Communication System Design Projects

KUNGLIGA TEKNISKA HÖGSKOLAN

PROFESSOR: DEJAN KOSTIC

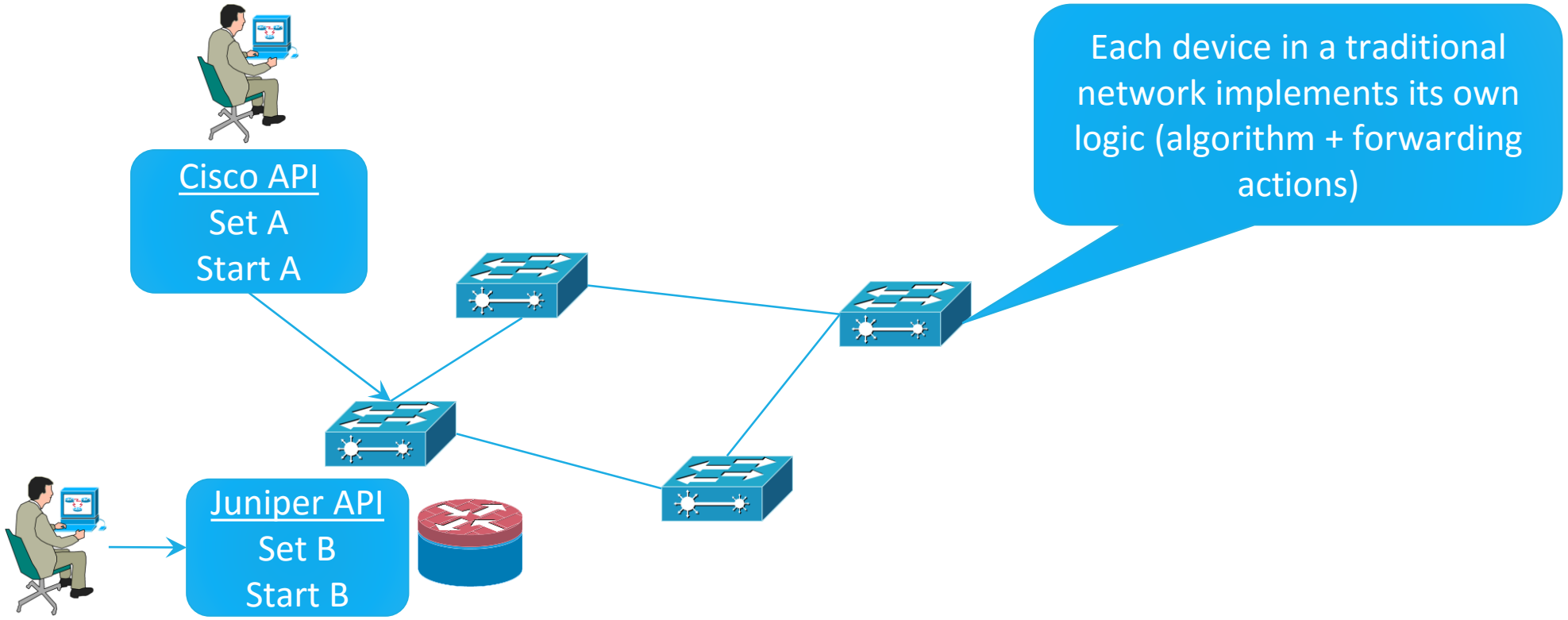
TEACHING ASSISTANT: GEORGIOS KATSIKAS

Traditional Vs. Modern Network Management

What is Network Management (NM)?

- A set of actions performed by network administrators in order to maintain the wellness of the network.
- Common networking problems:
  - Hardware:
    - e.g. Malfunctioning device/cable
  - or mostly Software:
    - e.g. Interface/service/daemon is down
- Common NM actions:
  - Hardware:
    - e.g. Replace malfunctioning device/cable
  - Software:
    - e.g. Configure and/or restart interface/service/daemon

