Redes de Computadores

SDN: Sistemas Operacionais de Rede

Prof. Rodrigo de Souza Couto

Bibliografia

- Esta aula é baseada nos seguintes trabalhos:
  - [1] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig. *Software-Defined Networking: A Comprehensive Survey*. Proceedings of the IEEE, 2015.
  - [2]N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an operating system for networks", ACM SIGCOMM Computer Communication Review, 2008.

Bibliografia

- Esta aula é baseada nos seguintes trabalhos (cont.):
  - [3] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Shenker, S., "*Onix: A Distributed Control Platform for Large-scale Production Networks*", USENIX OSDI, 2010.
  - [4] Apresentação disponível em https://www.usenix.org/legacy/events/osdi10/stream/koponen/index.html

Visão Geral da Arquitetura SDN

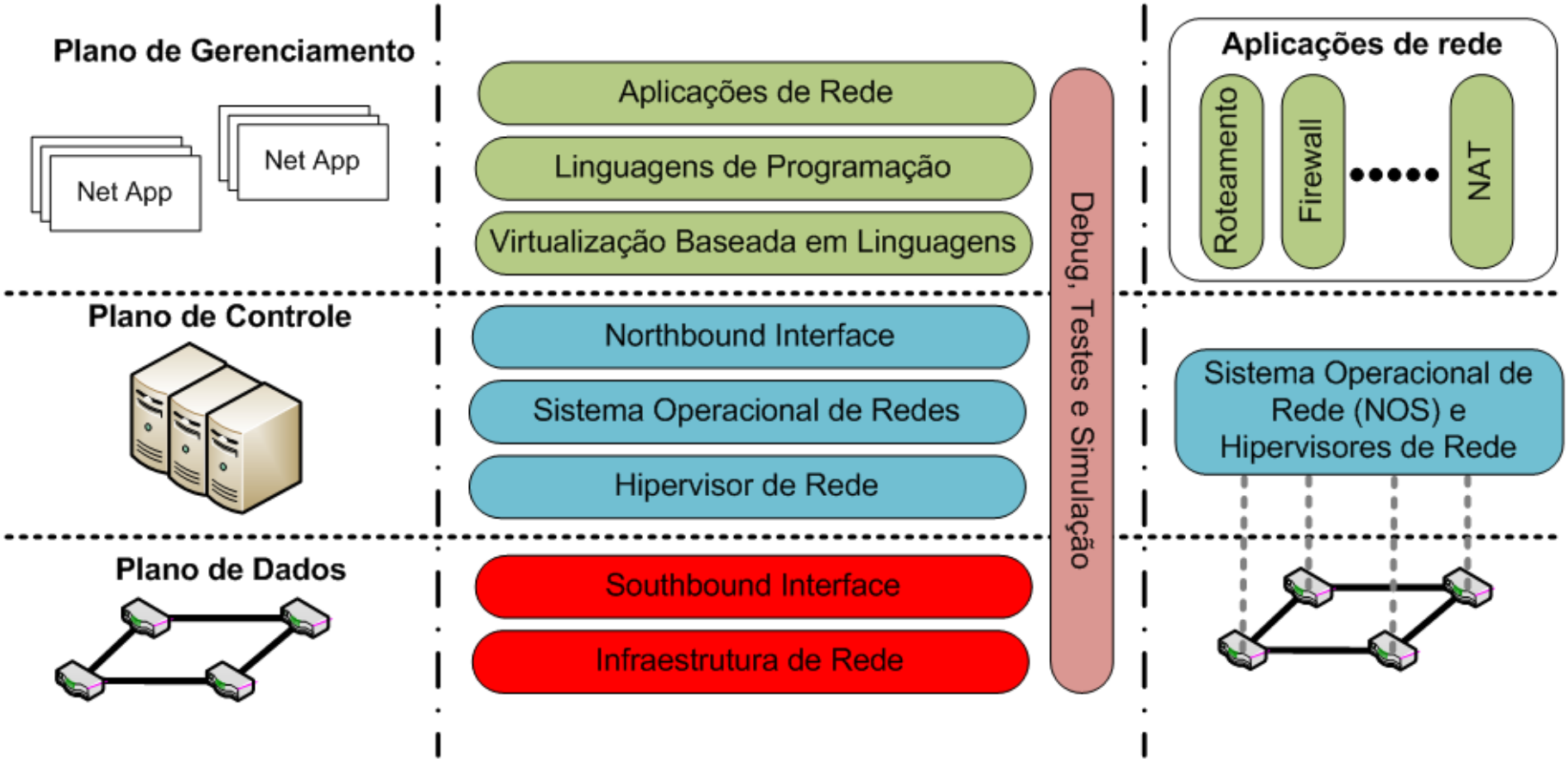


Figura adaptada de [1]

Visão Geral da Arquitetura SDN

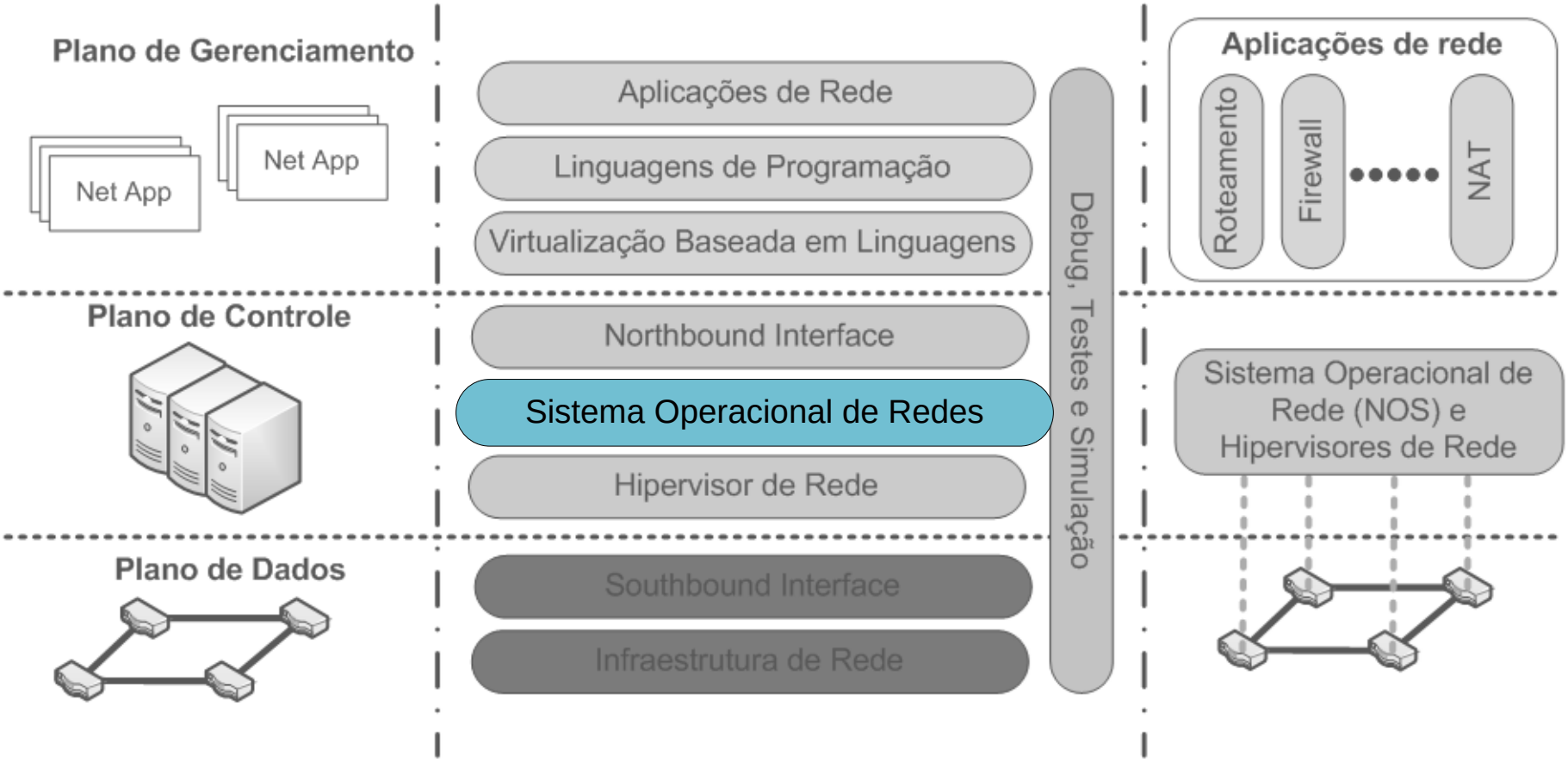


Figura adaptada de [1]

Sistema Operacional de Máquina

- Fornece abstrações para as aplicações acessarem o *hardware*
  - APIs de programação de alto nível
- Gerencia o acesso ao *hardware*
  - Uso do disco rígido, adaptador de rede, CPU e memória
- Facilita o desenvolvimento e a portabilidade de aplicações
  - Tarefas complicadas são realizadas pelo S.O.

NOS – Network Operating System

- Provê uma API para gerenciar a rede
- Permite que operações de gerenciamento de rede executam como programas centralizados
  - Programas se comunicam com o NOS
  - Por exemplo, decisões de roteamento podem ser calculadas com o conhecimento da topologia
- Permite que programas sejam escritos com abstrações de alto nível
  - Por exemplo, nome de usuário e de host ao invés de IP e MAC
  - Permite que aplicações de gerenciamento sejam independentes da tecnologia de rede (Ethernet, WiFi, etc.)

Controladores Centralizados

- Entidade única que gerencia toda a rede
- Vantagem
  - Arquitetura simples
- Desvantagens
  - Ponto único de falha
  - Limitações da escalabilidade
    - Um único controlador pode não conseguir gerenciar redes muito grandes
    - Alguns controladores aplicam mecanismos de multithread para melhorar desempenho
- Exemplos
  - NOX, NOX-MT, Maestro, Beacon e Floodlight

Controladores Distribuídos

- Diversos controladores gerenciam a rede
- Vantagens
  - Mais tolerantes a falhas
  - Maior capacidade de escalar
- Desvantagem
  - Complexidade da infraestrutura
- Exemplos
  - Onix, Hyperflow, HP VAN SDN, ONOS, SMaRt-Light

Controladores Distribuídos Tipos

- Cluster de nós
  - Diversos controladores em um mesmo local
    - P.ex. para oferecer alta vazão em rede de centro de dados
- Nós fisicamente distribuídos
  - Controladores distribuídos por uma WAN
    - P.ex. para oferecer tolerância a falhas
- Híbrido
  - Clusters de nós fisicamente distribuídos
    - P.ex. provedor de nuvem quere gerencia múltiplos centros de dados

Controladores Distribuídos - Consistência

- Fraca
  - Informações (p.ex. novos fluxos) da rede são trocadas eventualmente entre controladores
    - Existe um período de tempo no qual os controladores possuem visões diferentes da rede
  - Implementada pela maioria dos controladores
- Forte
  - Qualquer alteração na rede é informada a todos os controladores
  - Gera sobrecarga na rede, mas torna as aplicações mais simples
  - Implementada pelo Onix, ONOS e SMaRtLight

Controladores Distribuídos Independentes

- Cada controlador gerencia um segmento da rede
- Reduzem o impacto da falha de um único Controlador
  - Apenas parte da rede perderá o gerenciamento
- Possibilita o uso de controladores centralizados

Funções de núcleo oferecidas paras as aplicações

- Gerenciador da topologia
  - Coleta a topologia da rede, oferecendo uma visão única para as aplicações
- Gerenciador de estatísticas
  - Coleta estatísticas dos fluxos da rede
- Gerenciador do dispositivo

Funções de núcleo oferecidas paras as aplicações (cont.)

