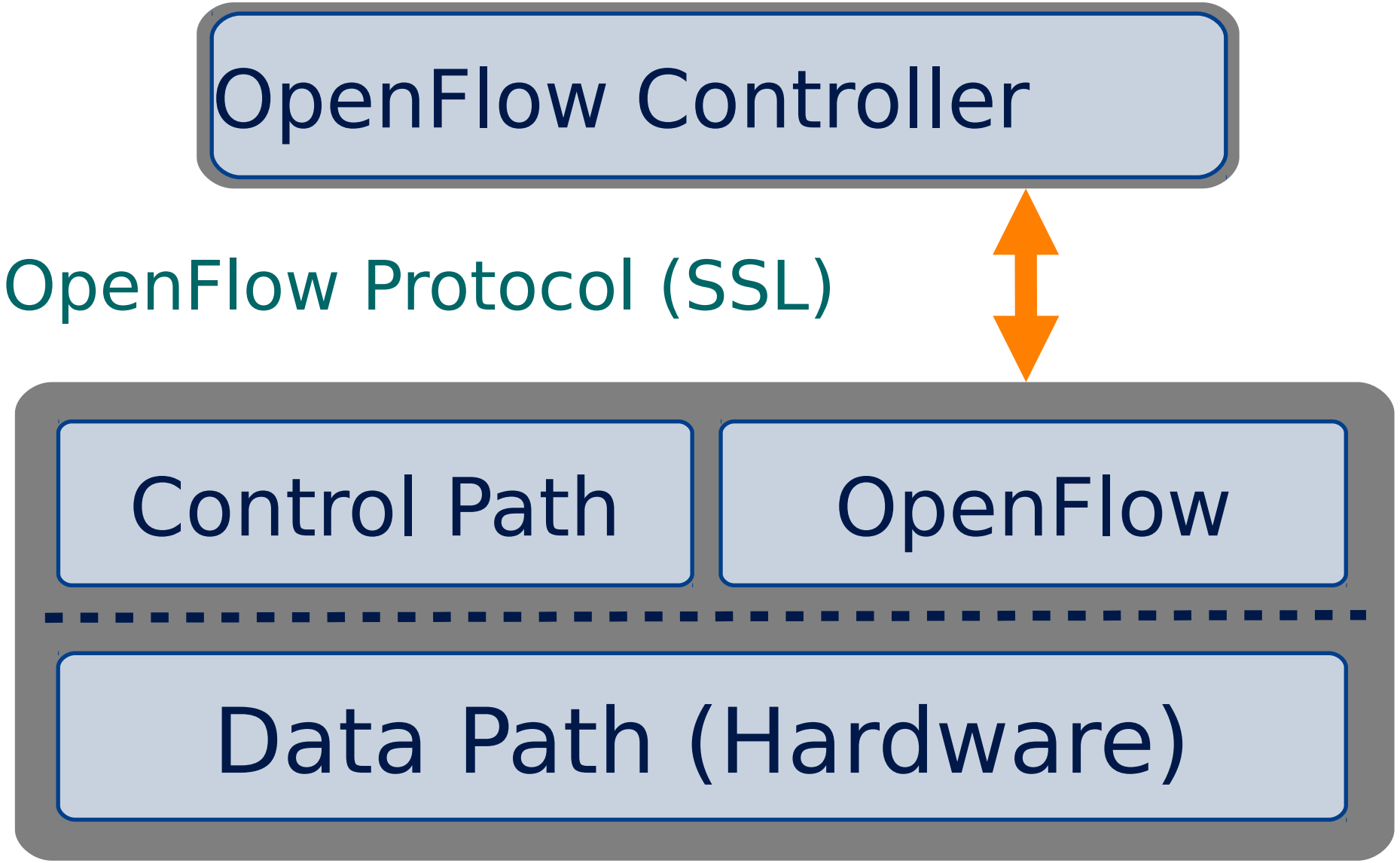
A “Rede definida por software”



