Advanced Topics in Communication Networks

Exercise session — Oct. 6, 2020


Edgar


Ege


Roland


Eric


Yannick

adv-net.ethz.ch

ETH zürich

MPLS exercise solutions are out!

Name	Last commit	Last update
..		
images	week 2 exercises	1 week ago
mpls_basics	added mpls exercise solutions	16 hours ago
mpls_stacked	added mpls exercise solutions	16 hours ago
README.md	lead them to use tcpdump instead pcaps	5 days ago

README.md

Multiprotocol Label Switching (MPLS)

In the previous week's tutorial, you learned how to forward packets by reading their MAC addresses and selecting the next hop from a match-action table. In today's exercise, you will see a different forwarding technique: Multiprotocol Label Switching. Your final objective will be to implement a basic version of MPLS forwarding in the data plane.

MPLS attaches labels to data packets to drive the packet forwarding decisions. That is, instead of forwarding packets based on their IP addresses, switches forward packets just by looking up the contents of the MPLS labels that are attached to packets.

As you will see, MPLS has numerous benefits such as the possibility of creating end-to-end circuits for all types of packets or an extensive support for traffic engineering.

MPLS Terminology

Before we begin, we need some vocabulary to understand the concepts better:

- Multiprotocol Label Switching (MPLS): A highly scalable, data-carrying mechanism that is independent of any data link layer protocol.
- Label Edge Router (LER): A router that operates at the edges of an MPLS network. An LER determines and applies the appropriate labels and forwards the labeled packets into the MPLS domain.
- Label Switch Router (LSR): A router that switches the labels that are used to route packets through an MPLS network. You can understand LSRs as *all* the MPLS-capable switches in the network. LERs are also LSRs.

MPLS exercise solutions are out!

Name	Last commit	Last update
..		
solution	added mpls exercise solutions	16 hours ago
sx-commands	week 2 exercises	1 week ago
basics.p4	week 2 exercises	1 week ago
p4app.json	week 2 exercises	1 week ago

Multiprotocol Label Switching (MPLS)

In the previous week's tutorial, you learned how to forward packets by reading their MAC addresses and selecting the next hop from a match-action table. In today's exercise, you will see a different forwarding technique: Multiprotocol Label Switching. Your final objective will be to implement a basic version of MPLS forwarding in the data plane.

MPLS attaches labels to data packets to drive the packet forwarding decisions. That is, instead of forwarding packets based on their IP addresses, switches forward packets just by looking up the contents of the MPLS labels that are attached to packets.

As you will see, MPLS has numerous benefits such as the possibility of creating end-to-end circuits for all types of packets or an extensive support for traffic engineering.

MPLS Terminology

Before we begin, we need some vocabulary to understand the concepts better:

- Multiprotocol Label Switching (MPLS): A highly scalable, data-carrying mechanism that is independent of any data link layer protocol.
- Label Edge Router (LER): A router that operates at the edges of an MPLS network. An LER determines and applies the appropriate labels and forwards the labeled packets into the MPLS domain.
- Label Switch Router (LSR): A router that switches the labels that are used to route packets through an MPLS network. You can understand LSRs as all the MPLS-capable switches in the network. LERs are also LSRs.

MPLS exercise solutions are out!

Name	Last commit	Last update
..		
solution	added mpls exercise solutions	16 hours ago
sx-commands	week 2 exercises	1 week ago
basics.p4	week 2 exercises	1 week ago
p4app.json	week 2 exercises	1 week ago

