Universität Stuttgart

Institute of Parallel and
Distributed Systems (IPVS)

Universitätsstraße 38
D-70569 Stuttgart

Lab-course / Fachpraktikum

Computer Communication: Software-defined Networking

Winter Term 20/21

Assignment 1

Mininet, Floodlight REST API

November 17th 2020

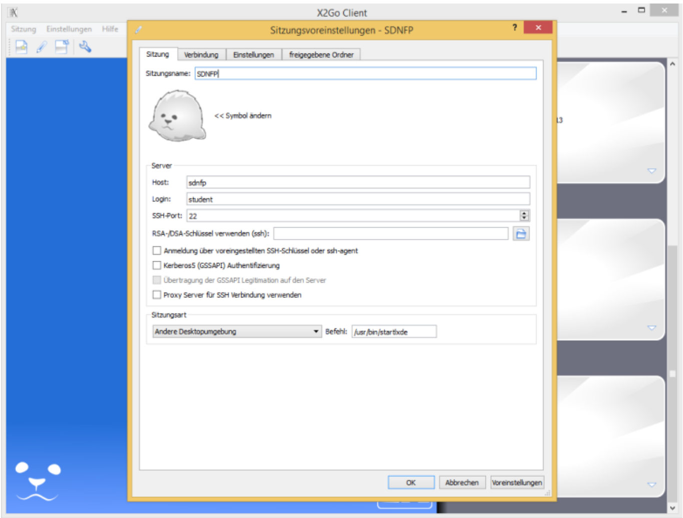
Sukanya Bhowmik, David Hellmanns

Overview

- **Setting up your working environment**
- Task 1
- Deadline and Submission

Setting up your Working Environment (1)

1. **Download & Install X2Go-Client**
   - http://wiki.x2go.org/doku.php/doc:installation:x2goclient
2. **Configure your client**
   - Host: sdnfp0**X**.infra.informatik.uni-stuttgart.de
     → **X=G+1** where G is your group no.
   - Login: student
   - SSH-Port: 22
   - Session: LXDE



Sitzungsname: SDNFP

Server

Host: sdnfp

Login: student

SSH-Port: 22

RSA-/DSA-Schlüssel verwenden (ssh):

