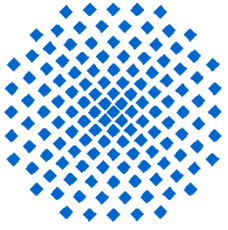


Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Lab-course / Fachpraktikum Computer Communication: Software-defined Networking

Summer Term 2020

Assignment 5

Network Programming

January 26th, 2021

Sukanya Bhowmik, David Hellmanns

Overview

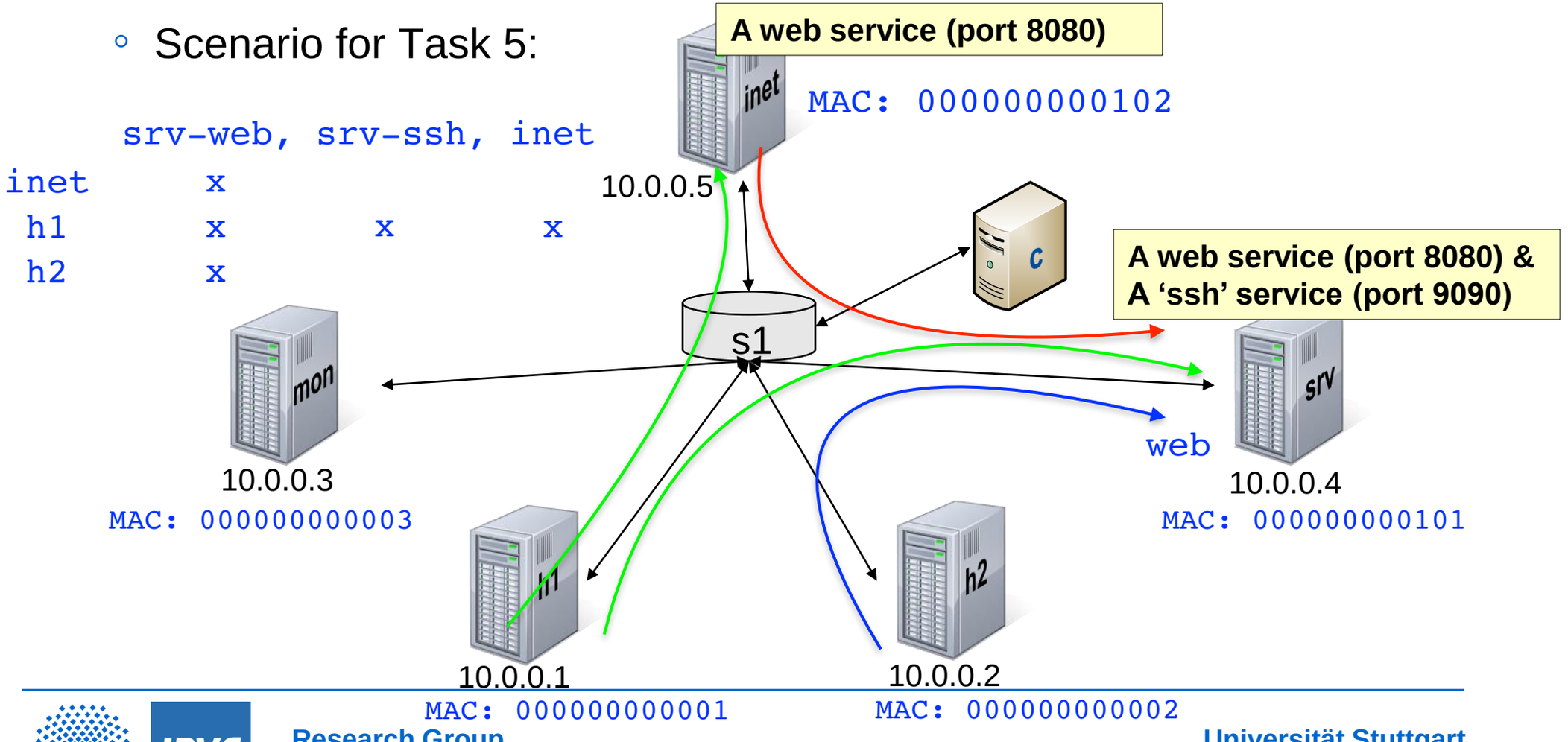
- **Task 5**
  - **Goals of this task**
  - 5.1 – Static Policies **[7 points]**
  - 5.2 – Dynamic Policies **[7 points]**
  - 5.3 – Policy Composition **[4 points]**
  - 5.4 – Comparison of Forwarding Policies **[2 points]**
- Deadline and Submission

