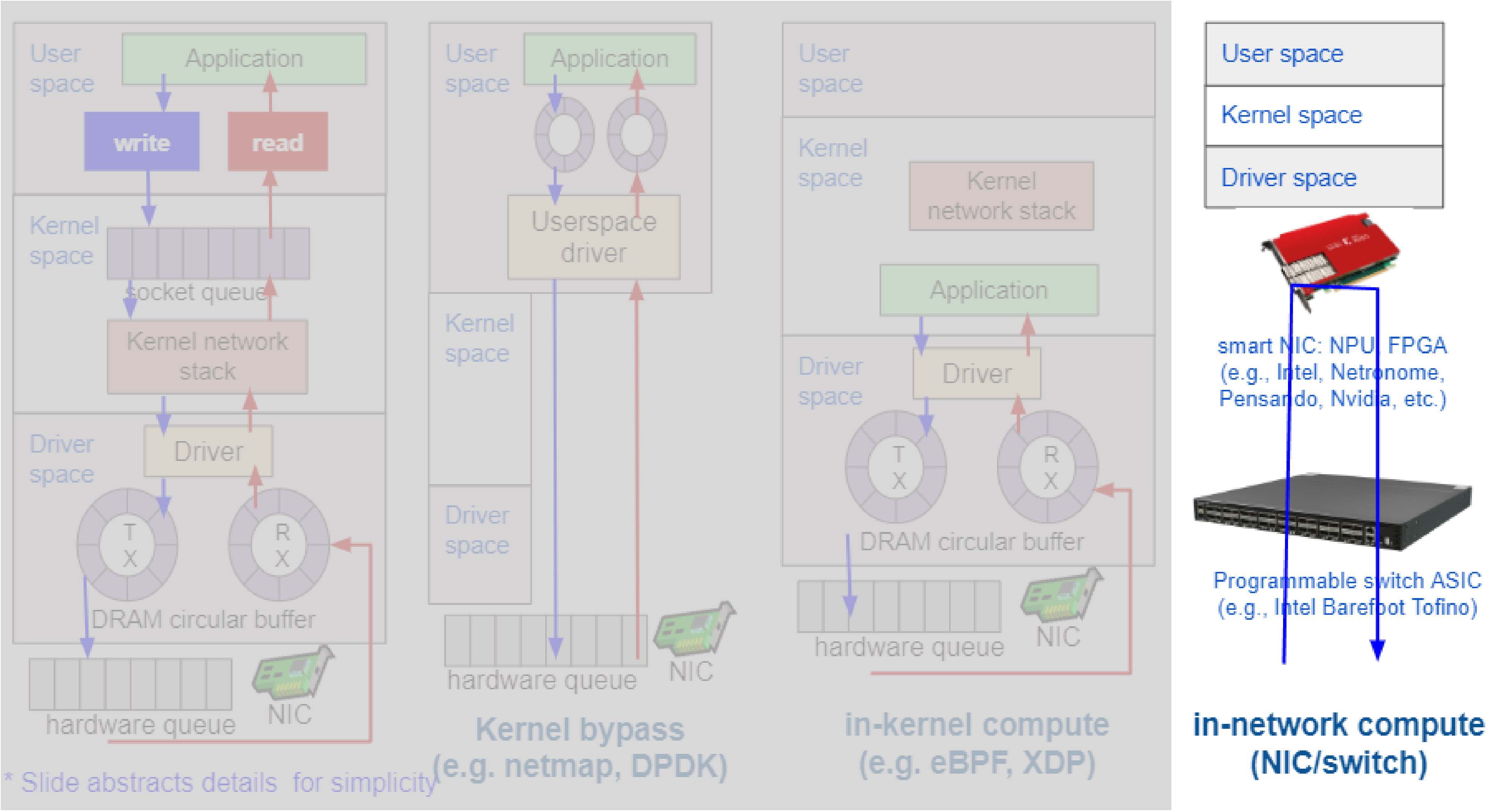
P4 Hands-on

Rinku Shah[1], Assistant professor, IIIT Delhi
Neeraj Kumar Yadav[1], Research Scholar, IIIT Delhi
Keshav Ghambir[2], Microsoft India
Harish S A[3], Research Scholar (PMRF), IIT Hyderabad

[1] IIIT Delhi
India

[2] Microsoft
India

[3] IIT Hyderabad
India



User space

Application

write

read

Kernel space

socket queue

Kernel network stack

Driver space

Driver

TX

RX

DRAM circular buffer

hardware queue

NIC

* Slide abstracts details for simplicity

User space

Application

Userspace driver

Kernel space

Driver space

hardware queue

NIC

Kernel bypass
(e.g. netmap, DPDK)

User space

Kernel space

Kernel network stack

Application

Driver space

Driver

TX

RX

DRAM circular buffer

hardware queue

NIC

in-kernel compute
(e.g. eBPF, XDP)

User space

Kernel space

Driver space

smart NIC: NPU, FPGA
(e.g., Intel, Netronome,
Pensando, Nvidia, etc.)

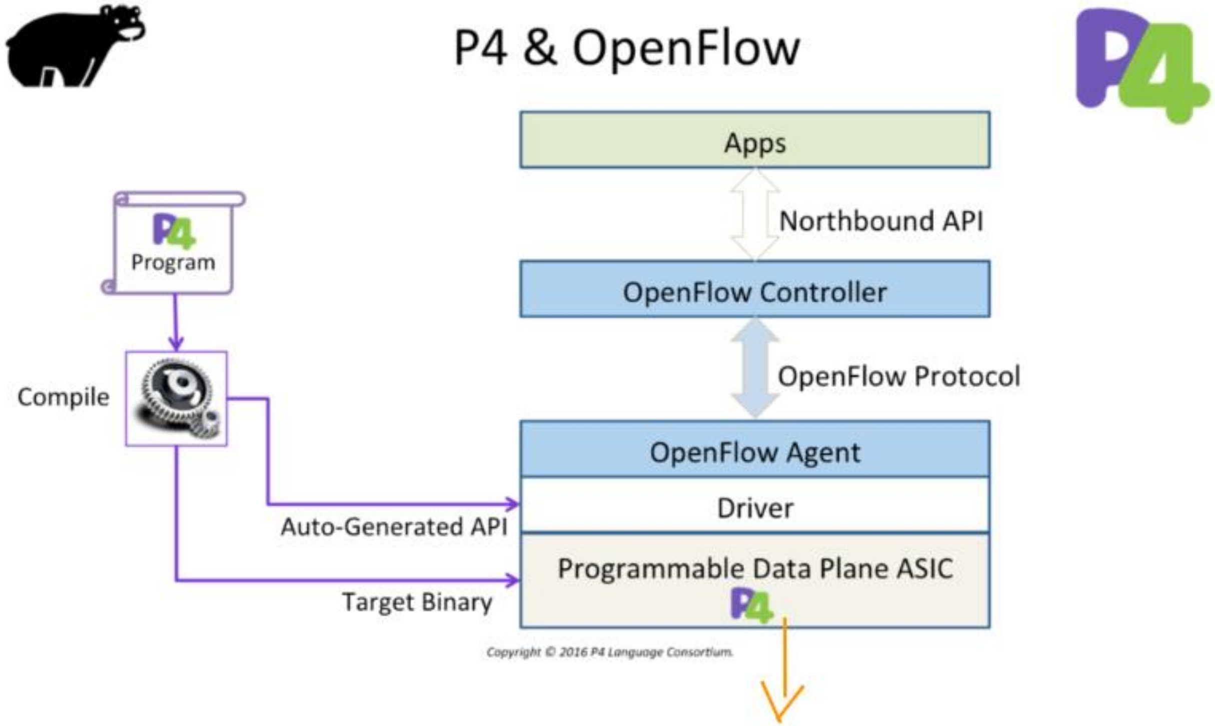
Programmable switch ASIC
(e.g., Intel Barefoot Tofino)

in-network compute
(NIC/switch)

SDN - Programmable Switches – P4

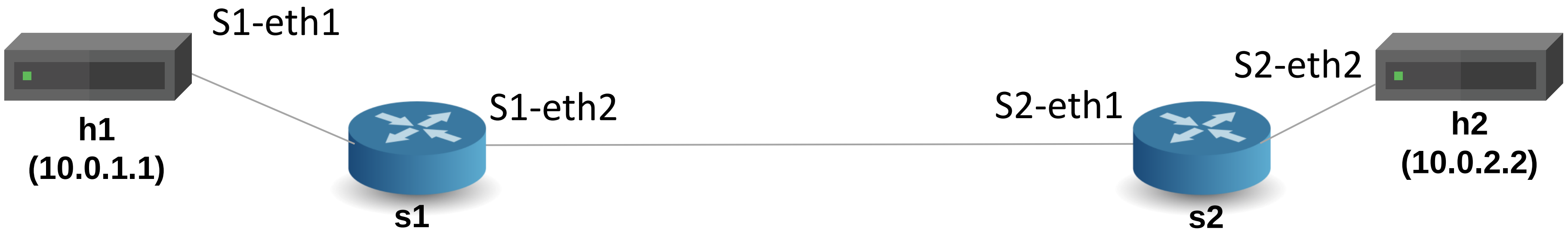
- Switches are programmable with a language called P4

- We can modify the way packets are processed at line rate which is very powerful!