- Encaminhamento por caminhos mais curtos
- Gerenciador de Notificações
  - Recebe eventos como mudanças de estado e alarmes
- Mecanismos de segurança
  - Isolamento entre as aplicações
    - Fluxo de aplicaçõs de menor prioridade não devem alterar fluxos de aplicações de maior prioridade

APIs Eastbound e Westbound

- Utilizadas para comunicação entre controladores distribuídos
  - Idealmente deveriam ser independentes do NOS

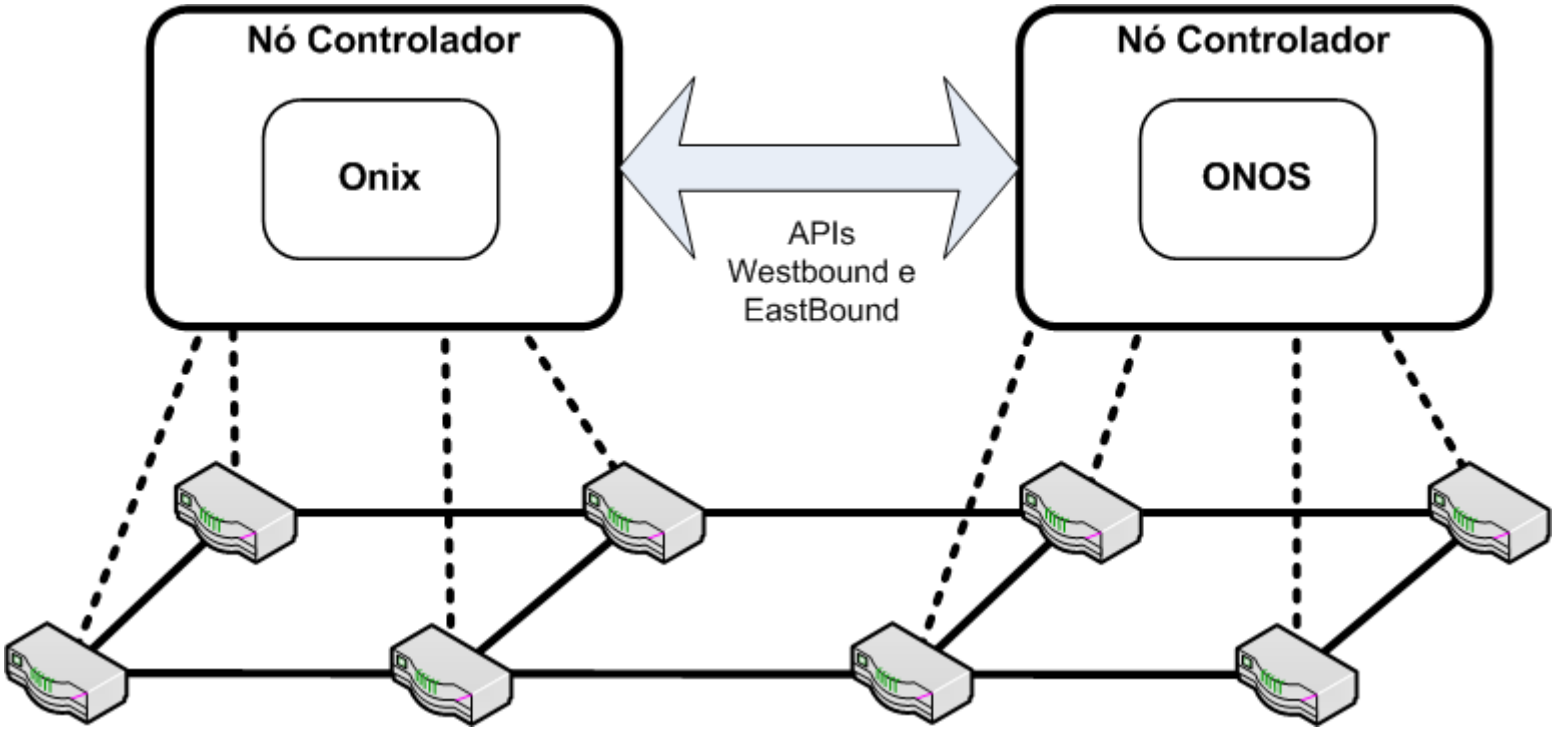


Figura adaptada de [1]

Controlador NOX

- Primeiro controlador OpenFlow
- Opera de forma centralizada
- Código aberto
  - Licença GPLv3
- Interface C++
- Executa em controladores Linux
- Suporta OpenFlow 1.0

NOX - Componentes

- Controlador
  - NOX
  - Aplicações
- Visão da Rede
  - Topologia
  - Localização de usuários
  - Entre outras informações coletadas dos comutadores

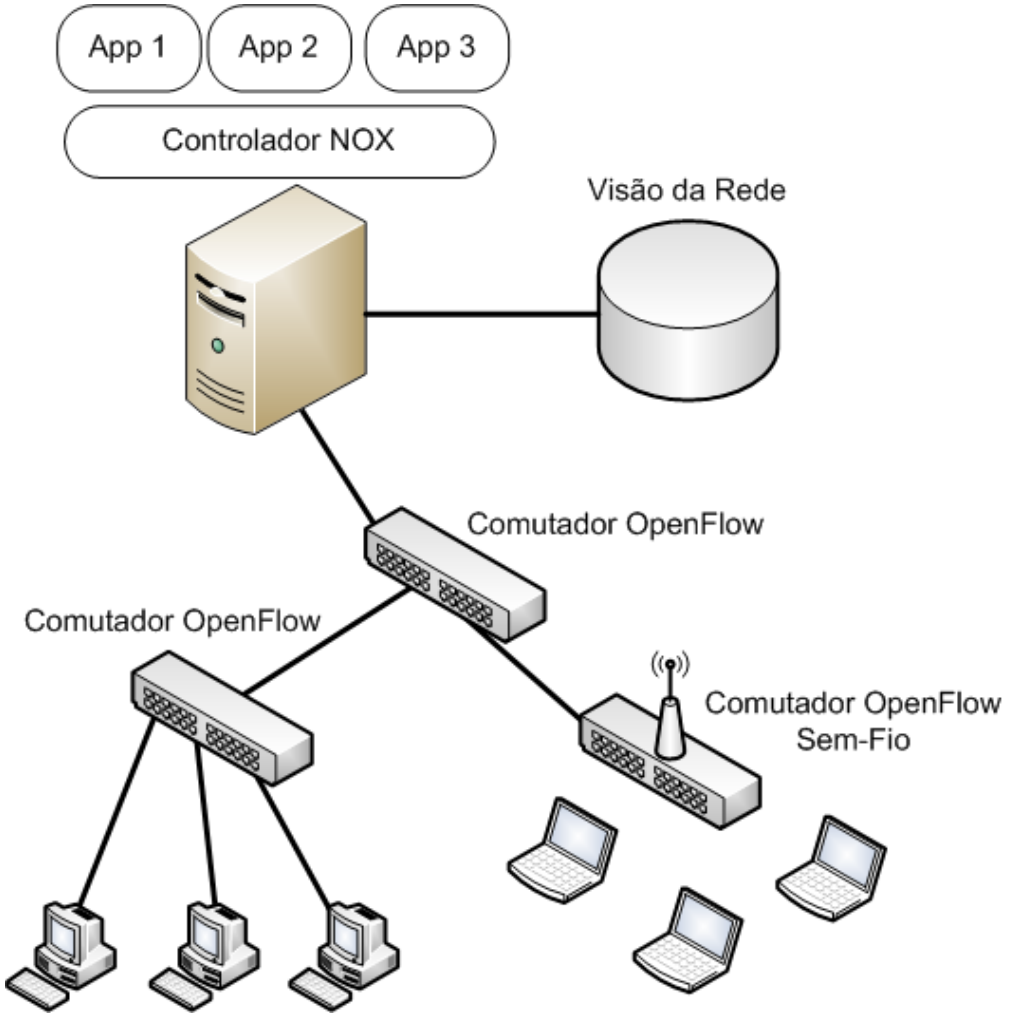
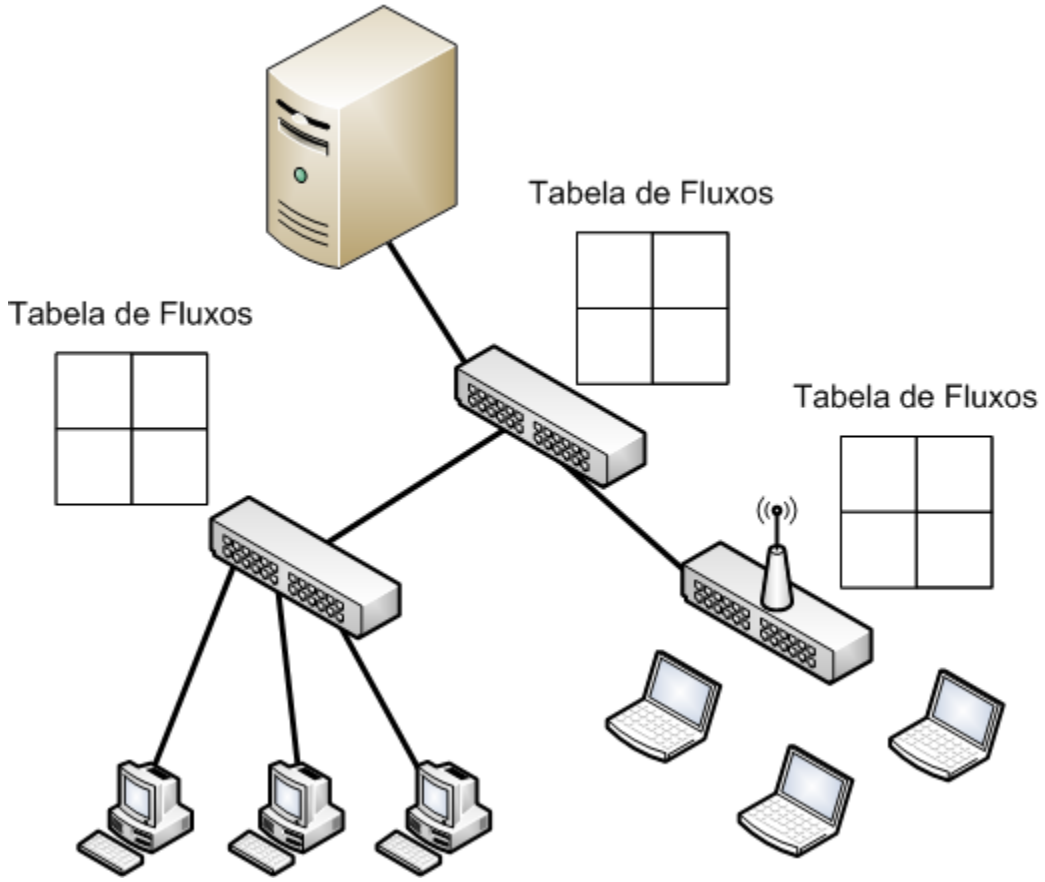


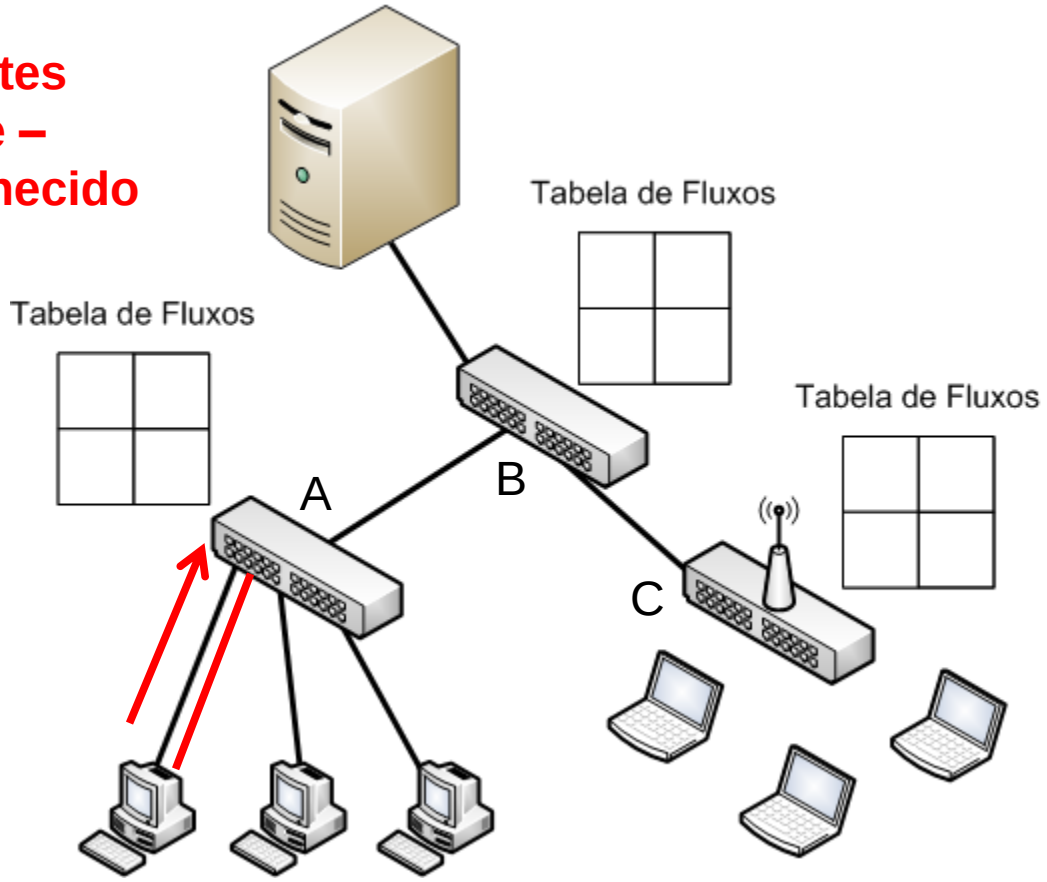
Figura adaptada de [1]

