

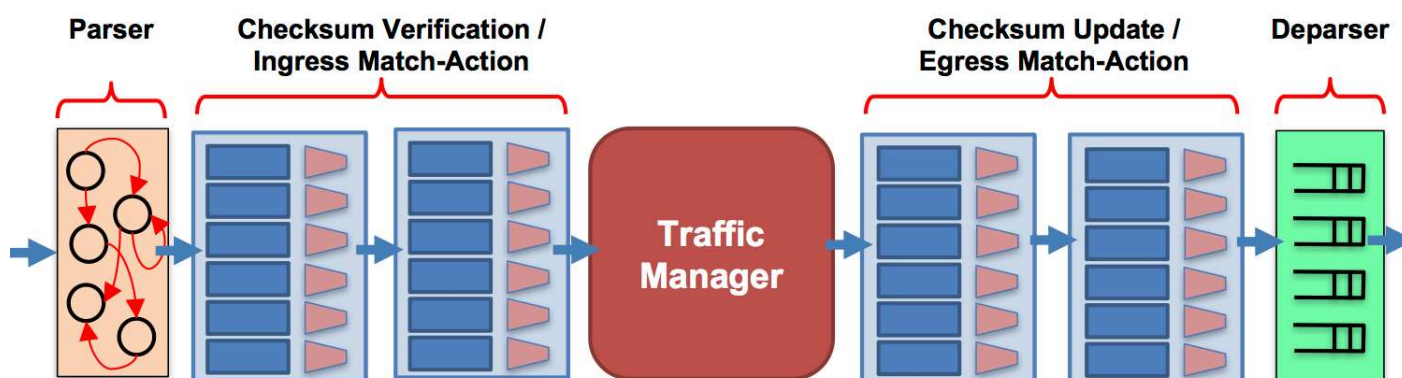
**CS344**

Construir um roteador de Internet

[Sobre](#) [Documentação](#) [Política](#) [Cronograma](#) [Código](#) [Fonte](#)  
[Equipes](#) [Palestras](#) [Piazza](#)

## Trabalhando com P4 no Mininet no BMV2

A P4.org desenvolveu um switch de software de código aberto chamado BMV2 (modelo comportamental versão 2), projetado para ser um alvo para programas P4. Ou seja, programas P4 podem ser compilados nele para configurar como ele processa pacotes. Cada alvo P4 suporta uma ou mais arquiteturas de alvo P4. A arquitetura de alvo suportada pelo BMV2 que usaremos nestes exercícios introdutórios é chamada de V1Model. Um diagrama do V1Model é mostrado abaixo:

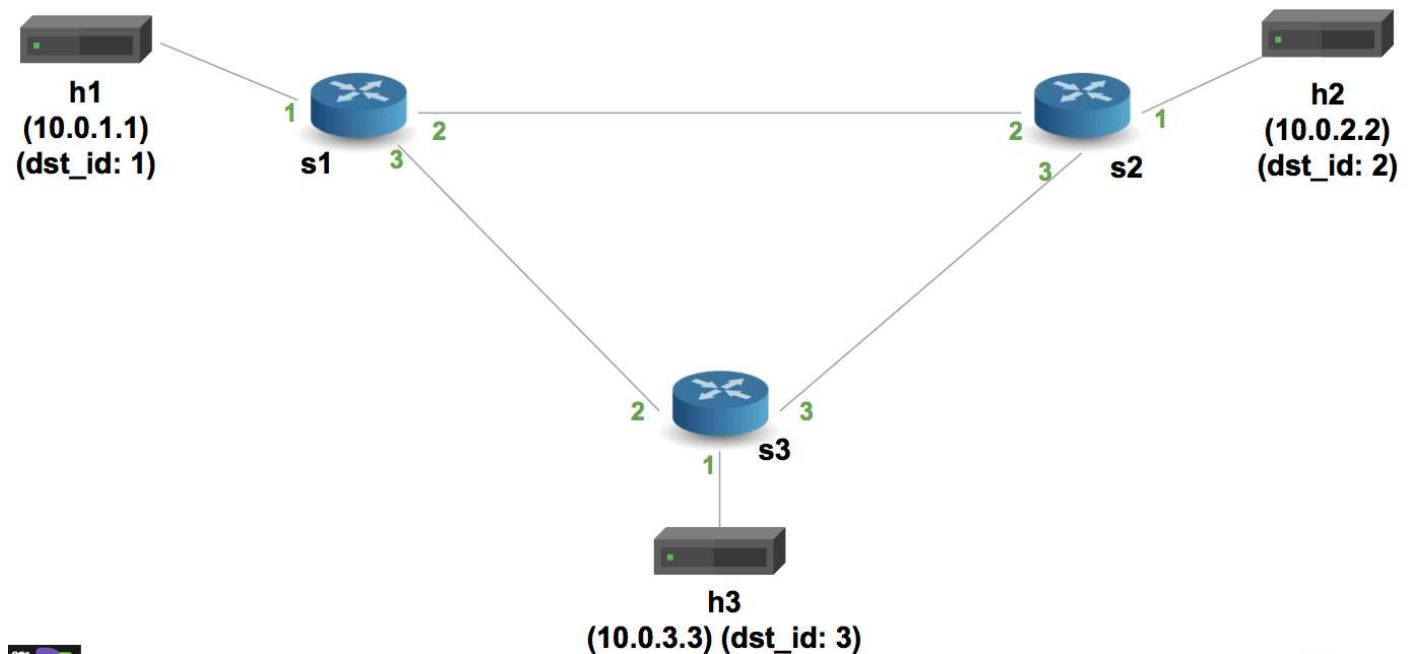


O V1Model consiste em seis componentes programáveis P4:

- Analisador
- Bloco de controle de verificação de soma de verificação
- Bloco de controle de processamento de Match-Action do Ingress
- Bloco de controle de processamento de ação de correspondência de saída
- Bloco de controle de atualização de soma de verificação
- Desmembrador

O P4.org definiu essa arquitetura de destino em um arquivo chamado `v1model.p4` e adicionou suporte a ela no compilador front-end de código aberto do P4, chamado `p4c`. Isso permite que o `p4c` compile programas P4 escritos para o V1Model em um arquivo JSON que pode ser usado para configurar o switch de software BMV2.

O P4.org elaborou um conjunto de exercícios tutoriais projetados para introduzir incrementalmente novos recursos da linguagem P4. Para cada exercício (com exceção do exercício P4Runtime), você escreverá um programa P4 direcionado ao V1Model e ao tipo `make` para compilar o programa e iniciar uma rede Mininet com três switches, onde cada switch executará seu programa P4. Cada exercício (novamente, com exceção do exercício P4Runtime) fornece um conjunto de entradas de tabela configuradas estaticamente para cada um dos três switches, portanto, nos concentraremos apenas no desenvolvimento do plano de dados. Aqui está a topologia que usaremos:



## Começando

Esta tarefa deve ser concluída em equipe. Ou seja, sua equipe terá um repositório contendo todas as suas soluções. Você pode decidir como dividir o trabalho, mas recomendamos fortemente que cada membro da equipe tente cada exercício. Esses exercícios são sua primeira oportunidade de adquirir experiência com o P4 e se aprofundar nas ideias fundamentais.

Você deve usar seu próprio laptop para esta tarefa e NÃO a máquina de desenvolvimento à qual sua equipe foi designada.

## Instalação usando Vagrant

1. Instale o [Vagrant](#) e o [VirtualBox](#) no seu laptop local.
2. Crie um fork do repositório [p4-mininet-tutorials](#) .
3. Clone seu fork no seu laptop local.

4. `cd` Acesse o diretório [p4-mininet-tutorials/P4D2\\_2018\\_East/vm](#) e execute `$ vagrant up`. O provisionamento da máquina virtual levará cerca de 30 minutos, então seja paciente e vá tomar um café :)
5. Após a conclusão do comando anterior, você deverá ver um prompt de login no Virtual Box. Efetue login com nome de usuário `p4` e senha `p4`. Em seguida, execute `$ sudo shutdown -r now`. Quando a máquina for reiniciada, você deverá ter uma área de trabalho gráfica com todos os softwares necessários para executar os exercícios pré-instalados.

