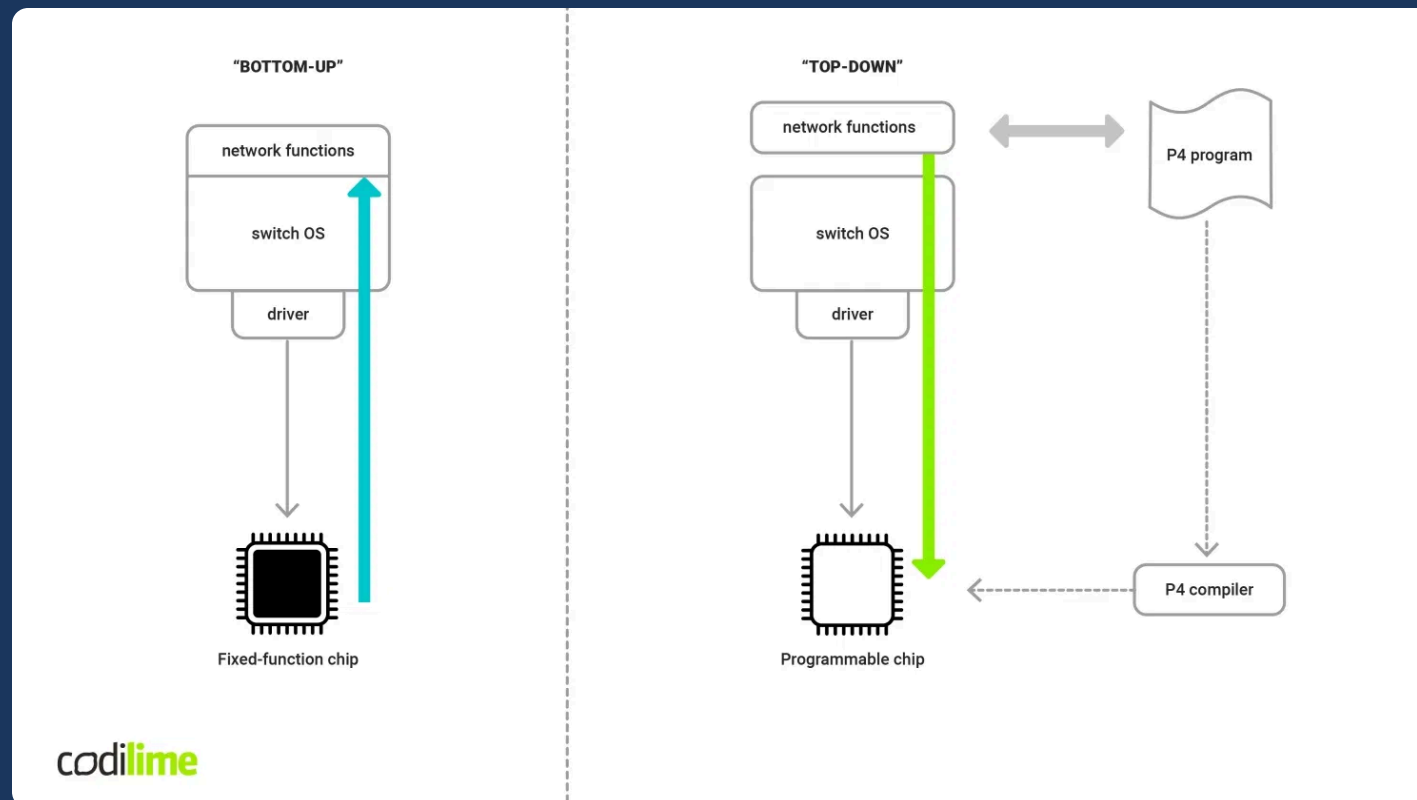


Introdução à Programação do Plano de Dados



Redes Tradicionais

Dispositivos de rede com funcionalidades fixas e proprietárias, limitando a inovação e adaptabilidade.