Lab 0: Hello World

A write that connects two ports



Let’s get the environment ready!

1. Install Hypervisor

- We need to first install virtualbox (or an equivalent hypervisor) on our system. Download the binary from the links below according to your host OS

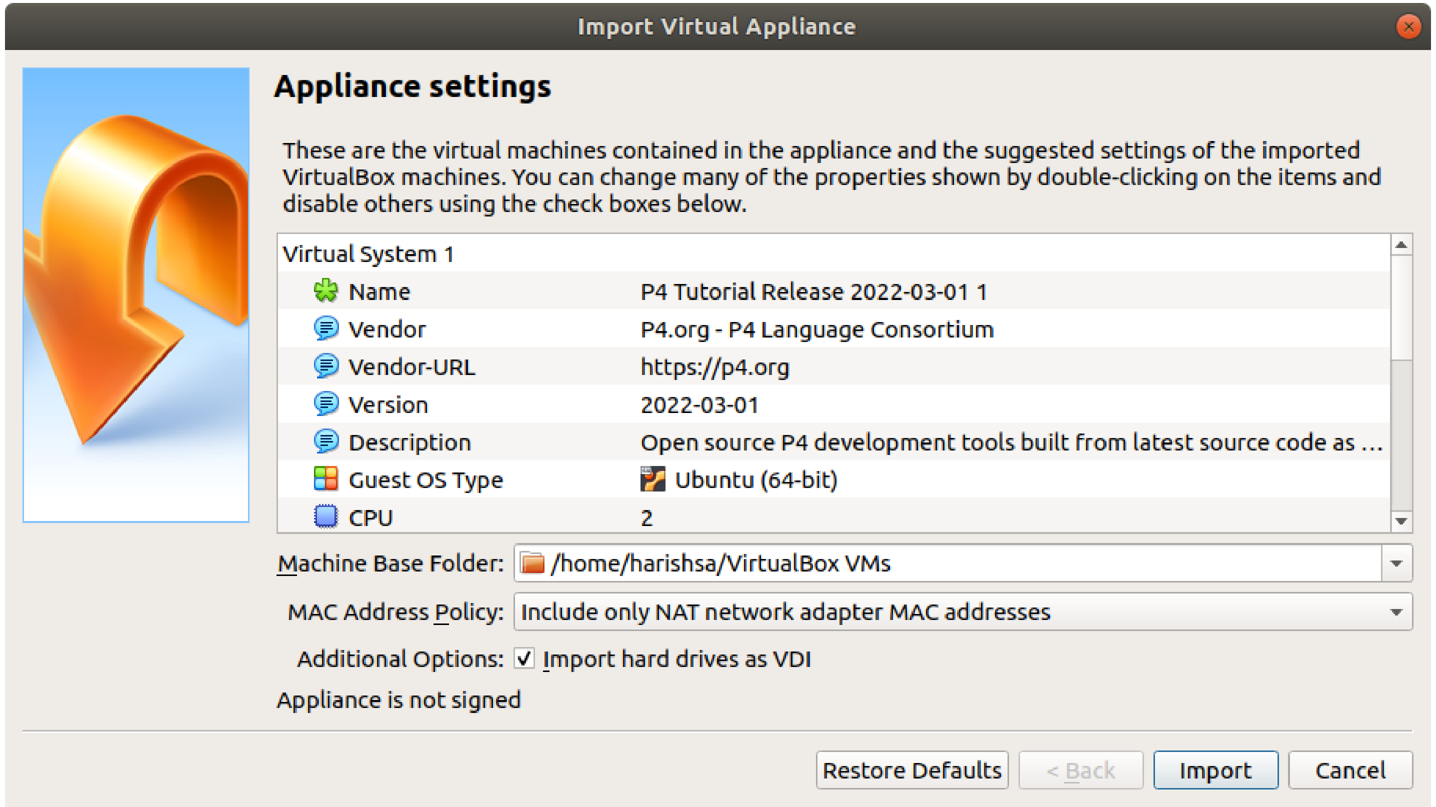
  Windows: https://www.virtualbox.org/wiki/Downloads
  Linux: https://www.virtualbox.org/wiki/Linux_Downloads

  The installation should be straightforward.
  Stick with the default settings.

2. Import VM Image

- Double click on the .ova file you just downloaded.
  Link: https://drive.google.com/file/d/1cgZh2wtyT878_GvndVKl-xeNojpB1Ms1/view

- Click on import and wait for the process to finish.



Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Virtual System 1	
Name	P4 Tutorial Release 2022-03-01 1
Vendor	P4.org - P4 Language Consortium
Vendor-URL	https://p4.org
Version	2022-03-01
Description	Open source P4 development tools built from latest source code as …
Guest OS Type	Ubuntu (64-bit)
CPU	2

Machine Base Folder: /home/harishsa/VirtualBox VMs

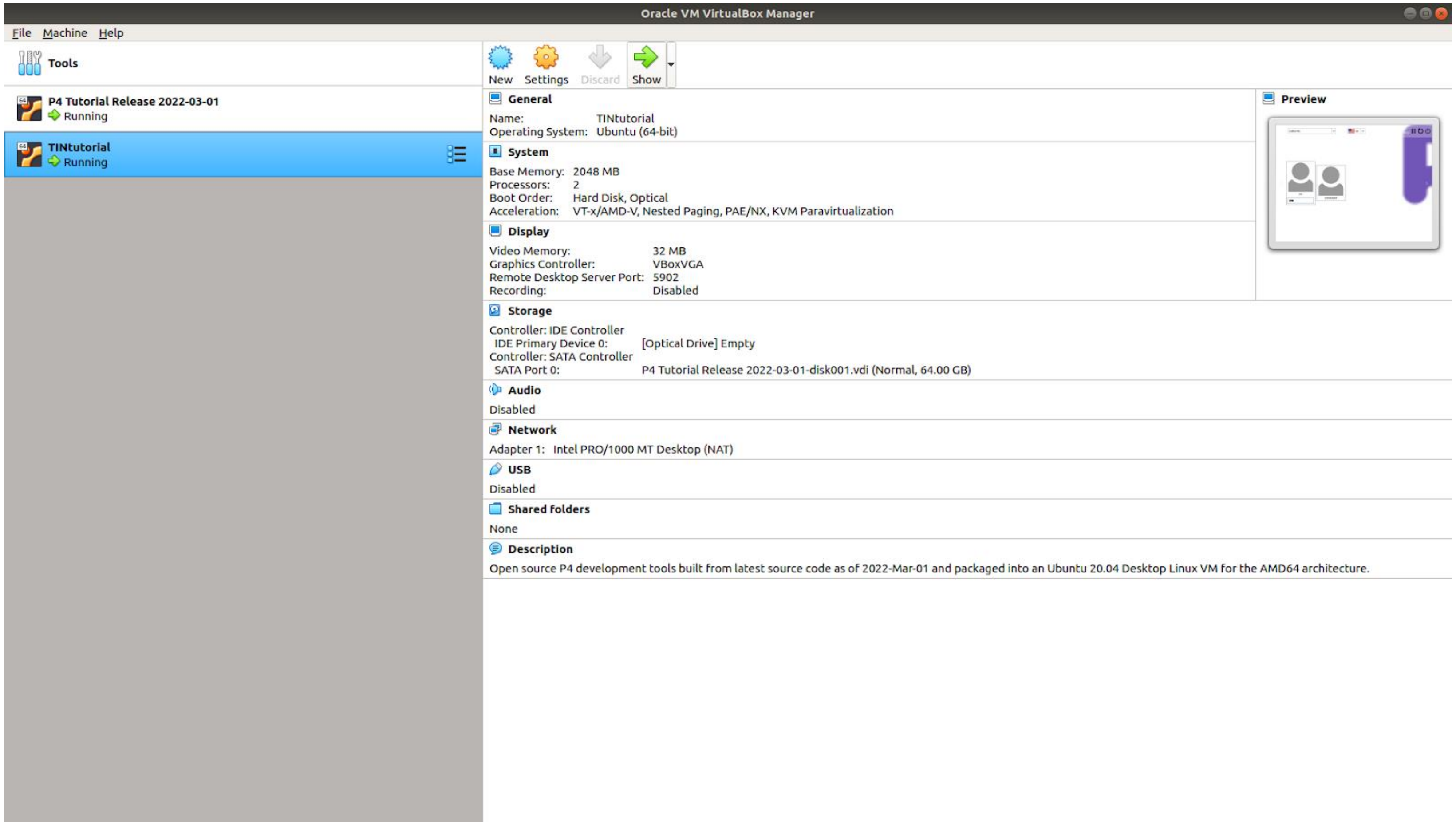
MAC Address Policy: Include only NAT network adapter MAC addresses

