TECHNISCHE UNIVERSITÄT BERLIN
Fakultät IV – Elektrotechnik und Informatik
Fachgebiet INET
Prof. Dr. Stefan Schmid

1st Assignment Network Protocols and Architectures, WS 25/26

Question 1: $(10 + 10 + 10 + 10 = 40$ points) *Question 1*

We consider the 3-layer Clos topology $\mathcal{C}(k)$ of degree $k \geq 2$, k even, that was presented in the lecture. It consists of k pods $P_i = (T_i, B_i)$, $1 \leq i \leq k$. Each pod contains $k/2$ top switches $T_i = \{t_{i,1}, t_{i,2}, ..., t_{i,k/2}\}$ and $k/2$ bottom switches $B_i = \{b_{i,1}, b_{i,2}, ..., b_{i,k/2}\}$. Furthermore, there are $k/2$ groups of $k/2$ backbone switches $A_\ell = \{a_{\ell,1}, a_{\ell,2}, ..., a_{\ell,k/2}\}$, $1 \leq \ell \leq k/2$. Finally, there are $k^2/2$ sets of $k/2$ hosts each: $H_{i,j}$, $1 \leq i \leq k, 1 \leq j \leq k/2$. (Bidirectional) links are inserted as follows:

- In every pod P_i, $1 \leq i \leq k$, links are inserted such that T_i and B_i form the partitions of a bipartite graph.
- Every bottom switch $b_{i,j}$, $1 \leq i \leq k$, $1 \leq j \leq k/2$, is connected to each of the hosts in $H_{i,j}$.
- In each backbone group ℓ, $1 \leq \ell \leq k/2$, and for every j, $1 \leq j \leq k/2$: the j-th backbone switch $a_{\ell,j}$ is connected to the j-th top router $t_{i,j}$ in each pod i, $1 \leq i \leq k$.

(a) Draw the $\mathcal{C}(6)$.

(b) Is the topology displayed in Figure 1 a 3-layer Clos topology? Explain your answer.

(c) How many switches are required to construct the $\mathcal{C}(k)$? What is the number of hosts? How many links are needed in total?

(d) Assume that we want to route a packet from some bottom switch $u \in \bigcup_{i=1}^{k} B_i$ to another bottom switch $v \in \bigcup_{i=1}^{k} B_i, v \neq u$. Show that, even if any set of $(k/2) - 1$ links are failed, there still exists a path from u to v that is a shortest-path in the original (failure-free) topology.

Question 2: $(20 + 20 + 20 = 60$ points) *Mininet*

For this tutorial you need to install Mininet. It is recommended to install Mininet in a virtualized environment. To (hopefully) simplify this process of provisioning a VM, you are provided with a **Vagrantfile**. In order to use this setup, you will need to install Vagrant and VirtualBox. The provided Vagrant file has been tested with Vagrant version 2.3.4 and VirtualBox version 7.0 on Ubuntu 22.04, Ubuntu 24.04, Debian 12, and Windows 11. We will use Vagrantfiles again later on in the course, so it is advisable to set it up. The provided Vagrantfile will create a Ubuntu VM, which comes with everything installed you'll need for this exercise. It also automatically creates a shared folder to your host system, which is located in the VM under `/vagrant`. In order to start or create the VM run the following command in the folder, which contains the Vagrantfile:

```
vagrant up
```

After successful provisioning you can access the VM using:

```
vagrant ssh
```

Once connected to the container, you can test if everything works and start to answer the questions below:

```
sudo mn --topo single,3 --controller=none --switch ovsbr
```

(a) Show the mac addresses of each host and all the interfaces of the switch. Copy and paste the output into your solution sheet and highlight the MAC address.

```
# Basic command syntax
<node> <command>
# Example
h1 ifconfig
```

(b) Show the forwarding address table on s1.

```
sh ovs-appctl fdb/show s1
```

Note: In case your output is empty try to run the command `pingall` first. Copy and paste the output into your solution sheet **and** explain the meaning of every column.

(c) Build a topology as shown in the figure below, using Mininet's python API. See the example for a reference implementation on how to add switches and hosts to the network. Once you create your topology file, you could run it with the command (example for `topo-2sw-2host.py` that defines mytopo):

```
sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo --test
    pingall
```

Now dump your topology from mininet, by running the follow two commands:

```
mininet> dump
mininet> links
```

Copy the output of both commands and paste it on the topology visualization tool. Attach the output graph visualization to your submission. *Note:* Sometimes this tool cannot produce the output. In that case try another browser or try again later. If this does not help, provide your python code and the outputs of the `dump` and `links` (just paste it into your solution sheet).

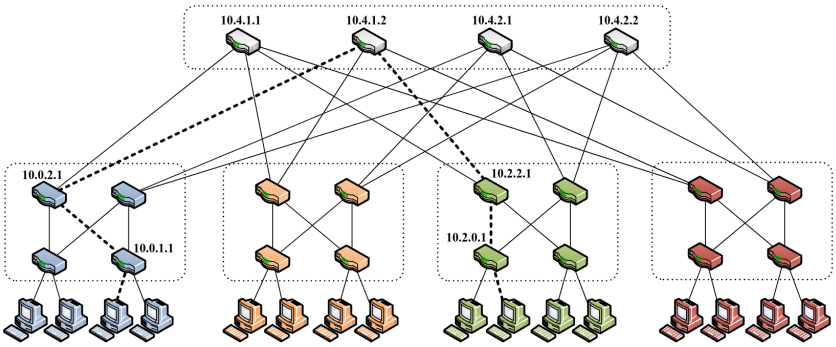

Figure 1: Fat Tree Topology, a common topology for data centers.

Due Date: Sunday, 26.10.2025, 23:59 pm

- **Upload your solutions as a PDF (no MS Office or OpenOffice files)** via ISIS: `https://isis.tu-berlin.de/course/view.php?id=44909`
- Submit in groups of 4 and put the names and Student ID numbers (Matrikelnummer) of **all** your group members on your solution!
- Only one student per group needs to upload the solution.