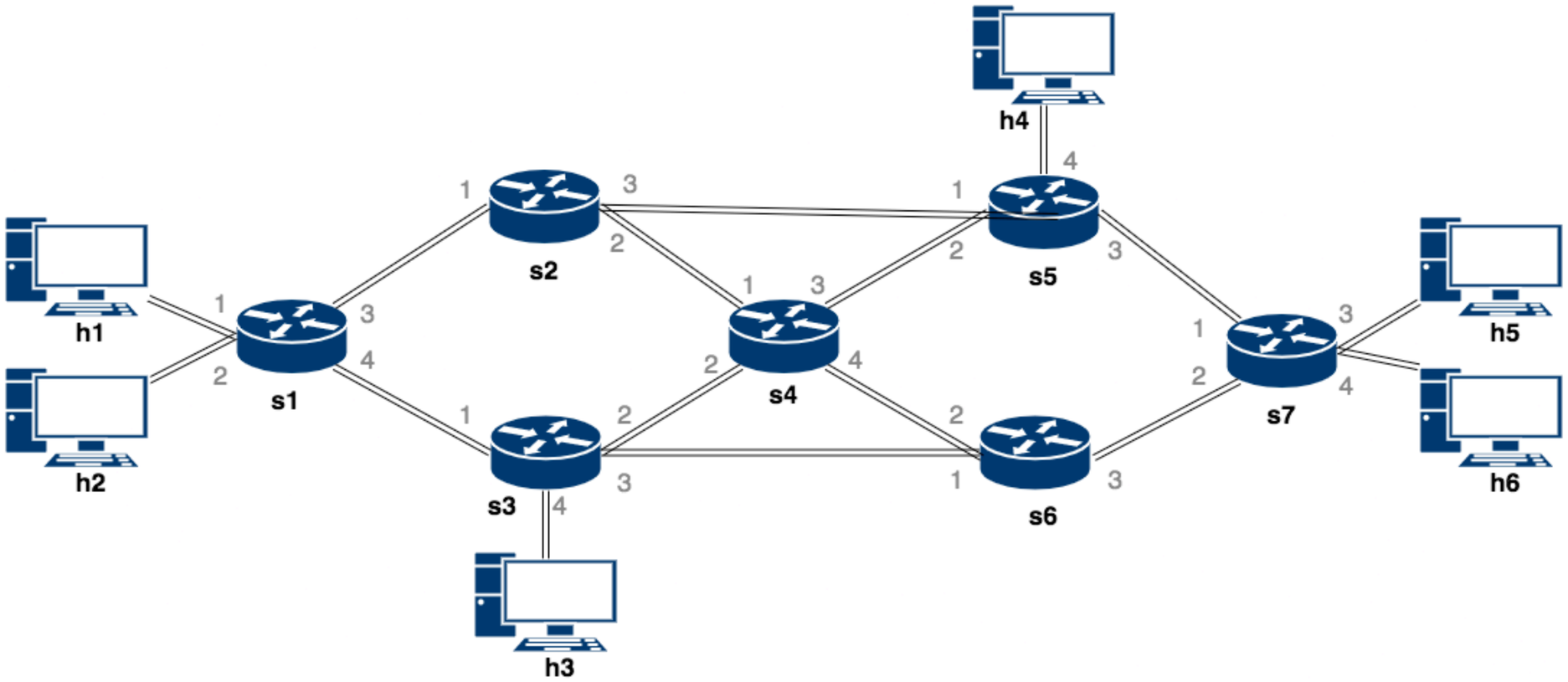
Multiprotocol Label Switching (MPLS)

Name	Last commit	Last update
..		
sx-commands	added mpls exercise solutions	16 hours ago
basics.p4	added mpls exercise solutions	16 hours ago
p4app.json	added mpls exercise solutions	16 hours ago

Before we begin, we need some vocabulary to understand the concepts better:

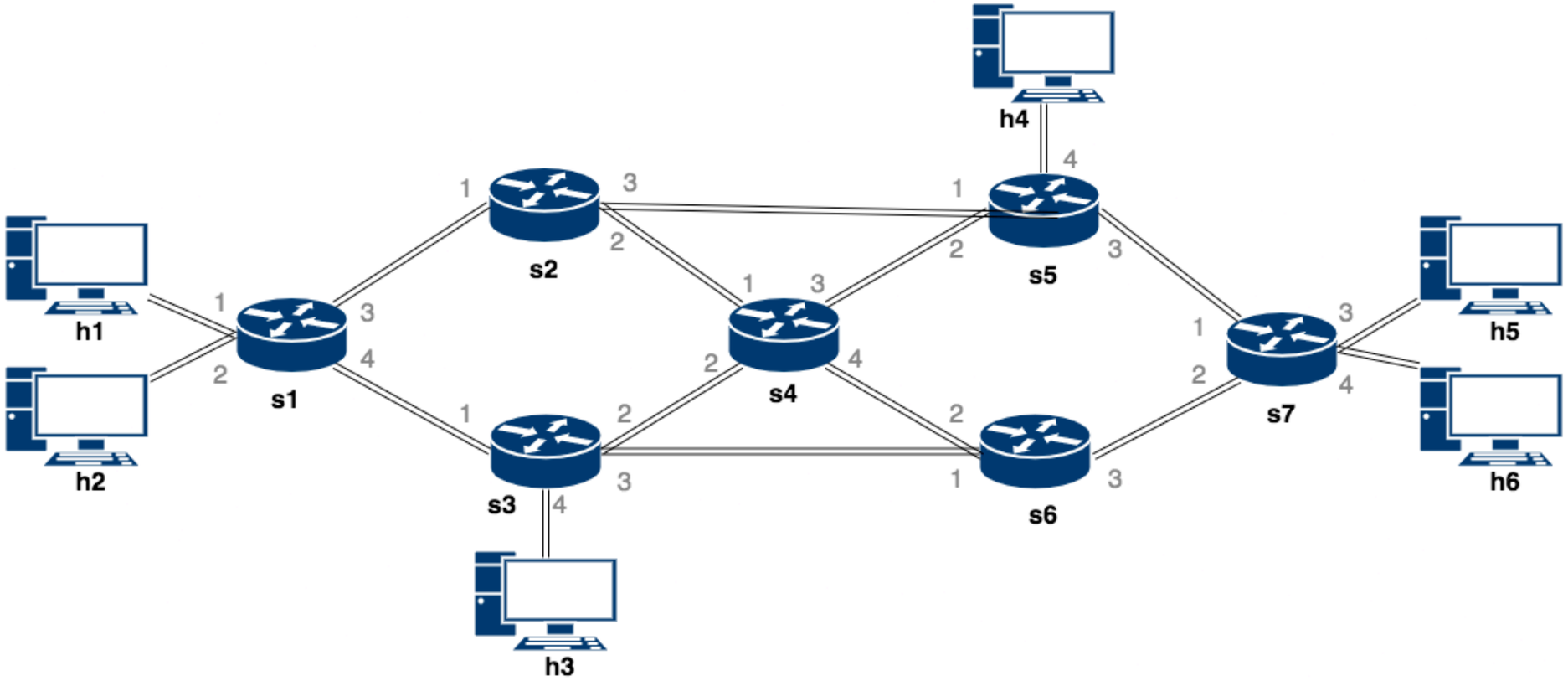
- Multiprotocol Label Switching (MPLS): A highly scalable, data-carrying mechanism that is independent of any data link layer protocol.
- Label Edge Router (LER): A router that operates at the edges of an MPLS network. An LER determines and applies the appropriate labels and forwards the labeled packets into the MPLS domain.
- Label Switch Router (LSR): A router that switches the labels that are used to route packets through an MPLS network. You can understand LSRs as all the MPLS-capable switches in the network. LERs are also LSRs.

