DevOps e Redes

SDN, NFV e Docker

Lucas Arbiza

LUCAS ARBIZA

Profissional

PoP-RS/RNP:

- Metropoa
- RSiX (PTT-RS)
- Rede Tchê

Acadêmico

Ciência da Computação:

- Mestrado UFRGS 2016
- Bacharelado UNIPAMPA 2011

CONTEÚDO

- DevOps
- SDN
- NFV
- Docker
- “Now show me the code!”

Photo by Eric Bailey; license: CC0

ORIGEM

2008

primeiras conversas – Agile 2008

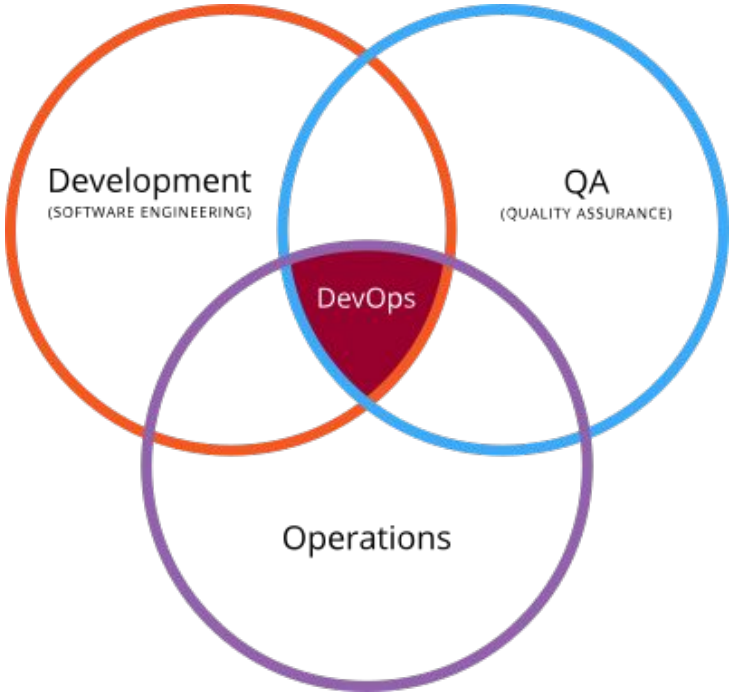
2009

Velocity, da O'Reilly – Allspaw e Hammond – termo DevOps

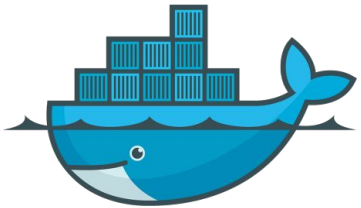
IDEIA: Agilidade para administração de infraestrutura

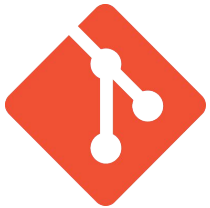
ORIGEM

Devs  Ops

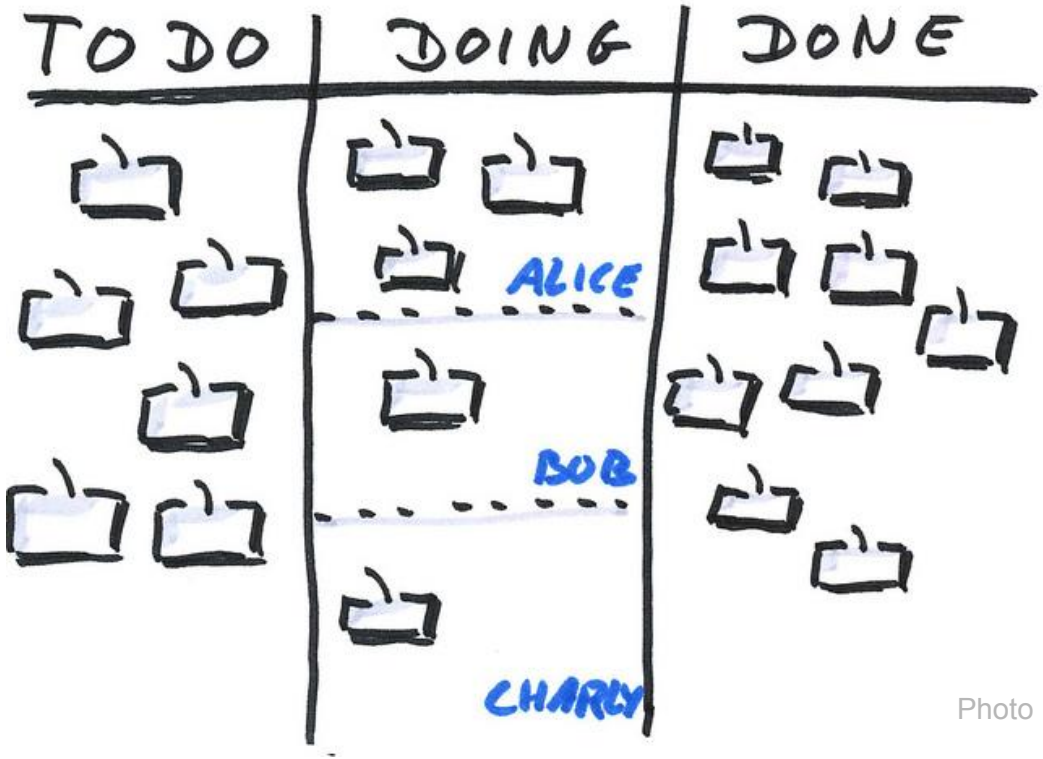


Agilidade para a infraestrutura





Agilidade para a infraestrutura



SDN, NFV e DevOps?
Tudo junto?

OpenFlow: Enabling Innovation in Campus Networks

March 14, 2008

Nick McKeown
Stanford University

Tom Anderson
University of Washington

Hari Balakrishnan
MIT

Guru Parulkar
Stanford University

Larry Peterson
Princeton University

Jennifer Rexford
Princeton University

Scott Shenker
University of California, Berkeley

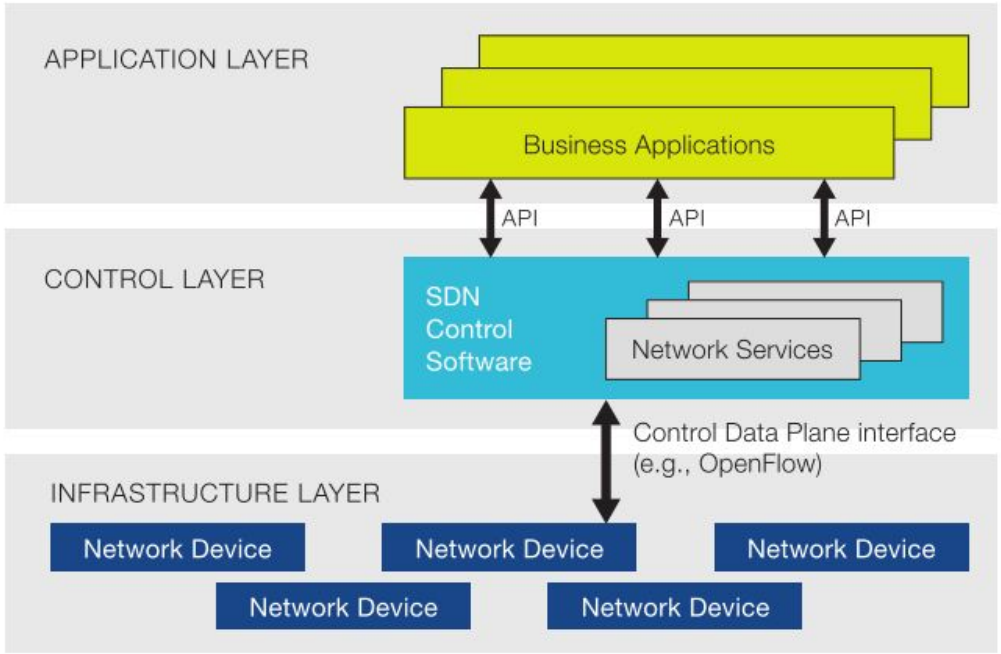
Jonathan Turner
Washington University in St. Louis