Exemplo de funcionamento do NOX



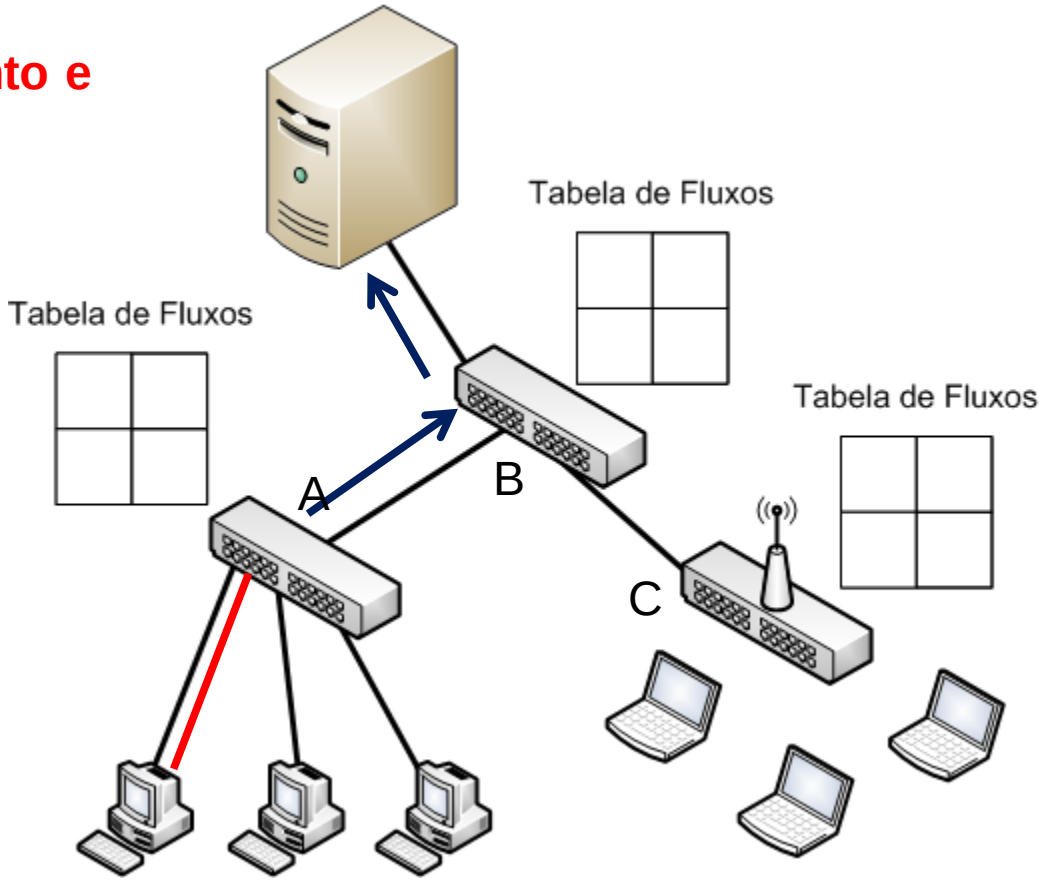
Exemplo de funcionamento do NOX

Envio de Pacotes por um Cliente – Fluxo Desconhecido



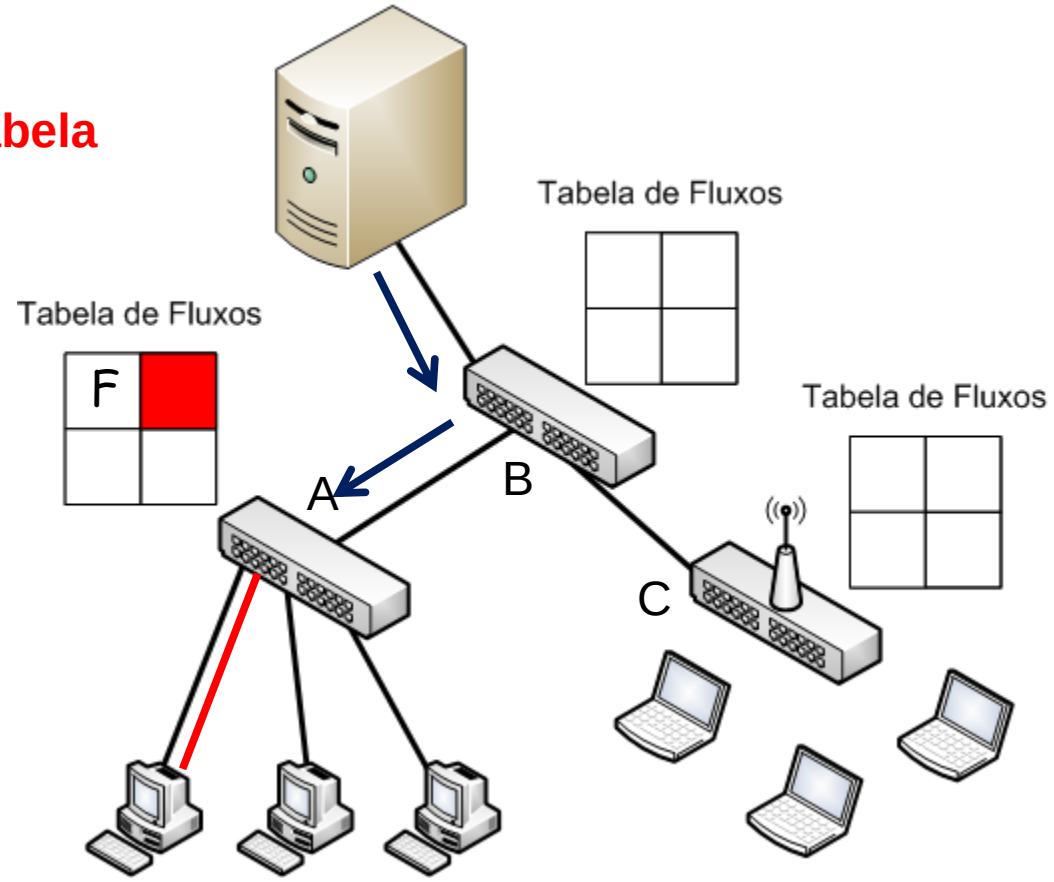
Exemplo de funcionamento do NOX

Encapsulamento e envio para o Controlador



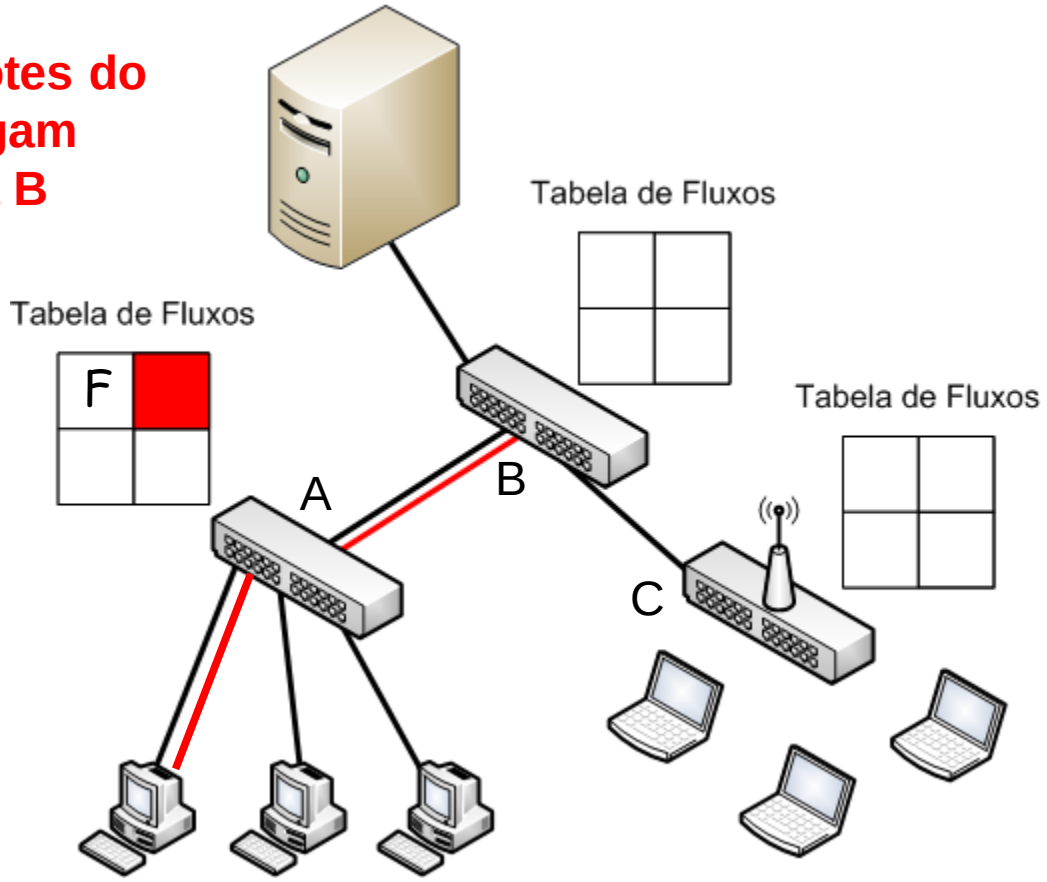
Exemplo de funcionamento do NOX

Controlador configura a Tabela de Fluxos do Comutador A



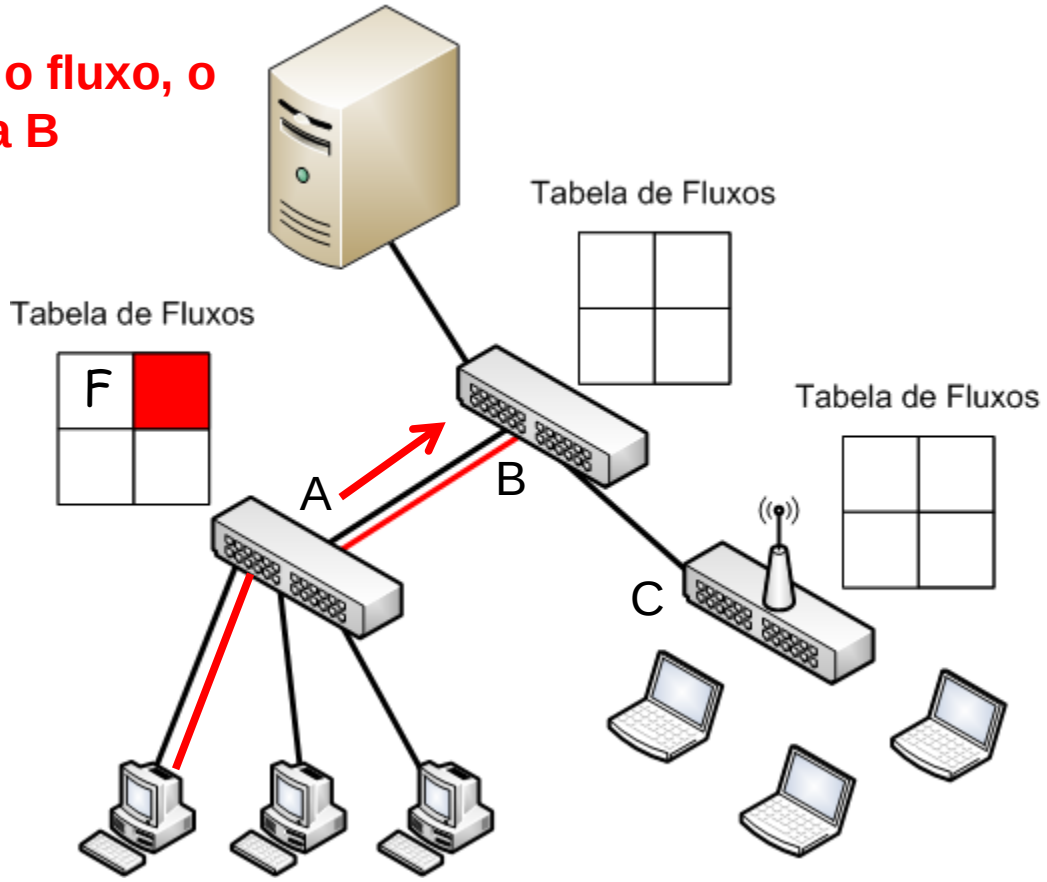
Exemplo de funcionamento do NOX

Todos os pacotes do fluxo que chegam em A irão para B