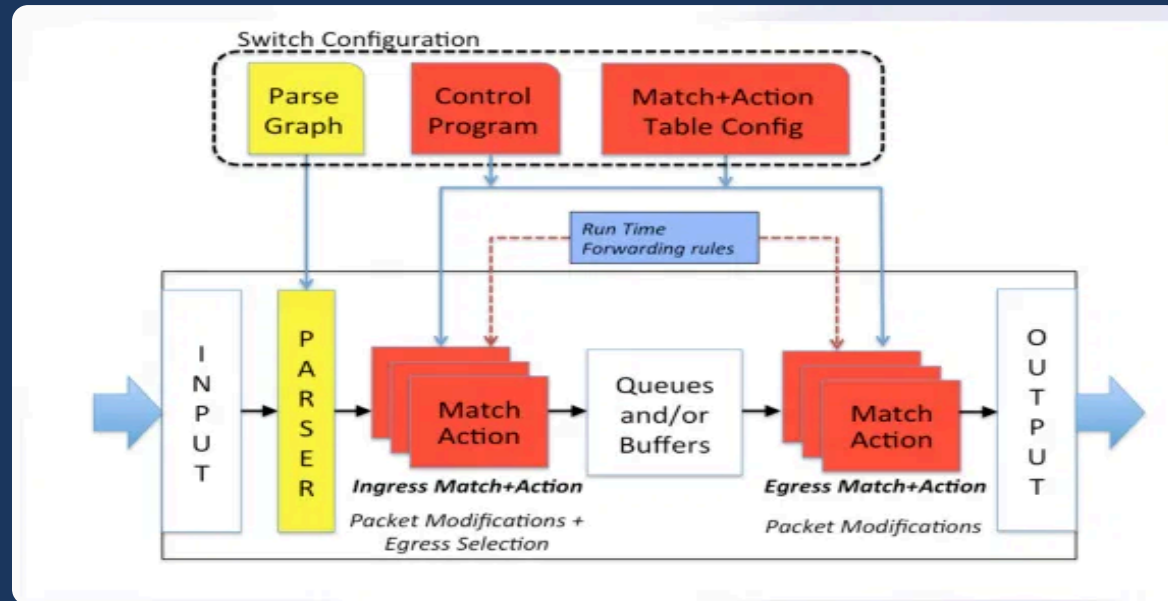
SDN

Separação entre plano de controle e plano de dados, mas ainda com limitações na programabilidade do plano de dados.

P4

Linguagem de domínio específico que permite programar o comportamento do plano de dados dos dispositivos de rede.

A Linguagem P4



O que é P4?

P4 (Programming Protocol-independent Packet Processors) é uma linguagem de domínio específico para programar o comportamento de processamento de pacotes em dispositivos de rede.

- ✓ Independente de protocolo
- ✓ Independente de alvo (target)
- ✓ Reconfigurável em campo