ABSTRACT

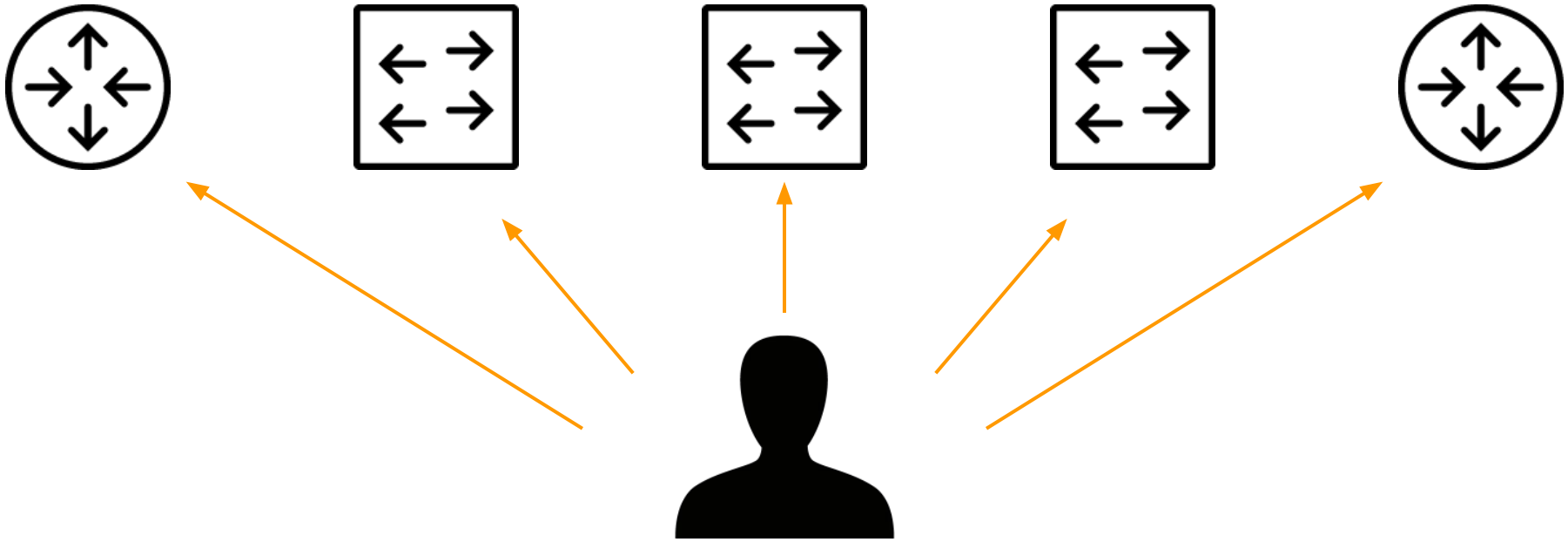
This whitepaper proposes OpenFlow: a way for researchers to run experimental protocols in the networks they use every day. OpenFlow is based on an Ethernet switch, with an internal flow-table, and a standardized interface to add

is almost no practical way to experiment with new network protocols (e.g., new routing protocols, or alternatives to IP) in sufficiently realistic settings (e.g., at scale carrying real traffic) to gain the confidence needed for their widespread deployment. The result is that most new ideas from the networking research community go untried and untested; hence

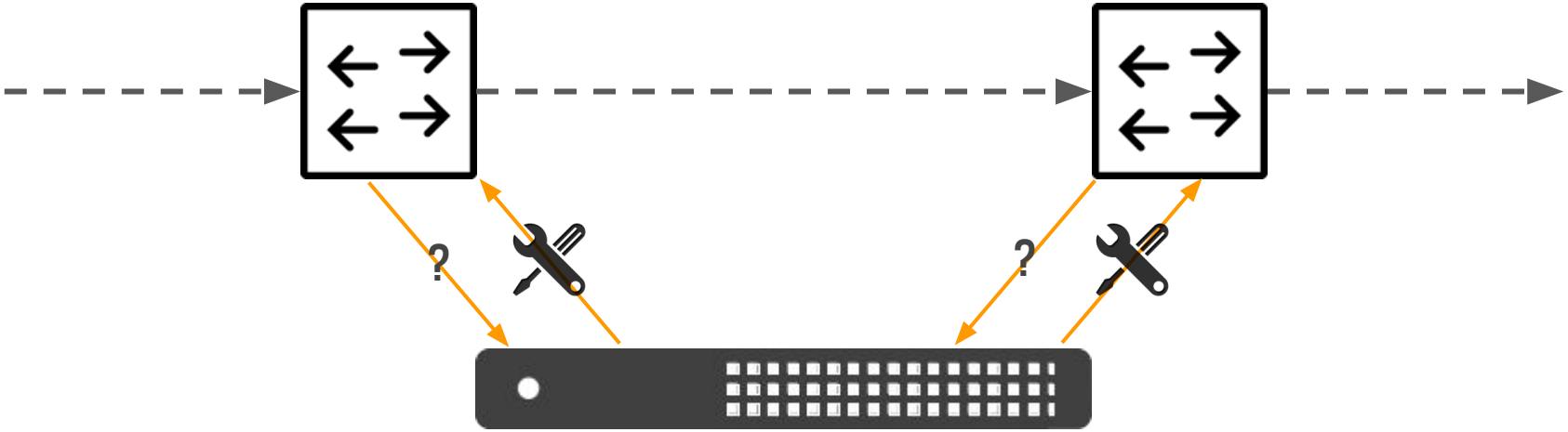
ARQUITETURA SDN



FUNCIONAMENTO: REDES TRADICIONAIS



FUNCIONAMENTO: REDES SDN



BENEFÍCIOS COM O USO DE SDN

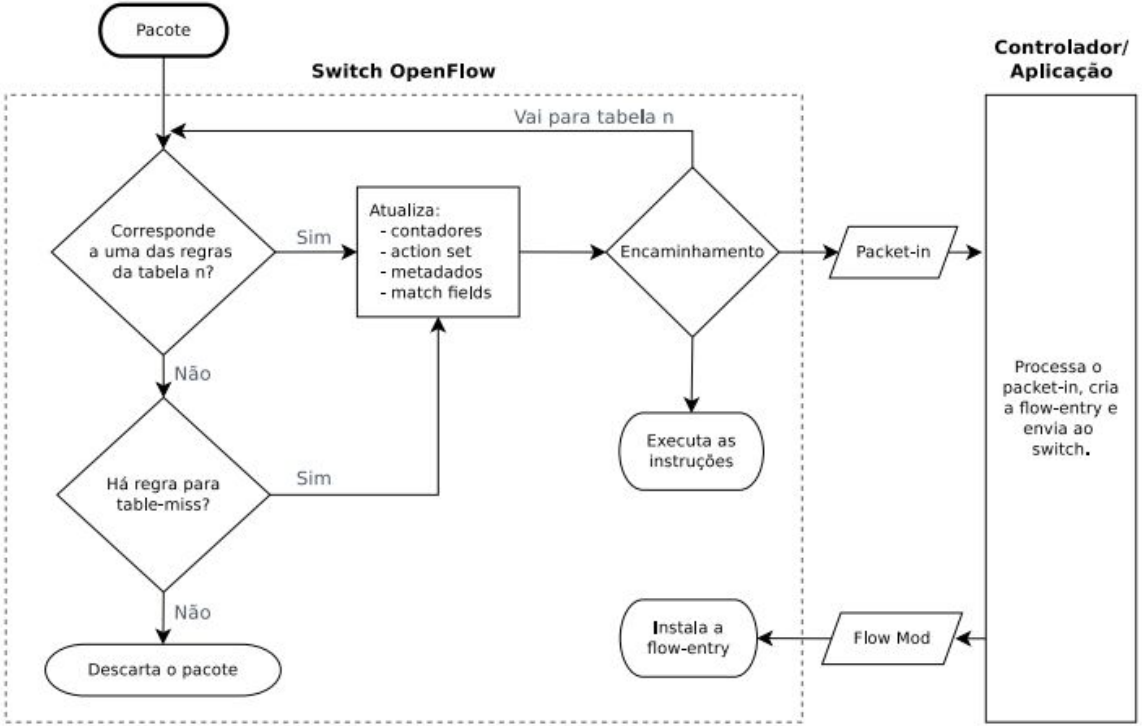
- Flexibilidade
- Programabilidade da rede
- Autonomia
- Abordagem centralizada

OPENFLOW

Flow entry:

Match fields	Priority	Counters	Instructions	Timeout	Cookie

OPENFLOW WORKFLOW



OPENFLOW

Controlador <> switch

- Configuração:
  - Proativa
  - Reativa (*packet in*)
- Negociação de conexão

OPENFLOW

Controlador <> switch

- Features
- Adicionar flows
- Remover flows (explicita ou *timeout*)
- Atualizar flows
- Read-state
- Port-status (enviada pelo switch)
- Error

OPENFLOW

Controlador <> switch

- Perda de conectividade com o controlador:
  - fail secure mode: *continua como um switch OpenFlow*
  - fail standalone mode: *opera como um switch L2 convencional*

OPENFLOW

Recursos do OpenFlow:

- Group table
- Meter table
- Counters

OPENFLOW

Recursos do OpenFlow -- Actions:

- Action set:
  - TTL
  - Tags
  - Sets
  - QoS
  - Group
  - Output
- Action list

OPENFLOW CONTROLLERS

- Nox
- Pox
- Nox 1.3
- OpenDaylight
- OpenContrail
- Floodlight
- Ryu
- FlowVizor
- Libfluid

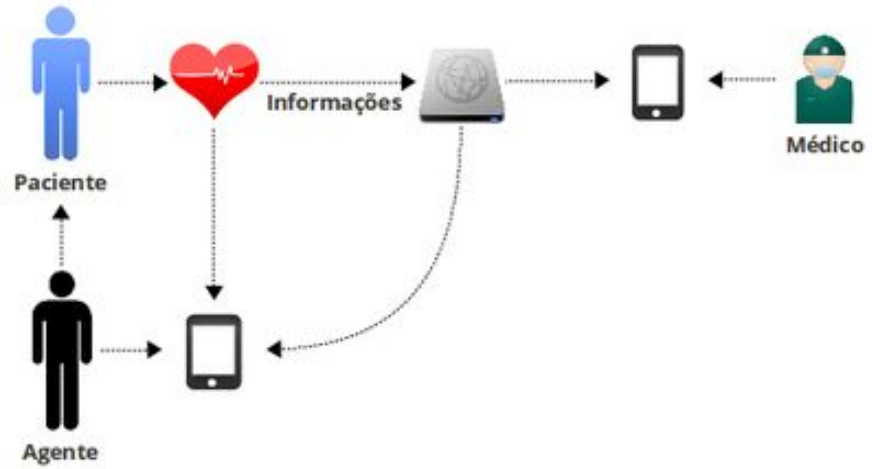
EXEMPLOS DE UTILIZAÇÃO

EXEMPLOS: Meu Mestrado

SDN no contexto de IoT : refatoração de middleware para monitoramento de pacientes crônicos baseada em software-defined networking

