**ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ**

**Διαχείριση Edge και Cloud δικτύων βασισμένων στο λογισμικό**

***Task 1: Setting Up Mininet with a Remote Controller (20 Marks)***

**Instructions:**

1. Install Mininet and an SDN controller of your choice (Ryu, POX, ONOS, or OpenDaylight).
   * Mininet installation: sudo apt-get install mininet
   * Ryu installation: pip install ryu
   * POX installation: Clone repository git clone https://github.com/noxrepo/pox.git
   * ONOS installation: Follow [ONOS setup guide](https://wiki.onosproject.org/display/ONOS/Quick+Start)
   * OpenDaylight installation: Follow [ODL setup guide](https://docs.opendaylight.org/en/latest/)
2. Start a simple Mininet topology (e.g., a tree topology):

sudo mn --topo=tree,depth=2 --controller=remote,ip=<CONTROLLER\_IP> --switch ovs

1. Verify connectivity using pingall.

**Deliverables:**

* Screenshot of the running Mininet topology.
* Output of pingall.

***Task 2: Implementing a Learning Switch in Ryu (20 Marks)***

**Instructions:**

1. Write a Python script to implement a learning switch using the **Ryu controller**.
2. The switch should dynamically learn MAC addresses and create forwarding rules.
3. Run the controller and test connectivity in Mininet.

**Deliverables:**

* Python script of the learning switch.
* Explanation of the logic behind learning switch behavior.
* Screenshot of Mininet traffic flow.

***Task 3: Traffic Engineering Using SDN (20 Marks)***

**Instructions:**

1. Implement custom OpenFlow rules to prioritize certain types of traffic:
   * Prioritize HTTP traffic by directing it to a specific port.
   * Direct ICMP (Ping) traffic to a different path.
   * Flood other type of traffic.
2. Use iperf and ping to measure performance under different configurations.
3. Analyze network latency and bandwidth before and after applying rules.

**Deliverables:**

* OpenFlow rules applied.
* Performance comparison with and without traffic engineering.
* Screenshots of test results.

***Task 4: OpenFlow Message Analysis with Wireshark (10 Marks)***

**Instructions:**

1. Install Wireshark: sudo apt-get install wireshark
2. Start Mininet and connect it to an SDN controller:
3. sudo mn --topo=tree,depth=2 --controller=remote,ip=127.0.0.1 --switch ovs
4. Start Wireshark and capture packets on the loopback interface (lo), filtering by OpenFlow messages:
   * Use the display filter: of (for OpenFlow messages)
5. Perform actions in Mininet such as pingall to generate OpenFlow traffic.
6. Capture and analyze the messages exchanged between the switch and the controller.
7. Identify key OpenFlow messages (e.g., Packet-In, Flow-Mod, Packet-Out).

**Deliverables:**

* Screenshots of OpenFlow message captures.
* Explanation of the observed OpenFlow message types.

***Task 5: Implementing Firewall Rules in POX (10 Marks)***

**Instructions:**

1. Use the **POX controller** to implement a simple firewall that blocks specific types of traffic.
2. Write a Python script (pox\_firewall.py) that:
   * Drops all TCP packets from a specific host.
   * Allows ICMP and UDP packets to pass.
3. Run the POX controller and apply the firewall rules.
4. Test connectivity using ping and iperf.

**Deliverables:**

* POX firewall Python script.
* Explanation of firewall behavior and observed traffic filtering.
* Screenshots (proof) of test results.

***Task 6: Implementing Failover Traffic Rerouting in OpenDaylight (20 Marks)***

**Instructions:**

**Step 1: Set Up OpenDaylight and Mininet**

1. **Start OpenDaylight (ODL) and enable OpenFlow features:**

./karaf

Inside OpenDaylight, install the required features:

feature:install odl-l2switch-switch odl-openflowplugin-flow-services odl-dlux-all

1. **Start Mininet with a Redundant Path Topology:**

sudo mn --topo=triangle --controller=remote,ip=<ODL\_IP>,port=6633 --switch ovs

* + This creates a **triangle topology** where there are two paths between any two hosts.

1. **Verify connection with OpenDaylight:**

sudo ovs-vsctl show

**Step 2: Set Up Primary and Backup Path Rules**

1. **Install the primary forwarding rule:**

sudo ovs-ofctl add-flow s1 "priority=10,ip,nw\_dst=10.0.0.2,actions=output:2"

* + This routes traffic to **Host 2** via **Port 2** on Switch 1.

1. **Install a backup rule with lower priority:**

sudo ovs-ofctl add-flow s1 "priority=5,ip,nw\_dst=10.0.0.2,actions=output:3"

* + If the primary path fails, traffic automatically switches to **Port 3**.

**Step 3: Test Failover Handling**

1. **Start a continuous ping:**

h1 ping 10.0.0.2

1. **Disable the primary link to simulate failure:**

sudo ovs-ofctl mod-port s1 2 down

* + This simulates a link failure.

1. **Observe the traffic switch to the backup path.**
   * The ping should recover after a short delay.
2. **Re-enable the primary link and watch the traffic return:**

sudo ovs-ofctl mod-port s1 2 up

**Deliverables:**

* Screenshots of OpenDaylight topology.
* Output showing pings before and after link failure.
* Explanation of failover behavior.