☐ Anmeldung über voreingestellten SSH-Schlüssel oder ssh-agent

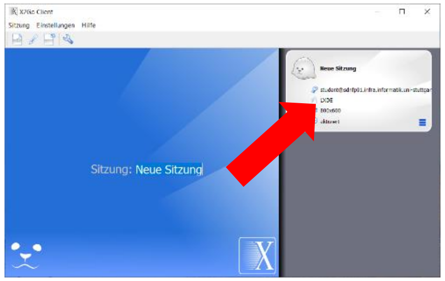
☐ Kerberos5 (GSSAPI) Authentifizierung

☐ Übertragung der GSSAPI Legitimation auf den Server

☐ Proxy Server für SSH Verbindung verwenden

Sitzungsart

Andere Desktopumgebung Befehl: /usr/bin/startlxde



Sitzung: Neue Sitzung

Neue Sitzung

student@sdnfp01.infra.informatik.uni-stuttgar

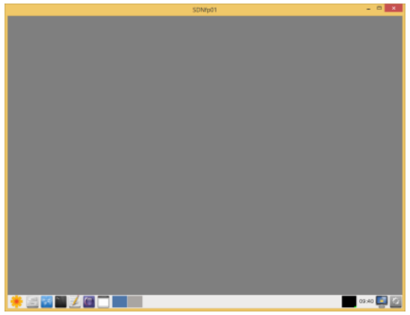
LXDE

800x600

aktiviert

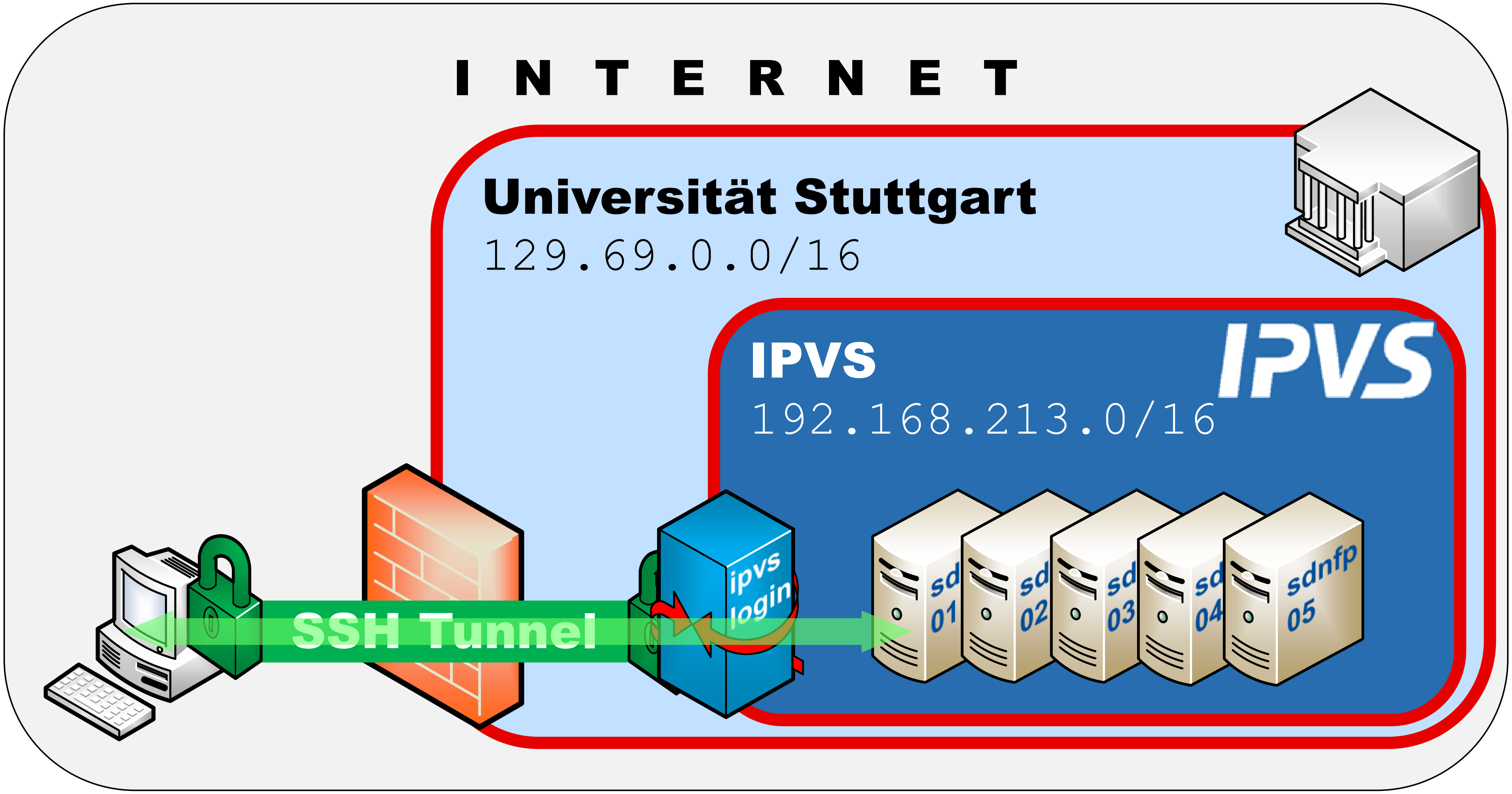
3. **Log into VM**
   - User: student (initial password provided via ILIAS)
   - **Change your password after first login!**
     → **shell:**

```
$ passwd
```



Setting up your Working Environment (2)

- **This only works from inside the IPVS network!**



Setting up your Working Environment (3)