Ethernet Switch/Router



Fonte: Nick McKeown (Stanford), SDN CIO Summit 2010



Fonte: Nick McKeown (Stanford), SDN CIO Summit 2010

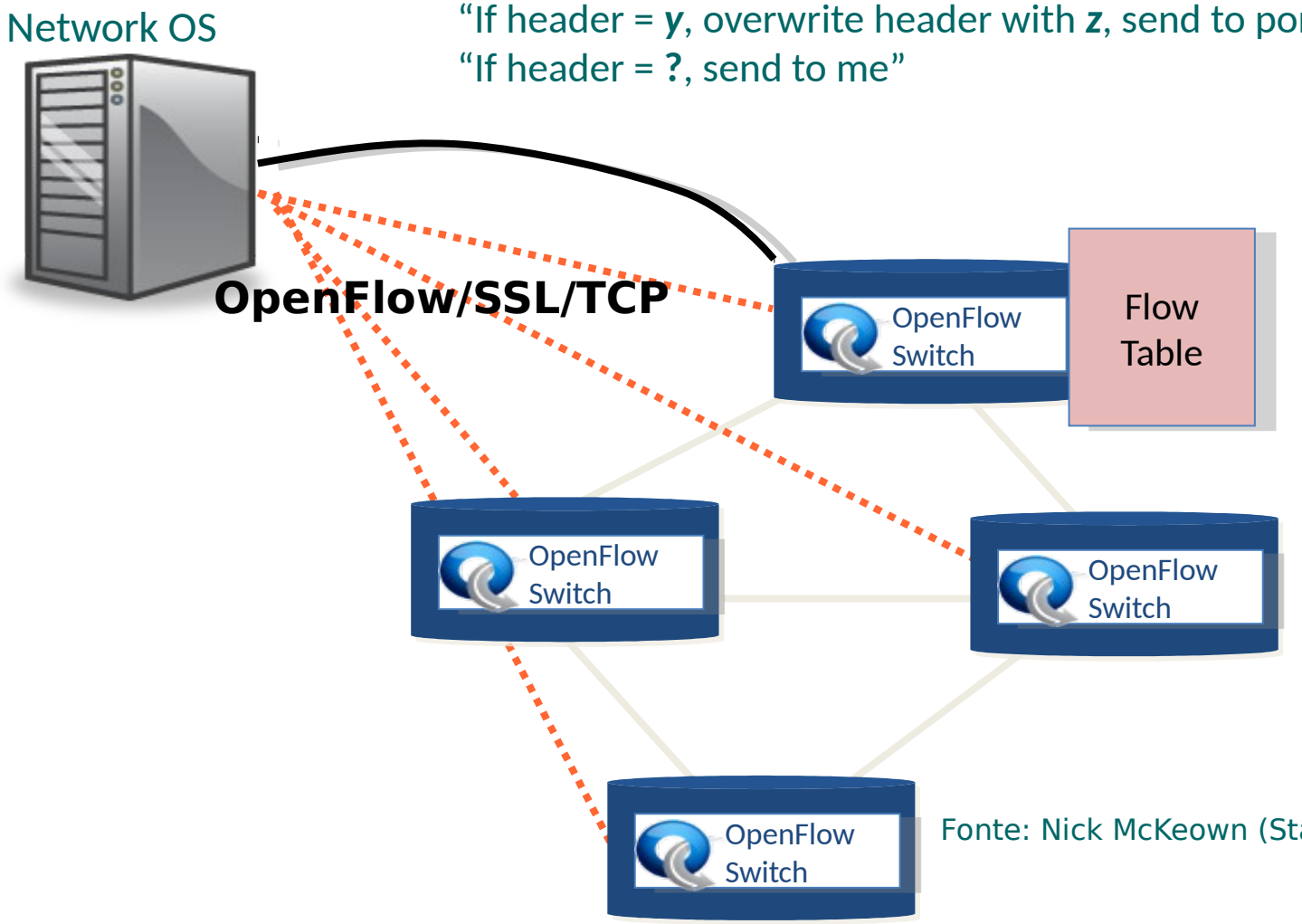
Exercício de fixação 1

- Como SDN/OpenFlow trata as seguintes questões?
  - Funcionalidades dependentes de fabricante – tempo de implantação;
  - Customizações são restritas aos parâmetros de configuração;
  - Rede com pouca flexibilidade.

Como o OpenFlow funciona?

Como o OpenFlow funciona

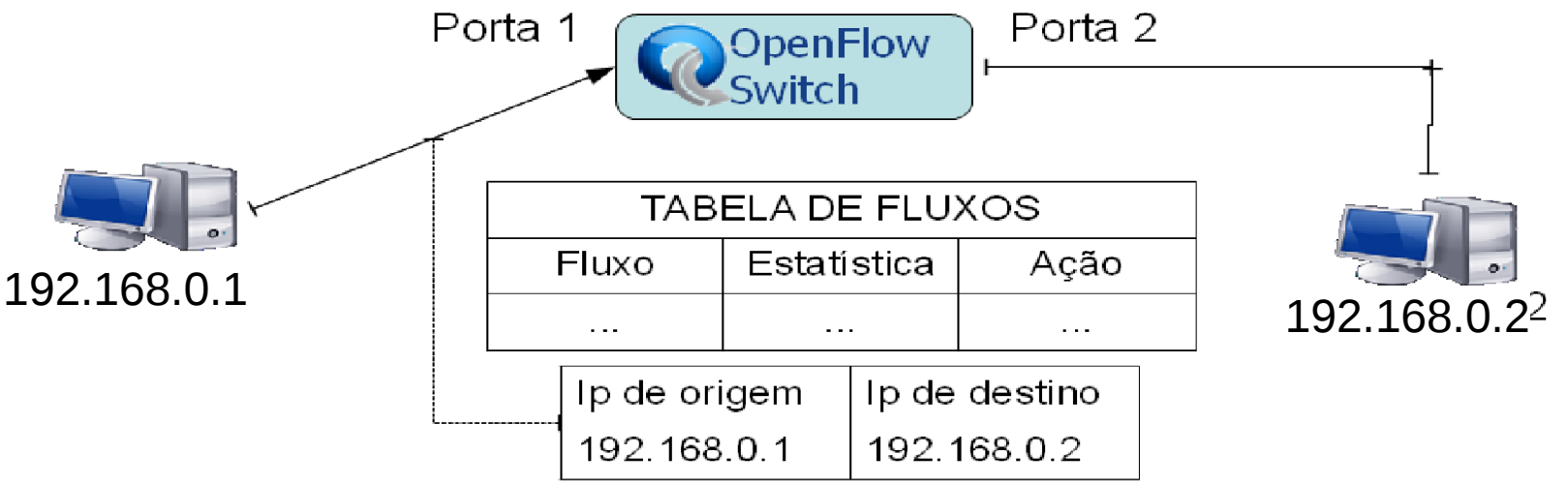
"If header = ***x***, send to port 4"
"If header = ***y***, overwrite header with ***z***, send to ports 5,6"
"If header = **?**, send to me"



Fonte: Nick McKeown (Stanford), SDN CIO Summit 2010

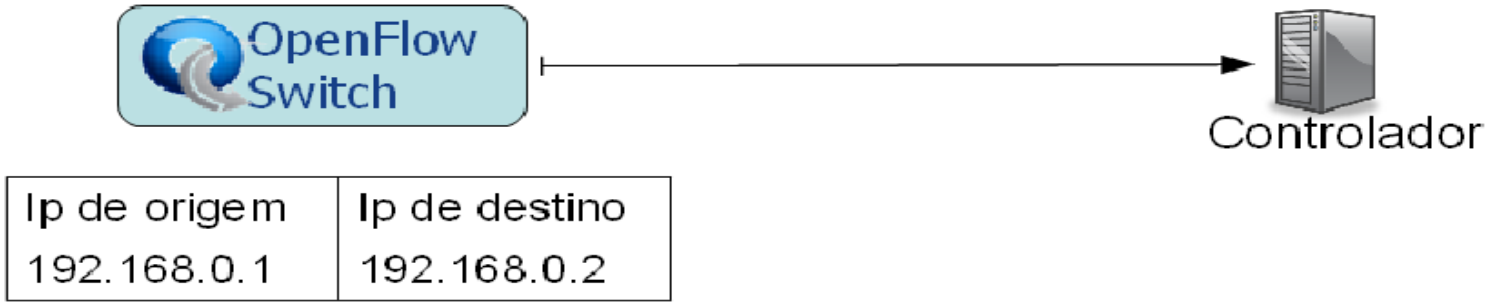
Como o OpenFlow funciona

- Quando o *switch* recebe um pacote ele deve compará-lo com sua tabela de fluxo



Como o OpenFlow funciona

- Se o cabeçalho não for compatível com nenhum fluxo em sua tabela, o *switch* deve encaminhar o cabeçalho (ou pacote todo) para o controlador



Como o OpenFlow funciona

- Ao receber o pacote, o controlador deve tomar uma decisão e
  - Enviar uma requisição de **mudança da tabela de fluxos do *switch***
  - Devolver o pacote (ou o cabeçalho + *buffer-id*)
  - Indicar uma **ação** para o pacote
  - Ou.. descartar