Principais Componentes

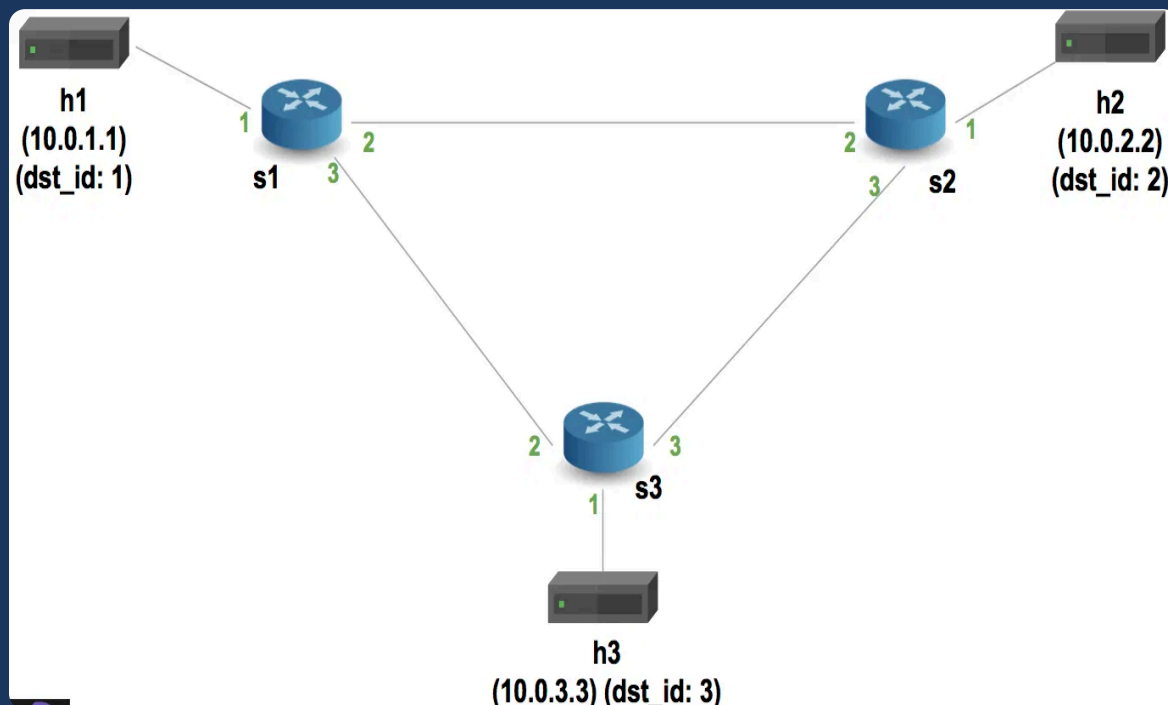
- **Cabeçalhos (Headers):** Definem os formatos dos cabeçalhos dos protocolos
- **Parsers:** Extraem campos dos pacotes de entrada
- **Tabelas:** Armazenam regras de encaminhamento
- **Ações:** Operações aplicadas aos pacotes
- **Controles:** Orquestram a sequência de operações
- **Deparsers:** Remontam os cabeçalhos para formar o pacote de saída

```
header ethernet_t {
    bit<48> dstAddr;
    bit<48> srcAddr;
    bit<16> etherType;
}

action forward(port_t port) {
    standard_metadata.egress_spec = port;
}

table ipv4_lpm {
    key = { hdr.ipv4.dstAddr: lpm; }
    actions = { forward; drop; }
    size = 1024;
    default_action = drop();
}
```

Arquitetura V1Model



O que é V1Model?

V1Model é a arquitetura alvo padrão suportada pelo BMv2, definindo um pipeline de processamento de pacotes com seis componentes programáveis em P4.

Definida no arquivo `v1model.p4`, esta arquitetura é amplamente utilizada para desenvolvimento e testes de programas P4.

Componentes Programáveis

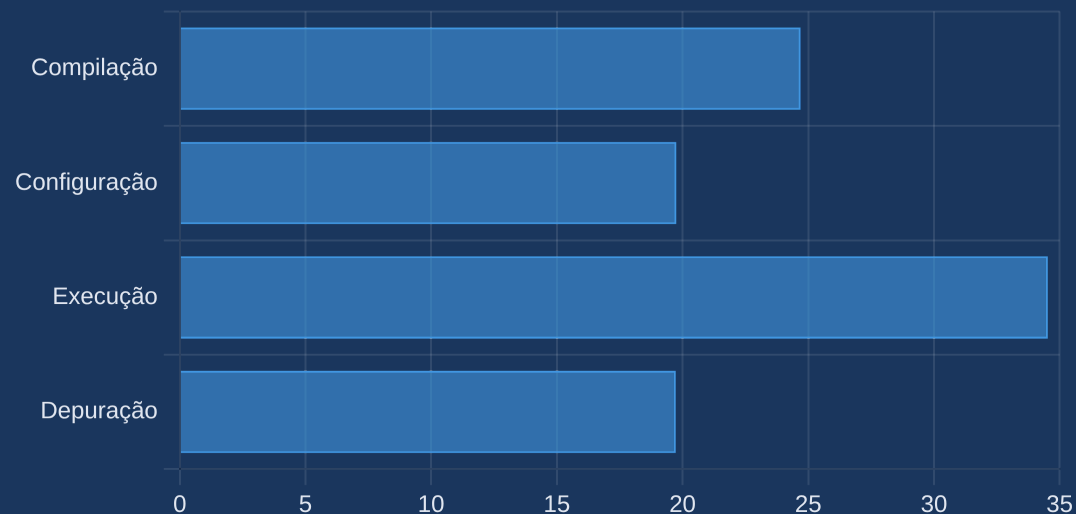
- 1 Parser**
Analisa os cabeçalhos dos pacotes de entrada e extrai os campos relevantes.
- 2 Verificação de Checksum**
Verifica a integridade dos pacotes recebidos.
- 3 Controle de Ingresso**
Processa os pacotes na entrada do switch, aplicando tabelas e ações.
- 4 Controle de Egresso**
Processa os pacotes na saída do switch, após decisões de encaminhamento.
- 5 Atualização de Checksum**
Recalcula checksums após modificações nos cabeçalhos.
- 6 Deparser**
Remonta os cabeçalhos para formar o pacote de saída.