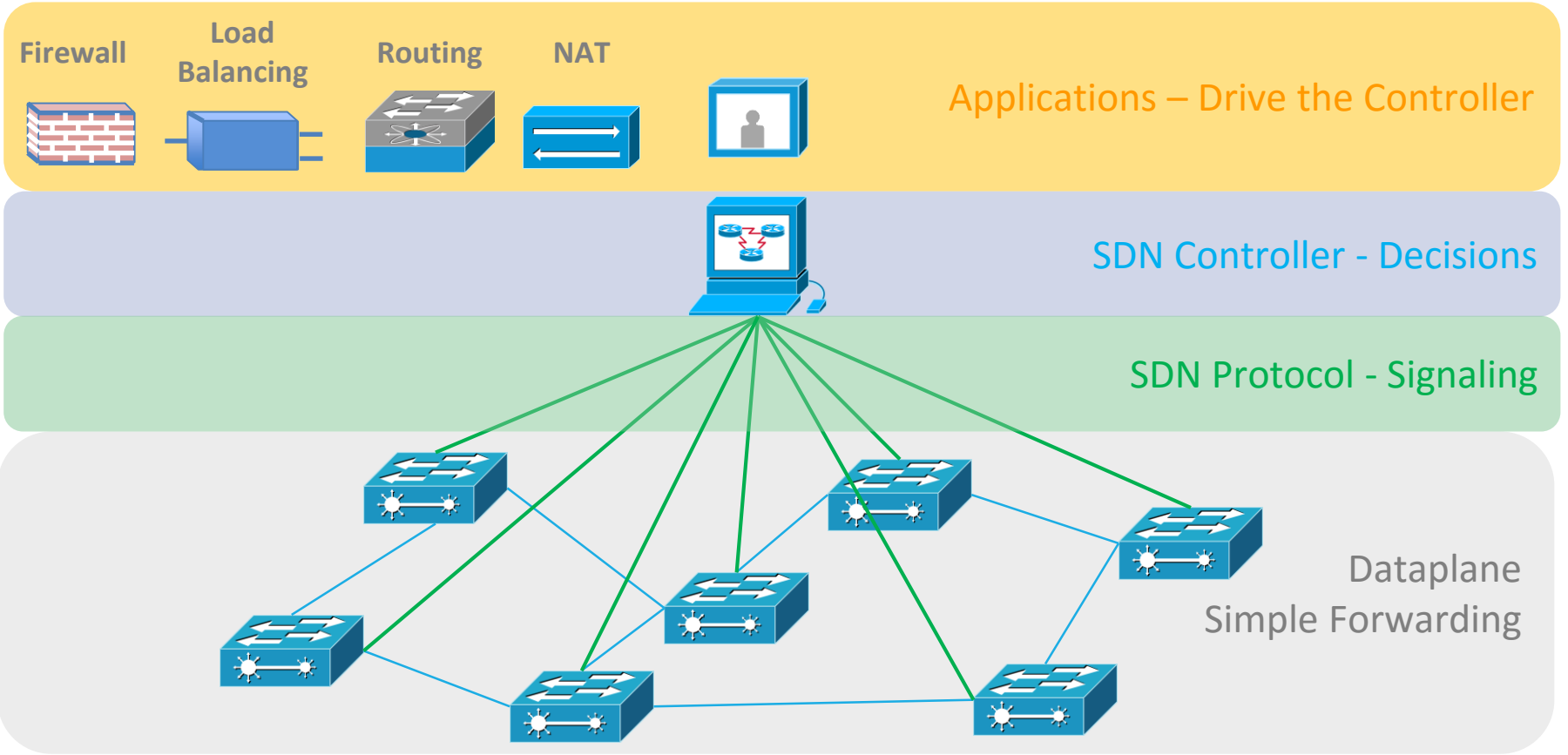
Traditional NM Approach



Modern NM Approach

Software Defined Networking

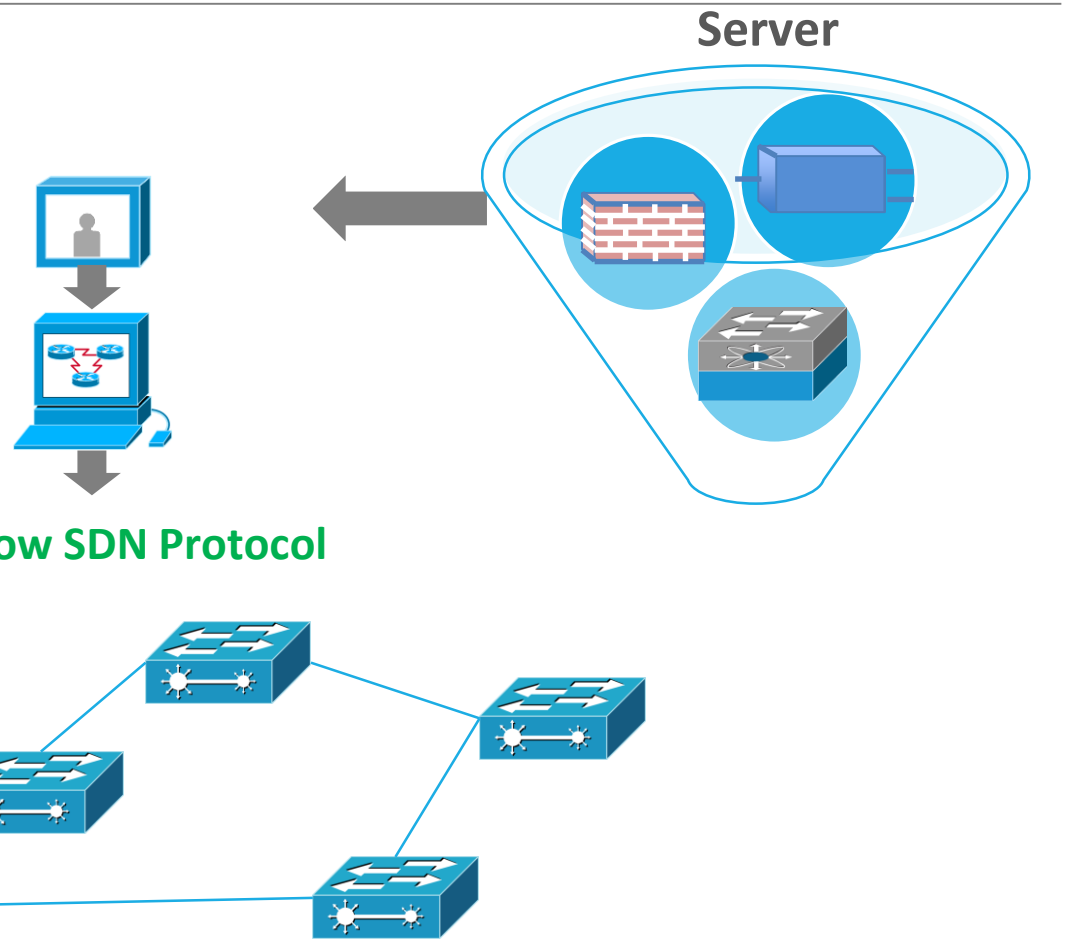
- Keep the SW simple.
- Separate control from dataplane.
- Controller and SW communicate over a protocol (e.g., OpenFlow [1]).
- Controller takes the decisions.
- Switches install the rules dictated by the controller.



Modern NM Approach

Network Functions Virtualization

- Routing, FW, LB, NAT are now VMs running in the cloud.
- Easy instantiation, management and migration.
- SDN + NFV form the networks of tomorrow.



SDN & NFV
Research & Industry

SDN Frameworks

- Research
  - Everything started with OpenFlow (OF) [1] in 2008
    - NOX was one of the first SDN controllers
    - Then POX, Beacon, Floodlight, Ryu
  - OpenFlow latest version is 1.5 (started from 1.0)
- Industry
  - OpenDaylight (ODL) as a successor of Floodlight
- Not all the controllers support all OF versions
- In this course we will use
  - **Ryu[4]:** (a) Supports up to OFv1.4, (b) modular, and (c) python-based.
  - **ODL[3]:** (a) Supports up to OFv1.3, (b) modular (Apache OSGi), and (c) java-based.

NFV Frameworks

- The idea of softwarizing the network elements began before 2000!
  - The hardware was not mature enough to support high-speed forwarding
  - Multi-core systems and datacenters raised the interest towards virtualizing Network Functions (NFs) on commodity hardware.
  - In combination with SDN it allows **flexible (and much cheaper) management** of network traffic.

- One of the most popular research NFV platforms is **Click[5]**:
  - Models the simplest packet operations (elements)
  - Enables a modular composition of these operations to build NFs
    - E.g., a Router is a "sequence" of elements such as:

    ... -> CheckIPHeader-> GetIPAddress -> IPLookup -> DecIPTTL -> ...

Project #3 - MODL

Distributed Monitoring Event Aggregation & Management for SDN.