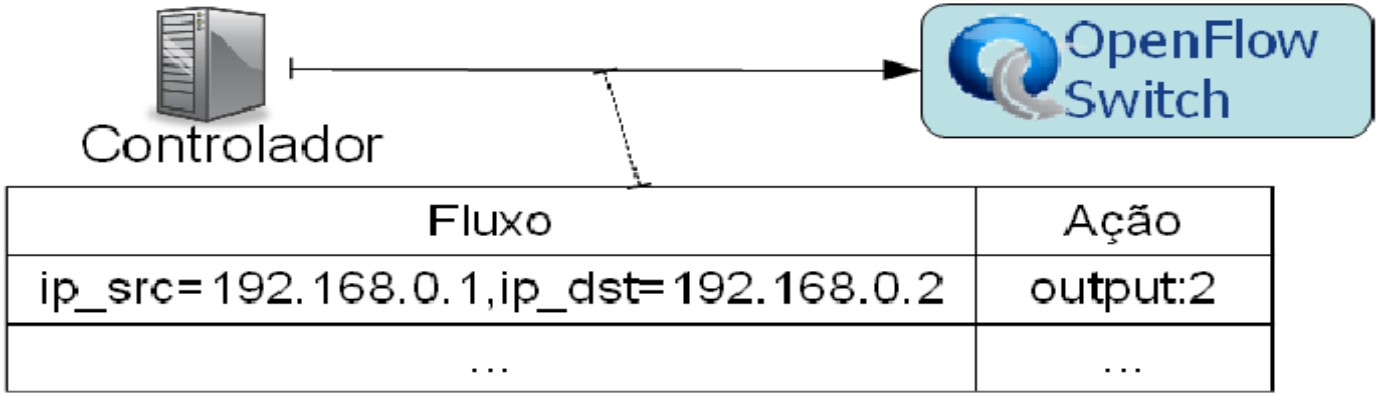
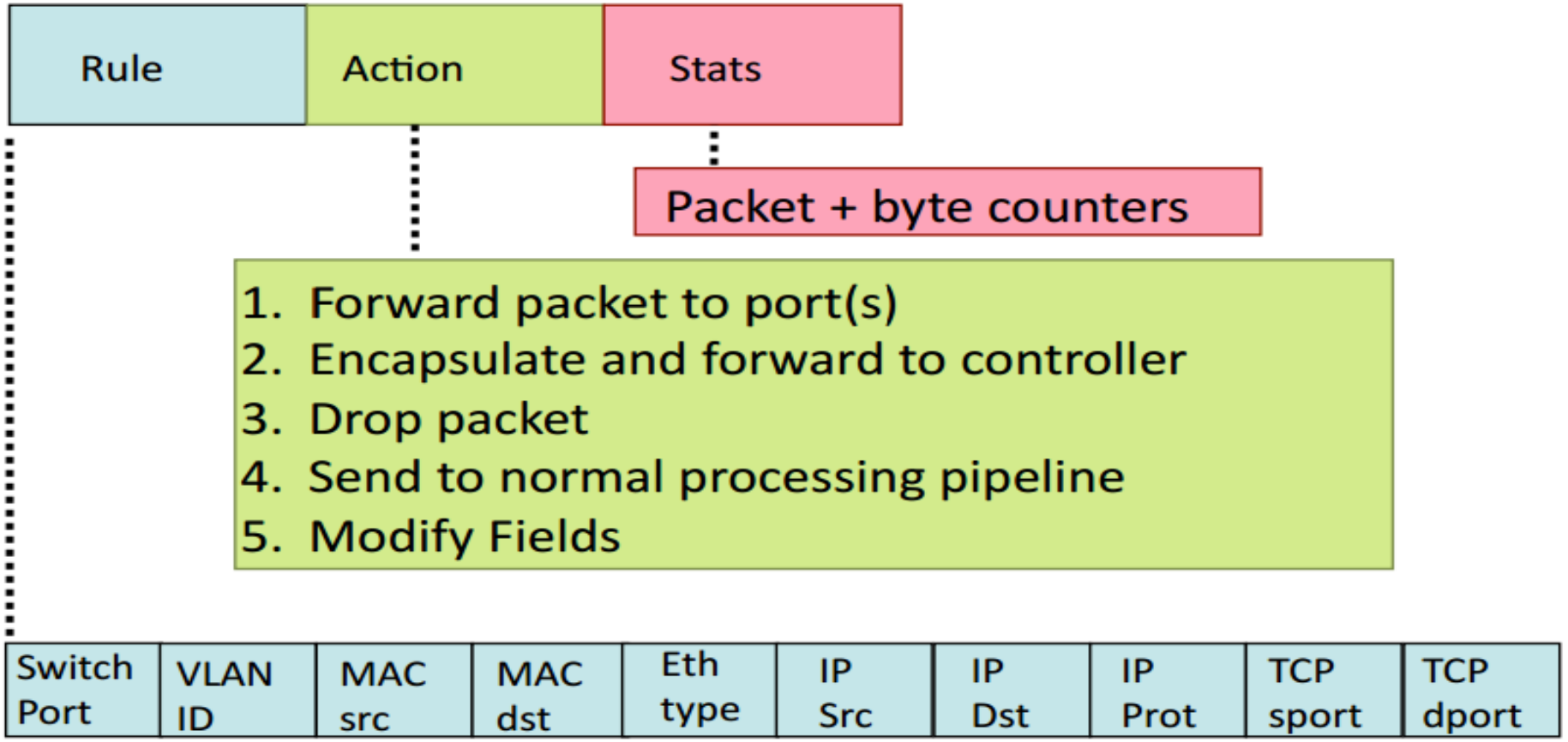


Tabela de fluxos



Exercício de fixação 2

- O que é um fluxo? O que é uma tabela de fluxos? Qual equivalente da tabela de fluxos em *switches* convencionais?
- Em qual camada do modelo TCP/IP o protocolo OpenFlow atua?
- Qual o papel do controlador?