Goals of this Task

Perform OpenFlow network programming on a higher level:

- Based on controller Pox, “push” flows via Pyretic
- Scenario for Task 5:

	srv-web,	srv-ssh,	inet
inet	x		
h1	x	x	x
h2	x		



Overview

- **Task 5**
  - Goals of this task
  - **5.1 – Static Policies**
  - 5.2 – Dynamic Policies
  - 5.3 – Policy Composition
  - 5.4 – Comparison of Forwarding Policies
- Deadline and Submission

Task 5.1 – Static Policies

Implement a static firewall as well as a simple monitor with Pyretic for the above-shown scenario, where …

FIREWALL

- … host `inet` can only access the web service on `srv`.
- … host `h1` has full access on all services, running on `srv` as well as on `inet`.
- … host `h2` can only access the web service on `srv` (no ssh on `srv` and no web on `inet`) .

MONITOR

- … `mon` prints every access to the ssh service on `srv` as well as all traffic from `h2`.

If needed, ARP-handling can be implemented by a flooding policy.

Task 5.1 – Static Policies

Implement your firewall / monitor solution within the provided files
~/ex5/simple_firewall.py, ~/ex5/simple_monitor.py

Test each module independently, start your pyretic module (e.g., firewall):

```
ex5$ pyretic.py -v low -m r0 simple_firewall
```

module dir included in `$PYTHONPATH`

`/opt/pyretic/pyretic.py` → `/opt/pyretic` is included in `$PATH`

module without suffix „`.py`“

Test your solution by bringing up a Mininet, starting web & ssh services on `inet` & `srv`, and a xterm for all nodes:

```
~$ cd ex5
ex5$ sudo ./mininet5.py
mininet> startservers
mininet> xterm h1 h2 srv inet mon
```

Task 5.1 – Static Policies

On `h1`, `h2`, and `inet` access web & ssh services through

`curl -sS -m 5 10.0.0.X:Y0Y0`,

On `mon` start monitoring with the packet capture tool

`tcpdump`

to print information about received packets.

Overview

- **Task 5**
  - Goals of this task
  - 5.1 – Static Policies
  - **5.2 – Dynamic Policies**
  - 5.3 – Policy Composition
  - 5.4 – Comparison of Forwarding Policies
- Deadline and Submission

Task 5.2 – Dynamic Policies

For this task, assume very high network load in the scenario (traffic to services at `inet` and `srv`), causing packets to be **indiscriminately dropped** at switch `s1`.

Implement a very crude Quality-of-Service module to prioritize traffic in the scenario by implementing a **dynamic** policy (using *query policy* and *callback handler* mechanism), which...

- ... counts packets, sent from/to a node
- ... dynamically blocks or forwards packets, based on the packet counter of the associated traffic
  - E.g.: Drop every 10th packet, sent from `h1`,
    drop every 5th packet, sent from `h2`

Task 5.2 – Dynamic Policies

Implement your solution within the provided file under

~/ex5/qos.py

Start your pyretic module:

```
ex5$ pyretic.py -v low -m r0 qos
```

Test your solution by bringing up a Mininet and start a xterm for nodes `h1`, `h2`, `inet`, and `srv` :

```
ex5$ sudo ./mininet5.py
mininet> xterm h1 h2 srv inet
```

Task 5.2 – Dynamic Policies

On `inet` and `srv`, run

`./udpreceiver 4000`,

receiving datagrams on port 4000

On `h1` and `h2`, run

`./udpsender 10.0.0.X 4000 300`,

sending 300 datagrams to node inet:4000 or node srv:4000.