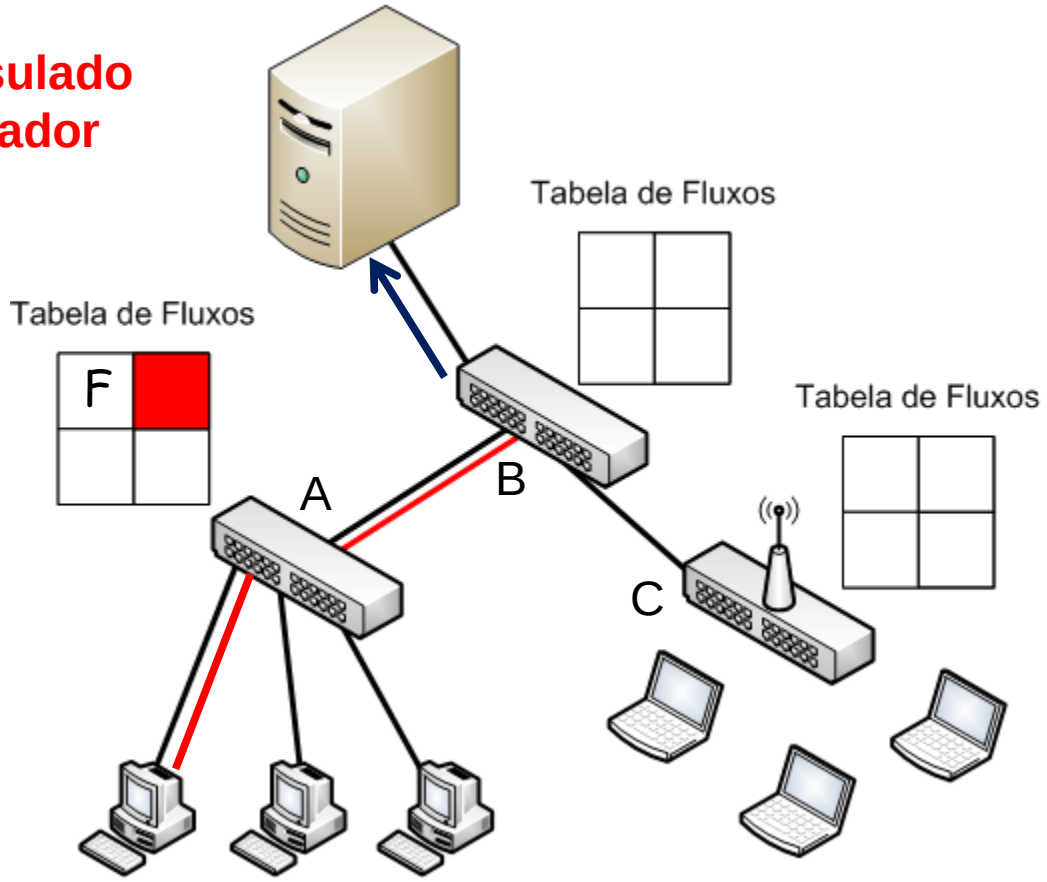
Exemplo de funcionamento do NOX

Após definido o fluxo, o pacote irá para B



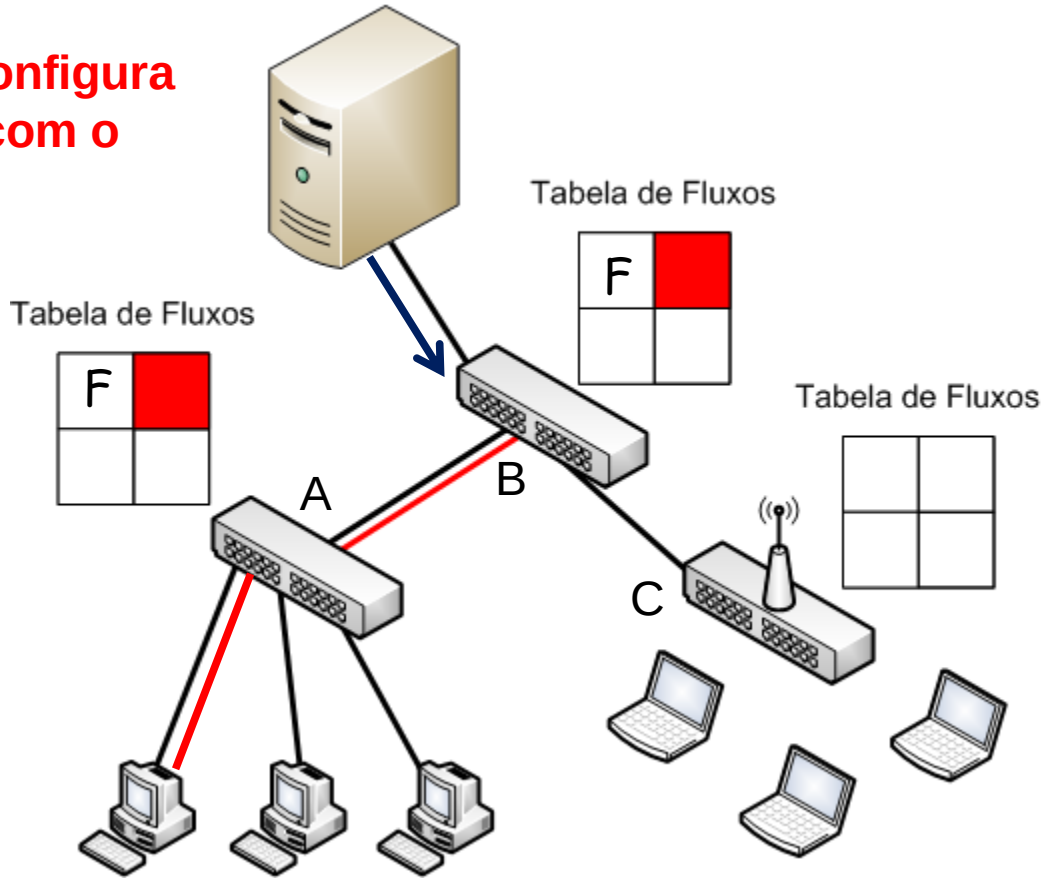
Exemplo de funcionamento do NOX

Pacote encapsulado para o Controlador



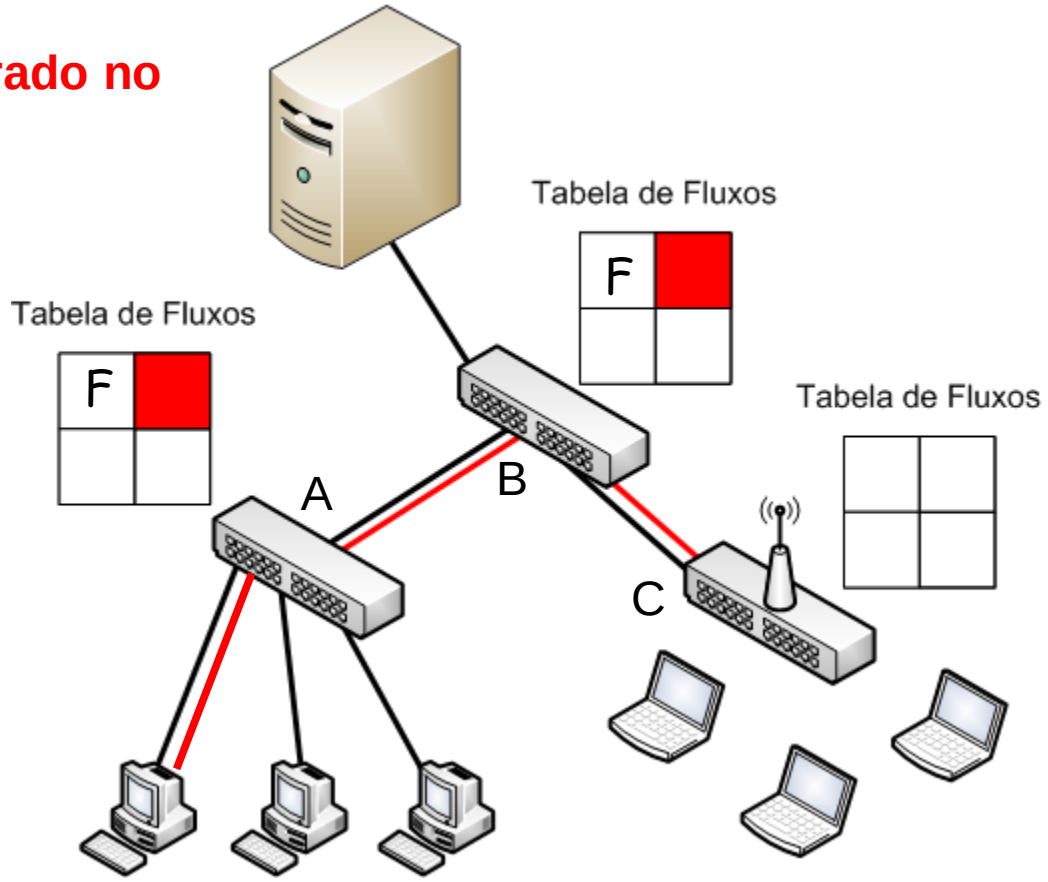
Exemplo de funcionamento do NOX

Controlador configura comutador B com o fluxo



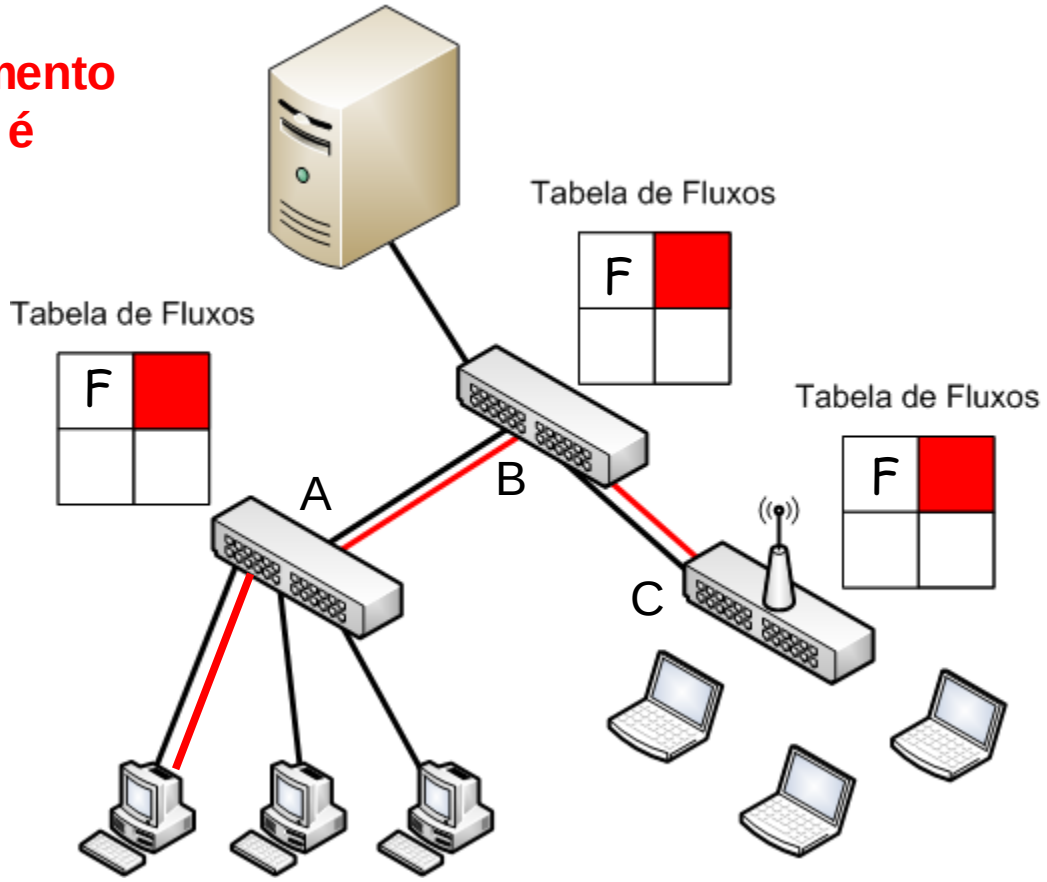
Exemplo de funcionamento do NOX

Fluxo configurado no Comutador B



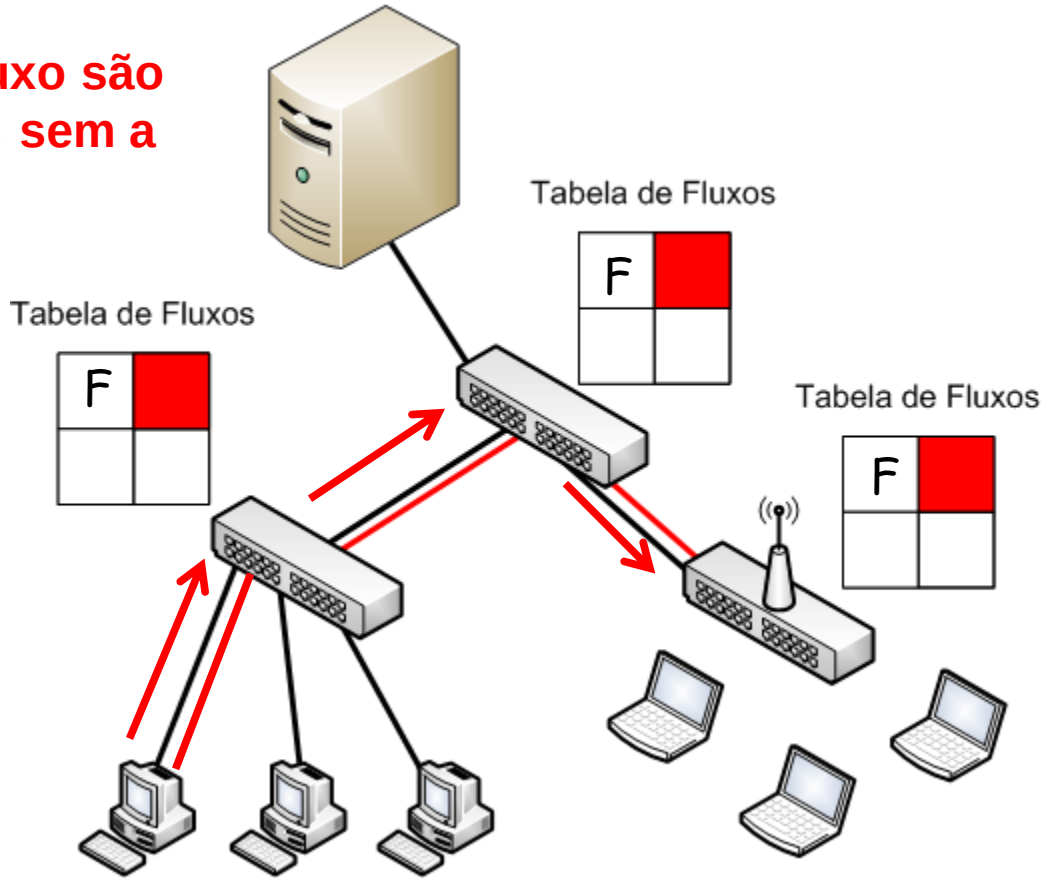
Exemplo de funcionamento do NOX

Após procedimento semelhante, C é configurado