Additional Options: ☑ Import hard drives as VDI

Appliance is not signed

3. Boot up the VM

- The imported VM would be visible on the left pane as shown. Click on it and click start on the right pane.



P4 Tutorial Release 2022-03-01
Running

TINtutorial
Running

General

Name: TINtutorial
Operating System: Ubuntu (64-bit)

System

Base Memory: 2048 MB
Processors: 2
Boot Order: Hard Disk, Optical
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display

Video Memory: 32 MB
Graphics Controller: VBoxVGA
Remote Desktop Server Port: 5902
Recording: Disabled

Storage

Controller: IDE Controller
IDE Primary Device 0: [Optical Drive] Empty
Controller: SATA Controller
SATA Port 0: P4 Tutorial Release 2022-03-01-disk001.vdi (Normal, 64.00 GB)

Audio

Disabled

Network

Adapter 1: Intel PRO/1000 MT Desktop (NAT)

USB

Disabled

Shared folders

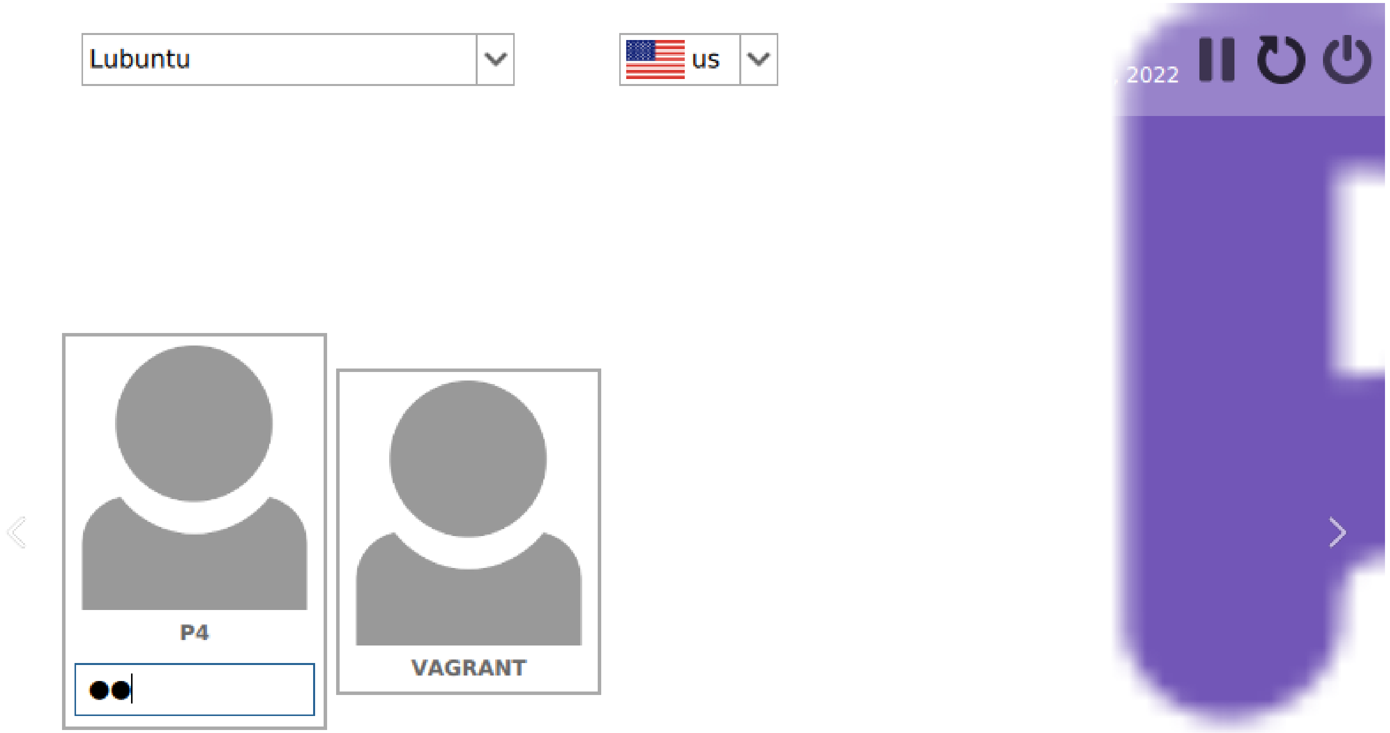
None

Description

Open source P4 development tools built from latest source code as of 2022-Mar-01 and packaged into an Ubuntu 20.04 Desktop Linux VM for the AMD64 architecture.

4. Log in to the VM

- Use password “p4”



5. Clone the repository

- Create a local copy of the hands-on material using git clone:

- Open Terminal and type the following command:

```
$ git clone https://github.com/pnl-iiitd/2023-P4-for-all-tutorial.git
```

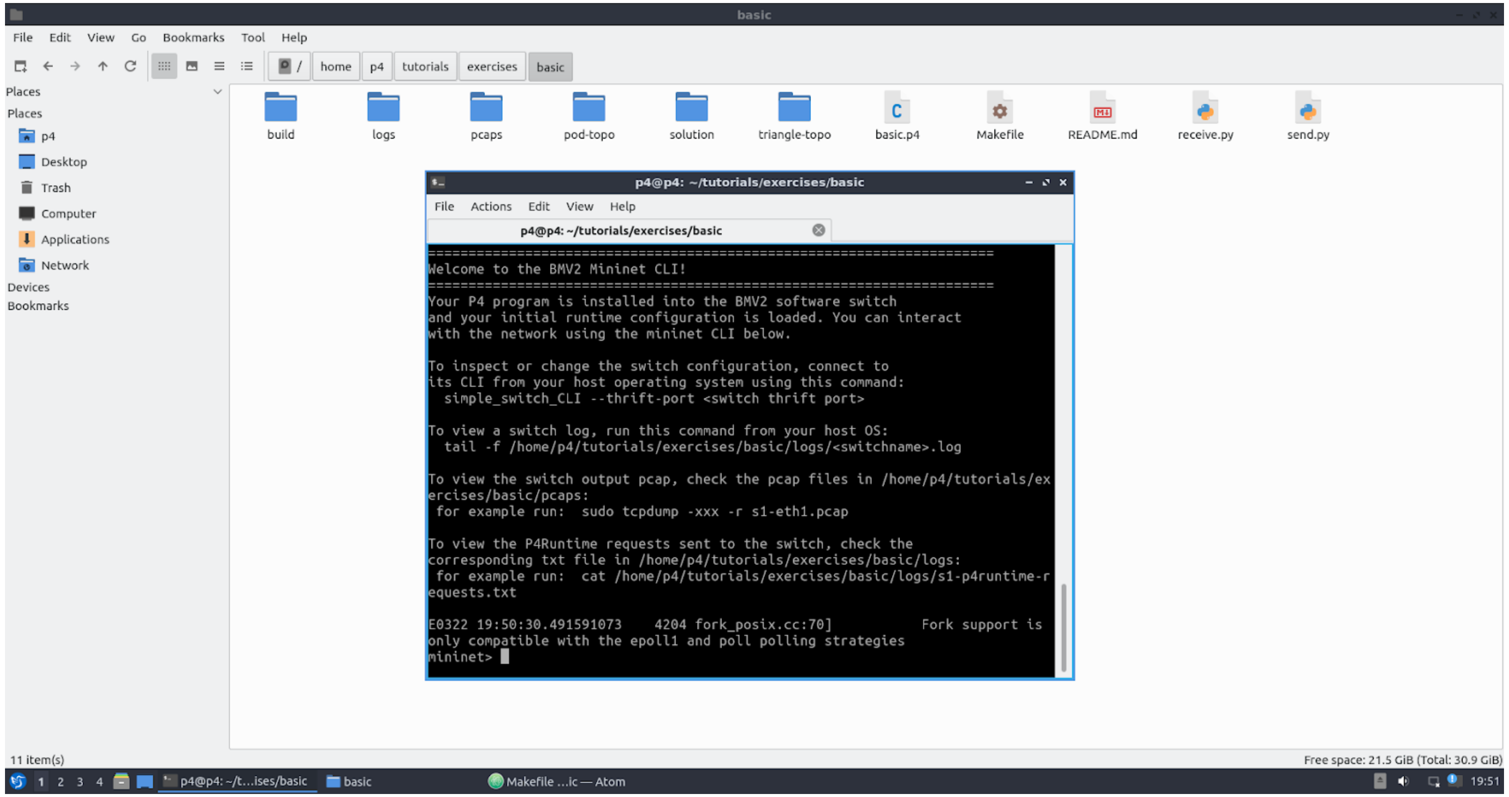
```
$ git clone pnl-iiitd/2023-P4-for-all-tutorial
```