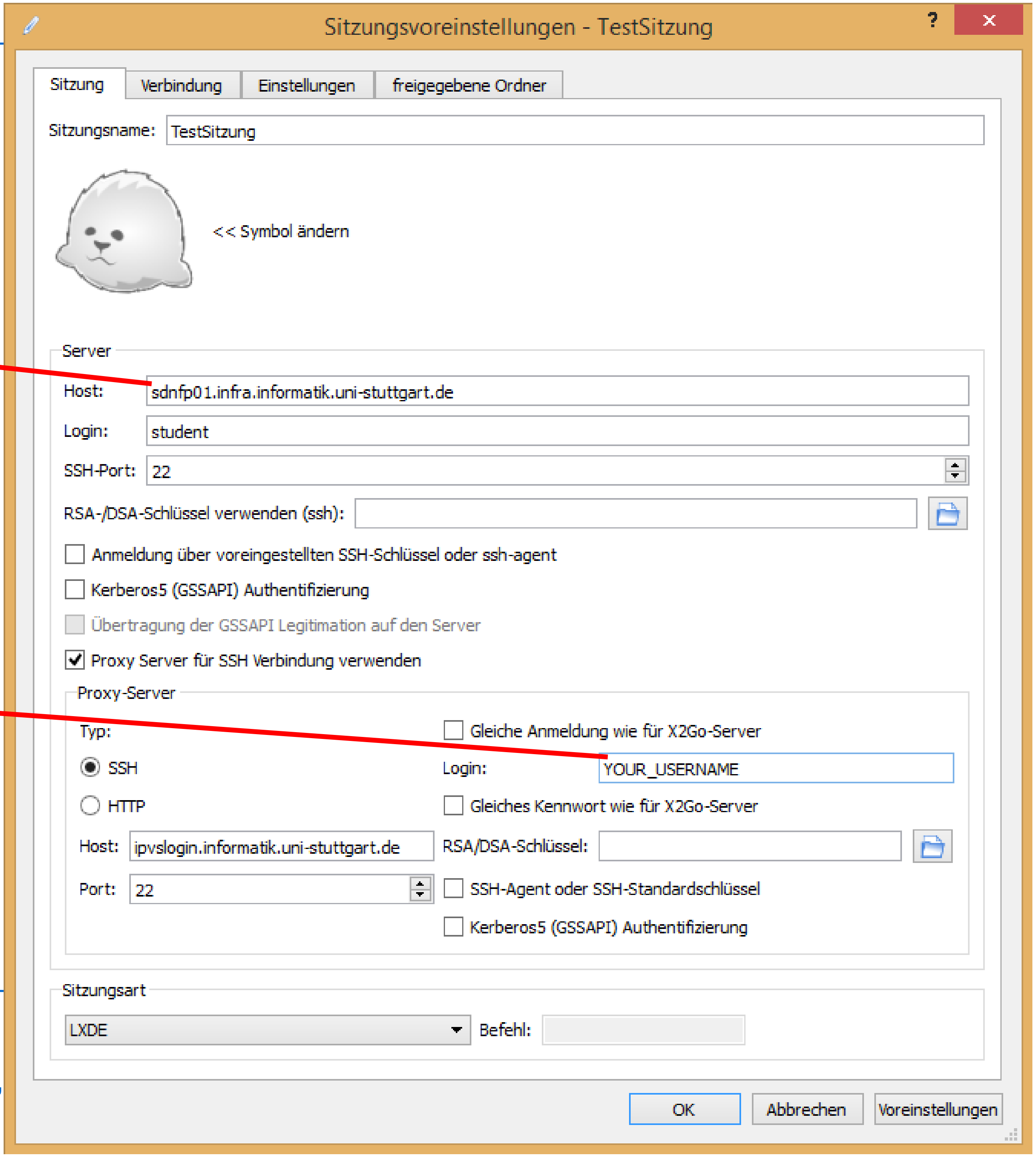
- **From outside computer science network :**
  - *either* connect over computer science VPN (ZDI, not RUS/TIK!)
    http://www.zdi.uni-stuttgart.de/vpn.html
  - *or* connect through SSH tunnel
    https://www.ipvs.uni-stuttgart.de/abteilungen/ifs/rechnerlabor/rechenbetrieb/NXaccess
    - X2Go: Enable “Use Proxy for SSH connection” in session preferences
      - Type: `SSH` Port: `22` Login: `[IPVS_USERNAME]`
      - Host: `ipvslogin.informatik.uni-stuttgart.de`
    - or e.g. under Linux: `ssh -L 9999:sdnfp0X:22 [user]@marvin`, then use `localhost:9999` as X2go host (cf. previous slide)
    - under Windows: use PuTTY (Connection → SSH → Tunnels
      - Source port: `9999` Destination: `sdnfp0X:22` "Local"