BMv2: Behavioral Model Version 2




O que é BMv2?

BMv2 (Behavioral Model Version 2) é um switch de software de código aberto desenvolvido pela P4.org, projetado para ser um alvo para programas P4.

Ele permite que desenvolvedores testem e validem seus programas P4 em um ambiente de software antes de implantá-los em hardware real.



Componentes do BMv2

-  **simple_switch:** O executável principal que implementa a lógica do switch e carrega o programa P4 compilado.
-  **Portas Virtuais:** Interfaces de rede virtuais que podem ser conectadas a outros dispositivos em um ambiente de emulação.
-  **P4Runtime:** API gRPC que permite que controladores externos interajam com o switch BMv2 para configurar tabelas e gerenciar o estado.

Uso Típico

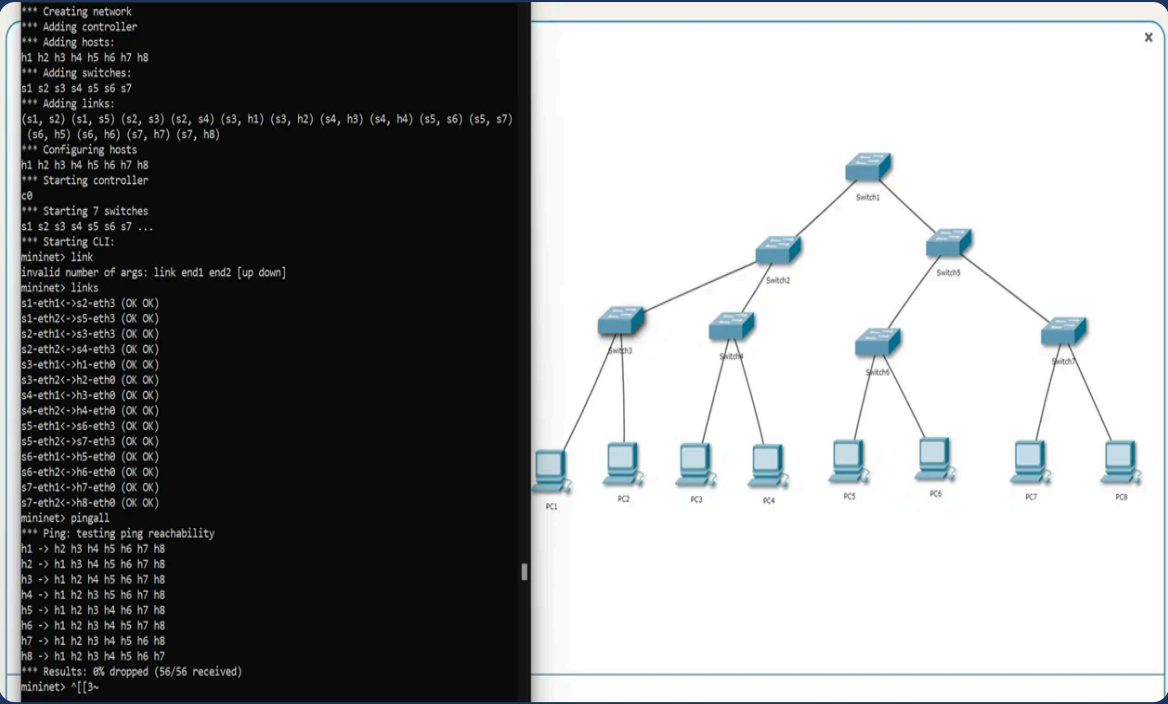
O BMv2 é frequentemente usado em conjunto com o Mininet para criar e testar topologias de rede virtuais com switches programáveis em P4.

```
# Compilar um programa P4 para BMv2
p4c --target bmv2 --arch v1model programa.p4 -o programa.json

# Iniciar o simple_switch com o programa compilado
simple_switch --log-console -i 0@veth0 -i 1@veth1 programa.json

# Configurar tabelas via CLI
simple_switch_CLI < comandos.txt
```

Mininet: Emulação de Redes



O que é Mininet?