This week: continue with the RSVP exercise

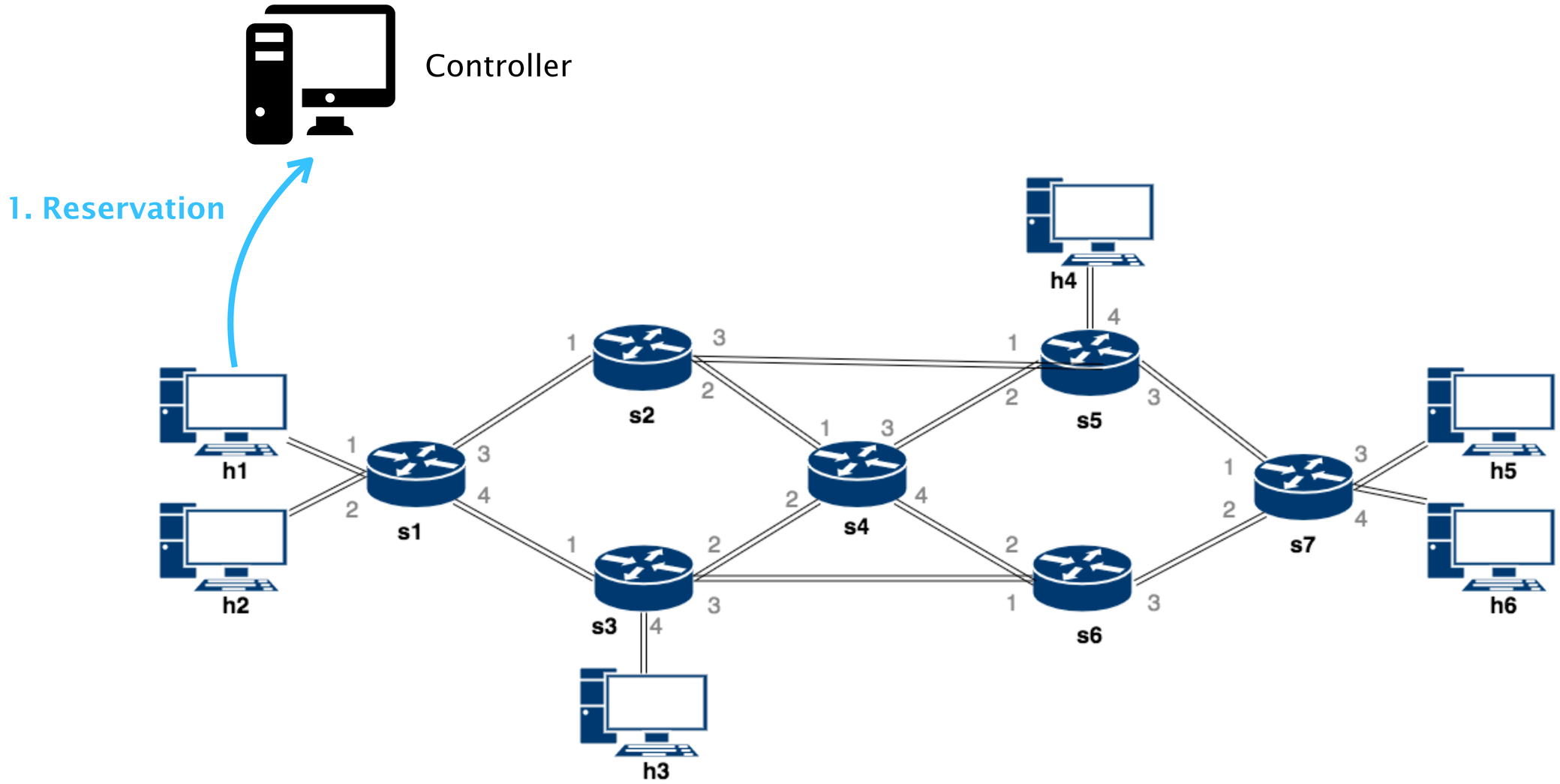