Hints:

- *You don't have to simulate the excessive network load, just assume it's there.*
- *Check the MAC Learning example from the tutorial if you have problems with dynamic policies!*

Overview

- **Task 5**
  - Goals of this task
  - 5.1 – Static Policies
  - 5.2 – Dynamic Policies
  - **5.3 – Policy Composition**
  - 5.4 – Comparison of Forwarding Policies
- Deadline and Submission

Task 5.3 – Policy Composition

Compose your modules (simple_*, qos) to implement two different network policies:

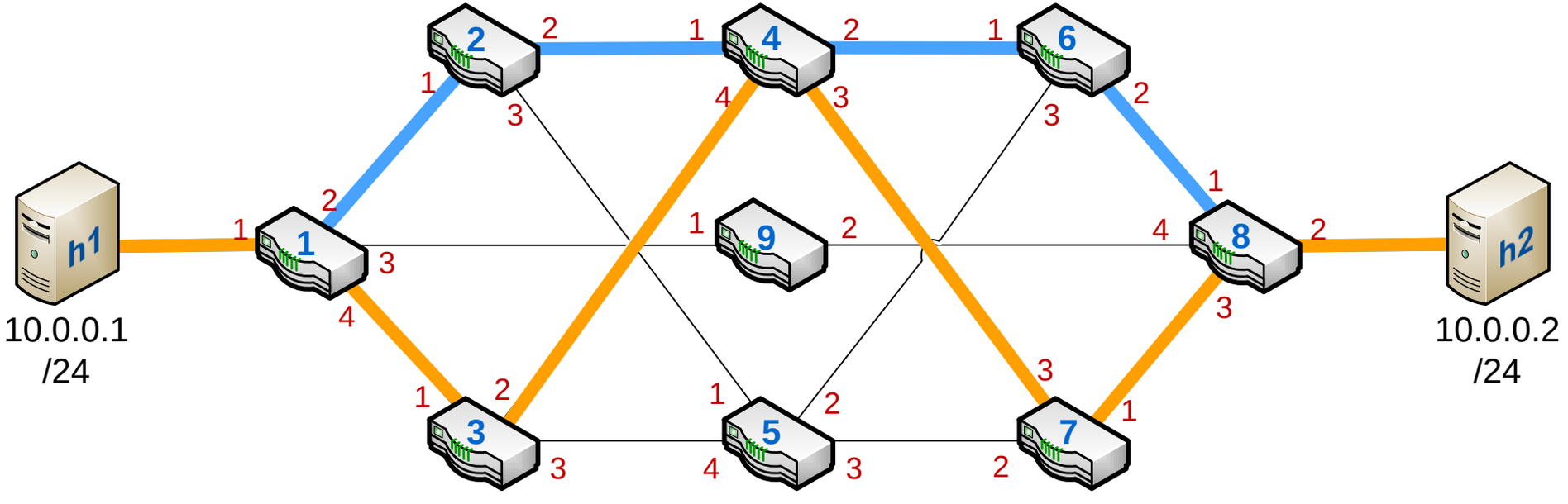
- A monitoring firewall, where only filtered traffic (traffic that remains after filtering) shall be monitored. Forwarding to end hosts is not to be influenced by monitoring. Describe, implement (→ File: ~/ex5/monitor_firewall.py) and verify possible policy compositions.
- A monitoring firewall with QoS capabilities. How can the QoS policy be meaningfully composed with the firewall and the monitoring policy? Describe, implement (→ File: ~/ex5/monitor_firewall_qos.py) and verify your composition(s).