- Verify that a new folder called 2023-P4-for-all-tutorial has been created in /home/p4

6. Environment Verification

- Open terminal and navigate to “/home/p4/2023-P4-for-all-tutorial/exercises/simple”
- Once inside the location execute the following command on the terminal:

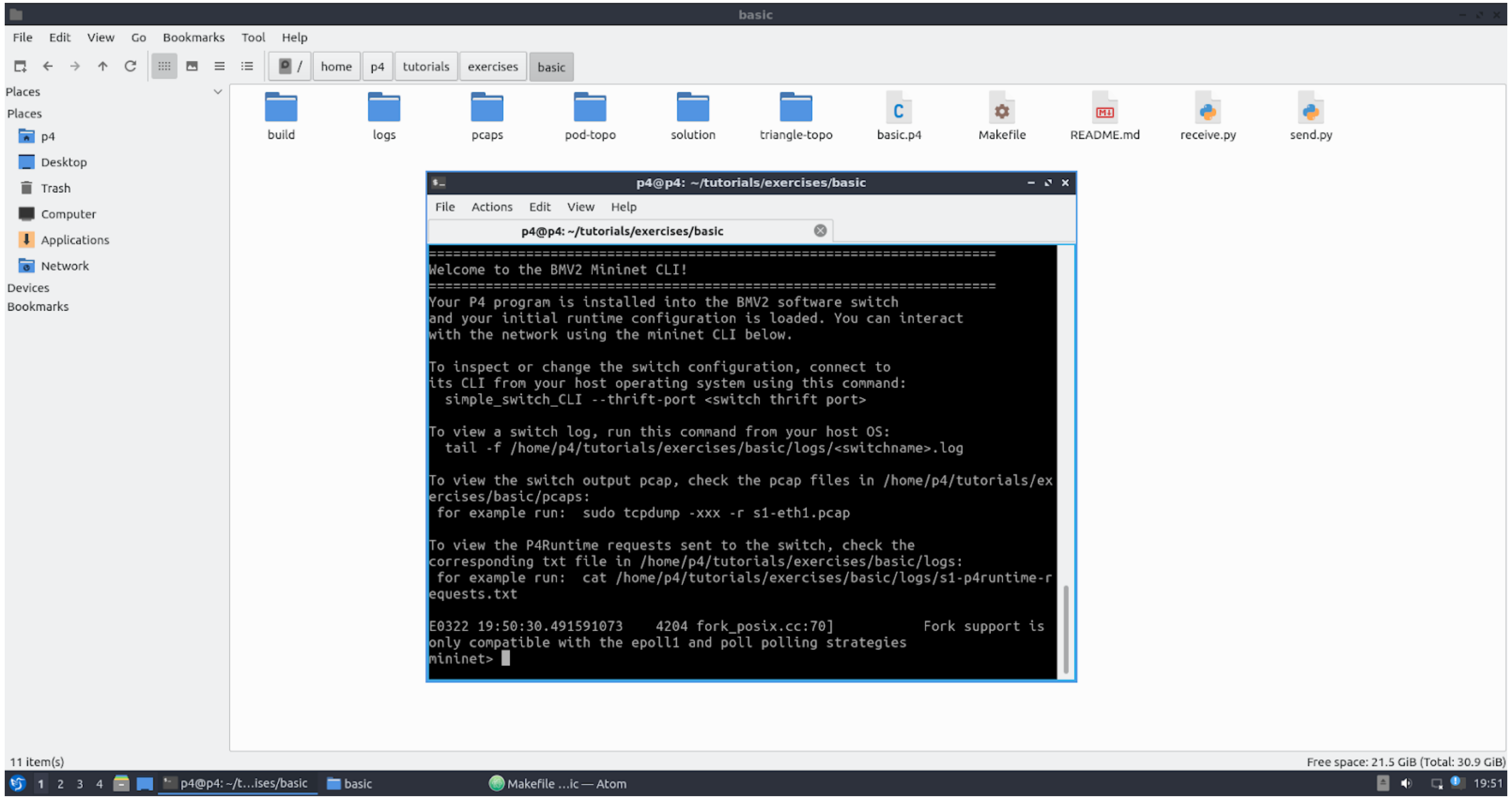
$ make



```
======================================================================
Welcome to the BMV2 Mininet CLI!
======================================================================
Your P4 program is installed into the BMV2 software switch
and your initial runtime configuration is loaded. You can interact
with the network using the mininet CLI below.

To inspect or change the switch configuration, connect to
its CLI from your host operating system using this command:
  simple_switch_CLI --thrift-port <switch thrift port>

To view a switch log, run this command from your host OS:
  tail -f /home/p4/tutorials/exercises/basic/logs/<switchname>.log

To view the switch output pcap, check the pcap files in /home/p4/tutorials/ex
ercises/basic/pcaps:
 for example run:  sudo tcpdump -xxx -r s1-eth1.pcap

To view the P4Runtime requests sent to the switch, check the
corresponding txt file in /home/p4/tutorials/exercises/basic/logs:
 for example run:  cat /home/p4/tutorials/exercises/basic/logs/s1-p4runtime-r
equests.txt

E0322 19:50:30.491591073    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> 
```

5. Environment Verification

- The screen shown below should be visible. We have now successfully started a mininet topology containing P4 compatible switches, hosts and links.