Setting up your Working Environment (4)

- **Preferred Method:**
  X2Go SSH Proxy

E.g., hostname "sdnfp02" for group 1

Your IPVS Account Name



Sitzungsvoreinstellungen - TestSitzung

?
×
Sitzung | Verbindung | Einstellungen | freigegebene Ordner
<< Symbol ändern
OK | Abbrechen | Voreinstellungen

Sitzungsname: TestSitzung

Server

Host: sdnfp01.infra.informatik.uni-stuttgart.de

Login: student

SSH-Port: 22

RSA-/DSA-Schlüssel verwenden (ssh):

□ Anmeldung über voreingestellten SSH-Schlüssel oder ssh-agent

□ Kerberos5 (GSSAPI) Authentifizierung

□ Übertragung der GSSAPI Legitimation auf den Server

☑ Proxy Server für SSH Verbindung verwenden

Proxy-Server

Typ:

● SSH

○ HTTP

Host: ipvslogin.informatik.uni-stuttgart.de

Port: 22

□ Gleiche Anmeldung wie für X2Go-Server

Login: YOUR_USERNAME

□ Gleiches Kennwort wie für X2Go-Server

RSA/DSA-Schlüssel:

□ SSH-Agent oder SSH-Standardschlüssel

□ Kerberos5 (GSSAPI) Authentifizierung

Sitzungsart

LXDE

Befehl:

Overview

- Setting up your working environment
- **Task 1**
  - Goals of this task
  - 1.1 – Remote Controller **[1 points]**
  - 1.2 – Simple Layer 1 VLANs **[3 points]**
  - 1.3 – Extended VLANs **[4 points]**
  - 1.4 – Mininet Python API and ARP Caches **[2 points]**
- Deadline and Submission

Goals of this Task

- Get to know Mininet
  - Use the Mininet console
  - Connect Mininet to the Floodlight controller
  - Use the Mininet Python API
- Get to know Floodlight’s RESTful web API
  - Install flow entries
  - Query the network state

Task 1.1 – Remote Controller (1)

- Fire up a terminal and start Mininet with an external controller:

  ```
  ~$ sudo mn --switch ovsk --controller remote,port=6653
  ```

- In the Mininet CLI:
  1. Send three pings from host 1 to host 2

     ```
     mininet> h1 ping -c3 h2
     ```

  2. Output the current flows over switch s1

     ```
     mininet> dpctl dump-flows
     ```

- Were the pings successful?
- Which flows are installed in the switch?
- Explain

Task 1.1 – Remote Controller (2)

- Now, in a new terminal, start the Floodlight controller:

```
~$ /opt/floodlight/floodlight-default.sh
```

- Wait until the controller has started up completely
(a few seconds until output settles down)

Task 1.1 – Remote Controller (3)

- When the controller has been started completely, go back to your first terminal with the Mininet CLI and repeat steps 1 and 2 from before:

```
mininet> h1 ping -c3 h2
mininet> dpctl dump-flows
```

- How has the output changed?
- What is the reason for this change?

- *Hint: make sure to run dpctl immediately after the ping is completed, otherwise you might miss something…*

Overview

- Setting up your working environment
- **Task 1**
  - Goals of this task
  - 1.1 – Remote Controller
  - **1.2 – Simple Layer 1 VLANs**
  - 1.3 – Extended VLANs
  - 1.4 – Mininet Python API and ARP Caches
- Deadline and Submission