This week: continue with the RSVP exercise

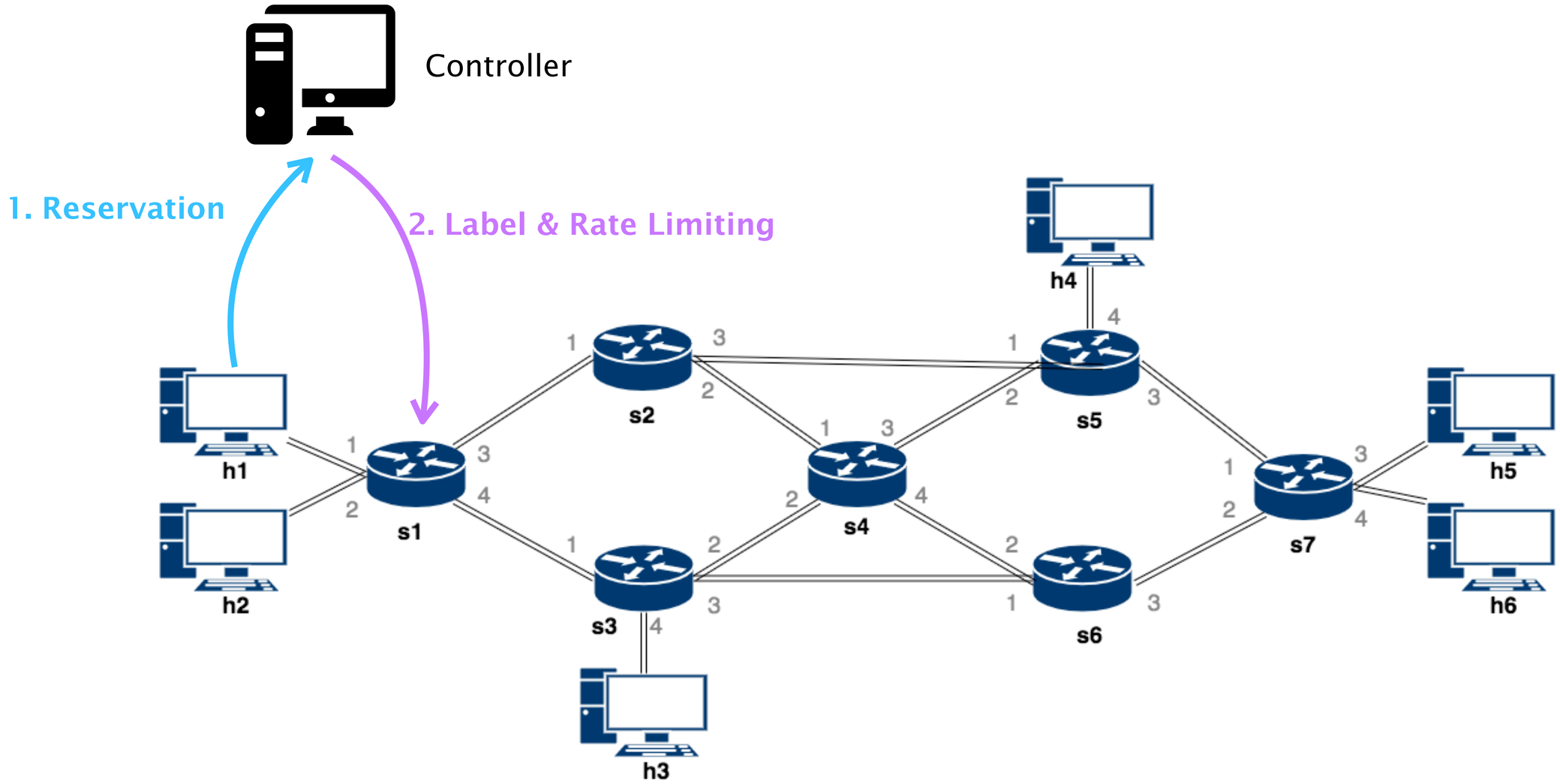
Controller