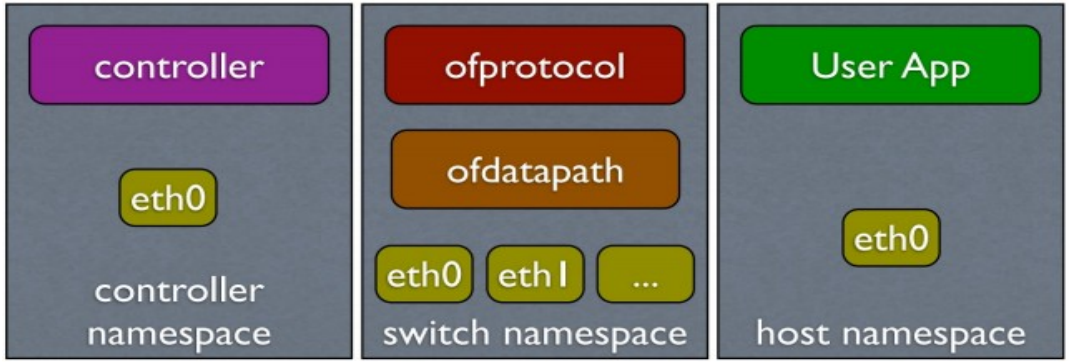
Open vSwitch

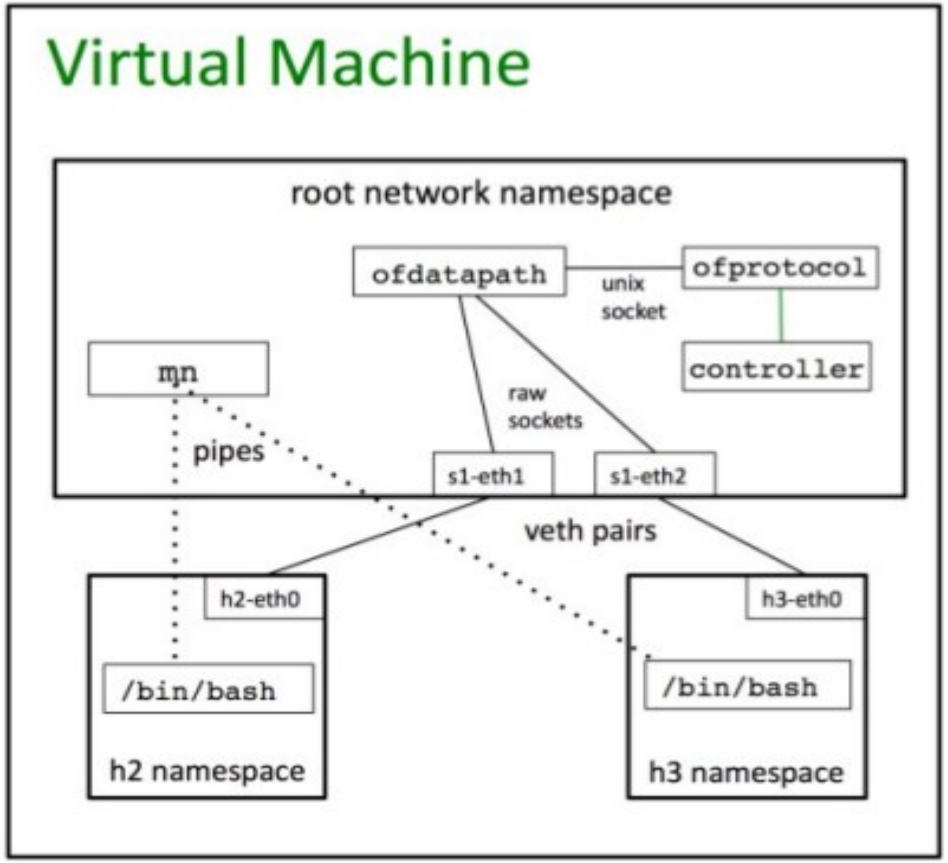
- Open vSwitch é um switch virtual *open source* tipicamente utilizando em ambientes de virtualização (locais e distribuídos)
- Também é utilizado em alguns equipamentos com hardware dedicado
- Muito utilizado em soluções SDN

Mininet

- Projeto que nasceu na Universidade de Stanford com o propósito de tornar mais fácil emular e testar novas soluções para SDN

- Arquitetura:





Controlador SDN e Aplicações

Exemplos de controladores

- NOX/POX
- Ryu
- Floodlight
- Kytos
- ONOS
- OESS
- OpenDayLight
- RouteFlow

Exemplos de controladores

- NOX/POX
- Ryu
- Floodlight
- Kytos
- ONOS
- OESS
- OpenDayLight
- RouteFlow

- Ryu:
  - Suporta OpenFlow 1.0, 1.1, 1.2, 1.3 e extensões da Nicira;
  - É um Framework para desenvolvimento de aplicações SDN, ao invés de um controlador monolítico
  - Diversos componentes: openstack, snort, REST, Topology manager, HA

Ryu Exemplo

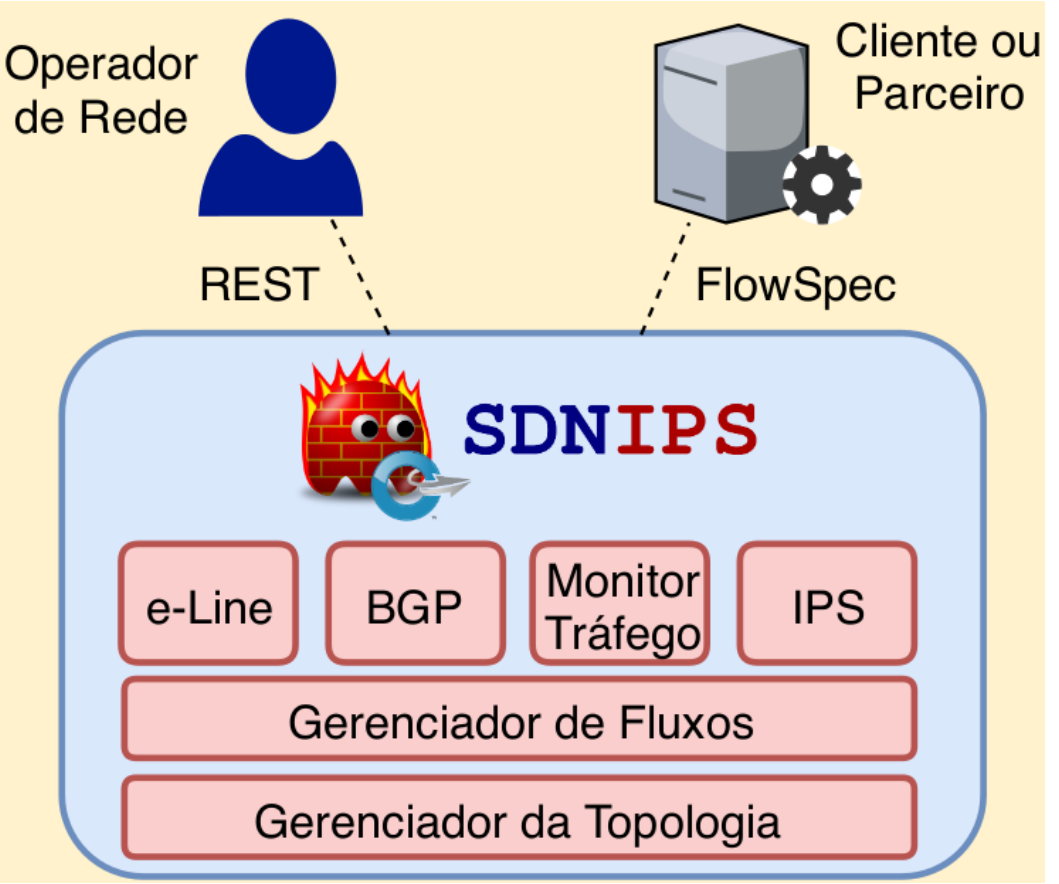
```
1 # ~/single-hub.py
2 from ryu.base import app_manager
3 from ryu.controller import ofp_event
4 from ryu.controller.handler import MAIN_DISPATCHER
5 from ryu.controller.handler import set_ev_cls
6 from ryu.ofproto import ofproto_v1_0
7
8 class SingleHub(app_manager.RyuApp):
9     OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]
10
11     def __init__(self, *args, **kwargs):
12         super(SingleHub, self).__init__(*args, **kwargs)
13
14     @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
15     def packet_in_handler(self, ev):
16         msg = ev.msg
17         dp = msg.datapath
18         ofp = dp.ofproto
19         ofp_parser = dp.ofproto_parser
20
21         actions = [ofp_parser.OFPActionOutput(ofp.OFPP_FLOOD)]
22         out = ofp_parser.OFPPacketOut(
23             datapath=dp, buffer_id=msg.buffer_id, in_port=msg.in_port,
24             actions=actions)
25         dp.send_msg(out)
```

Bibliotecas e Apps de apoio

- NetworkX
  - Estrutura de dados para armazenamento de informações da topologia de rede
- LLDP
  - Aplicação de apoio para descoberta de enlaces, baseado no padrão IEEE 802.1AB
- REST
  - Aplicação de apoio para Northbound API
- Topology Viewer
  - Aplicação de apoio para visualização da rede
- ...