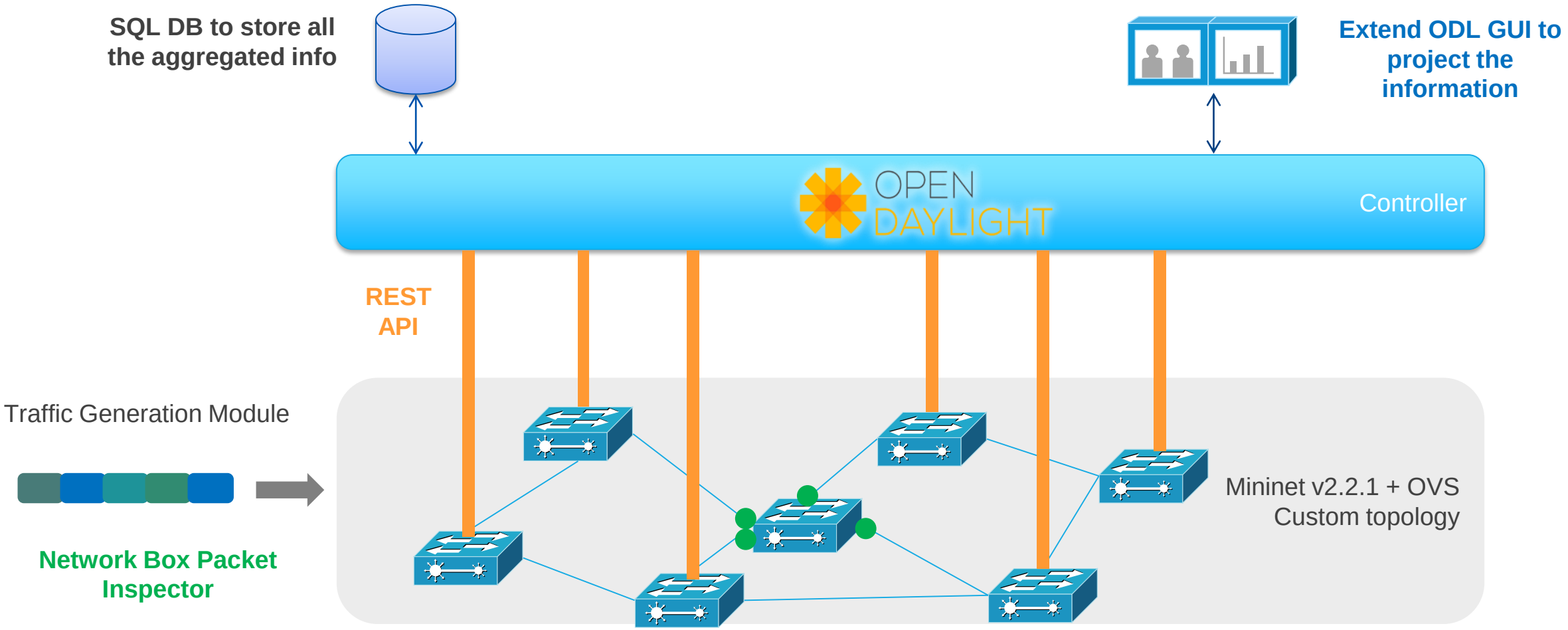
Distributed Monitoring Event Aggregation & Management for SDN

- **Problem**
  - Current SDN environments constrict their monitoring capabilities towards gathering packet and byte counters from the dataplane as well as several QoS metrics (OF 1.3).
- **Motivation**
  - To obtain the real network view and plan network management we need more precise information that reveals the state of each device in the network.
- **Approach**
  - Install distributed, lightweight monitoring modules across all the devices of an SDN topology and gather statistics for:
    - bandwidth, dropped packets, RTTs (per switch-to-controller),
    - overused/underused links, PL indicators,
    - expired TTLs, TCP retransmissions,
    - CPU/Memory/cache/NIC and per port usage statistics for switches.

Distributed Monitoring Event Aggregation & Management for SDN

- **Tasks & Tools**
  - Implement the distributed monitoring in C++ and make it controller agnostic using REST API. Focus on the performance of the ecosystem (light and fast).
  - OpenDaylight will be favored as the basic controller.
  - Develop side tools on the controller side (Extending ODL GUI) to effectively project the information.
  - Process the monitored data and generate meaningful and user friendly notifications for the network operator. Enable graphical queries of several parameters from the DB.
- **Required skills**
  - C++, Java, Advanced Networking, Linux, Web Services, Web GUI, SQL, mininet network emulator [2], OVS, Wireshark, OpenDaylight, SDN principles.

