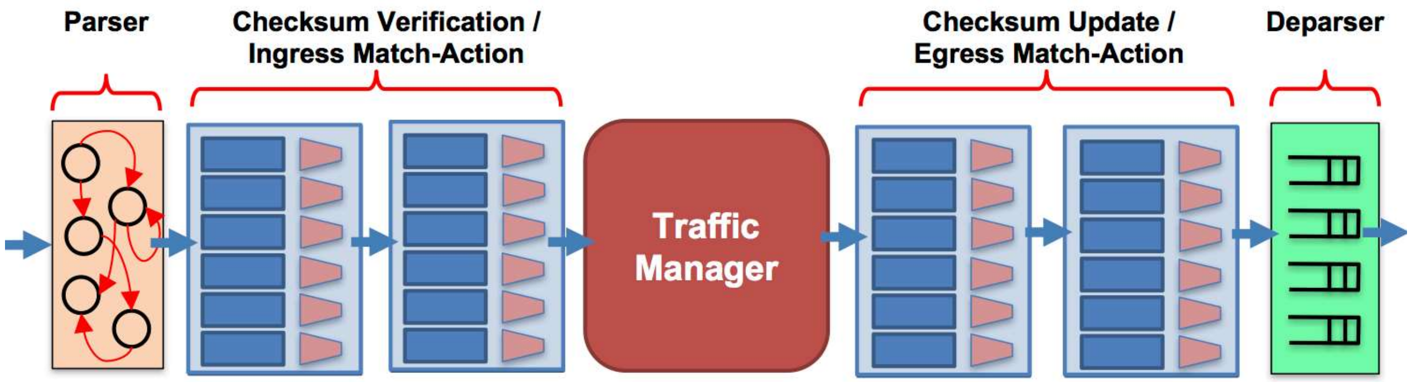
# Working with P4 in Mininet on BMV2

P4.org has developed an open source software switch called BMV2 (behavioral model version 2) designed to be a target for P4 programs. That is, P4 programs can be compiled onto it to configure how it processes packets. Every P4 target supports one or more P4 target architectures. The target architecture supported by BMV2 that we will be using for these introductory exercises is called the V1Model. A diagram of the V1Model is shown below:

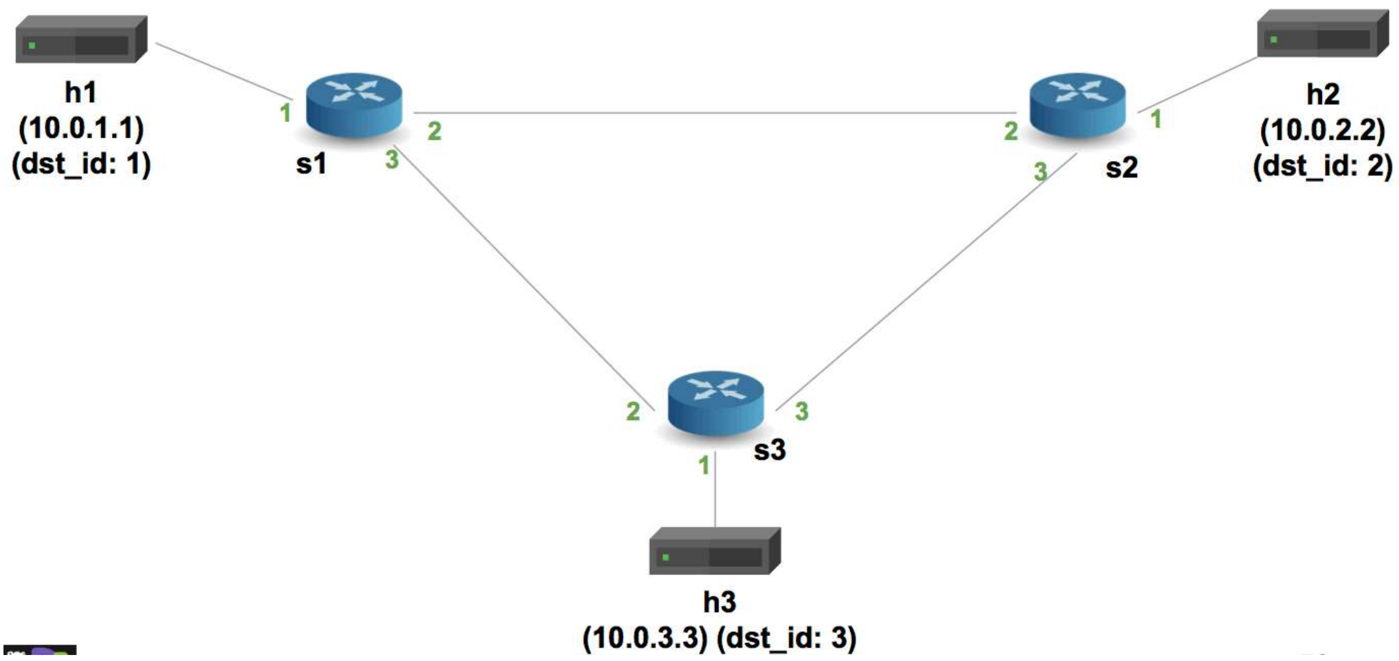

Parser
Checksum Verification / Ingress Match-Action
Traffic Manager
Checksum Update / Egress Match-Action
Deparser

The V1Model consists of six P4 programmable components:

- Parser
- Checksum verification control block
- Ingress Match-Action processing control block
- Egress Match-Action processing control block
- Checksum update control block
- Deparser

P4.org has defined this target architecture in a file called v1model.p4 and added support for it to the open source P4 front end compiler called p4c. This allows p4c to compile P4 programs written for the V1Model into a json file that can be used to configure the BMV2 software switch.

P4.org has put together a set of tutorial exercises designed to incrementally introduce new features of the P4 language. For each exercise (with the exception of the P4Runtime exercise), you will write a P4 program targeting the V1Model and type `make` to compile the program and launch a Mininet network with three switches where each switch is running your P4 program. Each exercise (again, with the exception of the P4Runtime exercise) provides a set of statically configured table entries for each of the three switches so we will be focused on data plane development only. Here is the topology that we will be using:



## Getting Started

This assignment is to be completed as a team. That is, your team will have one repository containing all of your solutions. You can decide how you would like to split up the work, but we highly encourage each teammate to attempt each exercise. These exercises are your first opportunity to gain experience with P4 and wrestle with the fundamental ideas.

You should use your own laptops for this assignment and NOT the developement machine to which your team has been assigned.

### Installation Using Vagrant

1. Install Vagrant and VirtualBox on your local laptop.
2. Create a fork of the p4-mininet-tutorials repository.
3. Clone your fork onto your local laptop.

4. `cd` into the p4-mininet-tutorials/P4D2_2018_East/vm directory and run `$ vagrant up`. It will take about 30 minutes for the virtual machine to provision so be patient, go grab a cup of coffee :)

5. Once the previous command completes you should see a login prompt within Virtual Box. Log in with username `p4` and password `p4`. Then run `$ sudo shutdown -r now`. When the machine reboots, you should have a graphical desktop machine with the all software required to run the exercises pre-installed.

### Directly Downloading the VM Image

If for some reason you cannot build the VM image on your own you can directly download the image from Google Drive.

1. Install VirtualBox on your laptop.

2. Download the VM image: P4 Tutorial 2018-03-05.ova.

3. Import the virtual machine into VirtualBox. Open VirtualBox, select “File > Import Appliance”, and navigate to the downloaded file.

4. Boot virtual machine. Select “P4 Tutorial 2018-03-05”, and click “Start”.

5. Create a fork of the p4-mininet-tutorials repository.

Clone your forked repository into `p4` user’s home directory on the VM and work from within your clone, *not* the `tutorials` folder preinstalled on the VM.

## The Exercises

The exercises are organized into four modules:

1. Introduction and Language Basics:
   - Basic Forwarding
   - Basic Tunneling
2. P4Runtime and the Control Plane:
   - P4Runtime
3. Monitoring and Debugging:
   - Explicit Congestion Notification
   - Multi-Hop Route Inspection
4. Advanced Data Structures:
   - Source Routing
   - Calculator
5. Dynamic Behavior:
   - Load Balancing

Refer to the `README` files within each exercise directory for a detailed description of the goals and how to perform the evaluations. Each exercise provides some starter code marked with `TODO` comments. Your goal is to complete the provided starter code and make sure that each exercise behaves as expected.

## Submission

After you have completed all of the exercises and would like to submit your work, simply add a tag to your latest commit and push the tag to your remote repository.

Submission process:

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

After the assignment deadline, the instructors will checkout the commit to which the `submission` tag points on your repository and make sure that all exercises are working properly.