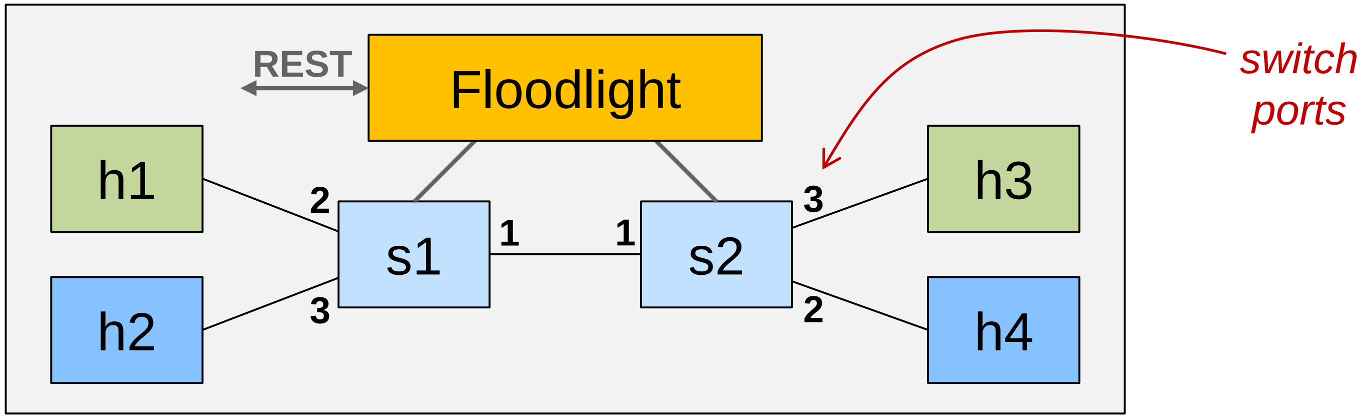
Task 1.2 – Simple Layer 1 VLANs (1)

- Start Floodlight controller with **noforwarding** configuration.

```
~$ /opt/floodlight/floodlight-noforwarding.sh
```

- When Floodlight is up and running, bring up a Mininet with the custom topology **task12topo**:

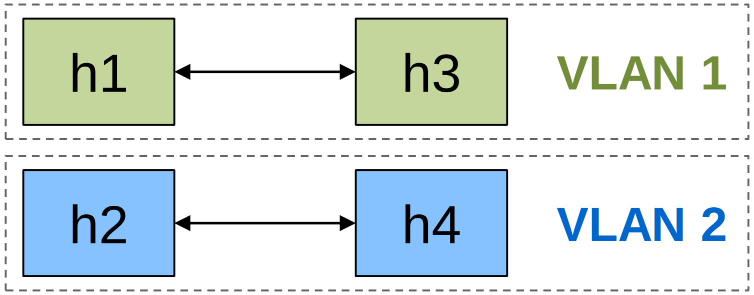
```
~$ sudo mn --mac --arp --switch ovsk
   --controller remote,port=6653
   --custom ex1/task12topo.py --topo task12topo
```

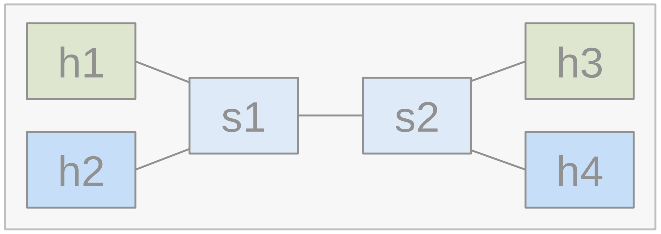


Mininet setup **task12topo** with desired VLANs

Task 1.2 – Simple Layer 1 VLANs (2)

- Set up two isolated VLANs between h1 and h3, and between h2 and h4, so that the hosts communicate as follows:





- Use the REST interface for the static flow entry pusher on http://localhost:8080/wm/staticentrypusher/json

- Do **not** match MAC or IP addresses (port-based forwarding)
  - Use VLAN IDs to distinguish source-destination pairs
  - Sanitize Ethernet frames on egress
    (Make sure that no VLAN ID tags are visible to the end hosts h1..4)

Task 1.2 – Simple Layer 1 VLANs (3)

- In the Mininet CLI, verify that these VLANs are mutually isolated by using the built-in pingall test:

```
mininet> pingall
```

Information:

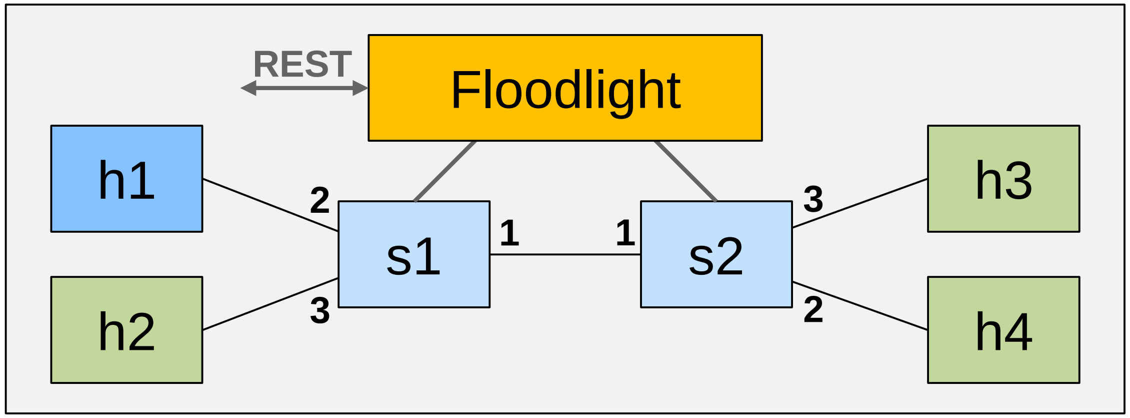
- Static Entry Pusher API Docs: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343518/Static+Entry+Pusher+API
- Strongly advised:
  - Please put all of your calls into a **single shell** (bash) **script: task12.sh**!
  - At the beginning of your script, **clear all static flow entries**!
  - *Hint: a flow entry can have* ***multiple actions*** *(e.g. modify header and output on port x)*

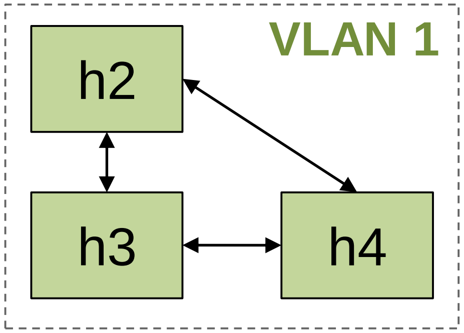
Overview

- Setting up your working environment
- **Task 1**
  - Goals of this task
  - 1.1 – Remote Controller
  - 1.2 – Simple Layer 1 VLANs
  - **1.3 – Extended VLANs**
  - 1.4 – Mininet Python API and ARP Caches
- Deadline and Submission

Task 1.3 – Extended VLANs

Create a bigger VLAN with the three hosts h2, h3, h4





Task 1.3 – Extended VLANs

Create a bigger VLAN with the three hosts h2, h3, h4

- Again, run Floodlight with noforwarding configuration:

```
~$ /opt/floodlight/floodlight-noforwarding.sh
```

- Use same topology as before, but **do not** use --arp option!

```
~$ sudo mn --mac --switch ovsk
   --controller remote,port=6653
   --custom ex1/task12topo.py --topo task12topo
```