SDN-IPS

- http://insert.ufba.br/sdn-ips



Programabilidade do dataplane com P4

Limitações de OpenFlow

- Limitado ao conjunto de cabeçalhos e campos disponíveis
- Sem suporte a protocolos customizados para encapsulamento
- Inflexibilidade do *pipeline* do hardware
- Sem suporte para In-band Network Telemetry
- Diferentes versões do protocolo entre equipamentos

Switches Whitebox

Equipamentos Homologados

	SuperMicro 5018A-TN7B	SuperMicro 5018D-FN8T	SuperMicro 5019D-FN8TP
Densidade de portas	7 x 1G RJ45	6 x 1G RJ45 2 x 10G SFP+	4 x 1G RJ45 4 x 10G (2x RJ45 + 2x SFP+)
Capacidade Agregada	7Gbps	26Gbps	44Gbps
Sistema Operacional	Fedora	Fedora	Fedora
Virtualization	Open vSwitch	Open vSwitch	Open vSwitch
Offload	DPDK	DPDK	DPDK
Preço (aproximado)	$550	$750	$1.500

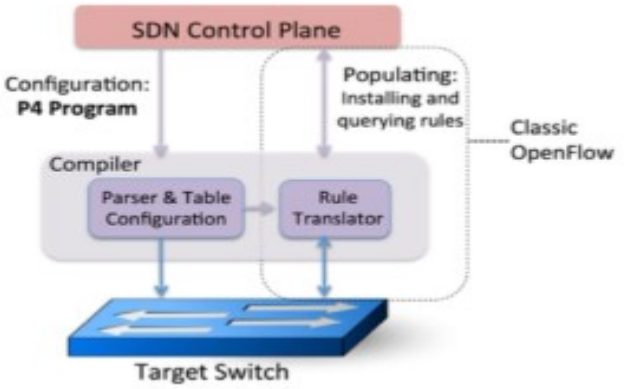
Fonte: Marcos Schwarz, WRNP’18

Network Disaggregation

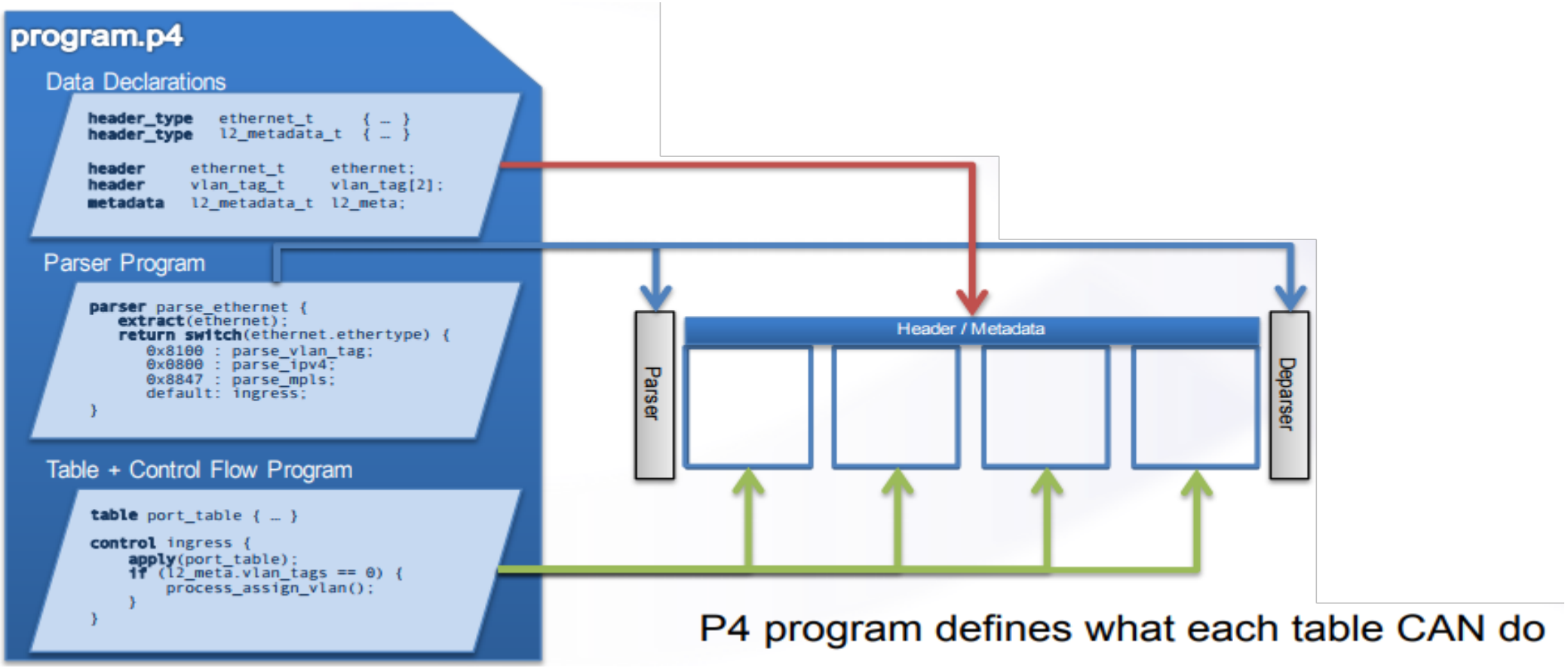
- Separação dos componentes de hardware e software em blocos lógicos
- Modularização das funcionalidades do dispositivo de rede
  - Forwarding agents, Chipset drivers, Operating System
- Arquitetura do switch customizável, pipeline reprogramável
- Inovação em todos os níveis da pilha de rede

P4 – Programabilidade do Dataplane

- Programas P4 definem o comportamento de encaminhamento no switch
- Dependências seriais expressas (ex: L3 Routing)
- Compilador P4 gera código específico para alvo
- OF pode ser integrado na consulta e instalação de regra