Exemplo de funcionamento do NOX

Pacotes do Fluxo são encaminhados sem a utilização do Controlador

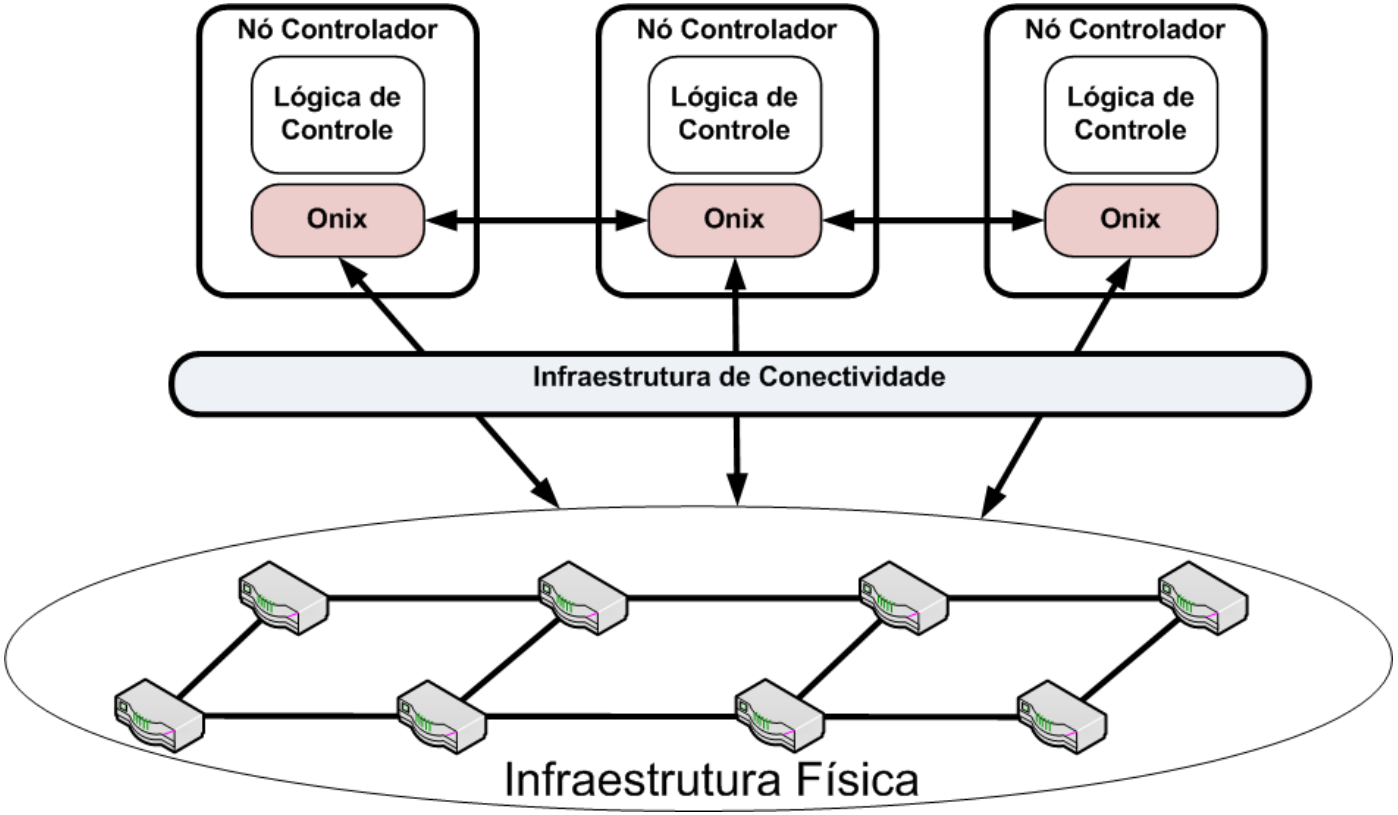


Controlador POX

- Baseado no NOX
- Opera de forma centralizada
- Código aberto
  - Licença GPLv3
- Interface Python
  - Mais simples, porém com pior desempenho que a interface C++ do NOX
- Executa em controladores Linux, Windows e MAC OS
- Suporta OpenFlow 1.0

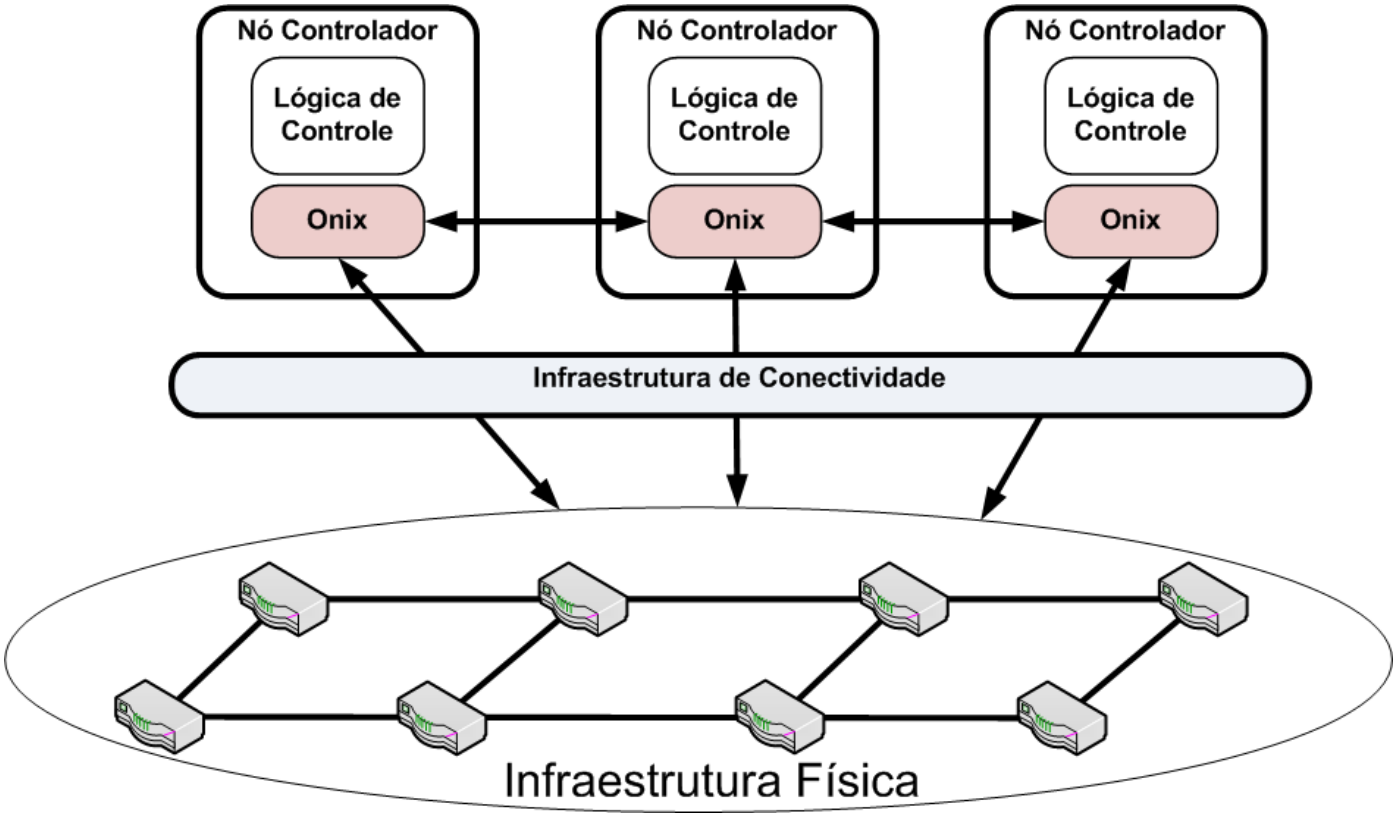
Controlador Onix

- Controlador distribuído
  - Lógica de controle centralizada é um sistema distribuído