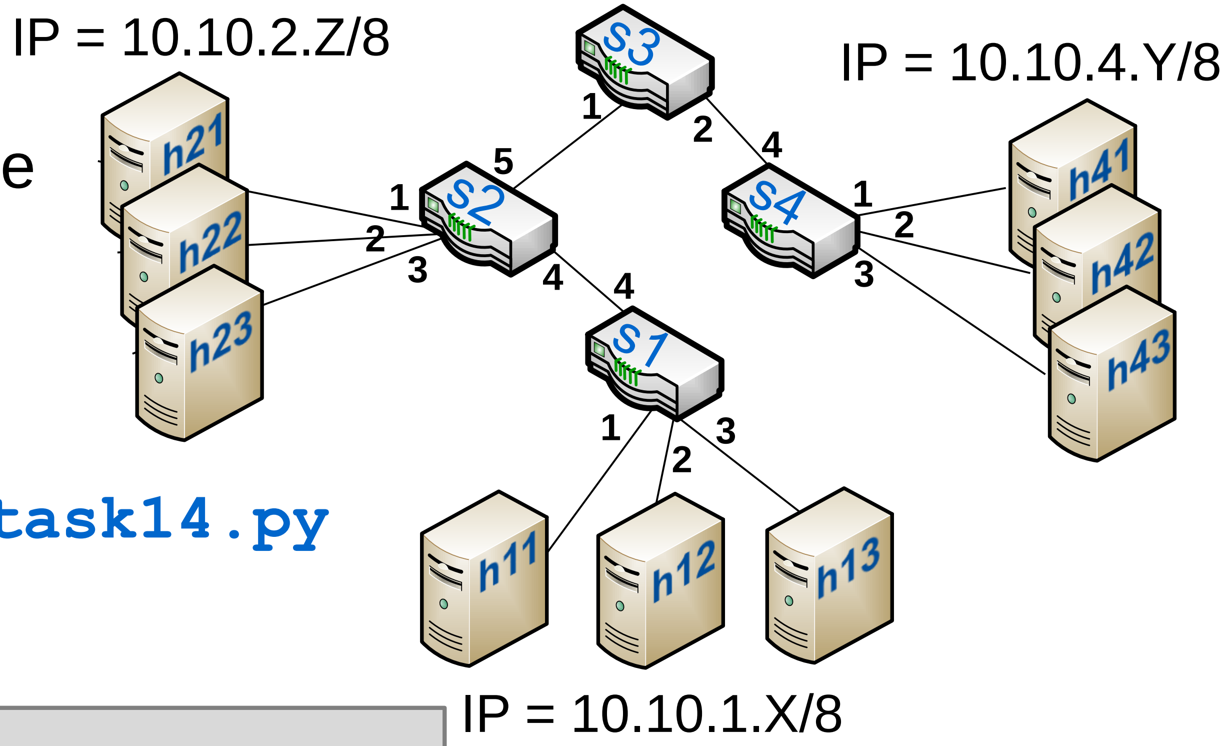
- Add suitable static forwarding table entries
  - Use static layer 2 forwarding rules
  - Pay attention that the ARP protocol works!
    - requires layer 2 broadcast, but keeping VLAN isolation
- Please put all of your calls into a **single shell** (bash) **script: task13.sh**!

Overview

- Setting up your working environment
- **Task 1**
  - Goals of this task
  - 1.1 – Remote Controller
  - 1.2 – Simple Layer 1 VLANs
  - 1.3 – Extended VLANs
  - **1.4 – Mininet Python API and ARP Caches**
- Deadline and Submission

Task 1.4 – Python API and ARP Caches (1)

Use Mininet's Python API to create the topology shown here and to statically fill hosts' ARP caches



