Communication System Design Projects

KUNGLIGA TEKNISKA HÖGSKOLAN

PROFESSOR: DEJAN KOSTIC

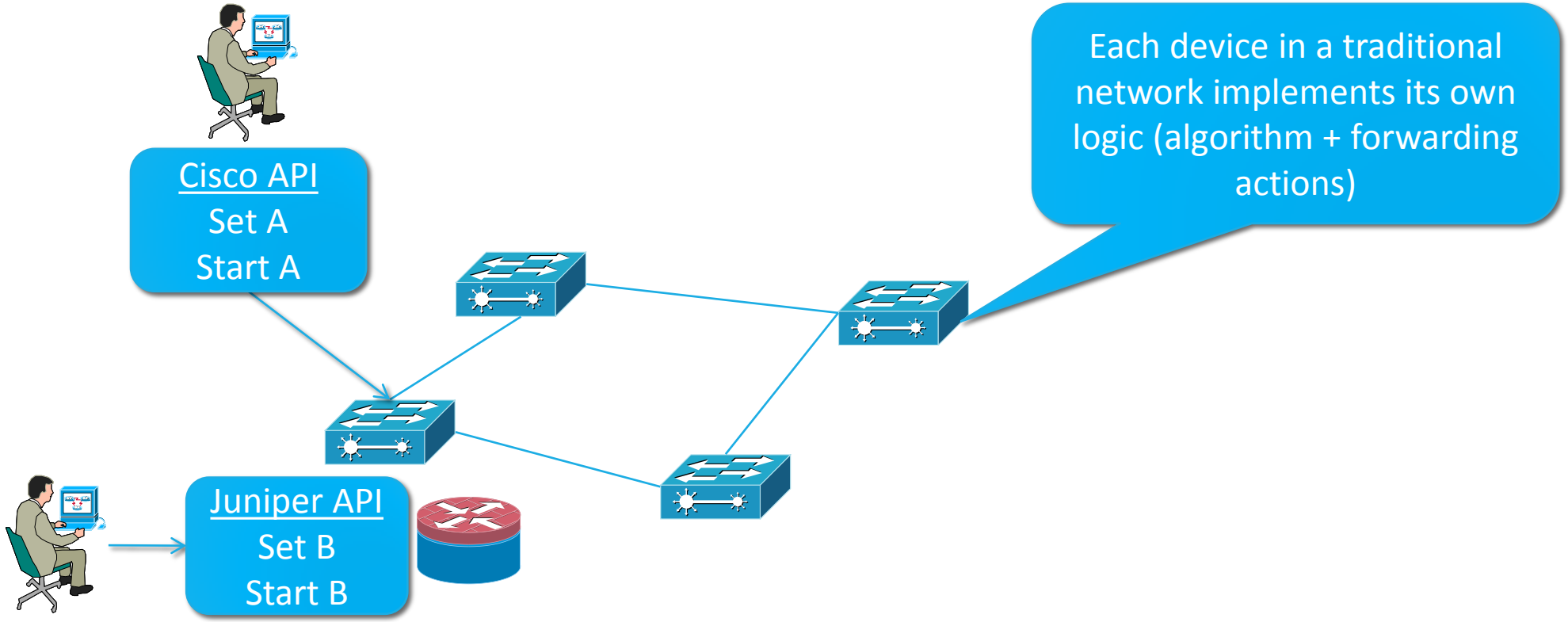
TEACHING ASSISTANT: GEORGIOS KATSIKAS

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`

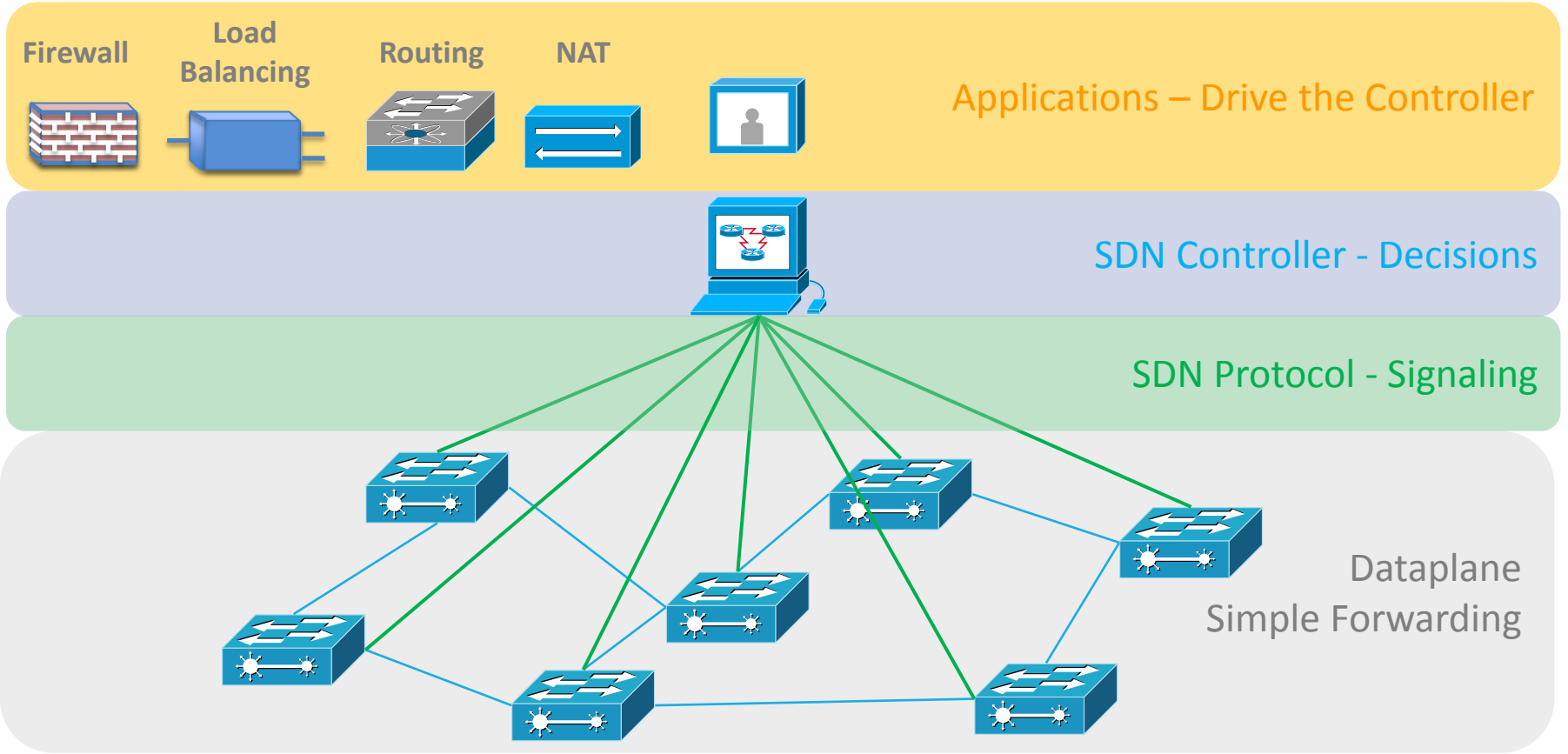