EXEMPLOS: SDN no cotexto de IoT (Mestrado)

CENÁRIO: Projeto REMOA



EXEMPLOS: SDN no cotexto de IoT

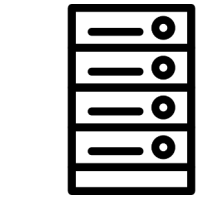
Não configuráveis

Configurável

EXEMPLOS: SDN no cotexto de IoT

Interconexão/Silo

FABRICANTE



EXEMPLOS: SDN no cotexto de IoT

Coleta de dados

```
POST /cgi-bin/measure HTTP/1.1
User-Agent: Withings UserAgent
Host: scalews.withings.net
Accept: */*
Content-Length: 233
Content-Type: application/x-www-form-urlencoded

action=store&sessionid=721-4ee232b1-50d6de38&macaddress=00:24:e4:01:36:10&userid=448489&meastim
e=1323446930&devtype=1&attribstatus=0&measures=%7B%22measures%22%3A%5B%7B%22value%22%
3A98550%2C%22type%22%3A1%2C%22unit%22%3A%2D3%7D%5D%7D
```

EXEMPLOS: SDN no cotexto de IoT

Gerência

~~ICMP~~
~~SNMP~~

~~ICMP~~
~~SNMP~~

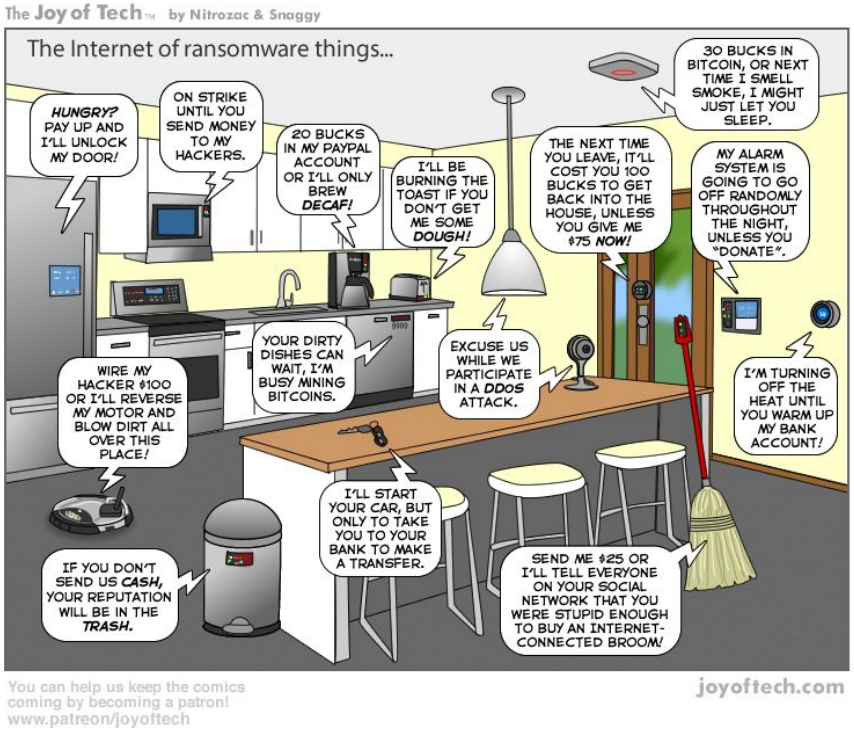
ICMP
~~SNMP~~

EXEMPLOS: SDN no cotexto de IoT

Segurança

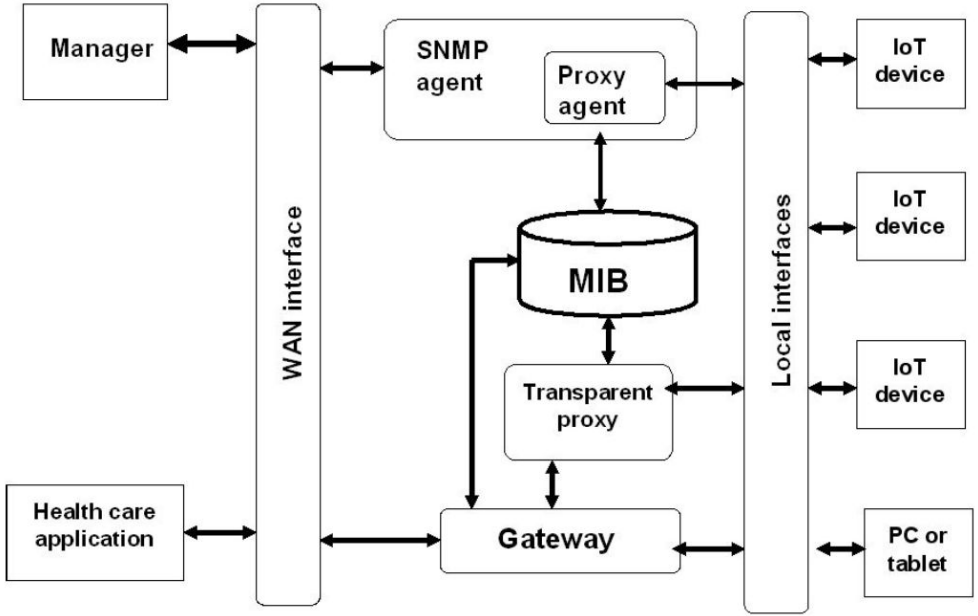
EXEMPLOS: SDN no cotexto de IoT

Segurança



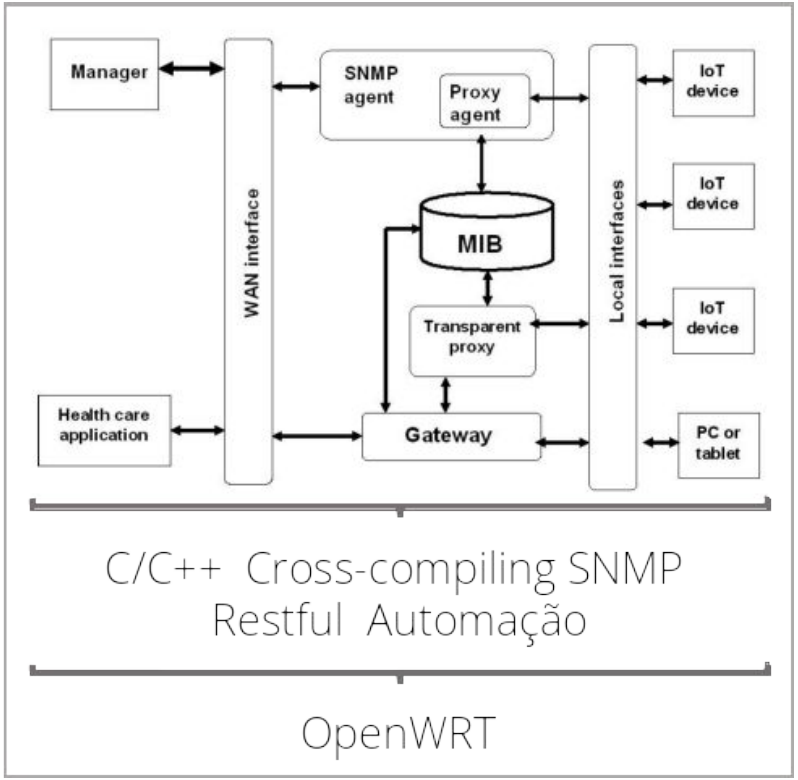
EXEMPLOS: SDN no cotexto de IoT

Middleware proposto



EXEMPLOS: SDN no cotexto de IoT

Problemas encontrados



EXEMPLOS: SDN no cotexto de IoT

REFATORAÇÃO

EXEMPLOS: SDN no cotexto de IoT

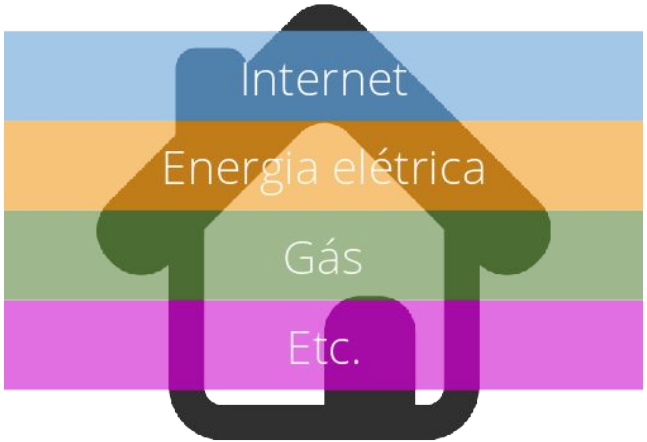
Chetty e Feamster (2012)