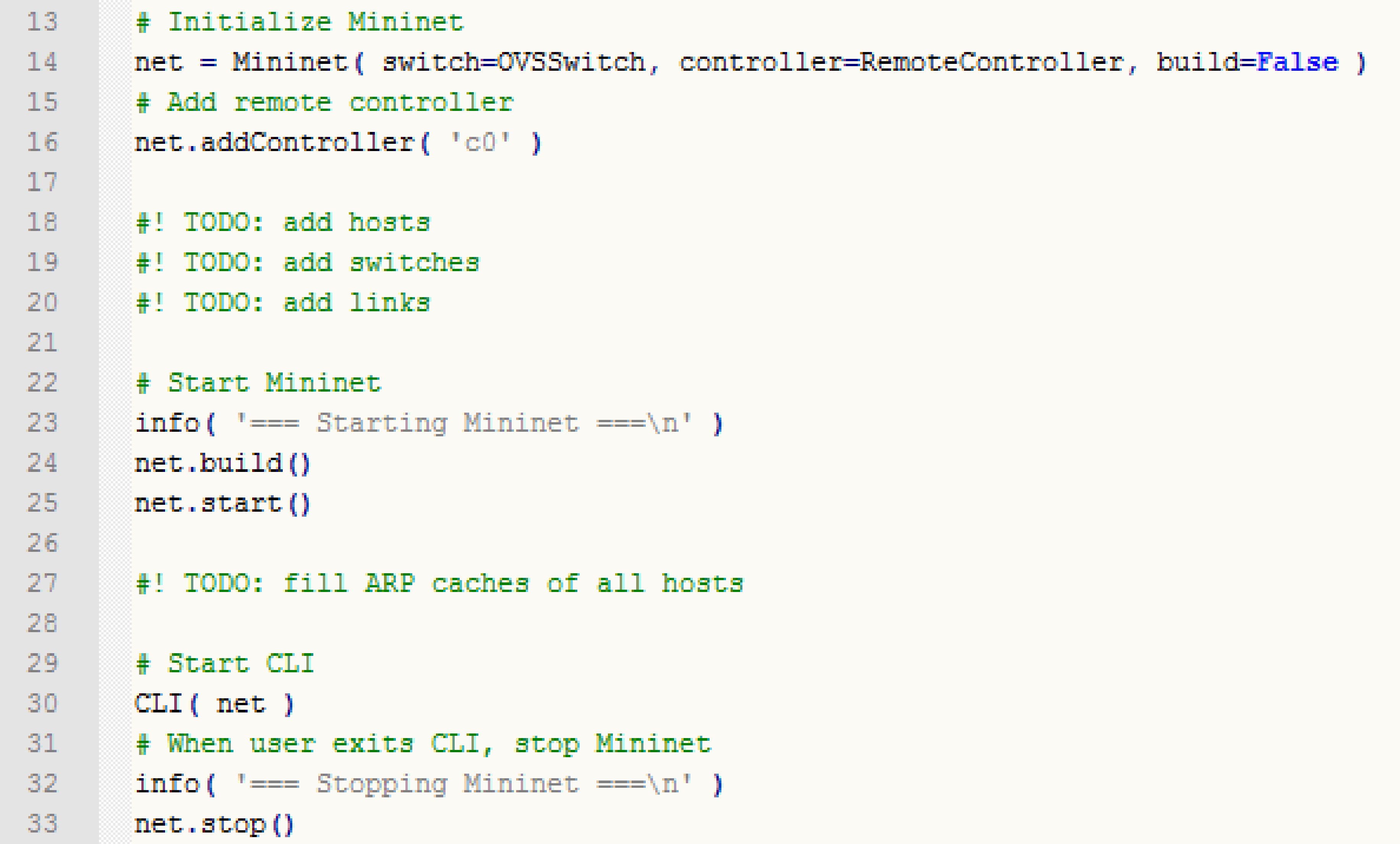
- Complete script in `~/ex1/task14.py`
- To test, run

```
~$ sudo ~/ex1/task14.py
```

Information:

- Mininet API Documentation: https://goo.gl/4QPTZe

Task 1.4 – Python API and ARP Caches (2)



```
# Initialize Mininet
net = Mininet( switch=OVSSwitch, controller=RemoteController, build=False )
# Add remote controller
net.addController( 'c0' )

#! TODO: add hosts
#! TODO: add switches
#! TODO: add links

# Start Mininet
info( '=== Starting Mininet ===\n' )
net.build()
net.start()

#! TODO: fill ARP caches of all hosts

# Start CLI
CLI( net )
# When user exits CLI, stop Mininet
info( '=== Stopping Mininet ===\n' )
net.stop()
```

`~/ex1/task14.py`

Task 1.4 – Python API and ARP Caches (3)

2. Start Mininet with your custom topology…

```
~$ sudo ~/ex1/task14.py
```

… and run the following commands in the CLI

```
mininet> net
mininet> py h11.MAC()
mininet> py h12.MAC()
mininet> py h13.MAC()
             ⋮
mininet> py h43.MAC()
mininet> h11 arp -n
mininet> h12 arp -n
mininet> h13 arp -n
             ⋮
mininet> h43 arp -n
```

(please document the outputs of these commands!)

Overview

- Setting up your working environment
- Task 1
- **Deadline and Submission**

Deadline and Submission

- When (submission deadline): December 1st, 2020 at 08:00am
- How: Via ILIAS system
  - One submission per group
- What: **One** Zip-file, containing:
  - **One PDF**-document
    → Naming convention: „<group_id>_<Name1><Name2><Name3>.zip”
    - Describing the commands you executed to solve the tasks
    - Showing the output
    - Brief explanation
  - Source files (extended or created scripts)
- Be prepared to show a live demo to the supervisor during the next meeting.