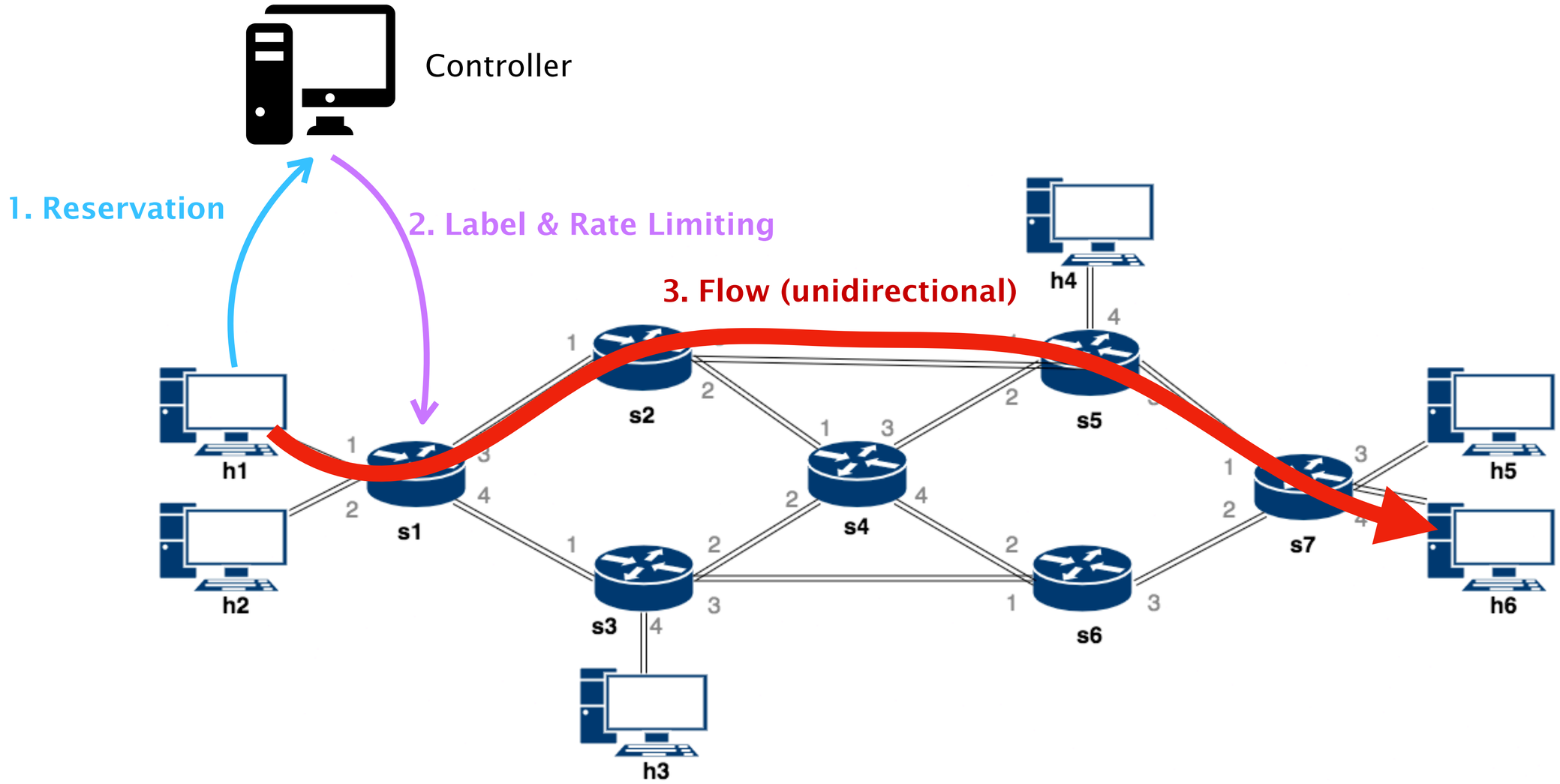
This week: continue with the RSVP exercise