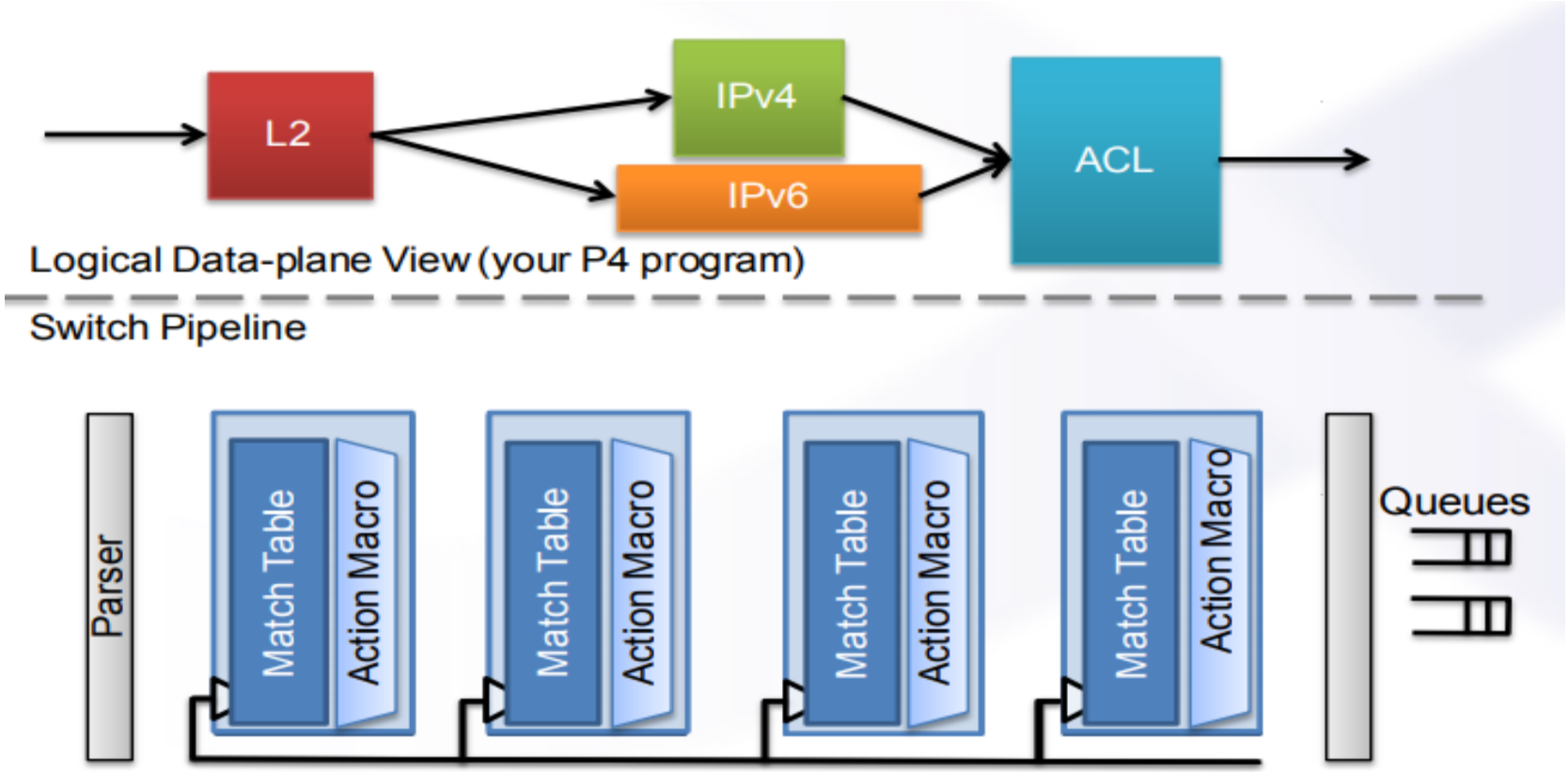


P4 – Programabilidade do Dataplane



Fonte: Nick McKeown, Networking Field Day'14

PISA - Protocol-Independent Switch Architecture



Fonte: Nick McKeown, Networking Field Day'14

Parsing P4

```
header vlan {
  fields {
    pcp : 3;
    cfi : 1;
    vid : 12;
    ethertype : 16;
  }
}
```

```
parser start {
  ethernet;
}

parser ethernet {
  switch(ethertype) {
    case 0x8100: vlan;
    case 0x9100: vlan;
    case 0x800: ipv4;
  }
}
```

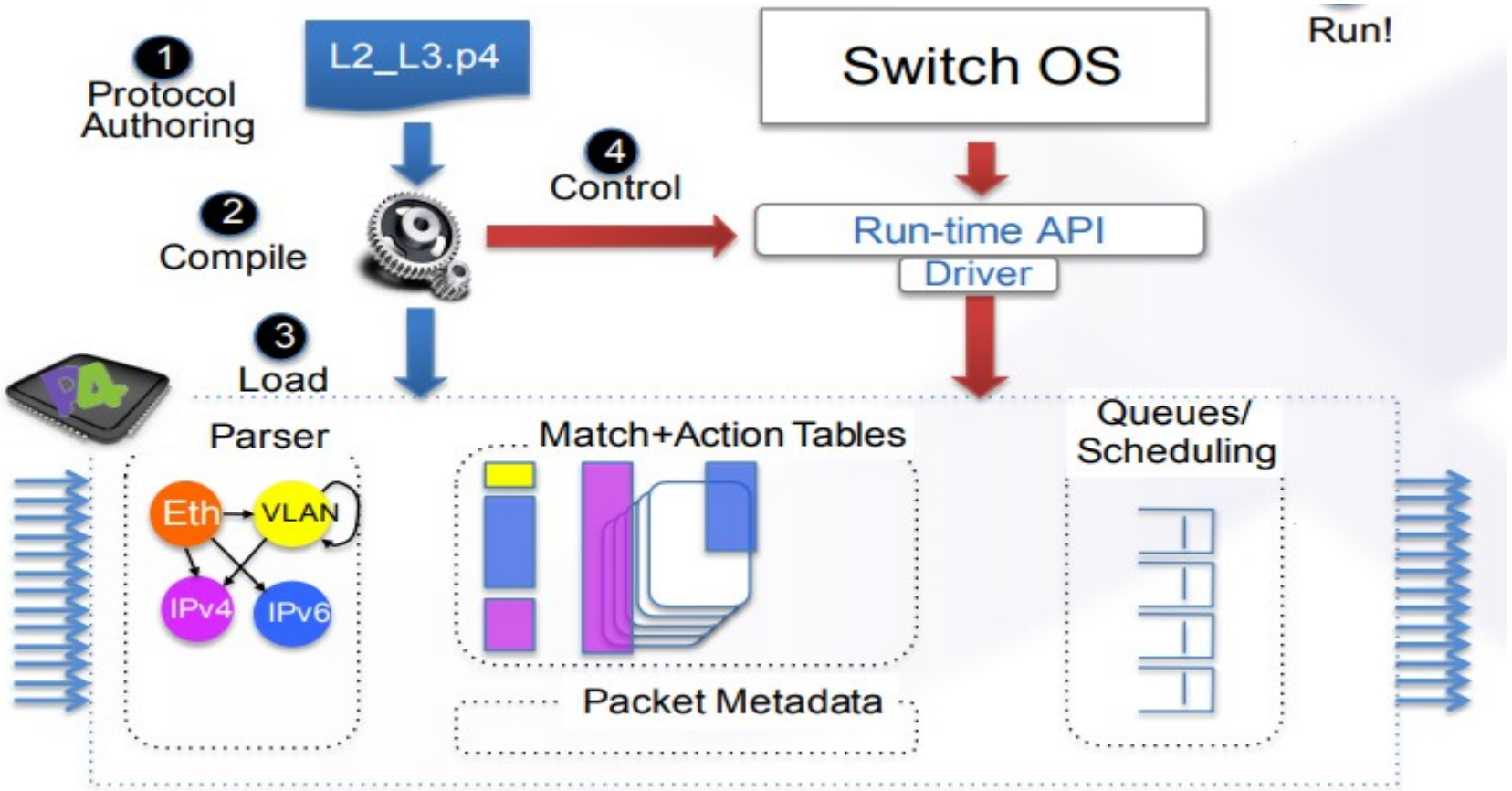
Actions P4

```
action add_mTag(up1, up2, down1, down2, egr_spec) {
  add_header(mTag);
  copy_field(mTag.ethertype, vlan.ethertype);
  set_field(vlan.ethertype, 0xaaaa);
  set_field(mTag.up1, up1);
  set_field(mTag.up2, up2);
  set_field(mTag.down1, down1);
  set_field(mTag.down2, down2);
}
```