Traditional NM Approach



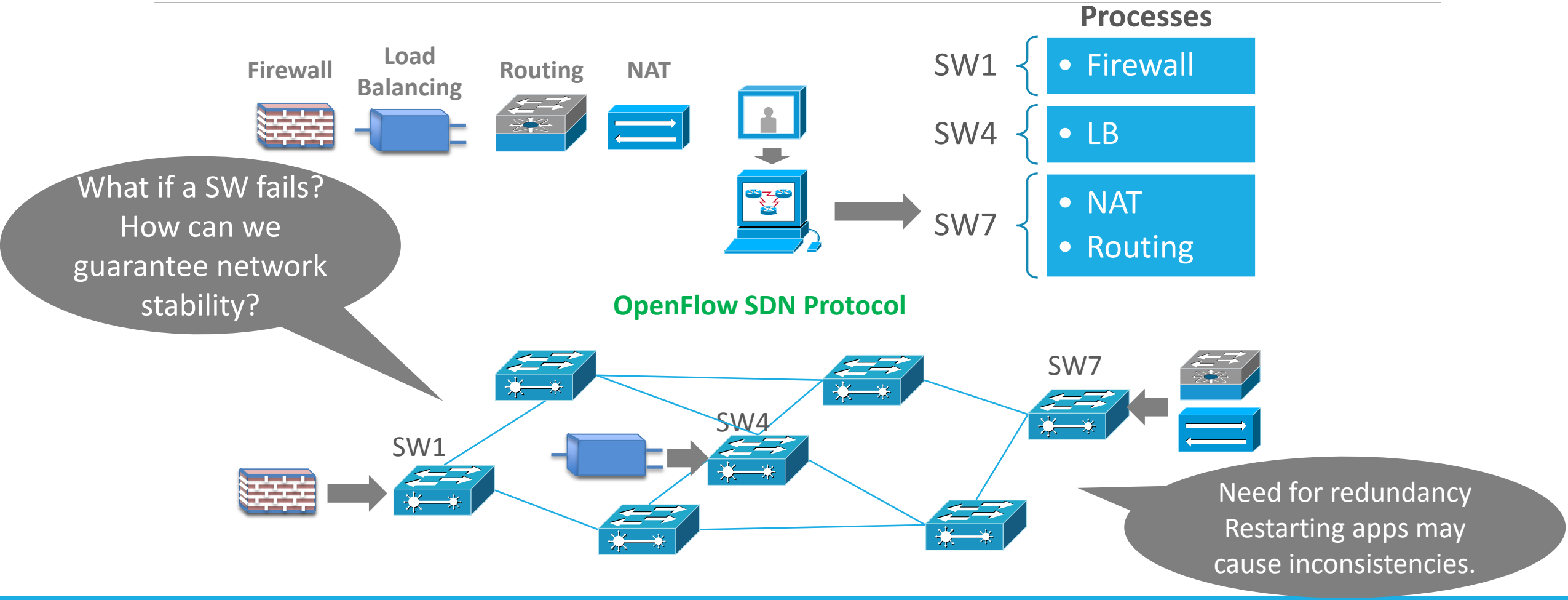
Modern NM Approach

Software Defined Networking

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



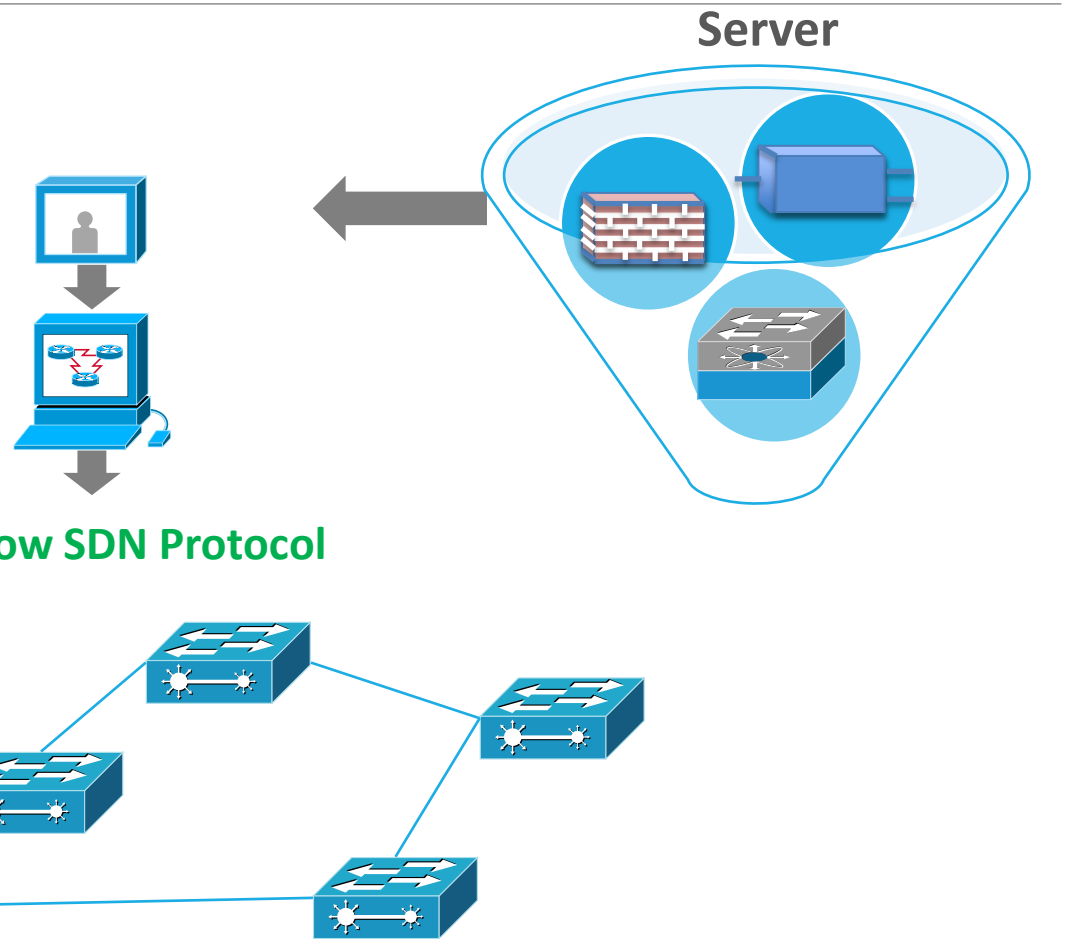
Modern NM Approach



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