Refatoração SDN

EXEMPLOS: SDN no cotexto de IoT

Yiakoumis et al. (2011)

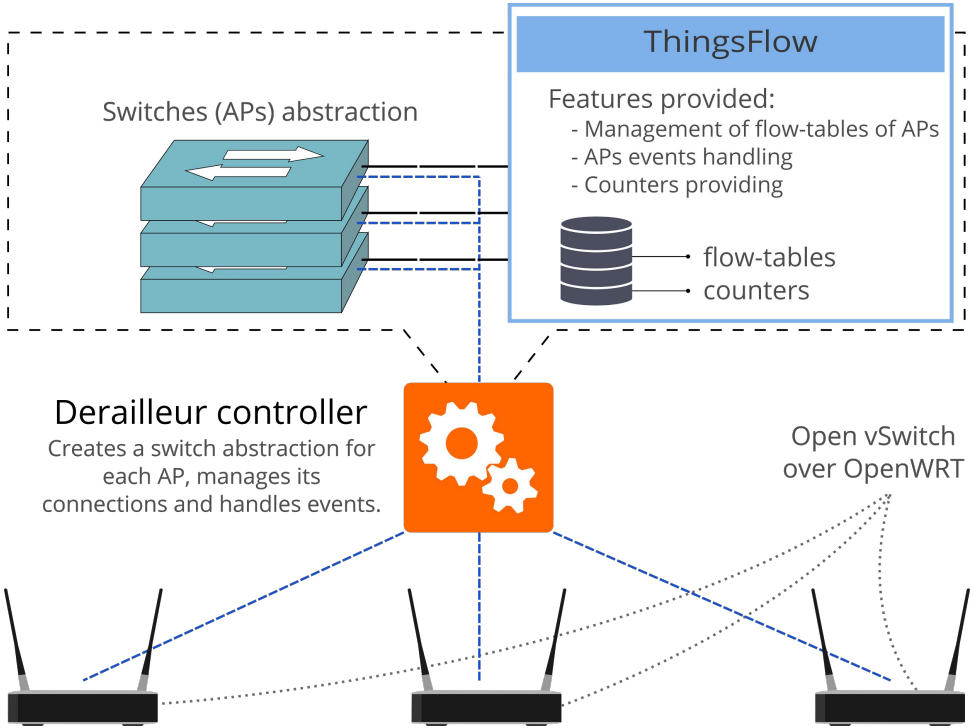
Divisão da rede
em “fatias”