P4 Match/Action

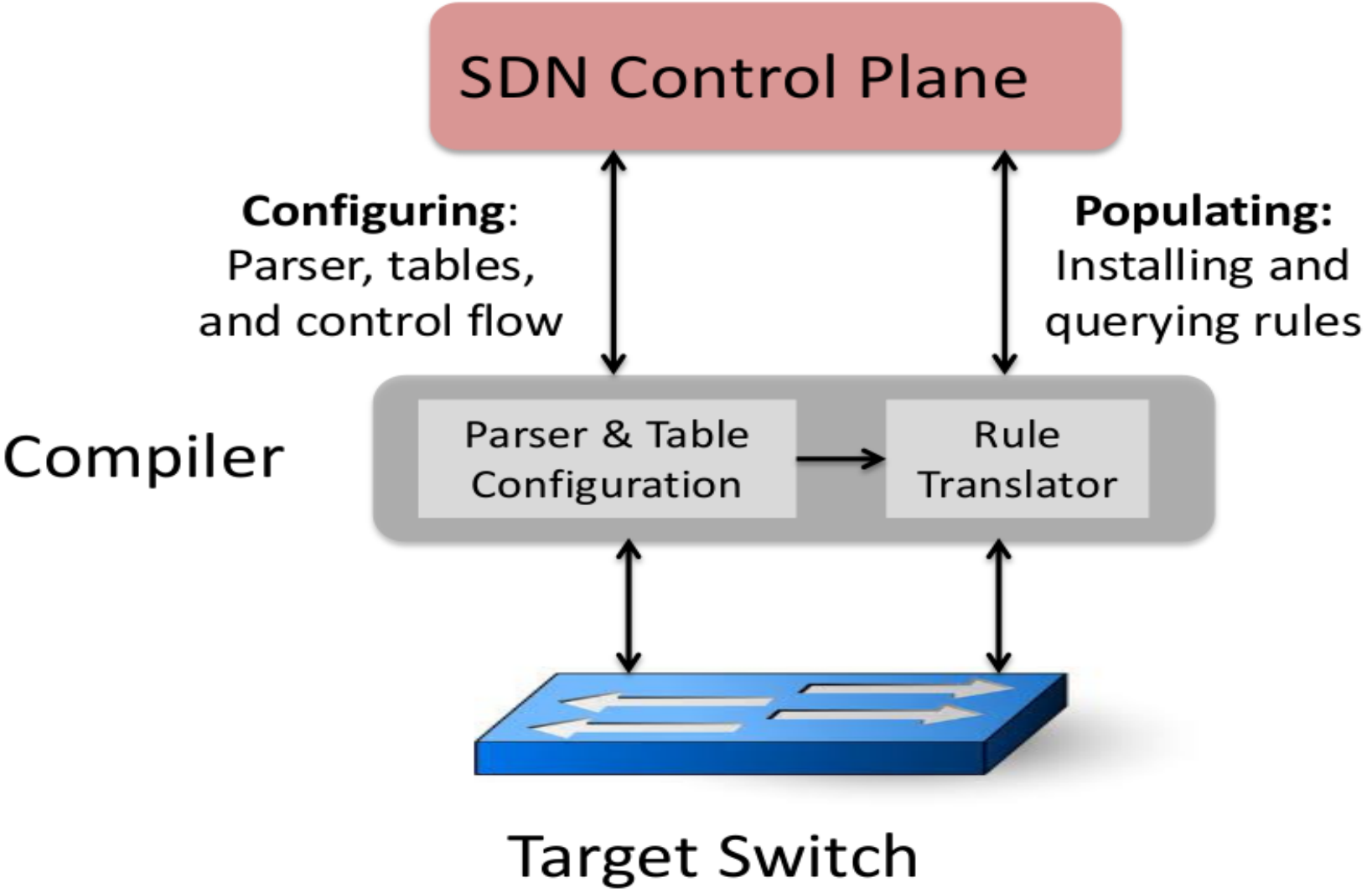
```
table mTag_table {
  reads {
    ethernet.dst_addr : exact; vlan.vid : exact;
  }
  actions {
    add_mTag;
  }
}
```

P4 Workflow



Fonte: Nick McKeown, Networking Field Day'14

Programabilidade completa da rede



Fonte: Marcos Schwarz, Kytos Summit'17

Aplicações P4

- Aplicações que requerem:
  - Mudança no *pipeline* do *switch*
  - Modificação dinâmica e sob demanda dos cabeçalhos de rede
- Exemplos:
  - INT – In-band Network Telemetry
  - Redes definidas por Resíduos com cabeçalhos de tamanho variável