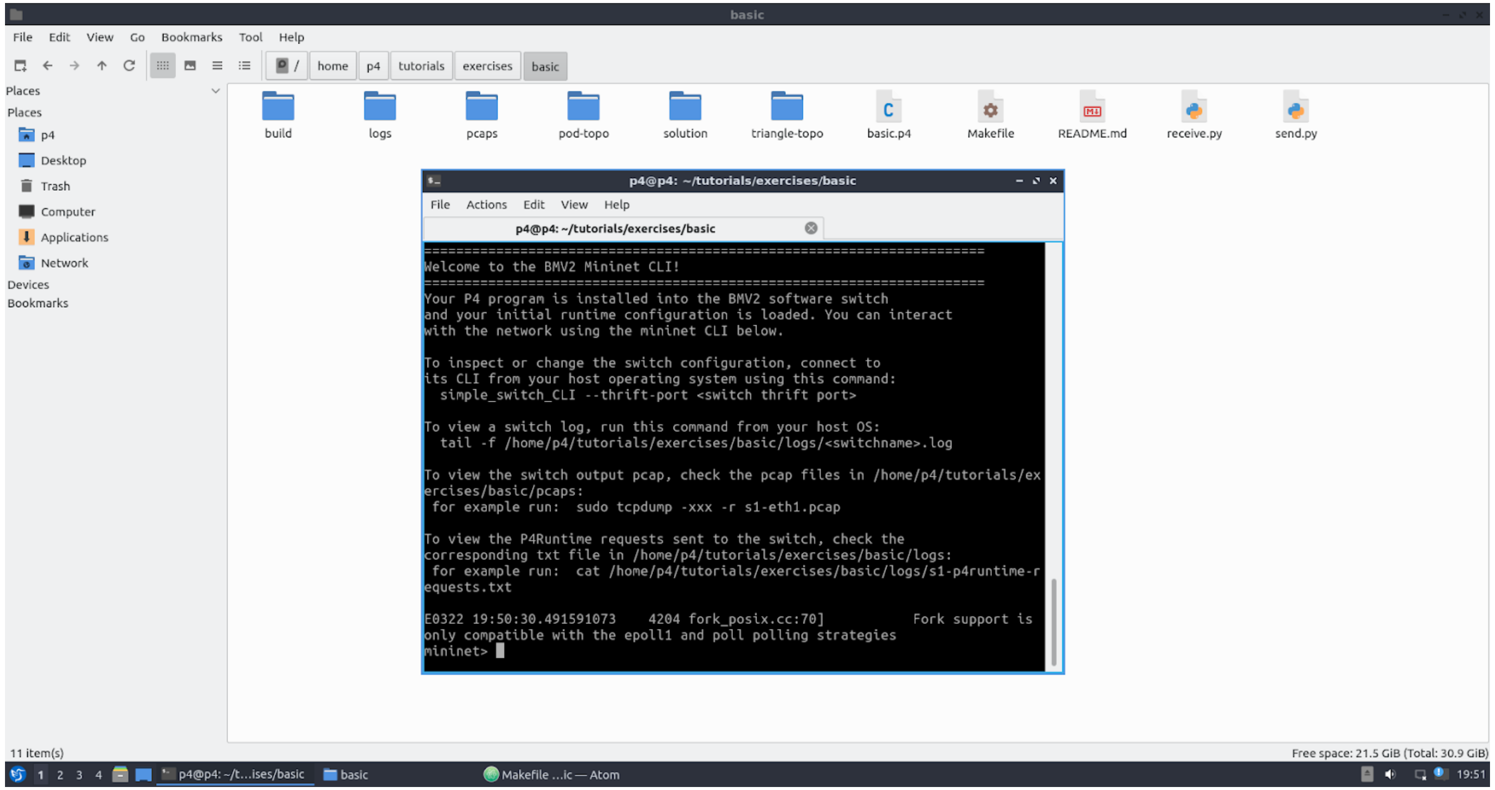
```
======================================================================
Welcome to the BMV2 Mininet CLI!
======================================================================
Your P4 program is installed into the BMV2 software switch
and your initial runtime configuration is loaded. You can interact
with the network using the mininet CLI below.

To inspect or change the switch configuration, connect to
its CLI from your host operating system using this command:
  simple_switch_CLI --thrift-port <switch thrift port>

To view a switch log, run this command from your host OS:
  tail -f /home/p4/tutorials/exercises/basic/logs/<switchname>.log

To view the switch output pcap, check the pcap files in /home/p4/tutorials/ex
ercises/basic/pcaps:
 for example run:  sudo tcpdump -xxx -r s1-eth1.pcap

To view the P4Runtime requests sent to the switch, check the
corresponding txt file in /home/p4/tutorials/exercises/basic/logs:
 for example run:  cat /home/p4/tutorials/exercises/basic/logs/s1-p4runtime-r
equests.txt

E0322 19:50:30.491591073    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> 
```

5. Environment Verification

- Next, inside the prompt type the following command:

  ```
  $ xterm h1
  ```

- A new xterm window as shown below should open up. Type “ifconfig” in the prompt and verify if the screenshot and your result are the same.



```
======================================================================
Welcome to the BMV2 Mininet CLI!
======================================================================
Your P4 program is installed into the BMV2 software switch
and your initial runtime configuration is loaded. You can interact
with the network using the mininet CLI below.

To inspect or change the switch configuration, connect to
its CLI from your host operating system using this command:
  simple_switch_CLI --thrift-port <switch thrift port>

To view a switch log, run this command from your host OS:
  tail -f /home/p4/tutorials/exercises/basic/logs/<switchname>.log

To view the switch output pcap, check the pcap files in /home/p4/tutorials/ex
ercises/basic/pcaps:
 for example run:  sudo tcpdump -xxx -r s1-eth1.pcap

To view the P4Runtime requests sent to the switch, check the
corresponding txt file in /home/p4/tutorials/exercises/basic/logs:
 for example run:  cat /home/p4/tutorials/exercises/basic/logs/s1-p4runtime-r
equests.txt

E0322 19:50:30.491591073    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> 
```

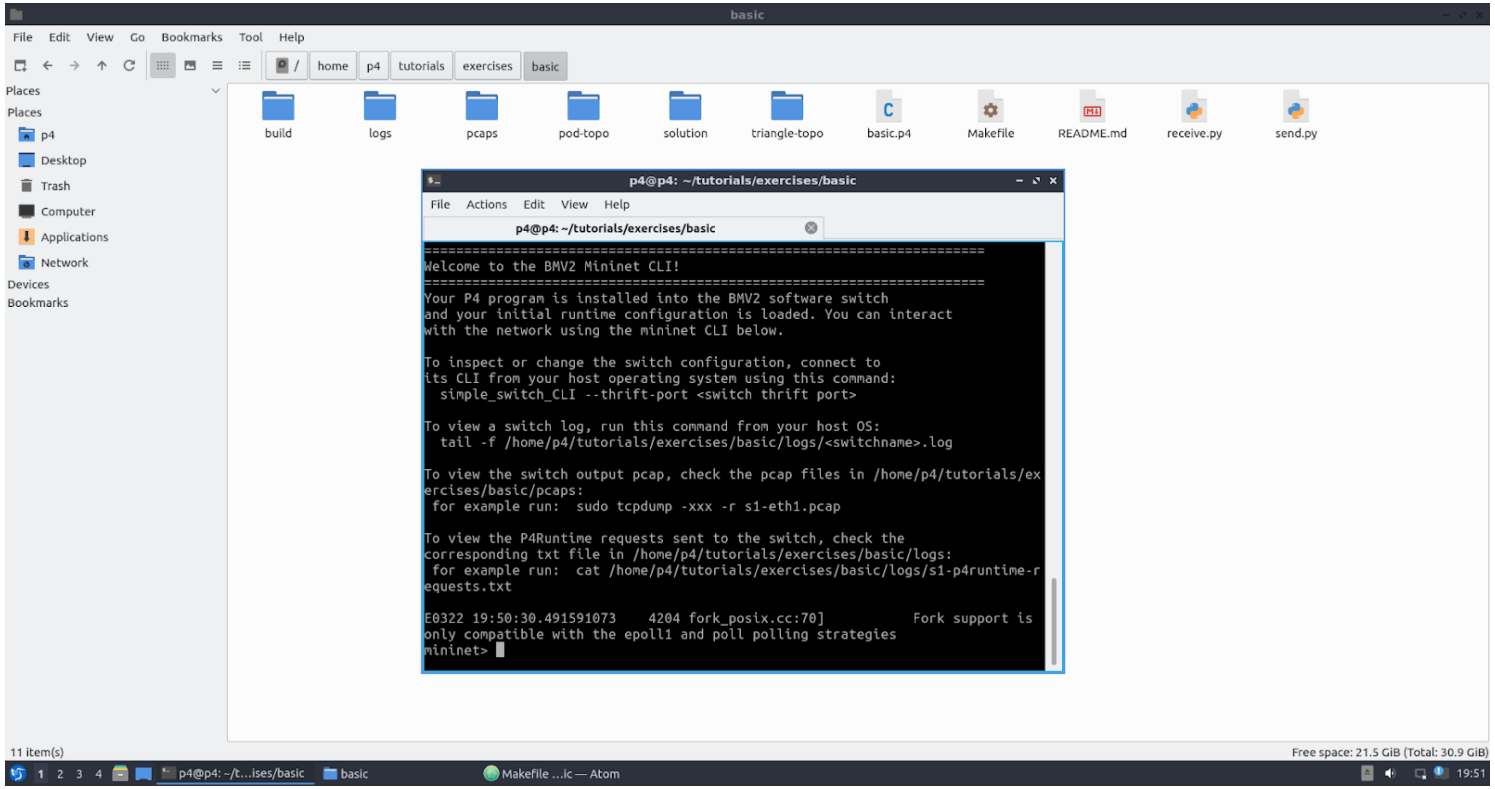
6. Wireshark

- Now we will generate packets and ensure that we can capture them on wireshark.