Distributed Monitoring Event Aggregation & Management for SDN



Distributed Monitoring Event Aggregation & Management for SDN

- **Specific details**
  - OpenDaylight Lithium [3] will be your SDN controller.
  - Mininet v2.2.1 with OVS 2.4 will emulate the network devices.
  - Restful API (WS) will be used to push info from switches to the controller.
  - Linux /proc, networking, process and HW management (i.e. Perf) will be used to gather the statistics.
  - A packet generation module will stress the network under different conditions to gather information from all the devices for an extended period. Standard mininet tools (iperf, netcat, ping) can be used.
  - A packet inspection module should capture the packets of any given network box (from every interface), correlate them in pairs (incoming-outgoing packets) and highlight the way they are modified by the box. Appropriate results will be stored to database tables as well.
  - SQL (MySQL) will be used to store the aggregated information (controller side) .
  - A post-processing module will turn the raw data into meaningful information and generate events.
  - ODL web-based GUI will be enhanced to project the processed info. A page should host the events/alerts that the network administrator should take into account (e.g. congested devices/links, etc.) and another one will host the monitoring statistics (e.g. over time). A 3rd page will be used to query the DB in a high level way.

Distributed Monitoring Event Aggregation & Management for SDN

- **A successfull outcome:**
  - Start your topology and SDN controller.
  - Start your monitoring module, setup connections with all the monitoring points/modules of your network.
  - Acquire live statistics from all the monitoring points, store to database and project to the Web based API (operator's panel).
    - The panel must visualize the topology (nodes + links) and the parameters you collected.
    - A history must be kept for each parameter to show its evolution over time.
  - Inject various traffic patterns from different domains. The impact must be captured and projected effectively in the operator's panel.
  - Cause some faults on purpose to prove that your monitoring module works. Your panel must blink!
  - After you have the database populated with several events/errors, show how a network operator can query the appropriate tab of the panel to get information about the network.

Project #4 – NFV ServiceChains

Flexible Deployment, Management and Monitoring of NFs to realize service chains

Flexible Deployment, Management and Monitoring of NFs to realize service chains

- **Problem**
  - Current NFV environments lack of a framework that enables easy deployment and chaining of lightweight NFs to provide advanced services to customers.
- **Motivation**
  - An automated framework is required to offer these services by bridging the SDN controller with a lightweight NFV platform, driven by high-level policies.

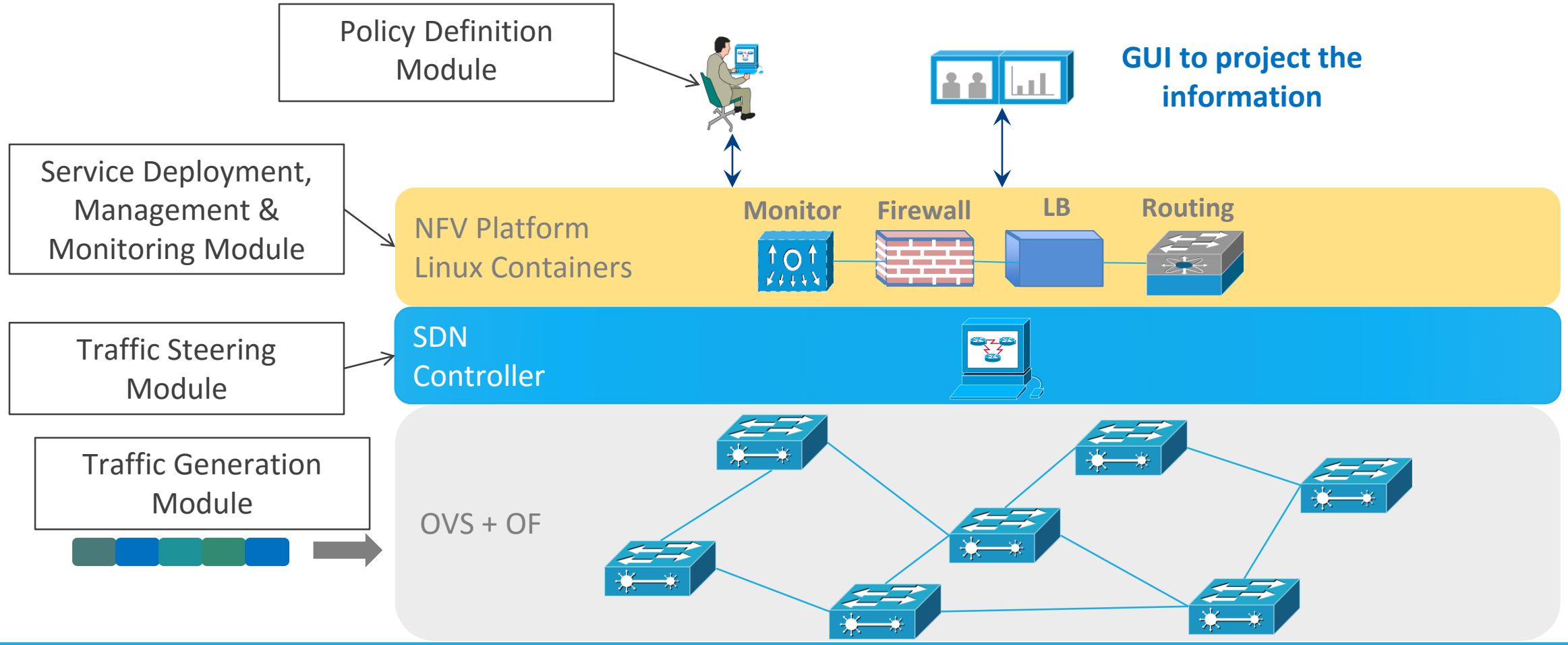
Flexible Deployment, Management and Monitoring of NFs to realize service chains