Mininet é um emulador de rede que permite criar rapidamente protótipos de redes complexas em um único computador. Ele utiliza virtualização leve (namespaces de rede do Linux) para criar uma rede virtual completa com hosts, switches, links e controladores.

```
# Exemplo de script Python para Mininet
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
```

```
# Criar rede
net = Mininet()
```

```
# Adicionar hosts e switches
h1 = net.addHost('h1')
h2 = net.addHost('h2')
s1 = net.addSwitch('s1')
```

```
# Criar links
net.addLink(h1, s1)
net.addLink(h2, s1)
```

```
# Iniciar rede
net.start()
CLI(net)
net.stop()
```

Principais Características

- ✔ **Virtualização Leve**
Usa namespaces de rede do Linux para criar hosts virtuais com isolamento de processos e recursos de rede.
- ✔ **Personalização**
Permite criar topologias personalizadas com diferentes tipos de switches, controladores e links.
- ✔ **Integração com SDN**
Suporta controladores SDN como OpenFlow e, com extensões, switches programáveis como BMv2.
- ✔ **API Python**
Oferece uma API Python para criar e gerenciar redes programaticamente.

Uso com P4 e BMv2

Para usar o Mininet com P4 e BMv2, é necessário:

- Estender o Mininet para suportar switches BMv2
- Compilar programas P4 para o formato JSON do BMv2
- Configurar o plano de controle para inserir regras nas tabelas P4
- Iniciar a topologia Mininet com os switches BMv2 executando os programas P4

Integração P4-BMv2-Mininet

Escrever o programa P4

- 1 Desenvolver o código P4 (.p4) que define o comportamento do plano de dados do switch.

Compilar o programa P4

- 2 Usar o compilador p4c para gerar um arquivo JSON e um arquivo P4Info a partir do código P4.

Configurar a topologia Mininet

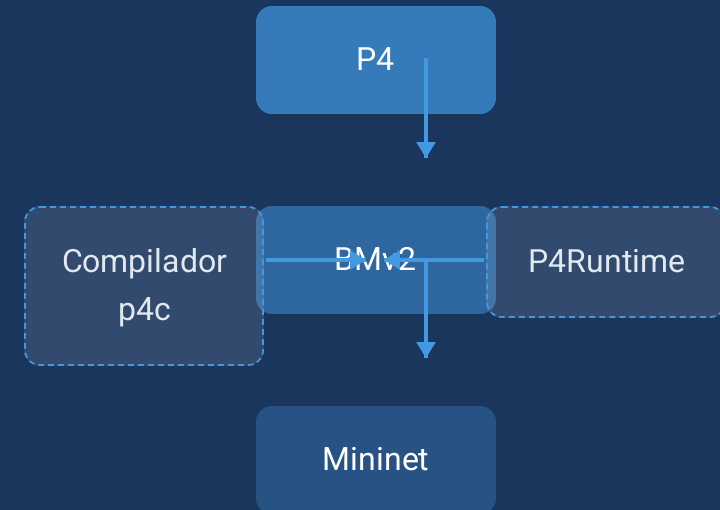
- 3 Criar um script Python que define a topologia da rede com switches BMv2 e hosts.

Carregar o programa no BMv2

- 4 Iniciar as instâncias do BMv2 no Mininet e carregar o arquivo JSON compilado.

Configurar o plano de controle

- 5 Inserir entradas nas tabelas de fluxo dos switches BMv2 via P4Runtime ou CLI.