- Open up a new terminal and type the following command:

  $ sudo wireshark &

- Once wireshark opens up, double click the interface “s1-eth1” as shown:

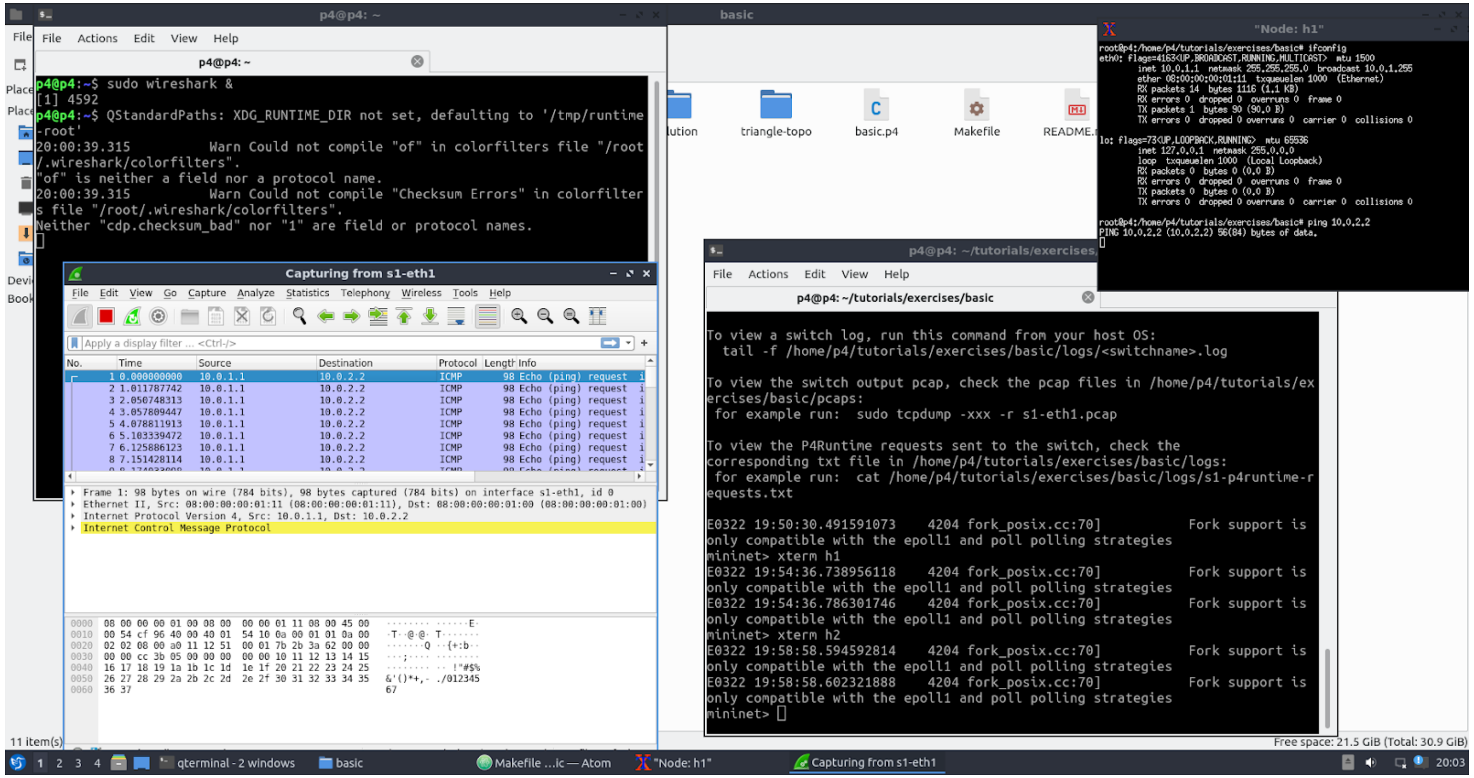


```
======================================================================
Welcome to the BMV2 Mininet CLI!
======================================================================
Your P4 program is installed into the BMV2 software switch
and your initial runtime configuration is loaded. You can interact
with the network using the mininet CLI below.

To inspect or change the switch configuration, connect to
its CLI from your host operating system using this command:
  simple_switch_CLI --thrift-port <switch thrift port>

To view a switch log, run this command from your host OS:
  tail -f /home/p4/tutorials/exercises/basic/logs/<switchname>.log

To view the switch output pcap, check the pcap files in /home/p4/tutorials/ex
ercises/basic/pcaps:
 for example run:  sudo tcpdump -xxx -r s1-eth1.pcap

To view the P4Runtime requests sent to the switch, check the
corresponding txt file in /home/p4/tutorials/exercises/basic/logs:
 for example run:  cat /home/p4/tutorials/exercises/basic/logs/s1-p4runtime-r
equests.txt

E0322 19:50:30.491591073    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> 
```

7. Test if all of it works

- Now go back to the xterm window and type the command:
  ping 10.0.2.2

- The wireshark screen will show ICMP packets captured



```
p4@p4:~$ sudo wireshark &
[1] 4592
p4@p4:~$ QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime
-root'
20:00:39.315          Warn Could not compile "of" in colorfilters file "/root
/.wireshark/colorfilters".
"of" is neither a field nor a protocol name.
20:00:39.315          Warn Could not compile "Checksum Errors" in colorfilter
s file "/root/.wireshark/colorfilters".
Neither "cdp.checksum_bad" nor "1" are field or protocol names.
```

```
root@p4:/home/p4/tutorials/exercises/basic# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.1.1  netmask 255.255.255.0  broadcast 10.0.1.255
        ether 08:00:00:00:01:11  txqueuelen 1000  (Ethernet)
        RX packets 14  bytes 1116 (1.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 90 (90.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@p4:/home/p4/tutorials/exercises/basic# ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
```

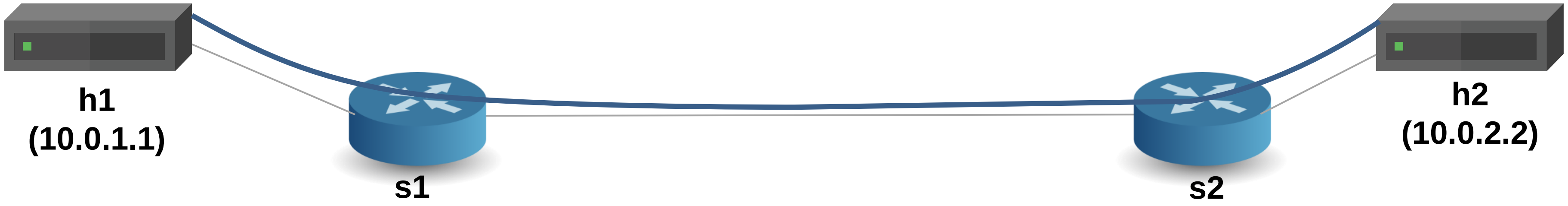
```
No.   Time          Source      Destination  Protocol  Length Info
    1 0.000000000   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    2 1.011787742   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    3 2.050748313   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    4 3.057809447   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    5 4.078811913   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    6 5.103339472   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    7 6.125886123   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i
    8 7.151428114   10.0.1.1    10.0.2.2     ICMP          98 Echo (ping) request  i

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s1-eth1, id 0
▸ Ethernet II, Src: 08:00:00:00:01:11 (08:00:00:00:01:11), Dst: 08:00:00:00:01:00 (08:00:00:00:01:00)
▸ Internet Protocol Version 4, Src: 10.0.1.1, Dst: 10.0.2.2
▸ Internet Control Message Protocol

0000  08 00 00 00 01 00 08 00  00 00 01 11 08 00 45 00   ········ ······E·
0010  00 54 cf 96 40 00 40 01  54 10 0a 00 01 01 0a 00   ·T··@·@· T·······
0020  02 02 08 00 a0 11 12 51  00 01 7b 2b 3a 62 00 00   ·······Q ··{+:b··
0030  00 00 cc 3b 05 00 00 00  00 00 10 11 12 13 14 15   ···;···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