- **Approach**
  - For a given client-server topology, we need to provide DNS, VoIP, Web, FTP, VLAN, etc. services.
  - Each service is realized using a chain of NFs (statically-defined policies).
  - We need a framework to deploy, manage and measure the performance of these service chains.
    - The services can be implemented as small modules that run on the clients and the servers.
    - The following NFs must be implemented: VLAN-enabled gateway router, Firewall, Load Balancer, Proxy.
    - SDN controller will be used to dynamically steer the traffic across virtualized NFs.
    - A monitoring module will be used to provide CPU and packet latency measurements of the chains.
  - The provided services should meet some criteria such as isolation, balanced load, security, etc.
  - A GUI should be build to project the topology and the active links together with the monitored info.

Flexible Deployment, Management and Monitoring of NFs to realize service chains

- **Tasks & Tools**
  - Design the core network that will interconnect 3 zones (2 client zones, 1 server zone).
  - Implement the framework and services in Python.
  - Implement the NFs using the Click language (v2.1). The NFs must run in Linux containers (Docker [7]).
  - Ryu/ODL will be your SDN controller, OF v.1.3 your SDN protocol and OVS v2.4 will be the dataplane. A GUI is required to project the relevant info.
  - Linux Perf will be used to monitor CPU and Click to monitor latency.
- **Required skills**
  - Python, Click, Advanced Networking, Linux, containers, OVS, Web Services, Web GUI, SDN principles.

Flexible Deployment, Management and Monitoring of NFs to realize service chains



Flexible Deployment, Management and Monitoring of NFs to realize service chains

- **A successfull outcome:**
  - Start your topology and SDN controller.
    - The controller must expose an Web based API to show the topology and enable the easy creation of policies.
  - Based on the given policies, start your NFV module that will deploy the required NFs.
  - The SDN controller must also be aware of these policies in order to setup the appropriate rules in the dataplane and steer the traffic appropriately.
  - Start traffic that targets the deployed services. As long as there is traffic passing through your NFs:
    - A monitoring module must be capturing the requested metrics. These metrics must be visualized in the Web GUI.
    - The traffic paths must be visualized.
  - Change policies and traffic patterns accordingly to show that your tool is able to tear down and boot NFs on demand. Show how these changes affect also the traffic paths (the SDN controller might setup new paths).

General Info

- **Facilities**
  - The course provides VM(s) for each team. The VM(s) will be equipped with the majority of the tools you need to run the project.
  - The VMs will be connected to the Internet to facilitate the installation of any other tools and let you share the project's source code via BitBucket.
- **Team Rules**
  - 5-6 persons are required to accomplish the projects.
  - You have to design the activities of the project, fit them into a timeplan and provide personal reports per week, regarding the progress of the tasks assigned to each person **individually**.
  - Each project gets a grade at the end, based on the team work, but your grades will be individual based on your contributions.

References

References

1) OpenFlow: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow/
2) Mininet Network Emulator: http://mininet.org/
3) OpenDaylight: http://www.opendaylight.org/
4) Ryu: http://osrg.github.io/ryu/
5) Click: http://www.read.cs.ucla.edu/click/
6) OpenVSwitch: http://openvswitch.org/
7) Docker: https://www.docker.com/


Q&A