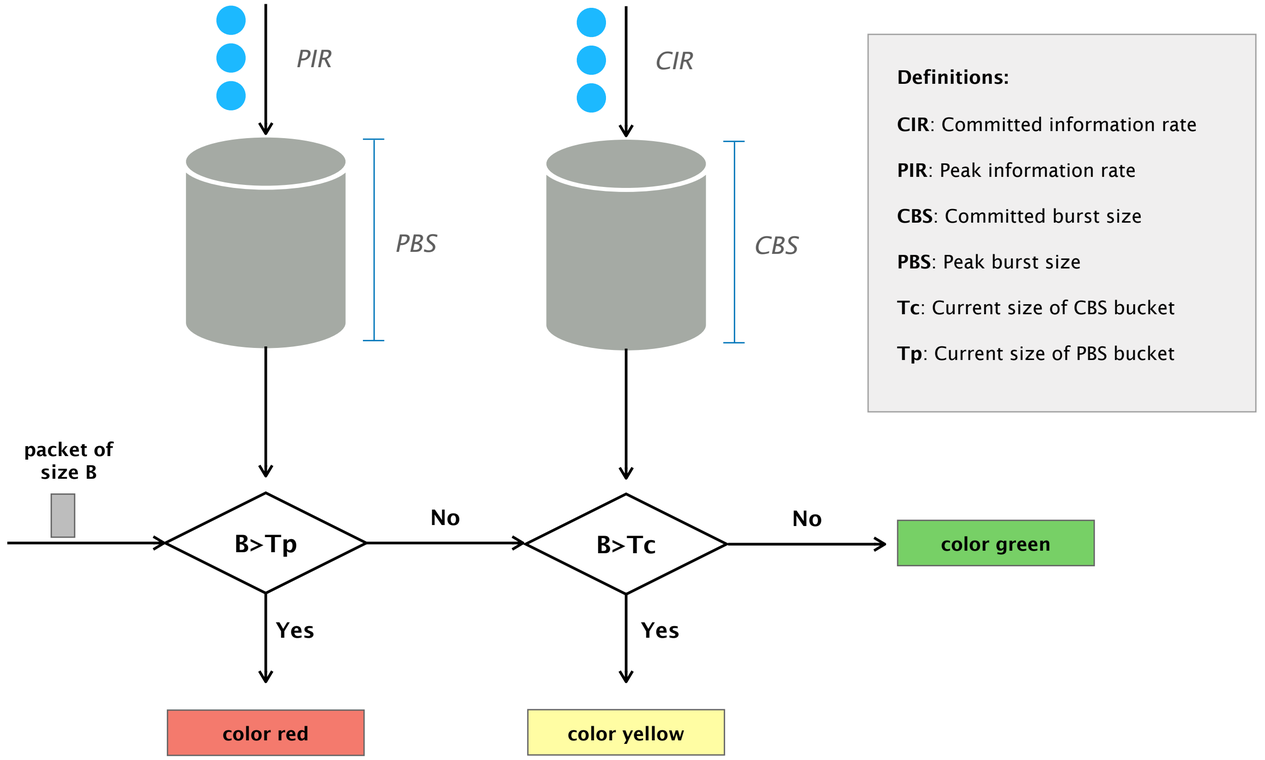
This week: continue with the RSVP exercise



This week: continue with the RSVP exercise



This week: meters to rate limit reservations



This week: add reservations with priorities

We proposed a possible algorithm

Task 2: Reservation with priorities

At this point, you should have a working RSVP centralized controller. However, we still did not use the `priority` attribute when performing the allocations. In this task, you will have to modify the `add_reservation` controller function such that some reservations have a higher priority over others.

When a high priority reservation comes and there is no space left many different things could be done. For example, previous reservations can be moved to make space, reservations with lower priority can be deleted to make space, etc. We propose one possible implementation, which will then be the one we will provide as a solution. Our proposed solution is not perfect, but does well enough while not needing a major change to the current implementation. In any case, feel free to implement a different algorithm.