```
To view a switch log, run this command from your host OS:
  tail -f /home/p4/tutorials/exercises/basic/logs/<switchname>.log

To view the switch output pcap, check the pcap files in /home/p4/tutorials/ex
ercises/basic/pcaps:
 for example run:  sudo tcpdump -xxx -r s1-eth1.pcap

To view the P4Runtime requests sent to the switch, check the
corresponding txt file in /home/p4/tutorials/exercises/basic/logs:
 for example run:  cat /home/p4/tutorials/exercises/basic/logs/s1-p4runtime-r
equests.txt

E0322 19:50:30.491591073    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> xterm h1
E0322 19:54:36.738956118    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
E0322 19:54:36.786301746    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> xterm h2
E0322 19:58:58.594592814    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
E0322 19:58:58.602321888    4204 fork_posix.cc:70]           Fork support is
only compatible with the epoll1 and poll polling strategies
mininet> 
```

What did we just do?

Simple Forwarding: topology

- We sent ICMP packets from h1 to h2



Lab 1: Basic Forwarding

Link

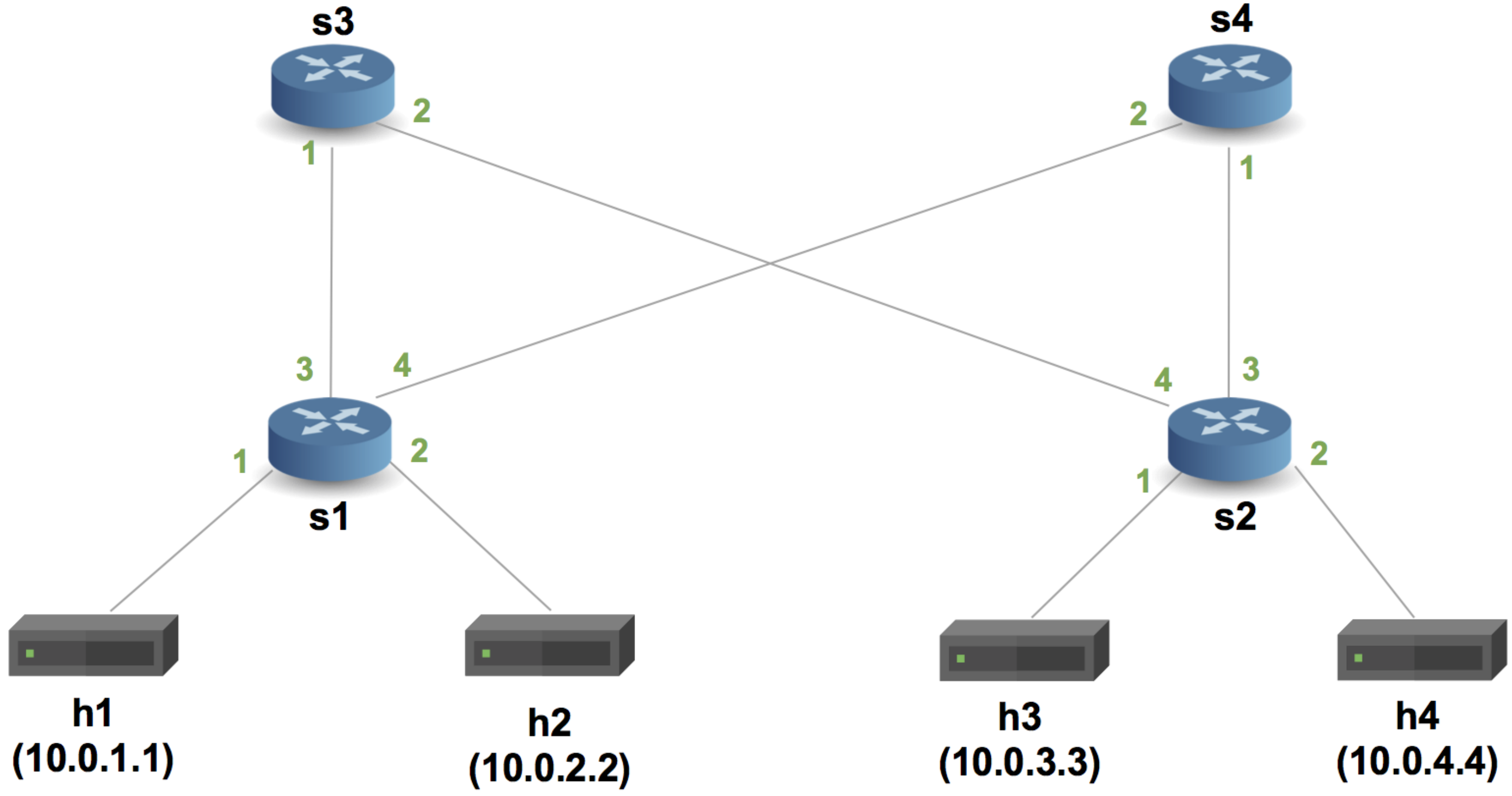
Basic Forwarding

- The objective of this exercise is to write a P4 program that implements basic forwarding. To keep things simple, we will just implement forwarding for IPv4.

- With IPv4 forwarding, the switch must perform the following actions for every packet:
  (i) update the source and destination MAC addresses,
  (ii) decrement the time-to-live (TTL) in the IP header, and
  (iii) forward the packet out the appropriate port.

Basic Forwarding: topology

- We will use the following topology for this exercise. It is a single pod of a fat-tree topology and henceforth referred to as pod-topo:



Basic Forwarding

- The skeleton P4 program, **basic.p4** initially drops all packets. We need to extend this skeleton program to properly forward IPv4 packets.

Before that, let's compile the incomplete basic.p4 and bring up a switch in Mininet to test its behavior!

Basic Forwarding

Commands:

```
> make run
```

- **This will:**
  - compile **basic.p4**, and
  - start the pod-topo in Mininet and configure all switches with the appropriate P4 program + table entries, and
  - configure all hosts with the commands listed in pod-topo/topology.json

Let’s run some more commands

Basic Forwarding

Commands:

```
mininet> h1 ping h2
```

```
mininet> pingall
```

Let’s bring up one of the device interface and run “ifconfig”

```
mininet> xterm h1
```

```
mininet> xterm h2
```

Based on the IP address, ping h2 from h1: ping 10.0.2.2

Unable to ping any hosts! Something is not fine!

```
make stop
```

```
make clean
```

Basic Forwarding

- The ping failed because each switch is programmed according to **basic.p4**, which drops all packets on arrival

Goal: Extend this file so it forwards packets

Basic Forwarding

- Replace all the /* TODO: …… */ in **basic.p4**
- Once filled, bring up mininet using make run
- Try h1 ping h2
- Bring up the respective xterms of the hosts and try pinging using the IP address

Now, let us dissect what we just did!

Basic Forwarding

- **Let us first look at the Makefile!**

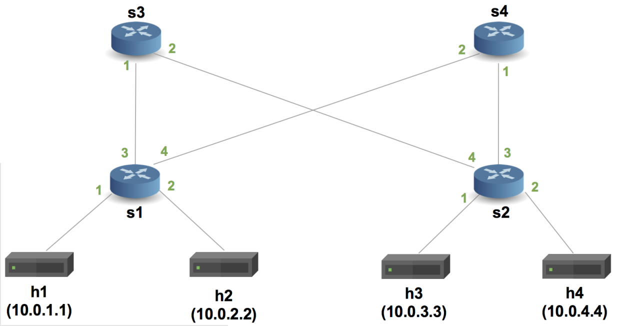
```
BMV2_SWITCH_EXE = simple_switch_grpc
TOPO = pod-topo/topology.json

include ../../utils/Makefile
```

Let’s look at topology.json

Basic Forwarding

topology.json