EXEMPLOS: SDN no cotexto de IoT

REFATORAÇÃO

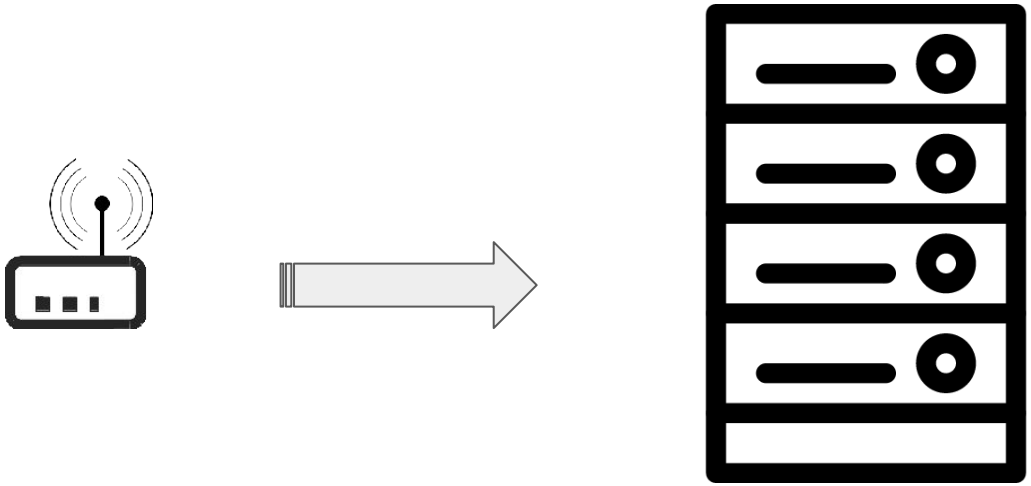
- Arquitetura



EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

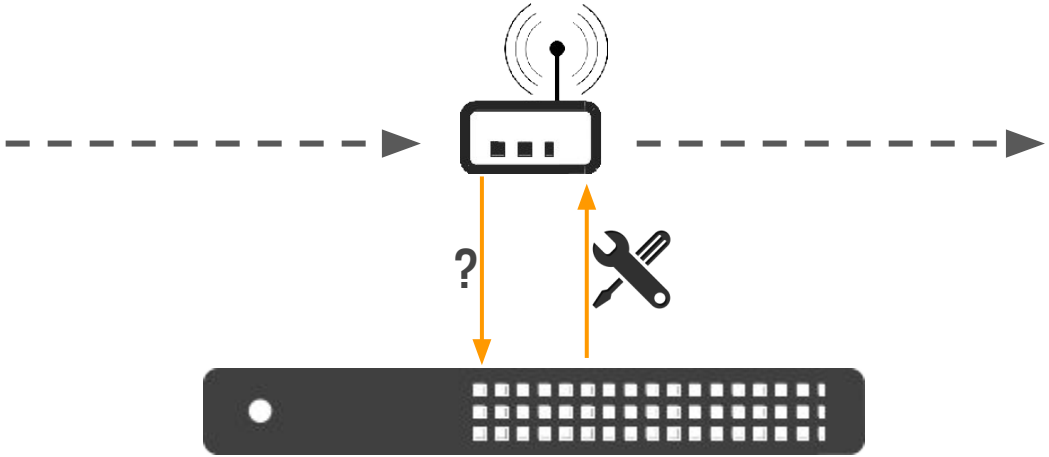
- Desenvolvimento



EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

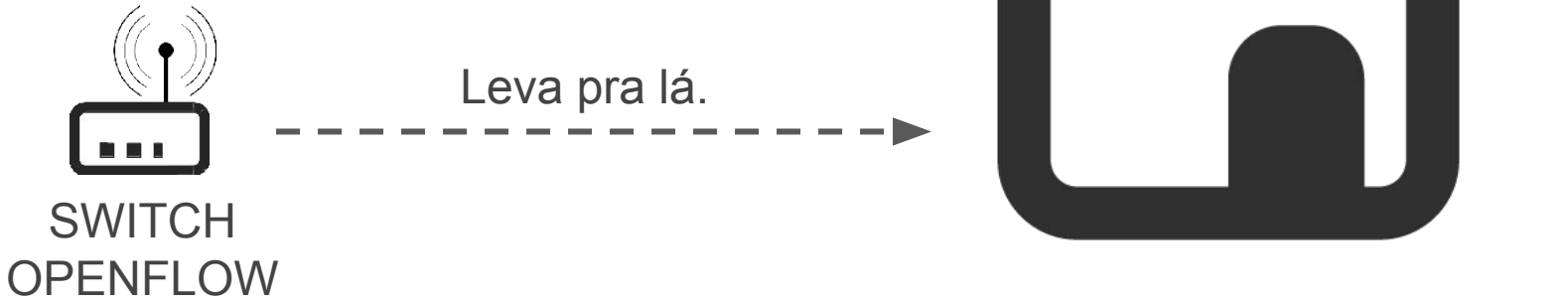
- Manutenção



EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

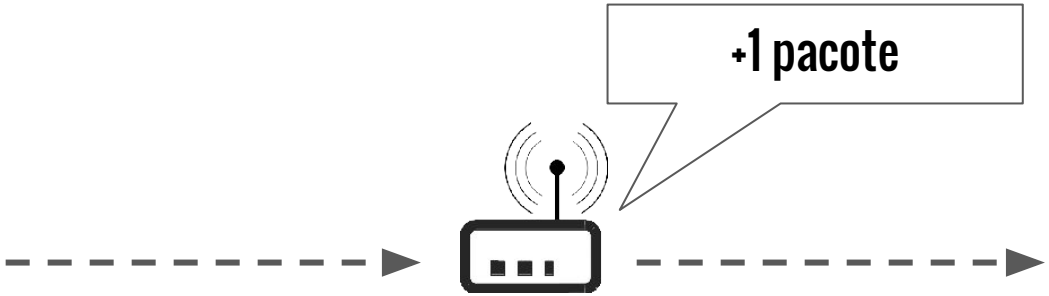
- Deployment



EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

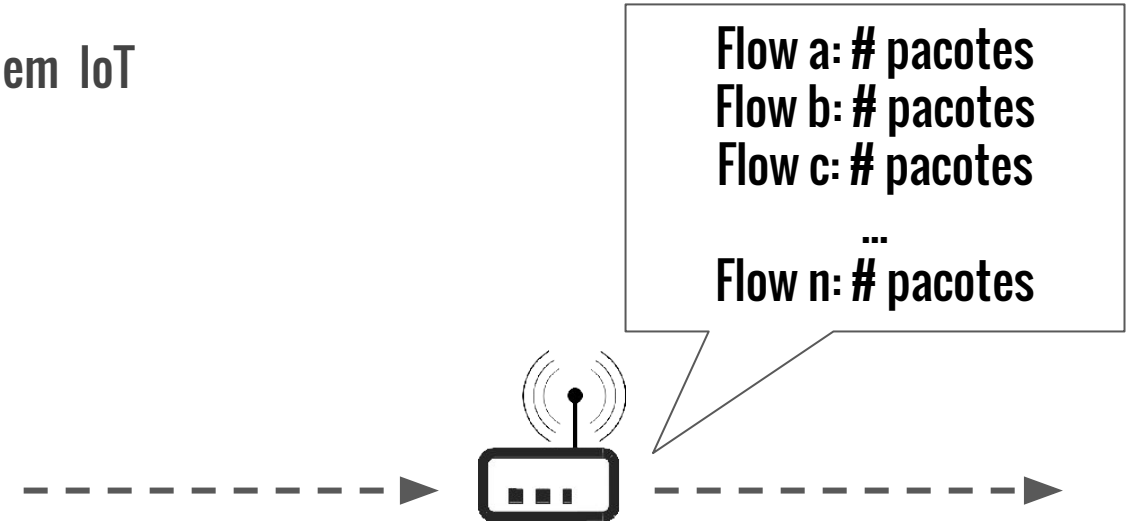
- Gerência para IoT



EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

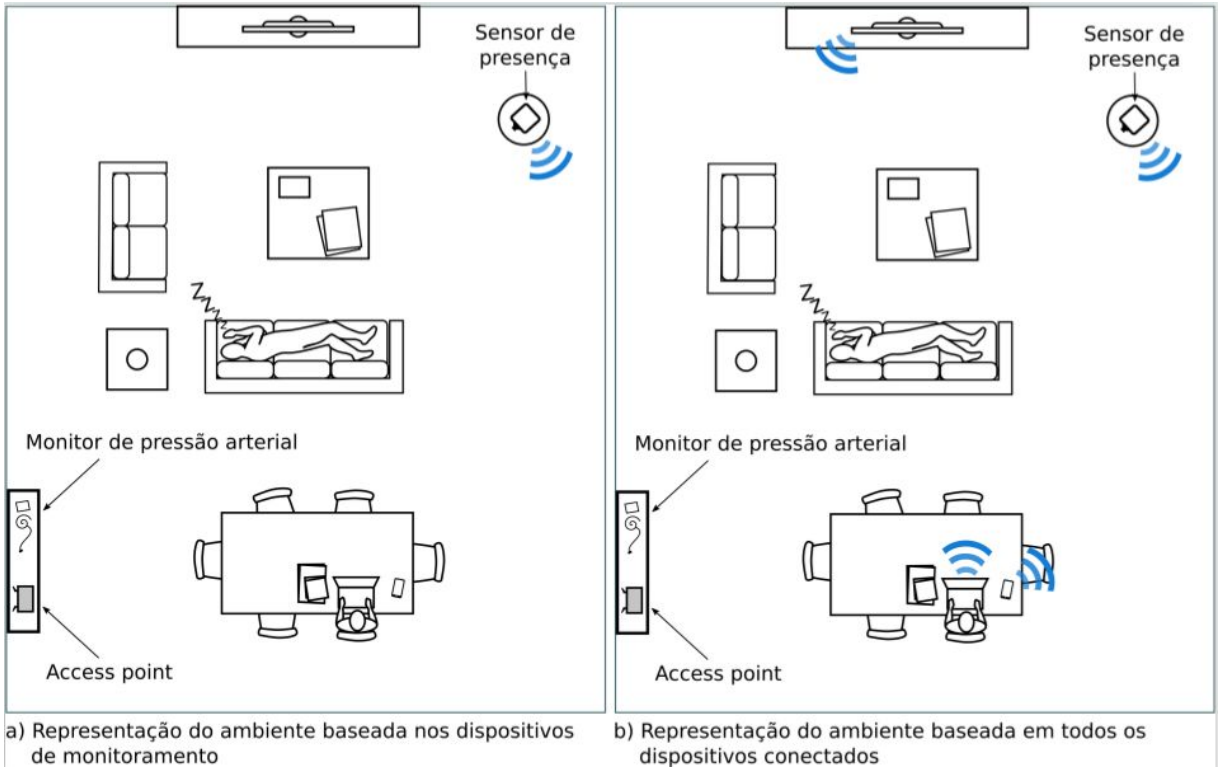
- Cooperação em IoT



EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

- Cooperação em IoT



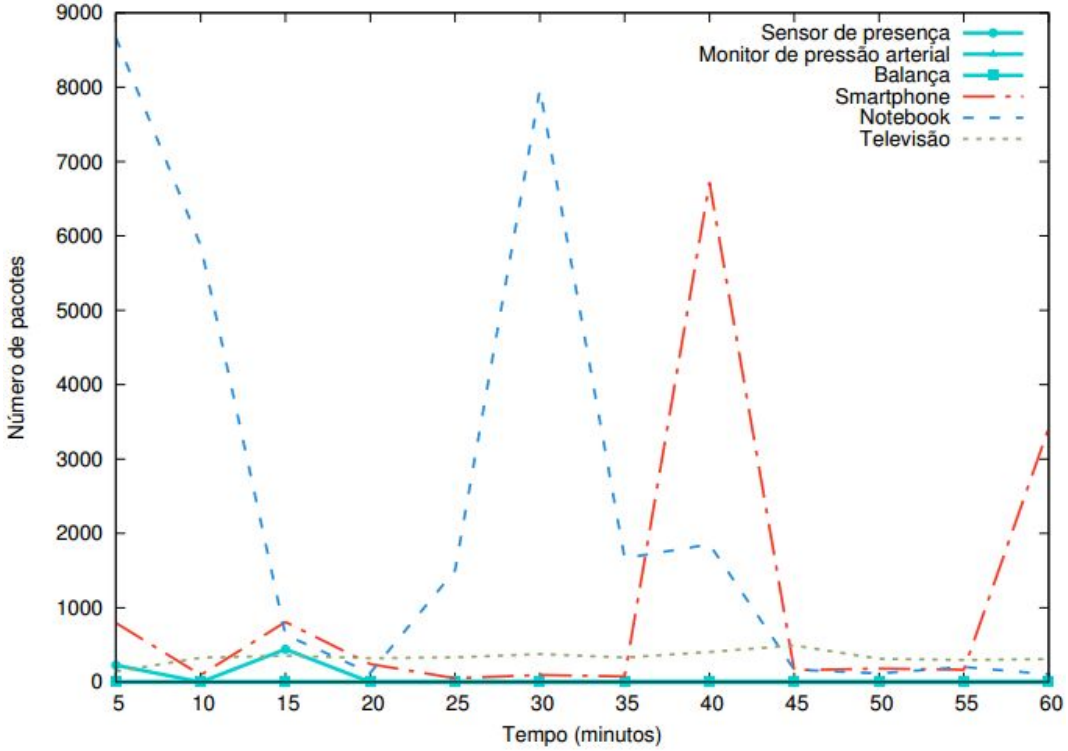
a) Representação do ambiente baseada nos dispositivos de monitoramento

b) Representação do ambiente baseada em todos os dispositivos conectados

EXEMPLOS: SDN no cotexto de IoT

BENEFÍCIOS

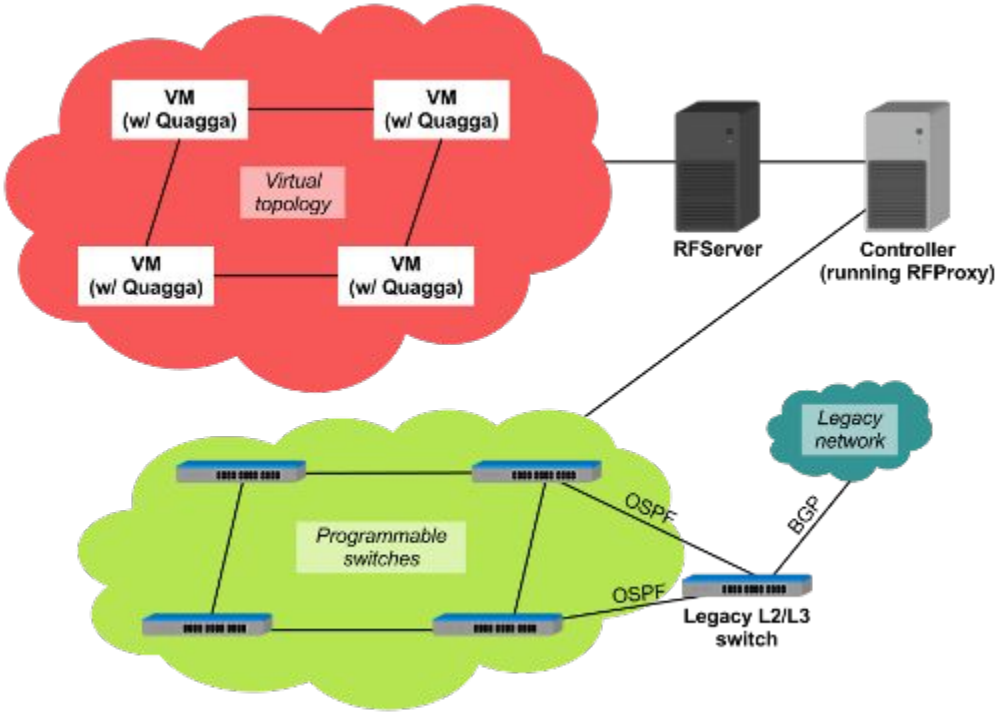
- Cooperação em IoT



EXEMPLOS: SDN no cotexto de IoT

PIPELINE

EXEMPLOS: RouteFlow



EXEMPLOS: OpenQoS

QoS tradicional:

- InfServ: reserva de recursos (*Resource Reservation Protocol*)
- DiffServ: filas e classificação

Proposta:

- Identificação de tráfego multimídia
- Roteamento por caminhos diferenciados

EXEMPLOS: OpenRoads

Problema: *handover*

Proposta:

- Envio do mesmo pacote para mais de uma base
- Usuário com duas placas de rede

EXEMPLOS: Gerência de redes

- Controlador: visão geral da rede
- Integração com outros protocolos e mecanismos de gerência

Exemplos:

- Gerência de acesso a recursos a partir de eventos e políticas de segurança
- Monitoramento de tráfego
- Autonomia e tomada de decisões

EXEMPLOS: iSDX

- Políticas L2/L3
- Definição de políticas por Json
- Coexistência com BGP

SDN substituirá tudo?