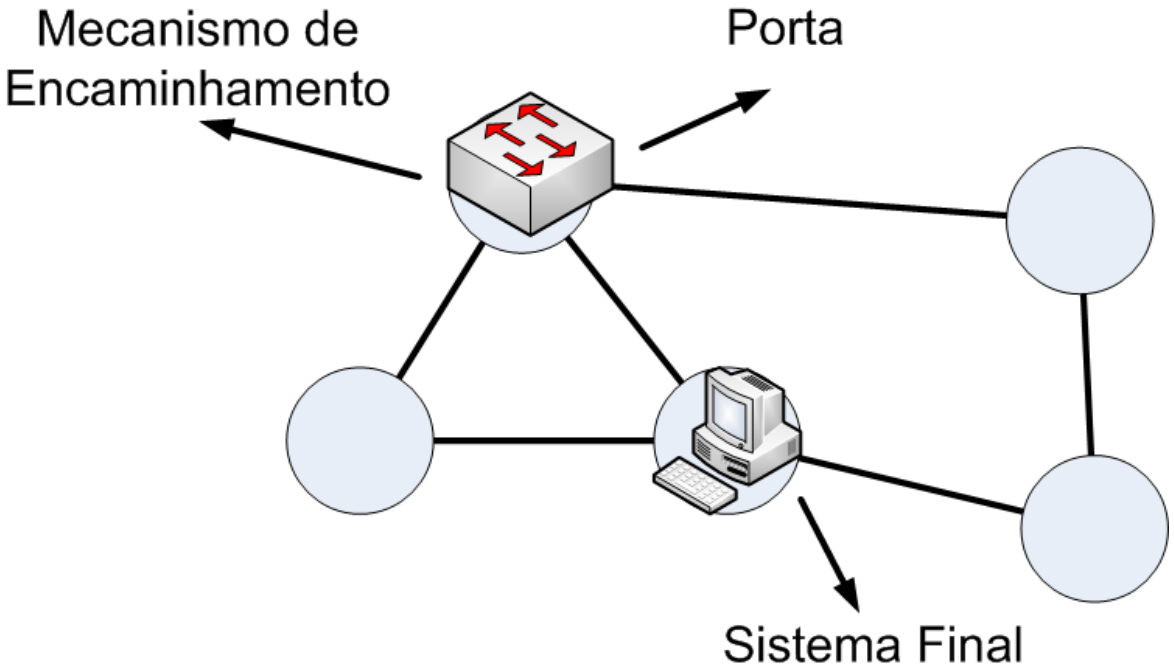
Controlador Onix

- Onix oferece uma API para manipular a rede
- Instâncias do Onix se coordenam para trocar a visão da rede



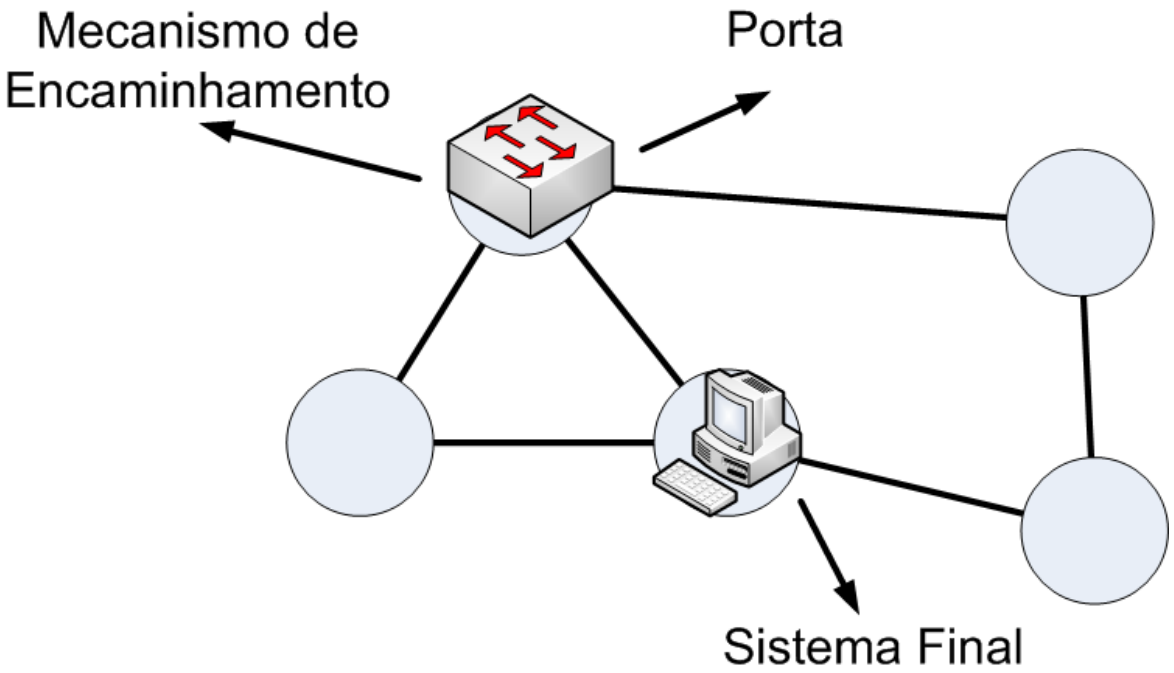
Onix API

- Desenvolvedores programam a partir de um grafo
- Cada nó representa uma entidades
  - Física ou lógicas (figura apresenta apenas físicas)
  - Aplicação pode realizar diversas ações sobre a entidade



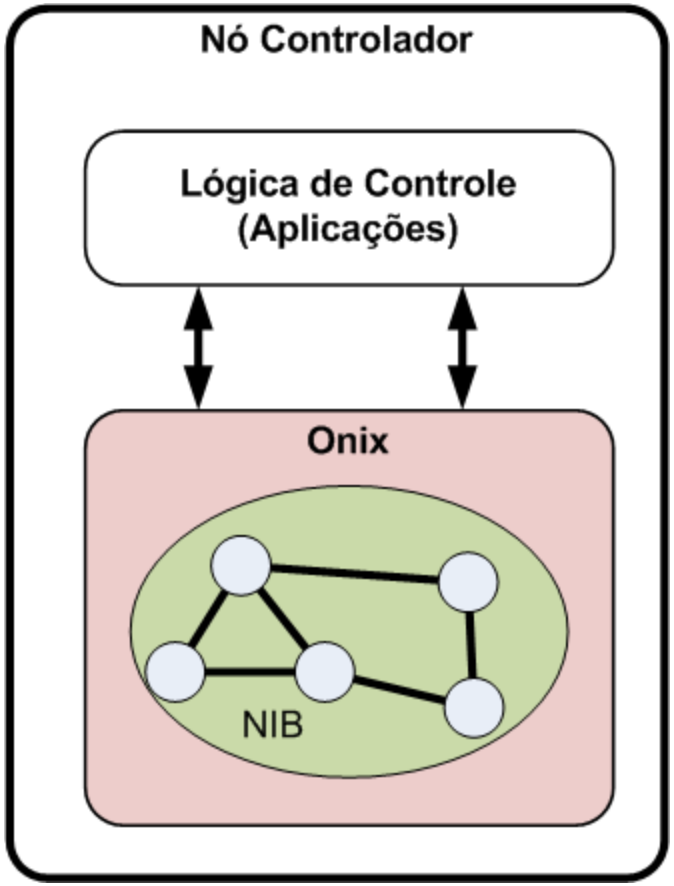
Onix API

- Exemplo de ações nos mecanismos de encaminhamento
  - Escrever entrada na tabela de fluxos
  - Listar as portas disponíveis
  - Registrar para receber atualizações



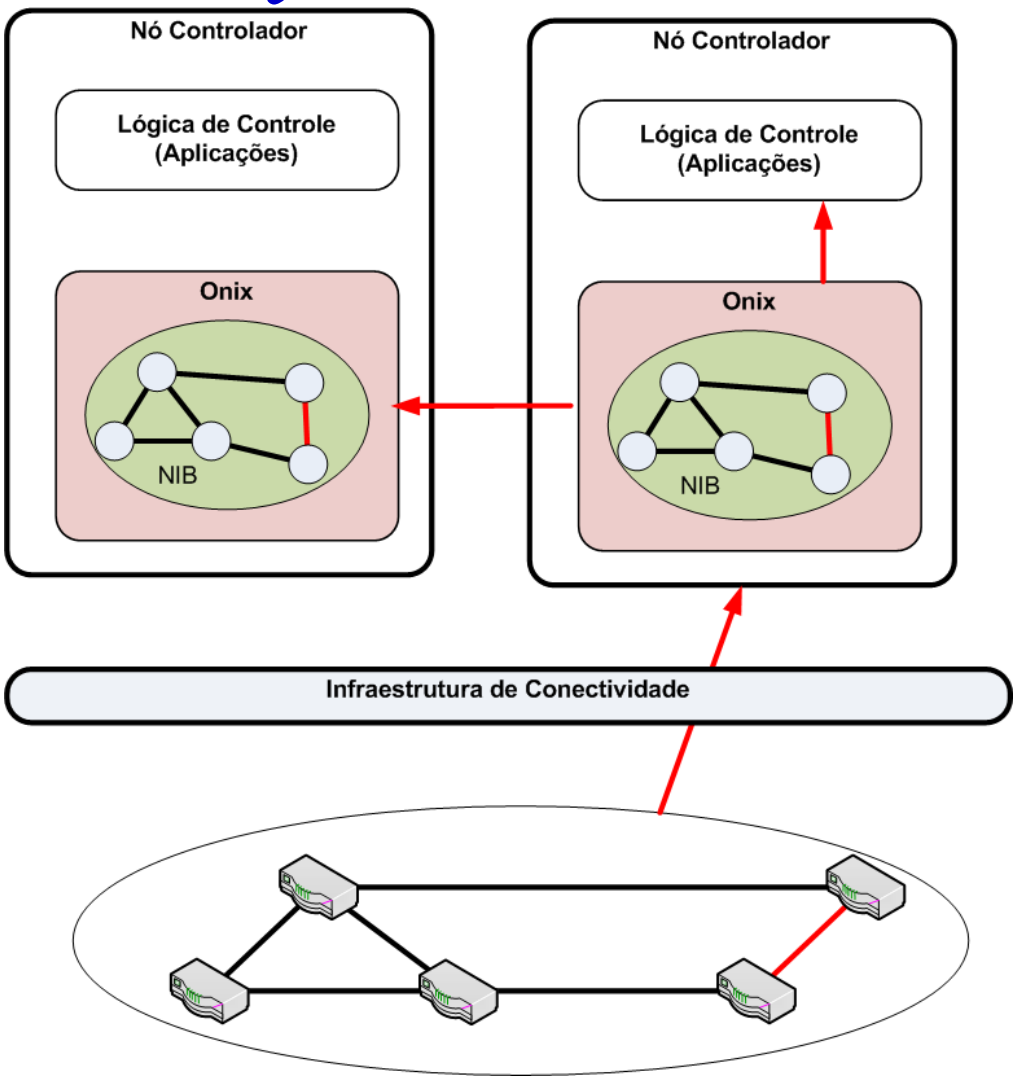
NIB (Network Information Base)

- Núcleo do modelo de controle do Onix
- Base da distribuição alcançada
- Aplicações de rede são desenvolvidas com comandos de escrita e leitura à NIB
- Onix provê escalabilidade e resiliência distribuindo informações da NIB entre múltiplas instâncias (controladores)