Benefícios da Integração

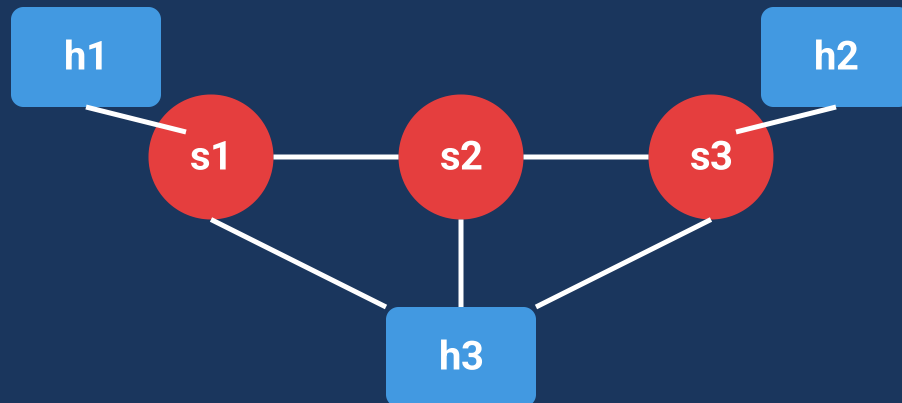
- **Desenvolvimento Rápido:** Ciclo rápido de desenvolvimento, teste e depuração.
- **Ambiente Controlado:** Testes em um ambiente virtualizado antes da implantação em hardware.
- **Flexibilidade:** Fácil modificação da topologia e do comportamento dos switches.
- **Automação:** Scripts podem automatizar todo o processo de teste.

Exercício: Encaminhamento Básico

Objetivo

Implementar um switch P4 que realiza encaminhamento básico de pacotes IPv4, incluindo:

- Atualização dos endereços MAC de origem e destino
- Decremento do TTL no cabeçalho IP
- Encaminhamento para a porta de saída apropriada



Topologia

Três hosts (h1, h2, h3) conectados a três switches (s1, s2, s3) em uma topologia triangular. Cada switch executa o programa P4 que implementamos.

Implementação

- 1 Definir cabeçalhos
Ethernet e IPv4
- 2 Implementar parser
Para extrair cabeçalhos Ethernet e IPv4
- 3 Definir ações
Para encaminhamento e descarte de pacotes
- 4 Criar tabela de encaminhamento
Usando endereço IP de destino como chave
- 5 Implementar deparser
Para remontar os pacotes de saída

```
action ipv4_forward(macAddr_t dstAddr, macAddr_t srcAddr, port_t port) {  
    // 1. Atualiza endereço MAC de destino  
    hdr.ethernet.dstAddr = dstAddr;  
  
    // 2. Atualiza endereço MAC de origem  
    hdr.ethernet.srcAddr = srcAddr;  
  
    // 3. Decrementa o TTL  
    hdr.ipv4.ttl = hdr.ipv4.ttl - 1;  
  
    // 4. Define a porta de saída  
    standard_metadata.egress_spec = port;  
}
```

Exercício: Funcionalidades Avançadas

Tunelamento Básico

Implementação de um mecanismo de tunelamento personalizado em P4, permitindo encapsular pacotes IPv4 para encaminhamento baseado em IDs de destino.

Cabeçalho de Túnel

Definição de um novo tipo de cabeçalho para encapsulamento.

Tabela de Encaminhamento

Tabela específica para encaminhamento baseado em IDs de túnel.

Calculadora em P4

Implementação de uma calculadora simples usando um protocolo personalizado em P4, capaz de realizar operações aritméticas básicas no plano de dados.

```
header calc_t {
    bit<8>  p;
    bit<8>  four;
    bit<8>  ver;
    bit<8>  op;
    bit<32> operand_a;
    bit<32> operand_b;
    bit<32> res;
}
```

Balanceamento de Carga

Implementação de balanceamento de carga usando Equal-Cost Multipath (ECMP) em P4, distribuindo pacotes entre múltiplos caminhos com base em um hash de 5-tuplas.

Função Hash

Utiliza campos como IP de origem/destino, portas TCP e protocolo para distribuir o tráfego.

```
hash(meta.ecmp_hash,
      HashAlgorithm.crc16,
      (bit<1>)0,
      { hdr.ipv4.srcAddr,
        hdr.ipv4.dstAddr,
        hdr.tcp.srcPort,
        hdr.tcp.dstPort,
        hdr.ipv4.protocol},
      num_nhops);
```

Outros Exercícios Avançados

Notificação Explícita de Congestionamento (ECN)

Implementação de ECN para sinalizar congestionamento sem descartar pacotes.

Roteamento por Origem

Implementação de roteamento onde a rota é especificada pelo remetente do pacote.

Firewall Básico

Implementação de um firewall stateful usando filtros de bloom em P4.