INT – In-band Network Telemetry

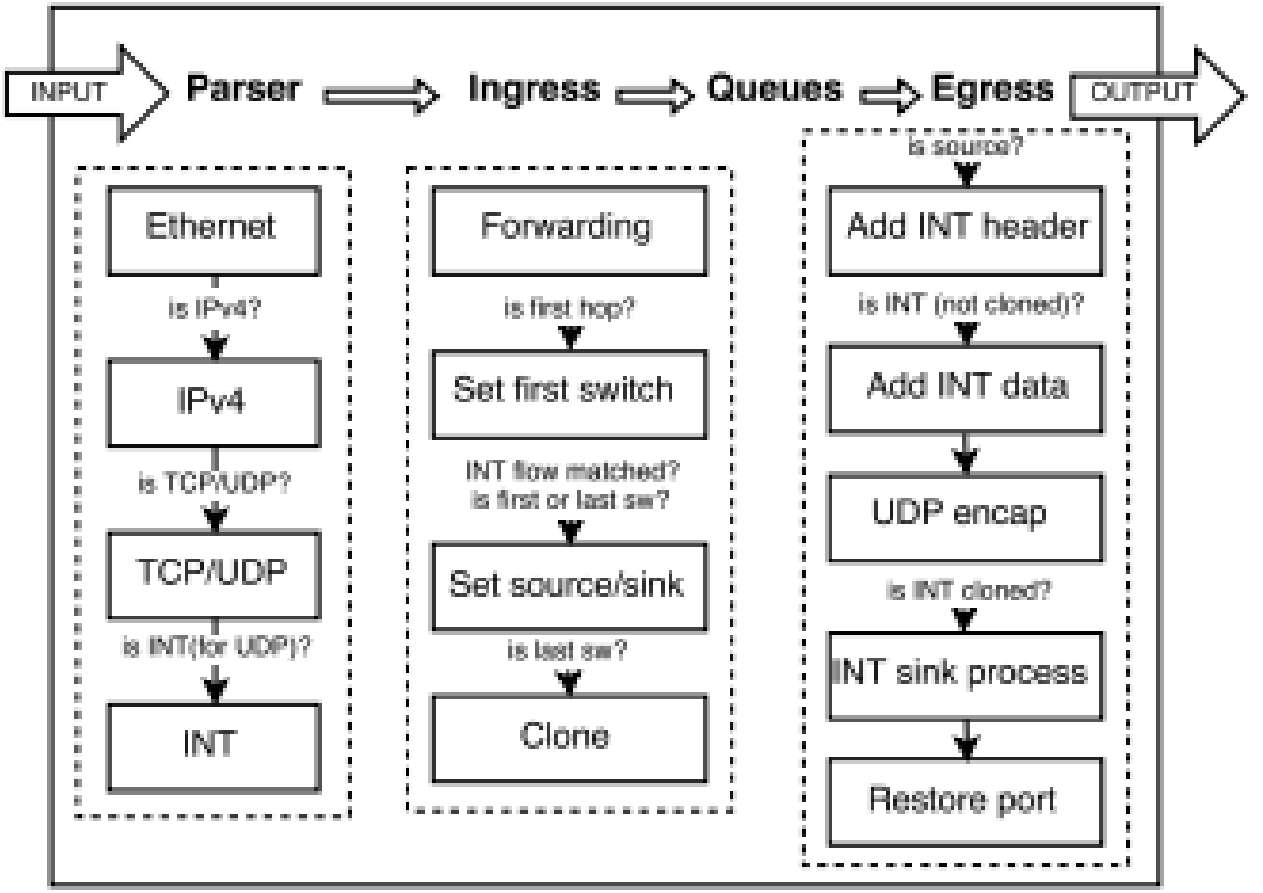
- Framework para coletar e reportar estatísticas e informações do estado da rede, através do *dataplane*, independente do *controlplane*
- Instruções de telemetria são “embarcadas” no cabeçalho dos pacotes
- Assim, os INT *traffic sinks* podem coletar e tratar/reportar as informações de telemetria
  - OAM
  - Controle *real-time*, sistema de *feedback*
  - Detecção de eventos e anomalias

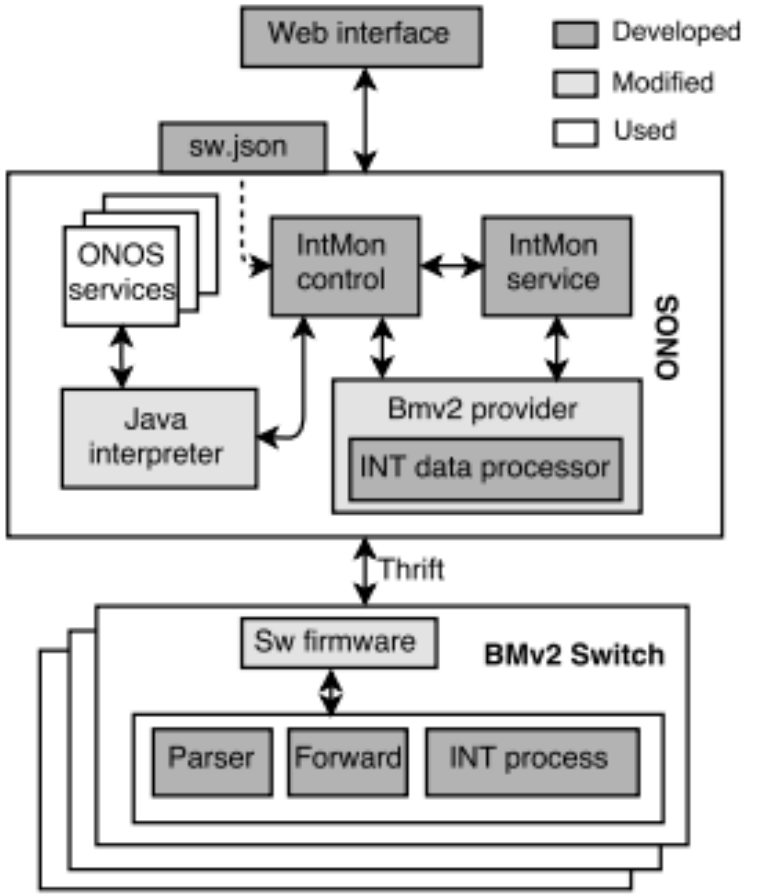
INT – In-band Network Telemetry

Diversas aplicações possíveis:

- *Troubleshooting* de rede
- Controle de congestionamento avançado
- Roteamento avançado
- Verificação de rede no *dataplane*

INT – In-band Network Telemetry





Fonte: Van Tu, N., *et al.*, 2017; IEEE NOMS

Exercício de Fixação

Qual a diferença entre programabilidade no *dataplane* e no *controlplane?*

Cite aplicações que demandam programabilidade no *dataplane*.

Que tipo de ambiente pode ser utilizado para experimentação?

Conclusões e desafios

Conclusões e Desafios para Internet do Futuro

- O crescimento da rede e a diversidade de aplicações requerem maior controle na orquestração da rede
  - Programabilidade no *dataplane* e no *controlplane*
- SDN/OpenFlow já incorporam formas de prover essa flexibilidade
- P4 (programação do *dataplane*) surge como uma possibilidade de inovação promissora
- Existem outras estratégias que discutem Internet do Futuro:
  - Information Centric Networks
  - Human Centric Networks

Referências

- Oficina SDN/OpenFlow – INSERT/UFBA (http://insert.ufba.br/oficina-sdn-openflow/)
- Marcos Schwarz. Network Disaggregation Overview. Kytos Summit’17
- Jennifer Rexford. Programming Abstractions for Future SDN Switches
- Nick McKeown. Programming The Network Data Plane in P4. SIGCOMM’16 Tutorial
- Nick McKeown. Why Does the Internet Need a Programmable Forwarding Plane. Networking Field Day 14
- Uwe Dahlmann. Introduction to Bare-metal Networking. Internet2 TechEx’16
- KREUTZ, Diego et al. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, v. 103, n. 1, p. 14-76, 2015.
- OLIVEIRA, Wallas F. et al. Aprovisionamento de QoS por Rótulos Programáveis para Redes Definidas por Resíduos. In: Simpósio Brasileiro de Redes de Computadores (SBRC). 2018.
- VAN TU, Nguyen; HYUN, Jonghwan; HONG, James Won-Ki. Towards ONOS-based SDN monitoring using in-band network telemetry. In: Network Operations and Management Symposium (APNOMS), 2017. IEEE, 2017. p. 76-81.