## Baixando diretamente a imagem da VM

Se por algum motivo você não conseguir criar a imagem da VM sozinho, poderá baixar a imagem diretamente do Google Drive.

1. Instale o [VirtualBox](#) no seu laptop.
2. Baixe a imagem da VM: [P4 Tutorial 2018-03-05.ova](#) .
3. Importe a máquina virtual para o VirtualBox. Abra o VirtualBox, selecione “Arquivo > Importar Appliance” e navegue até o arquivo baixado.
4. Inicialize a máquina virtual. Selecione “P4 Tutorial 2018-03-05” e clique em “Iniciar”.
5. Crie um fork do repositório [p4-mininet-tutorials](#) .

Clone seu repositório bifurcado no `p4` diretório inicial do usuário na VM e trabalhe dentro do seu clone, *não* na `tutorials` pasta pré-instalada na VM.

## Os Exercícios

Os exercícios estão organizados em quatro módulos:

1. Introdução e noções básicas da linguagem:
  - [Encaminhamento básico](#)
  - [Tunelamento básico](#)
2. P4Runtime e o Plano de Controle:
  - [P4Tempo de execução](#)
3. Monitoramento e Depuração:
  - [Notificação explícita de congestionamento](#)
  - [Inspeção de Rota Multi-Hop](#)
4. Estruturas de Dados Avançadas:
  - [Roteamento de origem](#)
  - [Calculadora](#)
5. Comportamento dinâmico:

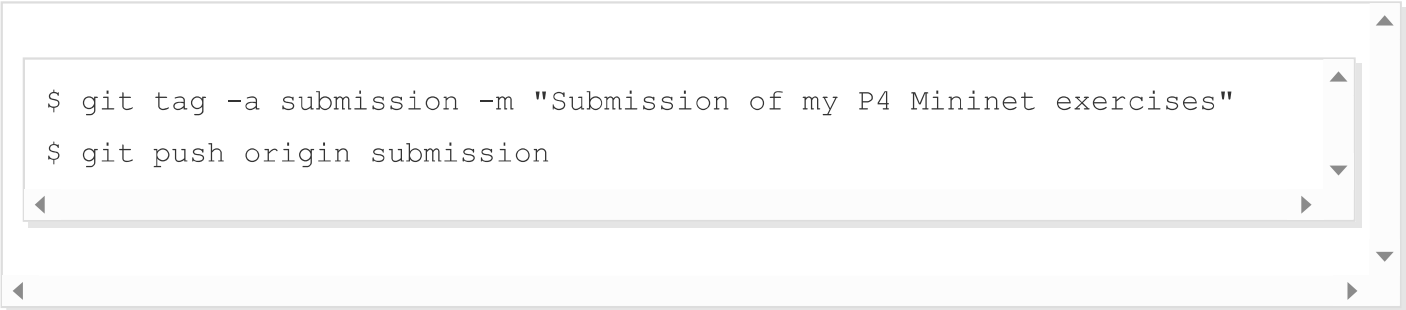
- **Balanceamento de carga**

Consulte os `README` arquivos dentro de cada diretório de exercícios para obter uma descrição detalhada dos objetivos e de como realizar as avaliações. Cada exercício fornece um código inicial marcado com `TODO` comentários. Seu objetivo é completar o código inicial fornecido e garantir que cada exercício se comporte conforme o esperado.

## Submissão

Depois de concluir todos os exercícios e desejar enviar seu trabalho, basta adicionar uma tag ao seu último commit e enviá-la ao seu repositório remoto.

Processo de submissão:

A screenshot of a terminal window with a light gray background. It contains two lines of text: the first line is "\$ git tag -a submission -m 'Submission of my P4 Mininet exercises'" and the second line is "\$ git push origin submission". The terminal has a standard scrollbar on the right side.

```
$ git tag -a submission -m "Submission of my P4 Mininet exercises"
$ git push origin submission
```

Após o prazo final da tarefa, os instrutores verificarão o commit para o qual a `submission` tag aponta no seu repositório e garantirão que todos os exercícios estejam funcionando corretamente.