Leading Activities

Global IT Concensous

PLATINUM MEMBERS

BROCADE, CISCO, CITRIX, ERICSSON, hp, IBM

JUNIPER NETWORKS, Microsoft, redhat

GOLD MEMBERS

NEC, vmware

SILVER MEMBERS

WIND, A10 Networks, ADVA Optical Networking, HUAWEI, INOCYBE TECHNOLOGIES, intel

ARISTA, AVAYA, ciena, midokura, nuagenetworks, ORACLE

ConteXtream, Coriant, CYAN, PANTHEON technologies, PLEXXI, PLUMgrid

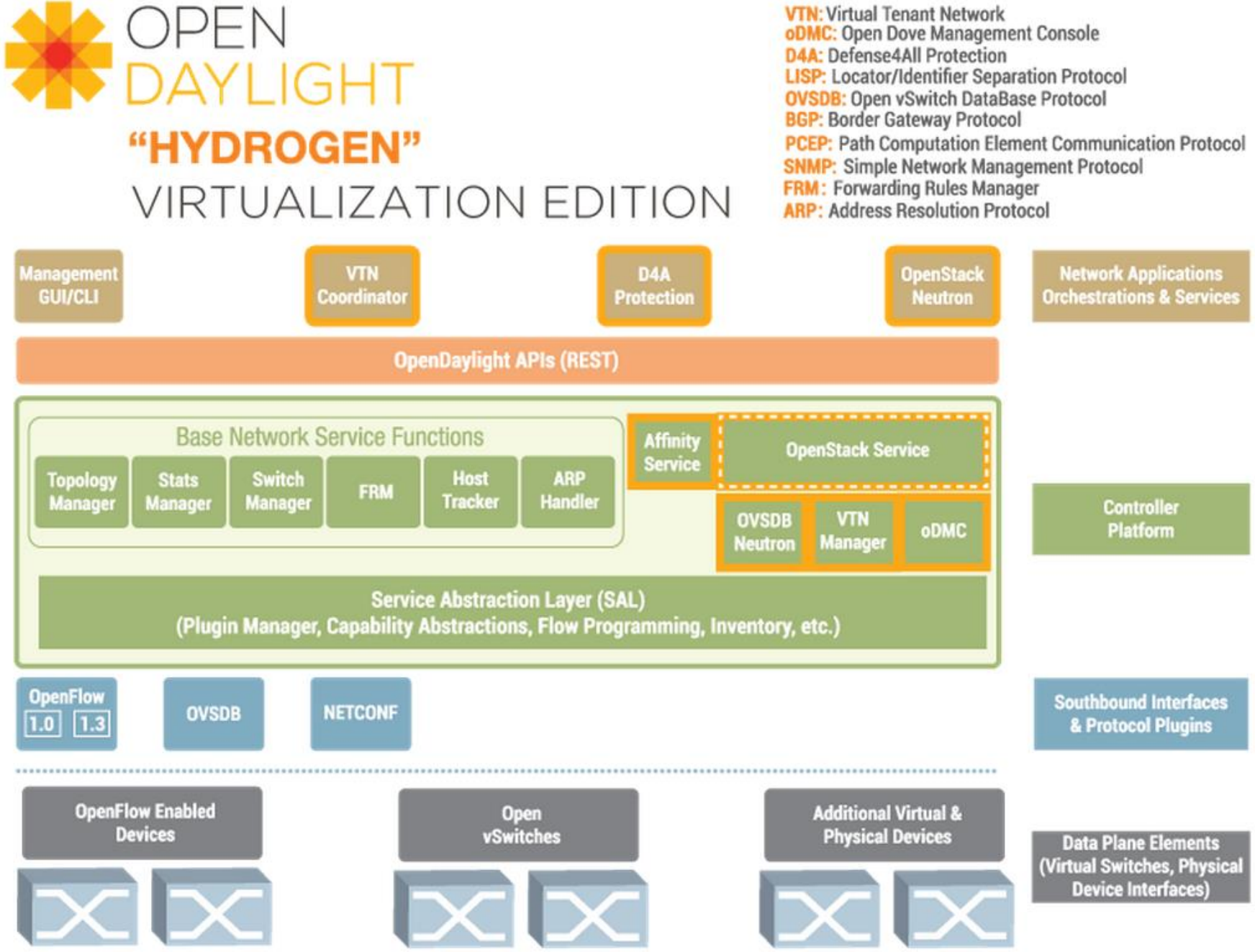
DELL, Extreme networks, FLEXTRONICS, QOSMOS Your Network is Information, radware, VERSA NETWORKS