We propose the following algorithm:

1. if the reservation (addition or modification) fits in the current network we do the same as before.
2. if the reservation (addition or modification) does not fit in the current network:
3. remove (`virtually`) all the reservations with a lower priority than the requested one. Hint: you don't need to remove them from switches, you can just remove its capacities.
4. Check if now, (without the low priority reservations) the requested reservation fits the network.
5. if it fits, add or modify it. Now, take all the reservations with lower priority that you `virtually` removed, and allocate them starting with the ones with the highest priority. Some will remain in the same path, some will be moved to a new path, and some, if they don't fit anymore, will be deleted.
6. if it does not fit, and there is an existing entry for this reservation, delete it.

> Hint: make sure `self.links_capacity` is updated properly in each of your steps. Sometimes, when you virtually remove reservations you will have to add the capacity to links.

Testing reservations with priorities

Test your final solution. First, start the topology and the controller.

This week: add reservations with priorities

We proposed a possible algorithm

Task 2: Reservation with priorities

At this point, you should have a working RSVP centralized controller. However, we still did not use the `priority` attribute when performing the allocations. In this task, you will have to modify the `add_reservation` controller function such that some reservations have a higher priority over others.

When a high priority reservation comes and there is no space left many different things could be done. For example, previous reservations can be moved to make space, reservations with lower priority can be deleted to make space, etc. We propose one possible implementation, which will not be the one we will provide as a solution. Our proposed solution is not perfect, but does well enough while not needing [illegible] Feel free to solve it as you want

Feel free to solve it as you want

We propose the following algorithm:

1. if the reservation (addition or modification) fits in the current network we do the same as before.
2. if the reservation (addition or modification) does not fit in the current network:
3. remove (`virtually`) all the reservations with a lower priority than the requested one. Hint: you don't need to remove them from switches, you can just remove its capacities.
4. Check if now, (without the low priority reservations) the requested reservation fits the network.
5. if it fits, add or modify it. Now, take all the reservations with lower priority that you `virtually` removed, and allocate them starting with the ones with the highest priority. Some will remain in the same path, some will be moved to a new path, and some, if they don't fit anymore, will be deleted.
6. if it does not fit, and there is an existing entry for this reservation, delete it.

> Hint: make sure `self.links_capacity` is updated properly in each of your steps. Sometimes, when you virtually remove reservations you will have to add the capacity to links.

Testing reservations with priorities

Test your final solution. First, start the topology and the controller.

How to debug my P4 code!

[p4-learning debugging docs]

146 lines (95 sloc) | 6.33 KB Raw Blame

Debugging and Troubleshooting

Monitoring Traffic

Sniffing traffic can be a very powerful tool when debugging your P4 program. Basic things like verifying if traffic crosses certain path, or if header fields look like expected can be easily achieved by just observing traffic. For that, there is a wide range of tools that can be used:

Pcap Files:

The `simple_switch` provides an option to save all the traffic that crosses its interfaces in a pcap file. To enable pcap logging when starting your switch, use the `--pcap=<output_dir>` command line option. For Example:

```
sudo simple_switch -i 0@<iface0> -i 1@<iface1> --pcap=<output_dir> <path to JSON file>
```

Pcap logging will create several files using the following naming: `<sw_name>-<intf_num>_<in|out>.pcap` .

P4 Utils Integration:

If you enable pcap in the `p4app.json` configuration file, switches will be started with the `--pcap` option and use as output dir `./pcap` .

Wireshark/Tshark:

Another option is to observe the traffic as it flows. For that you can use tools like `tshark` and its GUI version `wireshark` . Wireshark is already installed in the VM, you can find its executable in the desktop.

To capture traffic with tshark run:

```
sudo tshark -i <interface_name>
```

Tcpdump:

Similarly, and if you prefer, you can use `tcpdump` (also already installed in the VM).

To capture traffic with tcpdump run (shows link-layer information and does not resolve addresses):

```
sudo tcpdump -l -enn -i <interface_name>
```

Time to work on the RSVP exercise

We are here to help you (also with previous exercises)

Question of general interest	Unmute in Zoom and speak or Slack chat (#exercises)
Individual question	Raise hand in Zoom You will be assigned to a private “breakout room” with a TA Come back to the main room afterwards