By Unknown, scanned by Polaco77 [Public domain], via Wikimedia Commons

Image from Spider-Man movie; Columbia Pictures Corporation, Marvel Enterprises; 2002

Funcionalidades da rede:

ipservicesk9-mz.122-55.SE10.bin
by Cisco

manda-chuva-na-rede.py
by Equipe de TI

Image from Spider-Man movie; Columbia Pictures Corporation, Marvel Enterprises; 2002

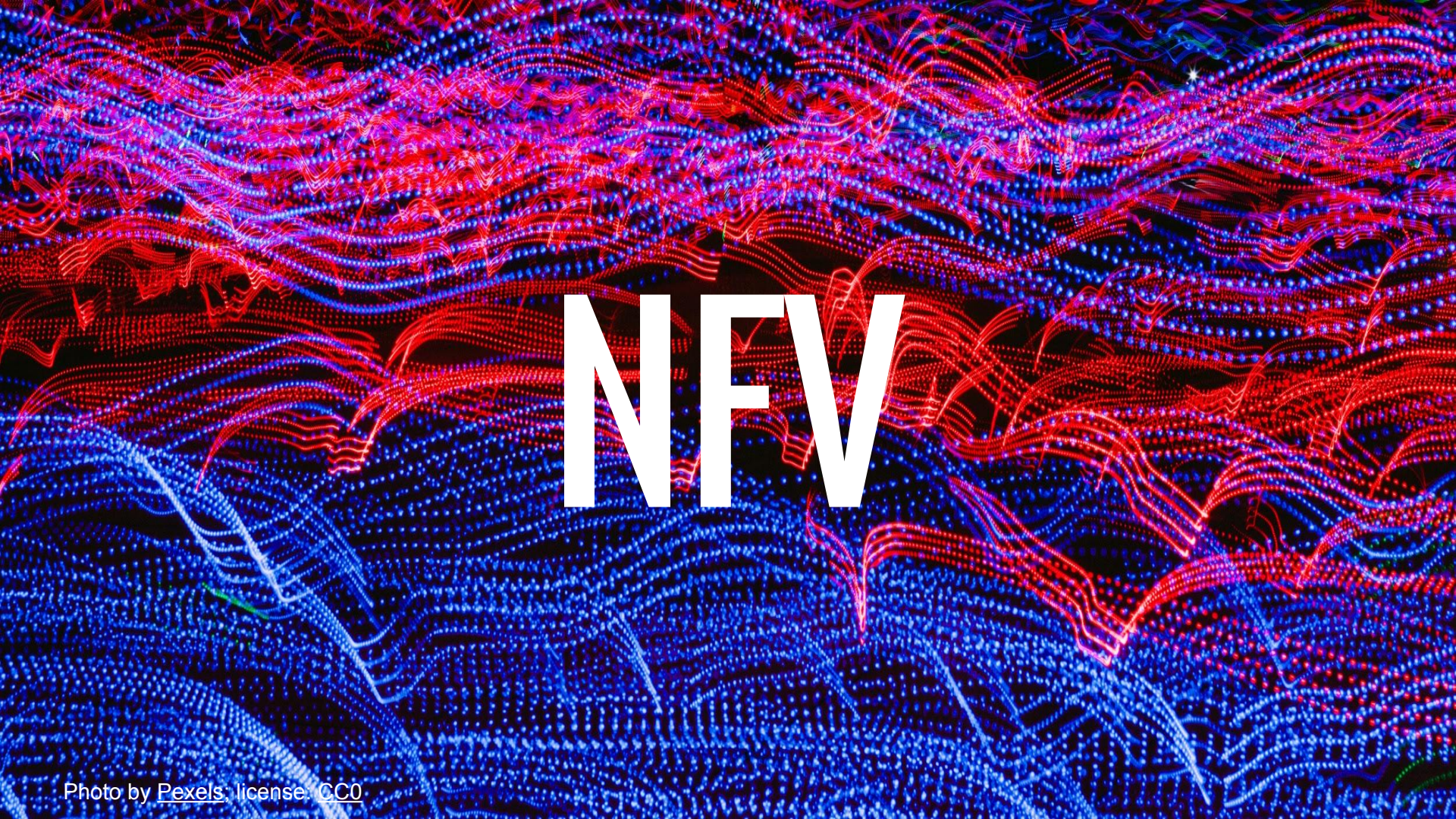


Photo by Pexels, license: CC0

NETWORK FUNCTIONS VIRTUALIZATION

Virtualização + Funções de rede

hummm...

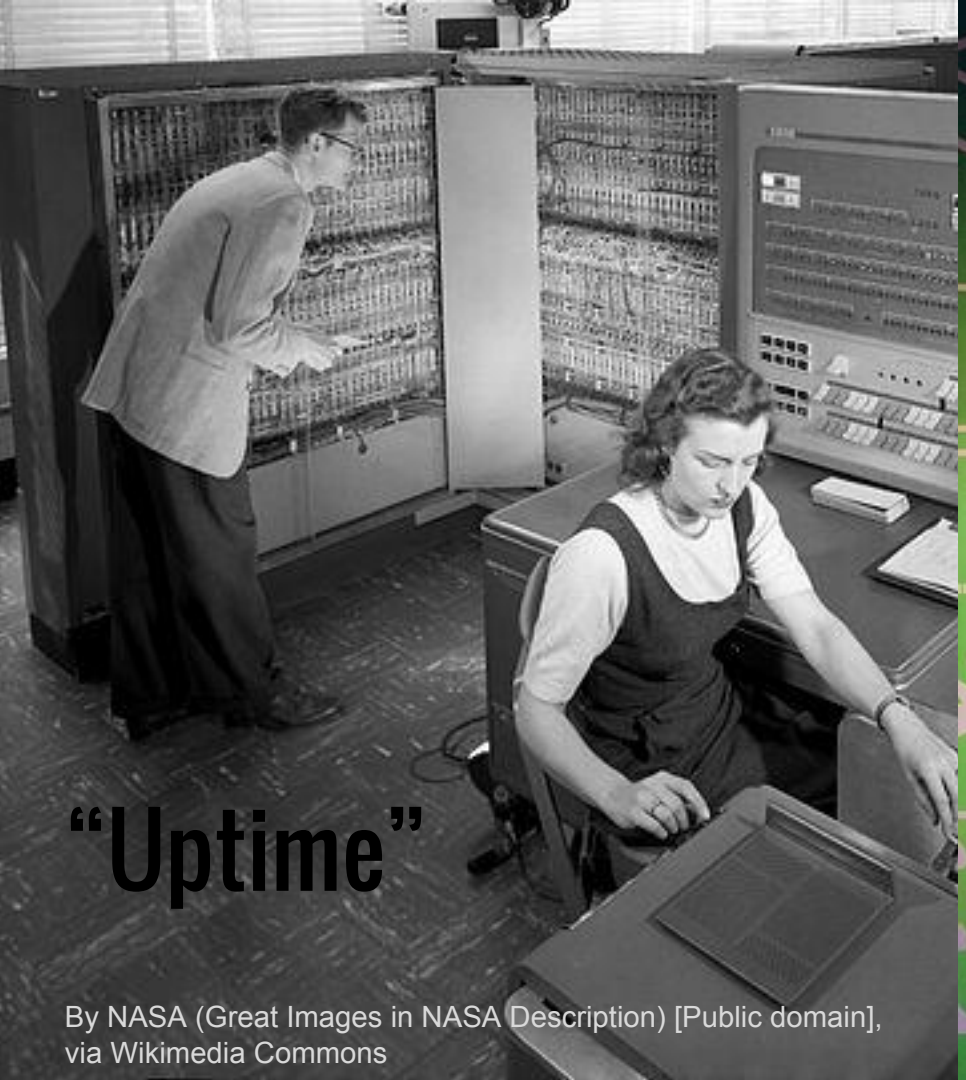
Seria uma máquina virtual?

Sim!!! E não.