FUJITSU, guavus, H3C, ZTE中兴

** Figures taken from [3]*

SDN

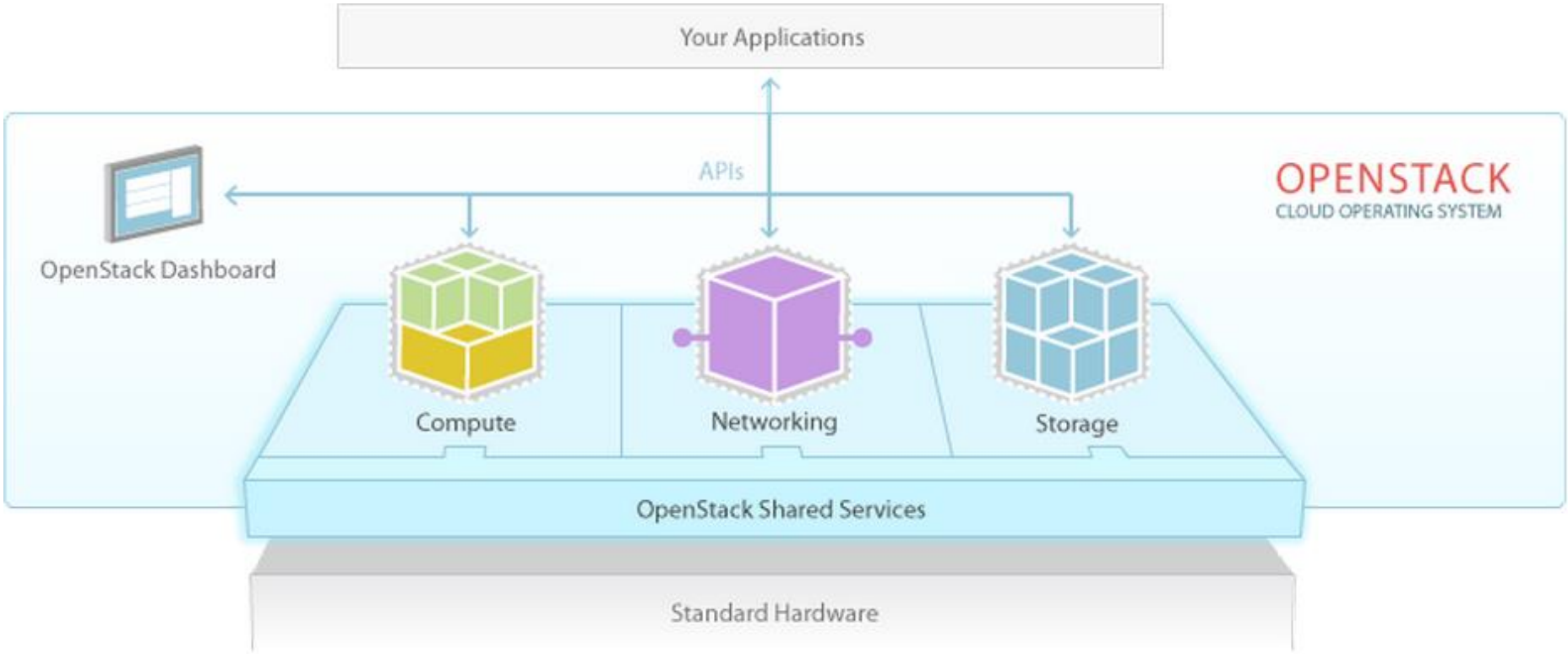
OpenDaylight Virtualization Edition	
Language	Java
Platform	Eclipse & OSGi