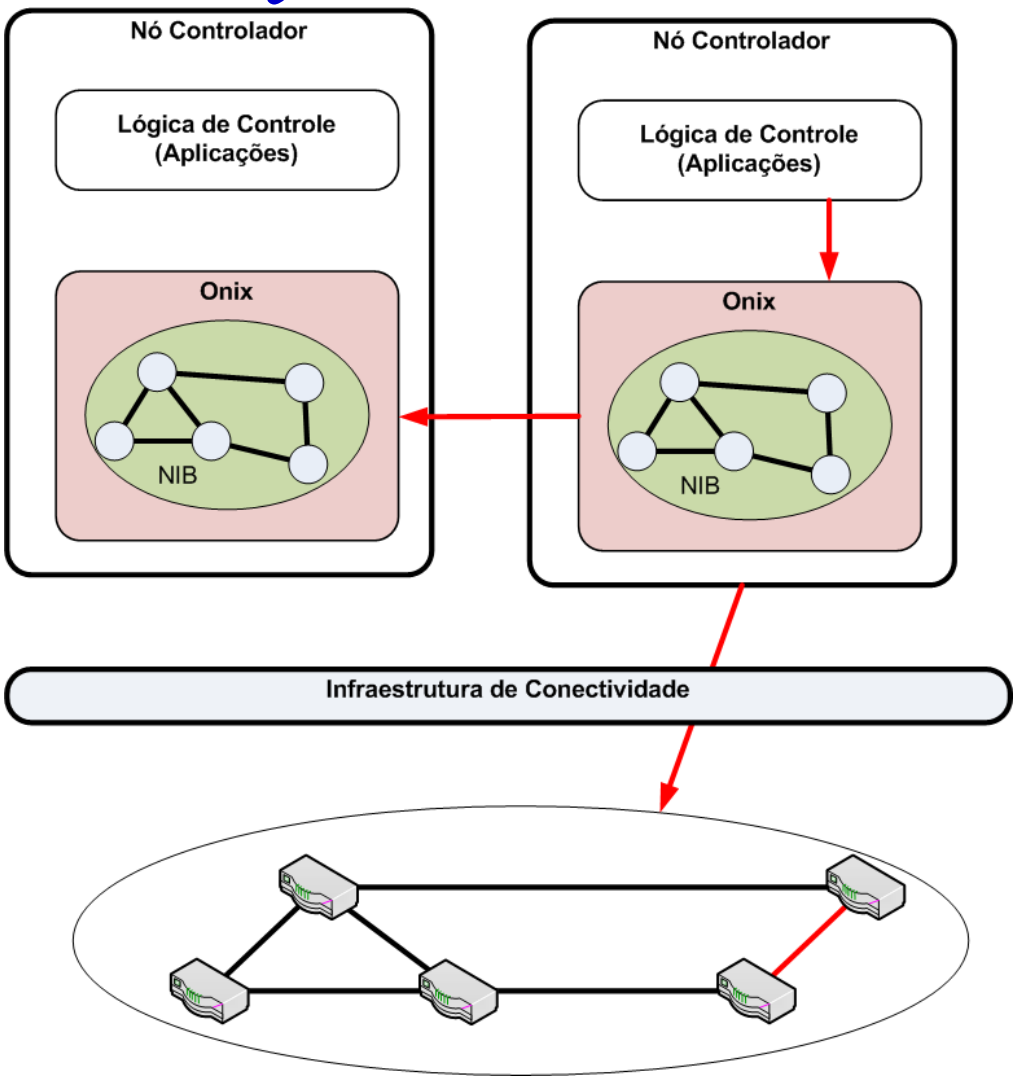
NIB (Network Information Base)

- Mudanças na rede são escritas na NIB do controlador
  - Também são propagadas para outros nós
  - Aplicações eventualmente recebem informação sobre a mudança



NIB (Network Information Base)

- Aplicações alteram a NIB
  - Alterações devem ser traduzidas para a infraestrutura física
  - Alterações podem ser propagadas para outras NIBs
- Tradução é realizada para uma SouthBound API
  - OpenFlow
  - OVSDB



Distribuição da NIB

- Dois tipos de consistências são suportados
  - Forte
    - Informações mais críticas e/ou que mudam menos frequentemente
      - P.ex. Topologia entre os comutadores
  - Fraca
    - Informações que mudam com maior frequencia e/ou que são menos críticas
      - P.ex. Mapeamento entre IP e MAC
- Aplicações definem os tipos de consistência para os estados que manipulam

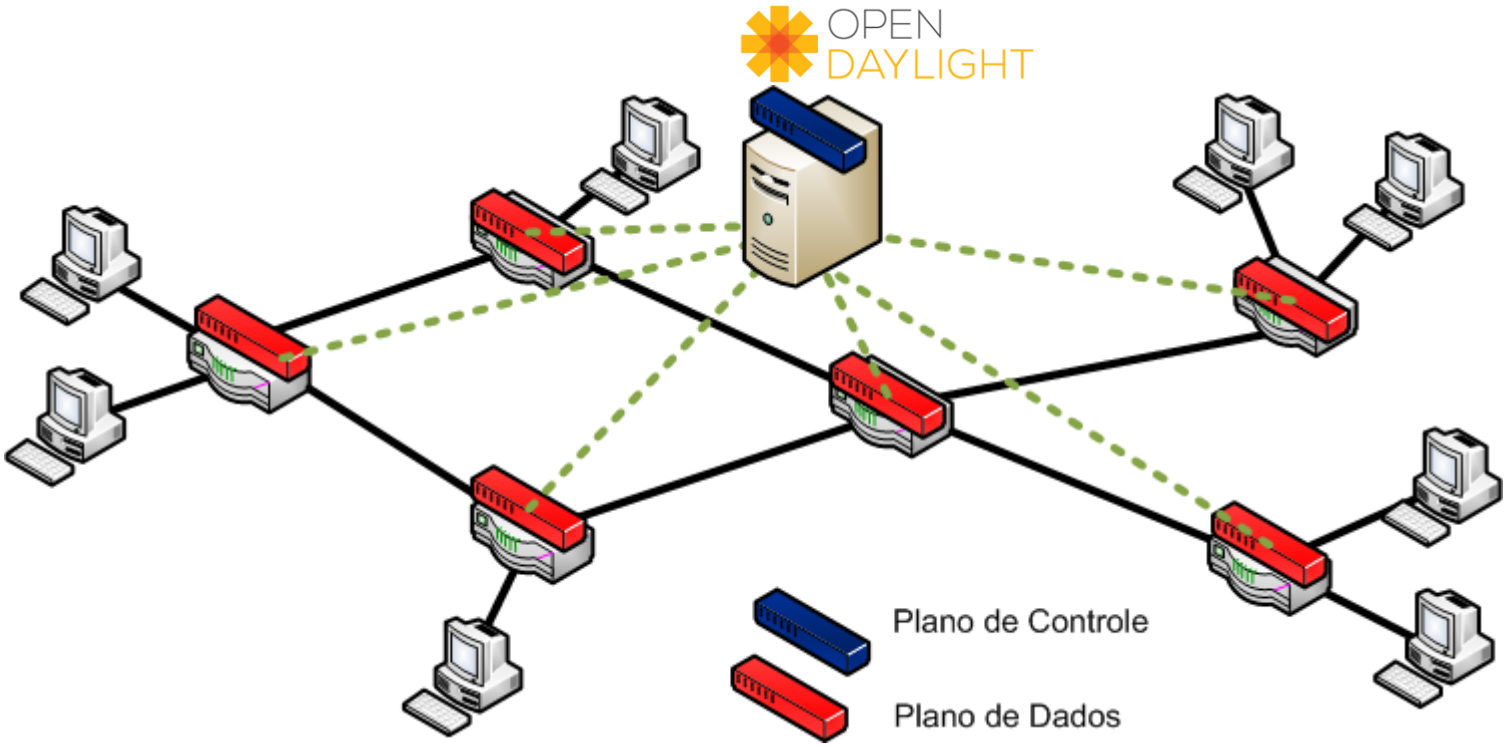
Implementação da distribuição da NIB

- Fraca
  - Utilização de uma DHT (*Distributed Hash Table*)
    - Em linhas gerais, cada nó possui um "pedaço" das informações de estado da rede
- Forte
  - Armazenamento em base de dados transacional (SQL)
    - Conteúdo da base de dados é replicado em cada nó, garantido a consistência

Southbound APIs utilizadas pelo Onix

- OpenFlow
  - Mudanças na NIB pelas aplicações são traduzidas em mensagens openflow para os comutadores
  - Mensagem openflow dos comutadores para o controlador sobre mudanças de estado alteram o conteúdo das NIBs
- OVSDB
  - Acesso a configurações de comutadores
  - Configuração de entidades lógicas nos comutadores
    - P.ex. túneis

OpenDaylight



Arquitetura Opendaylight

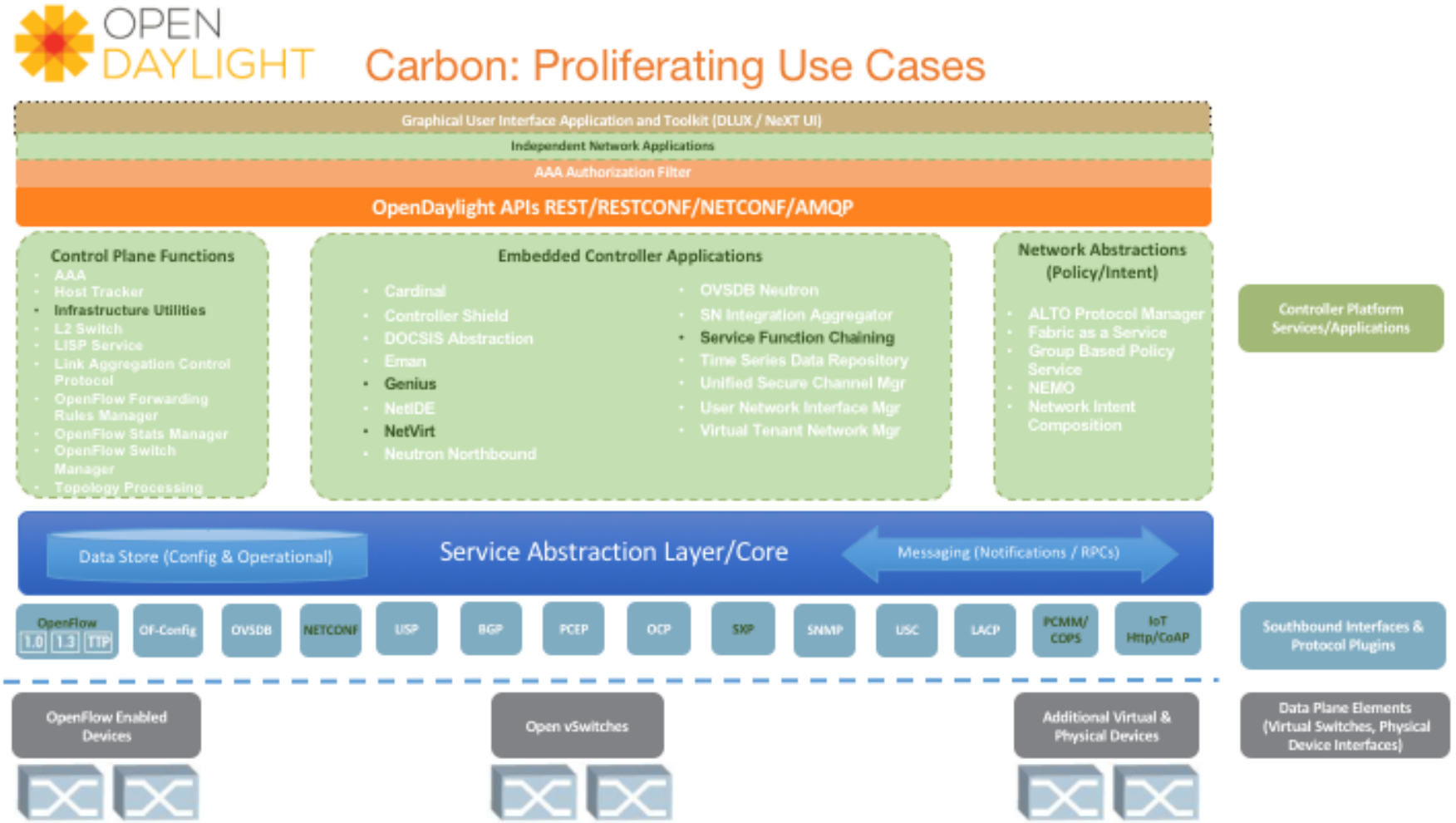
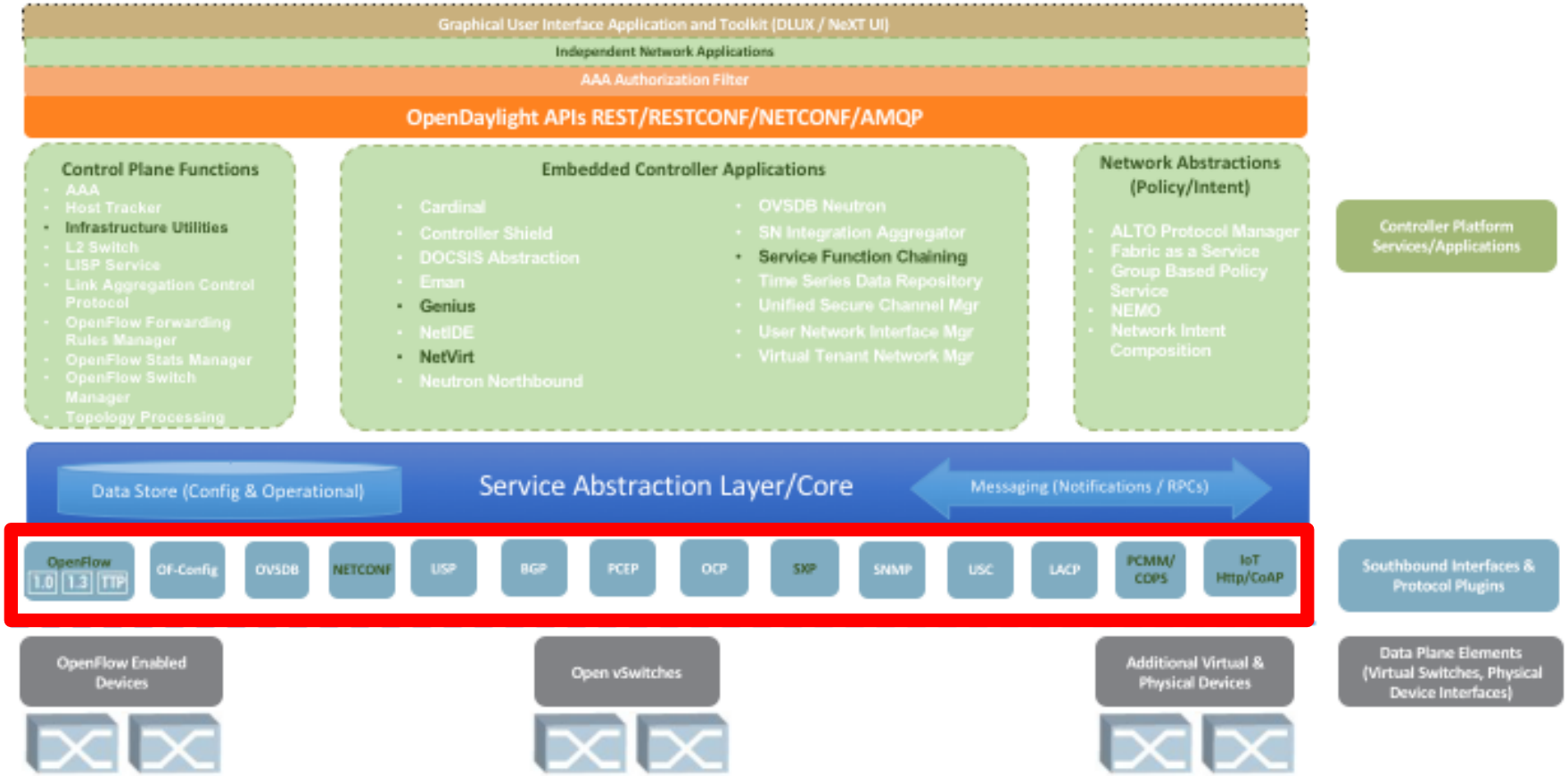