Use the setup from Task 5.2 to test the qos part and use the curl-cmd (Slide 7) to test monitor and firewall.

Overview

- **Task 5**
  - Goals of this task
  - 5.1 – Static Policies
  - 5.2 – Dynamic Policies
  - 5.3 – Policy Composition
  - **5.4 – Comparison of Forwarding Policies**
- Deadline and Submission

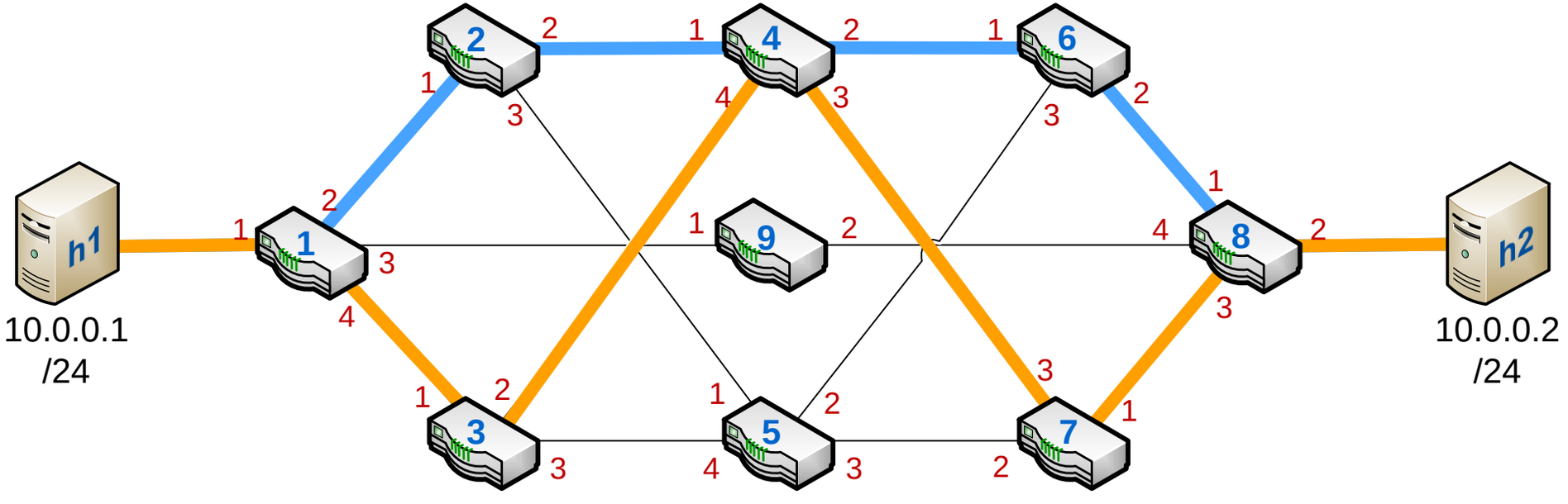
Task 5.4 – Comparison of Forwarding Policies



Implement forwarding for the blue and orange flows (unidirectional from h1 to h2) using separate static pyretic modules (blue.py, orange.py). Optimize for minimal number of policies.

- *Mind: topology similar but not equal to Task 3.1*
- Just **assume** the given topology. You don't have to implement it.

Task 5.4 – Comparison of Forwarding Policies



Analysis:

- Compare your pyretic module implementations (blue flow, orange flow) w.r.t. space complexity.
- Explain and try to generalize your conclusions.
  - Which aspect is Pyretic not able to fully abstract from?

Overview

- Task 5
- **Deadline and Submission**

Deadline and Submission

- When (submission deadline): February 9th 2021 at 08:00am
- How: Via ILIAS system
  - One submission per group
  1. One document (PDF)
     - Describing the commands you executed to solve the tasks
     - Showing the output
     - Explanation
  2. Archive of your solutions
- Be prepared to show a live demo to the supervisor during the next meeting