NETWORK FUNCTIONS VIRTUALIZATION

NFV is a really simple concept (network services packaged in VM format), what makes it complex is all the infrastructure you need around it.

Ivan Pepelnjak



“Uptime”

By NASA (Great Images in NASA Description) [Public domain], via Wikimedia Commons

“Replacement”

Photo by Nkululeko Masondo; license: CC BY-SA

NETWORK FUNCTIONS VIRTUALIZATION

- Rodar serviços em hardware de uso geral
- Implantar, remover e escalar funções facilmente
- Implantar funções onde elas são necessárias
- Entrega orquestrada e automatizada de serviços

Conteúdo de Cisco, “ NFV - Network Functions Virtualization”

NETWORK FUNCTIONS VIRTUALIZATION

BENEFÍCIOS:

- Menos custos com hardware específico
- suporte ao modelo "*pay-as-you-grow*"
- Menor custo de *datacenter*
- Reduz o tempo de entrega/produção
- Escala

NETWORK FUNCTIONS VIRTUALIZATION

Qual a relação com SDN?

Orquestração -- aplicação de SDN

SDN é essencial para NFV?

Não.

NETWORK FUNCTIONS VIRTUALIZATION

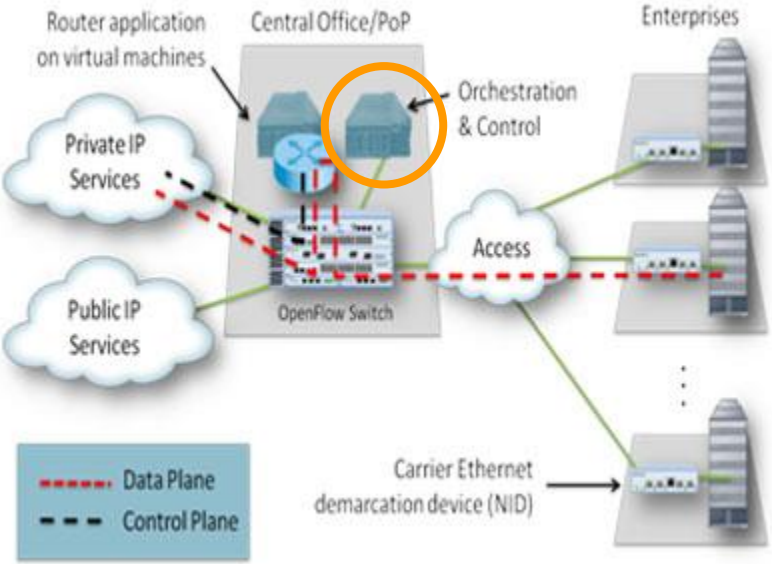


Imagem de SDxCentral, “What is NFV – Network Functions Virtualization – Definition?”

CONTAINERS SÃO:

- (+-) 1 processo -- não é VM
- Finitos
- Imutáveis (cuidado com os dados!)

DOCKER CONTAINERS: Rodando um container

```
## Rodando um bash em um container

$ docker run -ti --rm arbiza/alpine-bash

.

.

.

$ exit
```

DOCKER CONTAINERS: Criando imagens

```
## arquivo: Dockerfile

FROM alpine:latest

RUN apk add --update bash

ENTRYPOINT ["/bin/bash"]
```

```
## No diretório que contém o Dockerfile acima

$ docker build -t <usuário/nome-imagem>:<tag> .
```

DOCKER CONTAINERS: Comandos

docker images

docker ps (-a)

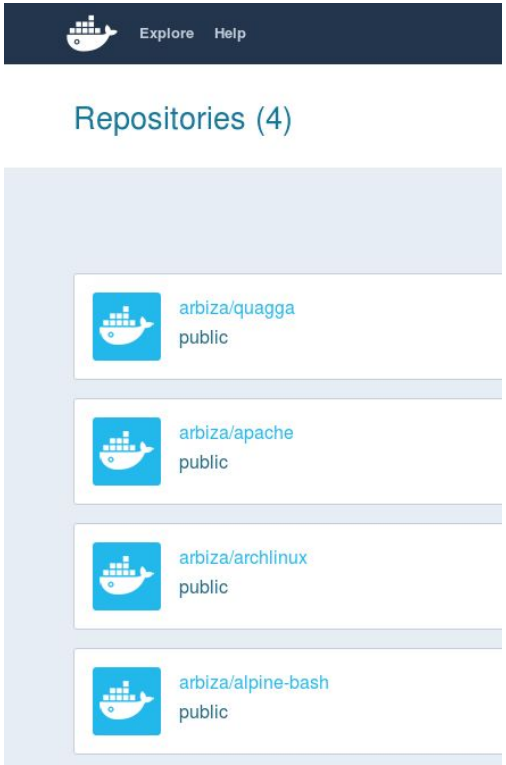
docker inspect ...

docker run [-ti|-d] [--name] [-v] <imagem> comando

docker start/stop/restart

docker commit <container> <nome imagem>:<tag>

DOCKER CONTAINERS: Docker Hub



Endereço: hub.docker.com

- docker push <usuário/imagem>

DOCKER CONTAINERS: Docker Compose

```
## Arquivo docker-compose.yml


version: '2'

services:

     www-volume:
          build: .
          container_name: www-volume
          volumes:
               - "./server-blocks:/etc/nginx/conf.d"
               - "/source/site1:/var/www/site1"
               - "/source/site2:/var/www/site2"


                  # continua ali ->
```

```
...
      www:
            image: nginx:1.10.0-alpine
            expose:
                  - "80"
                  - "443"
            ports:
                  - "80:80"
                  - "443:443"
            restart: always
            container_name: www
            volumes_from:
                  - www-volume
            volumes:
                  - "/etc/nginx"
            depends_on:
                  - www-volume
```

DOCKER CONTAINERS: Docker Compose

```
## No arquivo onde está o docker-compose.yml

$ docker-compose -d up

$ docker-compose down

$ docker-compose restart

$ docker-compose run www
```

Now show me the code!

LAB NO NOTE: Ferramentas

- Open vSwtch
- Docker
- Docker Compose
- ovs-docker
- Ryu Controller
- Terminal “bom” (Terminator, Tmux, etc.)

LAB NO NOTE: Criando uma bridge/switch

```
### TERMINAL ROOT

# ovs-vsctl add-br ovs-sw

# ovs-vsctl set-controller ovs-sw tcp:127.0.0.1:6653

# ovs-vsctl set-fail-mode ovs-sw secure
```

LAB NO NOTE: Verificando bridge-switch

```
### TERMINAL ROOT

### verificando detalhes da bridge

# ovs-vsctl show

#

### verificando flows

# ovs-ofctl dump-flows ovs-sw
```

LAB NO NOTE: Criando containers

```
### TERMINAL USER

### Criando diretório para os arquivos de exemplo

$ mkdir <nome diretório>

$ cd <nome diretório>

$ vim docker-compose.yml
```

LAB NO NOTE: Criando containers

```
### arquivo docker-compose.yml

version: '2'

services:

    host:
    image: arbiza/alpine-bash
    network_mode: "none"
    cap_add:
            - ALL
```

LAB NO NOTE: Criando containers

```
### TERMINAL HOST 1

$ docker-compose run --rm --name h1 host
```

LAB NO NOTE: Criando containers

```
### TERMINAL HOST 2

$ docker-compose run --rm --name h2 host
```

LAB NO NOTE: Verificando containers

```
### TERMINAL USER

## No diretório criado

$ docker-compose ps

## Em qualquer diretório

$ docker ps
```

LAB NO NOTE: Conectando containers

```
### TERMINAL ROOT

# ovs-docker add-port ovs-sw eth0 h1

# ovs-docker add-port ovs-sw eth0 h2
```

LAB NO NOTE: Conectando containers

```
### TERMINAL ROOT

# ovs-vsctl show



### TERMINAL CONTAINERS

# ip link sh
```

LAB NO NOTE: Configurando containers

```
### TERMINAL HOST 1

# ip addr add 2001:db8::1/64 dev eth0

# ip addr add 10.0.0.1/24 dev eth0

# ip addr
```

LAB NO NOTE: Configurando containers

```
### TERMINAL HOST 2

# ip addr add 2001:db8::2/64 dev eth0

# ip addr add 10.0.0.2/24 dev eth0

# ip addr
```

LAB NO NOTE: Configurando containers

```
### TERMINAL HOST 1 - Ping Host 2

### O esperado é que não funcione.

# ping 2001:db8::2/64

# ping 10.0.0.2/24
```

LAB NO NOTE: Instalando o controlador Ryu

Instalar dependências:

- python-eventlet
- python-routes
- python-webob
- python-paramiko
- python-oslo-config

LAB NO NOTE: Instalando o controlador Ryu

Instalar OUTRAS dependências (TALVEZ):

- python-oslo-config
- python-msgpack

LAB NO NOTE: Instalando o controlador Ryu

```
### TERMINAL USER

$ git clone git://github.com/osrg/ryu.git

$ cd ryu

$ sudo python ./setup.py install
```

LAB NO NOTE: Instalando o controlador Ryu

```
### TERMINAL USER

### Rodando no diretório clonado

$ ryu-manager ryu/app/simple_switch.py
```

LAB NO NOTE: Rodando o controlador Ryu

```
### TERMINAL ROOT

### Antes do ping

# ovs-ofctl dump-flows ovs-sw
```