Figura retirada de https://www.opendaylight.org/what-we-do/current-release

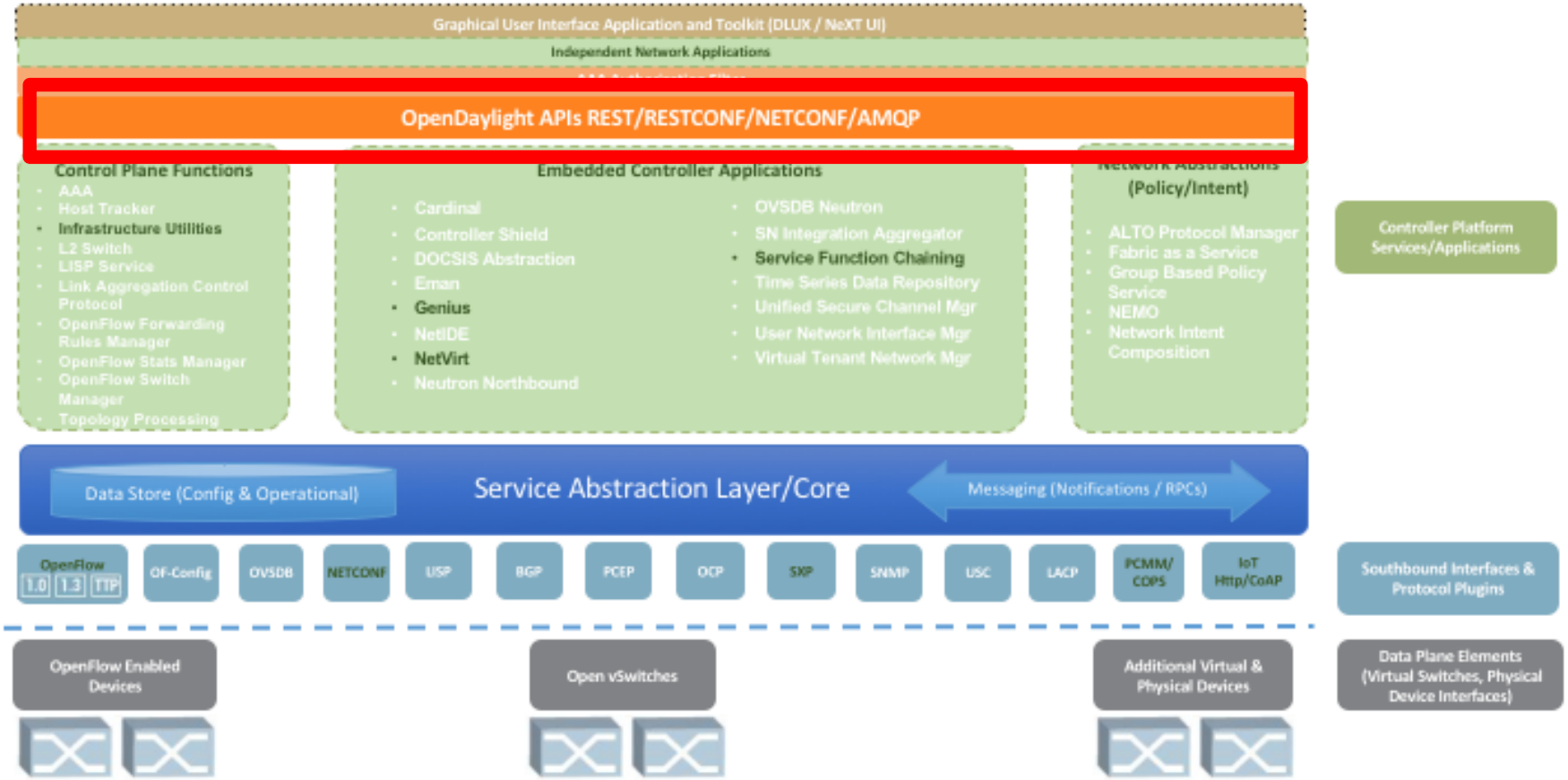
Arquitetura Opendaylight

- Suporte a diferentes Southbound APIs
  - OpenFlow, BGP, SNMP, NETCONF, etc.

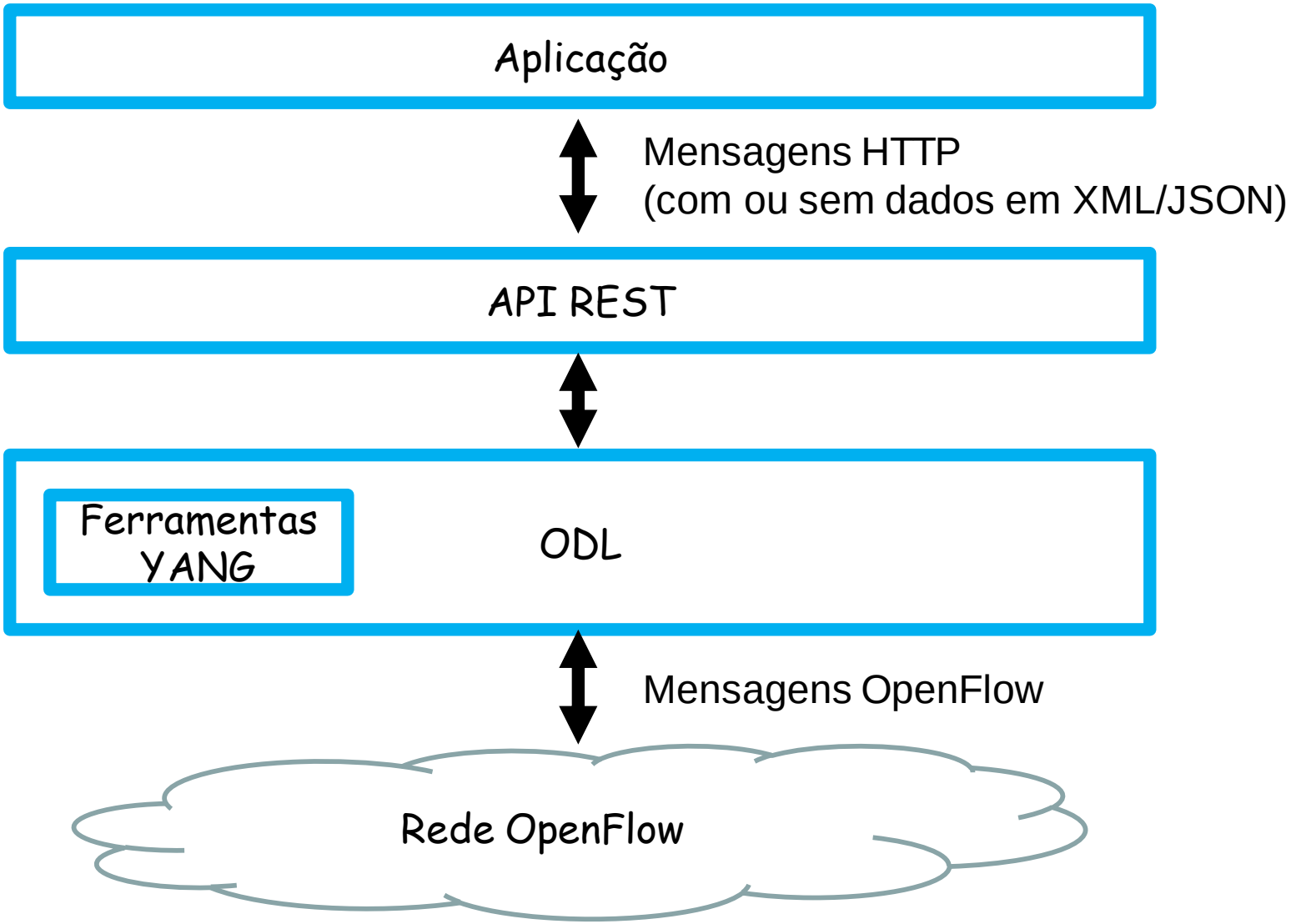


Arquitetura Opendaylight

- Utilização de APIs para desenvolvimento de aplicações
  - Ex: API REST é acessada por requisições HTTP



Visão Geral de uma aplicação utilizando a API REST



Modelo YANG

- Linguagem para modelar interações com dispositivos de rede
  - P.ex. configuração remota de uma interface de rede
- Modelo definido pelo IETF para configuração do protocolo NETCONF
- Mensagens na linguagem YANG podem ser traduzidas para o formato JSON e XML
  - Envio por requisições HTTP
- Para gerenciar dispositivos de rede no OpenDaylight é possível enviar ao controlador mensagens XML no formato YANG
- Mais infos: https://tools.ietf.org/html/rfc6020

O que é XML (eXtensible Markup Language)?

- Linguagem utilizada para descrever diferentes tipos de dados
- Exemplo: Receita de pão em XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos" tempo_de_cozimento="1 hora">
  <titulo>Pão simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
  </instrucoes>
</receita>
```

Retirado de https://pt.wikipedia.org/wiki/XML

Exemplo de uma mensagem em XML noo modelo YANG

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
    <strict>false</strict>
    <instructions>
        <instruction>
            <order>0</order>
            <apply-actions>
                <action>
                    <order>0</order>
                    <dec-nw-ttl/>
                </action>
            </apply-actions>
        </instruction>
    </instructions>
    <table_id>2</table_id>
    <id>124</id>
    <cookie_mask>255</cookie_mask>
    <installHw>false</installHw>
```

Exemplo de uma mensagem em XML noo modelo YANG

```
<installHw>false</installHw>
   <match>
      <ethernet-match>
         <ethernet-type>
            <type>2048</type>
         </ethernet-type>
      </ethernet-match>
      <ipv4-destination>10.0.1.1/24</ipv4-destination>
   </match>
   <hard-timeout>12</hard-timeout>
   <cookie>1</cookie>
   <idle-timeout>34</idle-timeout>
   <flow-name>FooXf1</flow-name>
   <priority>2</priority>
   <barrier>false</barrier>
</flow>
```