```
{
    "hosts": {
        "h1": {"ip": "10.0.1.1/24", "mac": "08:00:00:00:01:11",
               "commands":["route add default gw 10.0.1.10 dev eth0",
                           "arp -i eth0 -s 10.0.1.10 08:00:00:00:01:00"] },
…
},
    "switches": {
        "s1": { "runtime_json" : "pod-topo/s1-runtime.json" },
….
},
    "links": [
        ["h1", "s1-p1"], ["h2", "s1-p2"], ["s1-p3", "s3-p1"], ["s1-p4", "s4-p2"],
        ["h3", "s2-p1"], ["h4", "s2-p2"], ["s2-p3", "s4-p1"], ["s2-p4", "s3-p2"]
    ]
}
```



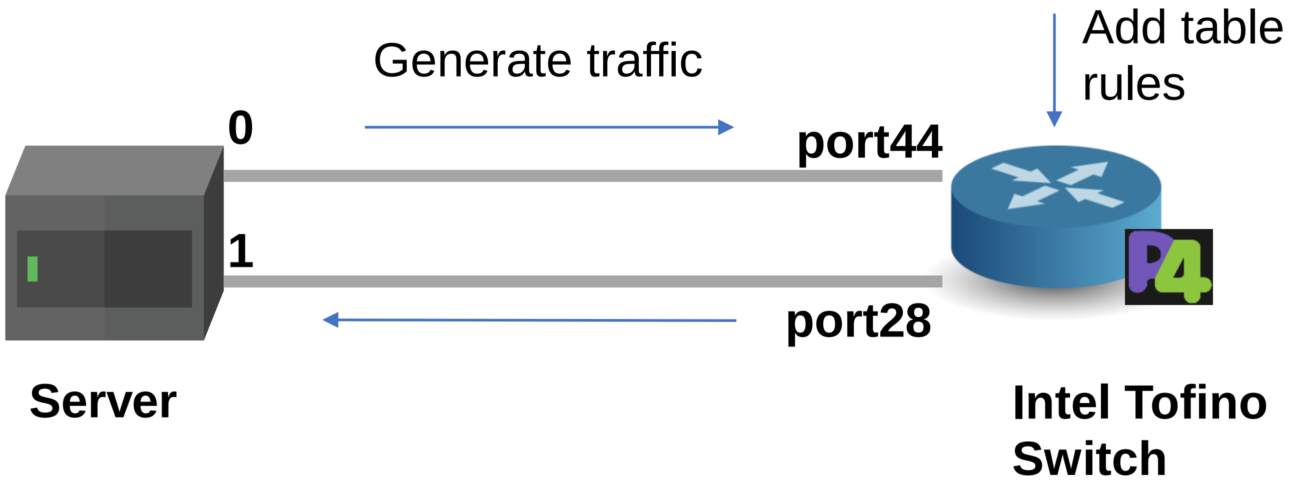
Let’s analyze one of the runtime files

Basic Forwarding

s1-runtime.json

```
"target": "bmv2",
  "p4info": "build/basic.p4.p4info.txt",
  "bmv2_json": "build/basic.json",
  "table_entries": [
    ……
    {
      "table": "MyIngress.ipv4_lpm",
      "match": {
        "hdr.ipv4.dstAddr": ["10.0.1.1", 32]
      },
      "action_name": "MyIngress.ipv4_forward",
      "action_params": {
        "dstAddr": "08:00:00:00:01:11",
        "port": 1
      }
    },
```

Tofino Demonstration



Tofino

- Initialize the switch
- Compile the code
- Add the ports: QSFP ports
- Now observe using BFRT the table rules and values, etc

Lab2: Basic Tunneling

Link

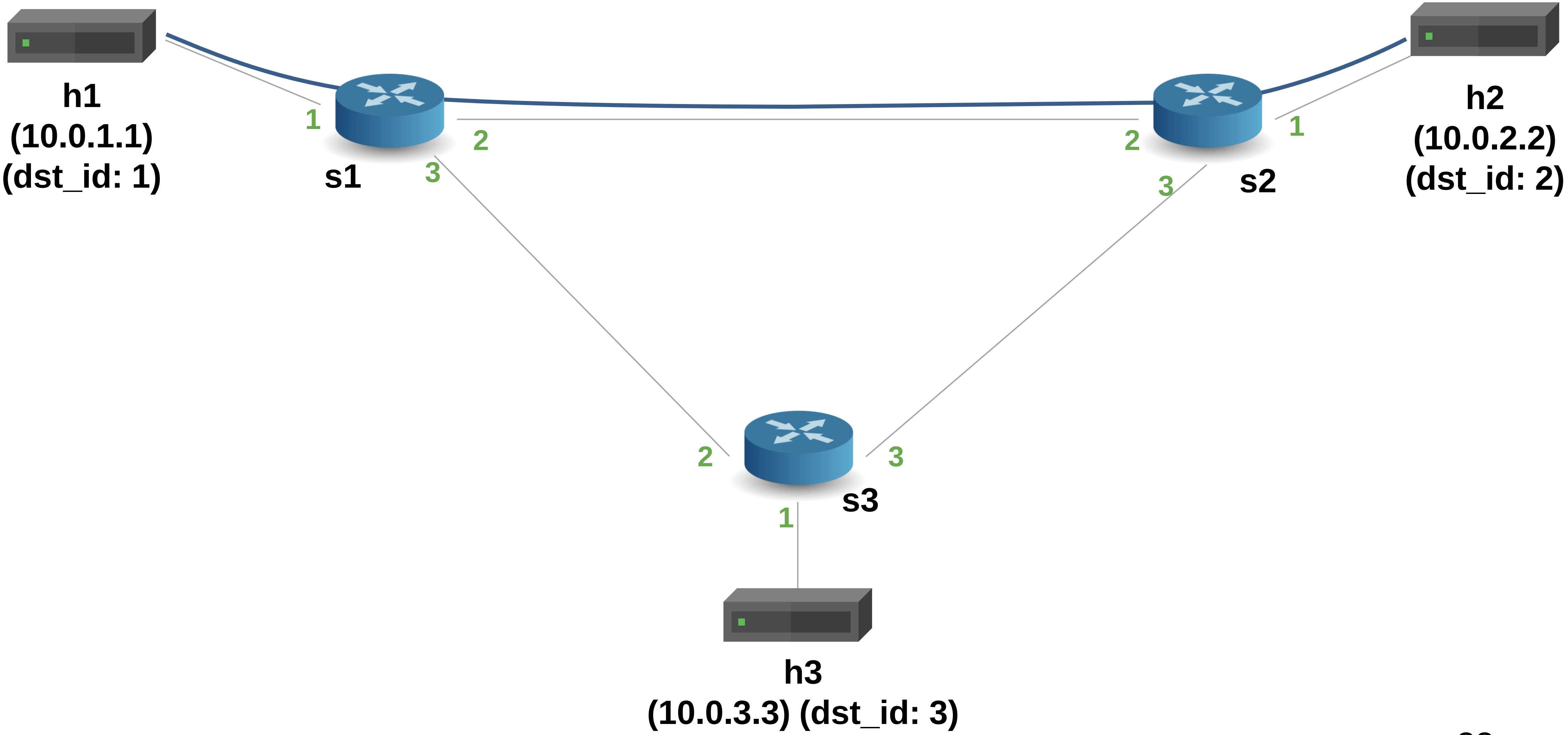
Basic Tunneling

- We will add support for a basic tunneling protocol to the IP router that you completed in the previous assignment. The basic switch forwards based on the destination IP address.

- Define a new header type to encapsulate the IP packet and modify the switch code, so that it instead decides the destination port using a new tunnel header.

- The new header type will contain a protocol ID, which indicates the type of packet being encapsulated, along with a destination ID to be used for routing.

Basic Tunneling: topology



Basic Tunneling

- Add support for basic tunneling to the basic IP router

- Define a new header type (`myTunnel`) to encapsulate the IP packet

- `myTunnel` header includes:
  - `proto_id` : type of packet being encapsulated
  - `dst_id` : ID of destination host

- Modify the switch to perform routing using the `myTunnel` header

Basic Tunneling TODO List

- Define `myTunnel_t` header type and add to `headers` struct
- Update parser
- Define `myTunnel_forward` action
- Define `myTunnel_exact` table
- Update table application logic in `MyIngress apply` statement
- Update deparser
- Adding forwarding rules

FAQs

- **Can I apply a table multiple times in my P4 Program?**
  - No (except via resubmit / recirculate)
- **Can I modify table entries from my P4 Program?**
  - No (except for direct counters)
- **What happens upon reaching the `reject` state of the parser?**
  - Architecture dependent
- **How much of the packet can I parse?**
  - Architecture dependent