LAB NO NOTE: Configurando containers

```
### TERMINAL HOST 1 - Ping Host 2

### O esperado é que funcione.

# ping 2001:db8::2/64

# ping 10.0.0.2/24
```

LAB NO NOTE: Rodando o controlador Ryu

```
### TERMINAL ROOT

### Depois do ping

# ovs-ofctl dump-flows ovs-sw

cookie=0x0, duration=3.028s, table=0, n_packets=4, n_bytes=472, idle_age=0,
in_port=2,dl_dst=e2:61:d9:55:8e:c1 actions=output:1

cookie=0x0, duration=3.027s, table=0, n_packets=3, n_bytes=354, idle_age=0,
in_port=1,dl_dst=fa:5f:0d:85:b3:8a actions=output:2
```

LAB NO NOTE: O OVS é um switch!

```
## Todos os comandos eu utilizei na prototipação para o MSc.

# ARP: encaminhamento normal
ovs-ofctl add-flow ovs "table=0, priority=100, dl_type=0x0806, actions=normal"

# Access Point: Input e Output
ovs-ofctl add-flow ovs "table=0, priority=10, dl_dst=2e:bb:00:f7:0c:44,
actions=resubmit(,1)"
ovs-ofctl add-flow ovs "table=0, priority=10, dl_src=2e:bb:00:f7:0c:44,
actions=resubmit(,2)"

# Table-miss
ovs-ofctl add-flow ovs "table=3, priority=0 actions=CONTROLLER:65535"
```

LAB NO NOTE: O OVS é um switch!

```
# SSH
ovs-ofctl add-flow ovs "table=3, priority=10, dl_type=0x0800, nw_dst=10.0.0.0/24,
nw_proto=6 , tcp_dst=22, actions=drop"

# Modificação de destino
ovs-ofctl add-flow ovs "table=4, priority=100, dl_type=0x0800,
dl_src=08:00:27:5e:a2:4d, nw_dst=8.8.8.8, nw_proto=6, tp_dst=80,
actions=mod_nw_dst=200.132.0.100"
```

LAB NO NOTE: Limpando tudo!

```
## Remoção da bridge:
# ovs-vsctl del-br ovs-sw

## Containers:
## Apenas sair do terminal ou executar no diretório
## onde está o arquivo docker-compose.yml
$ docker-compose down

## Controlador:
$ Ctrl^C
```

REFERÊNCIAS

Cisco, " NFV - Network Functions Virtualization", Online.
http://www.cisco.com/c/en/us/solutions/service-provider/network-functions-virtualization-nfv/index.html

Guto Carvalho, "O que é DevOps afinal?", 2013. Online.
http://gutocarvalho.net/octopress/2013/03/16/o-que-e-um-devops-afinal/

H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, "OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks," in Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific, 2012, pp. 1–8.

L. M. R. Arbiza, L. M. R. Tarouco, L. M. Bertholdo, and L. Z. Granville, "SDN-Based Service Delivery in Smart Environments," in Intelligent Distributed Computing IX, P. Novais, D. Camacho, C. Analide, A. E. F. Seghrouchni, and C. Badica, Eds. Guimarães, Portugal: Springer International Publishing, 2016, pp. 475–484.

L. M. R. Arbiza, L. M. Bertholdo, C. R. P. dos Santos, L. Z. Granville, and L. M. R. Tarouco, "Refactoring Internet of Things Middleware Through Software-defined Network," in Proceedings of the 30th Annual ACM Symposium on Applied Computing, 2015, pp. 640–645.

REFERÊNCIAS

L. M. R. Arbiza, “SDN no contexto de IoT : refatoração de middleware para monitoramento de pacientes crônicos baseada em software-defined networking,” Universidade Federal do Rio Grande do Sul, 2016.

L. M. R. Tarouco, L. M. Bertholdo, L. Z. Granville, L. M. R. Arbiza, F. Carbone, M. Marotta, and J. J. C. de Santanna, “Internet of Things in Healthcare : Interoperatibility and Security Issues,” in IEEE International Conference on Communications, International Workshop on Mobile Consumer Health Care Networks, Systems and Services, 2012, pp. 6121–6125.

M. Chetty and N. Feamster, “Refactoring network infrastructure to improve manageability: a case study of home networking,” SIGCOMM Comput. Commun. Rev., vol. 42, no. 3, pp. 54–61, 2012.

N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

“OpenFlow Switch Specication: Version 1.3.2.” The Open Network Foundation, ONF, 2013.

REFERÊNCIAS

RouteFlow, http://routeflow.github.io/RouteFlow/

R. Gomes and L. A. Bianchin. " Docker para desenvolvedores", 2016. Online. https://leanpub.com/dockerparadesenvolvedores

S. Noble. "Network Function Virtualization or NFV Explained", 2015. Online. Network Function Virtualization or NFV Explained

"SDX, A Software Defined Internet Exchange Point". Online. http://sdx.cs.princeton.edu/

SDxCentral. "What is NFV – Network Functions Virtualization – Definition?". Online. https://www.sdxcentral.com/nfv/definitions/whats-network-functions-virtualization-nfv/

"Software-Defined Networking: The New Norm for Networks." The Open Network Foundation, p. 12, 2012.

REFERÊNCIAS

“Using all wireless networks around me”. Stanford's OpenFlow Channel on YouTube, 2010. https://www.youtube.com/watch?v=ov1DZYINg3Y

Y. Yiakoumis, K.-K. Yap, S. Katti, G. Parulkar, and N. McKeown, “Slicing Home Networks,” in Proceedings of the 2Nd ACM SIGCOMM Workshop on Home Networks, 2011, pp. 1–6.


?
KEEP
CALM
AND
ASK
QUESTIONS

CONTATO

Lucas Arbiza

lucas@arbiza.com.br

LICENSE

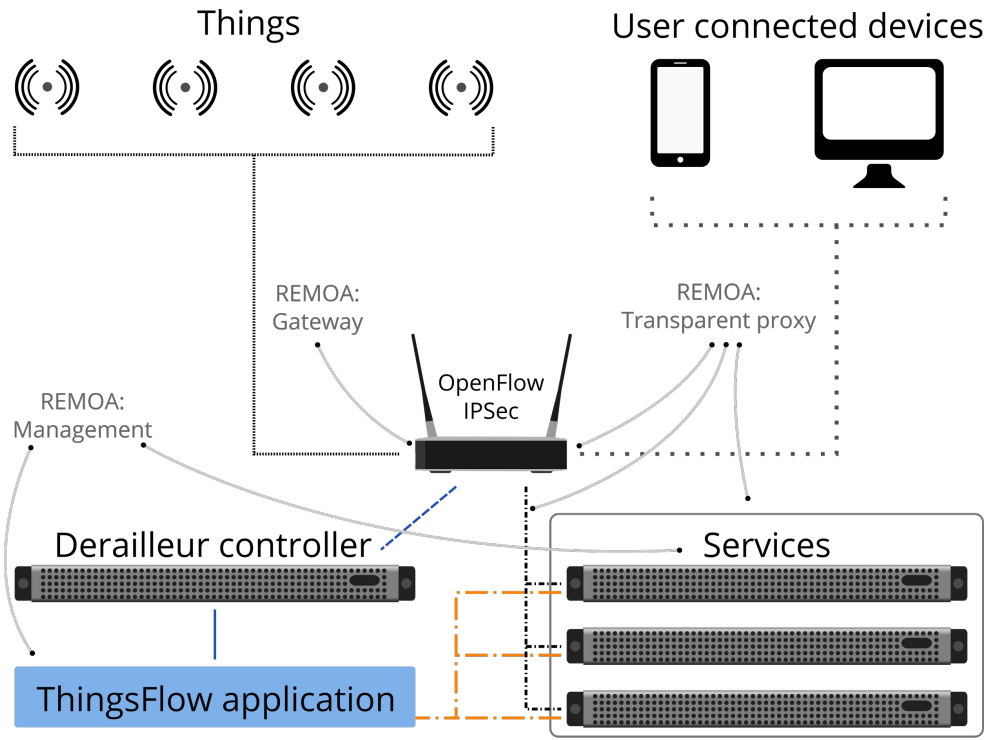
Slides backup

OPENFLOW

```
struct ofp_packet_in {
    struct ofp_header header;
    uint32_t buffer_id;      /* ID assigned by datapath. */
    uint16_t total_len;      /* Full length of frame. */
    uint8_t reason;          /* Reason packet is being sent (one of OFPR_*) */
    uint8_t table_id;        /* ID of the table that was looked up */
    uint64_t cookie;         /* Cookie of the flow entry that was looked up. */
    struct ofp_match match;  /* Packet metadata. Variable size. */
    //uint8_t pad[2];        /* Align to 64 bit + 16 bit */
    //uint8_t data[0];       /* Ethernet frame */
};
OFP_ASSERT(sizeof(struct ofp_packet_in) == 32);
```

EXEMPLOS: SDN no cotexto de IoT

REFATORAÇÃO



EXEMPLOS: SDN no cotexto de IoT

RECURSOS UTILIZADOS

CONTROLADOR E APLICAÇÃO:

- C++(11)
- Boost++
- libfluid

ACCESS POINT:

- OpenWRT
- Open vSwitch