** Figure taken from [3]*

NFV

OpenStack Networking	
Language	Python, Bash
Platform	REST OpenStack Neutron



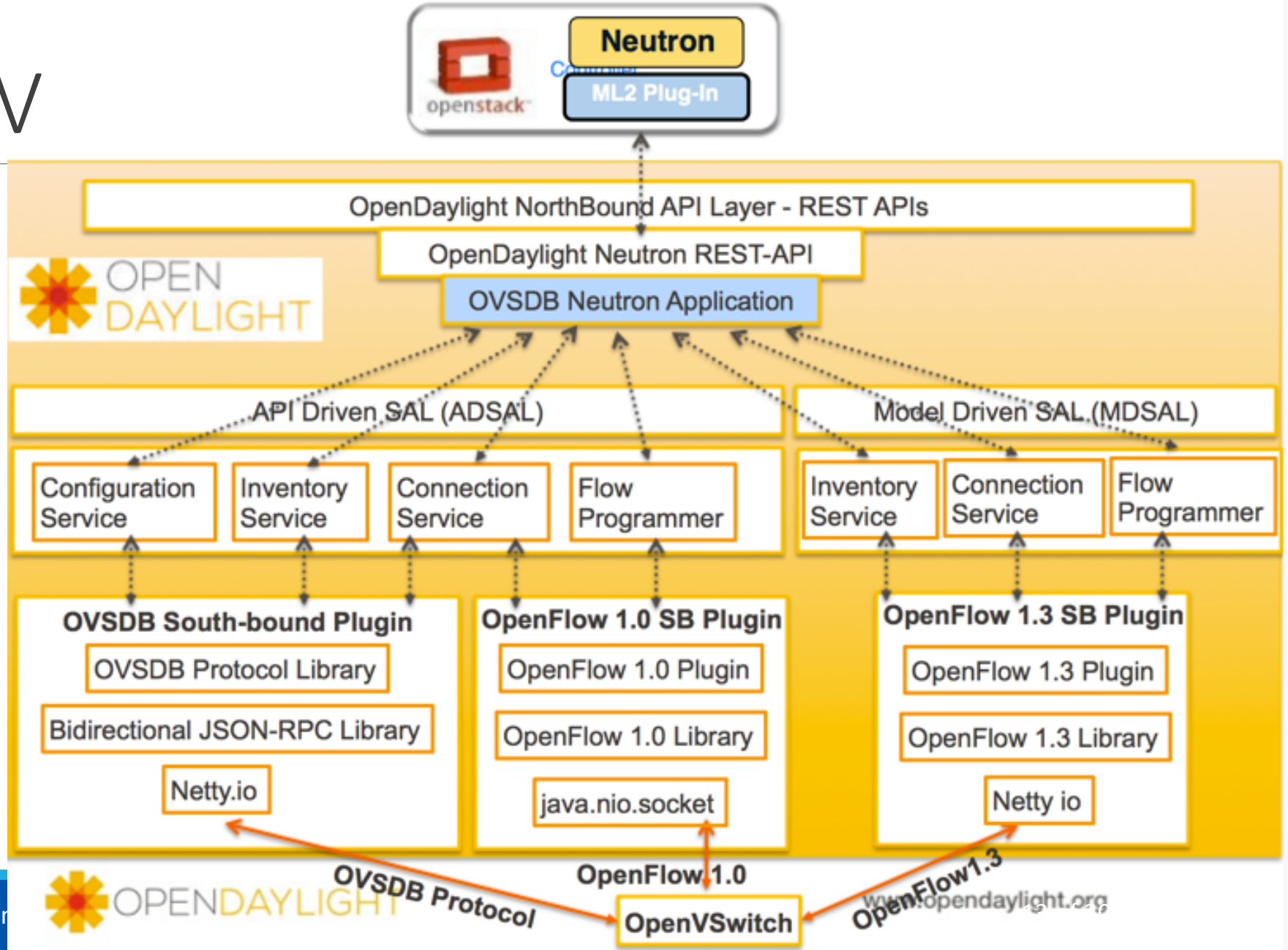
** Figure taken from [4]*

SDN + NFV

SDN + NFV

DevStack

Fedora 20 all-in-one VM released



** Figure taken from [5]*

SDN + NFV

Workflow to follow

- Abstract physical topology by creating high-level Virtual Tenant Networks (OpenStack Nodes)
- Instantiate VMs into OpenStack nodes
- Applications run in VMs and call OpenStack Networking API through REST
- OpenStack then talks to OpenDaylight to configure the real switches
- Two GUIs: OpenStack for virtual and OpenDaylight for physical network

Project #1 - MODL

DISTRIBUTED MONITORING EVENT AGGREGATION & MANAGEMENT FOR SDN.

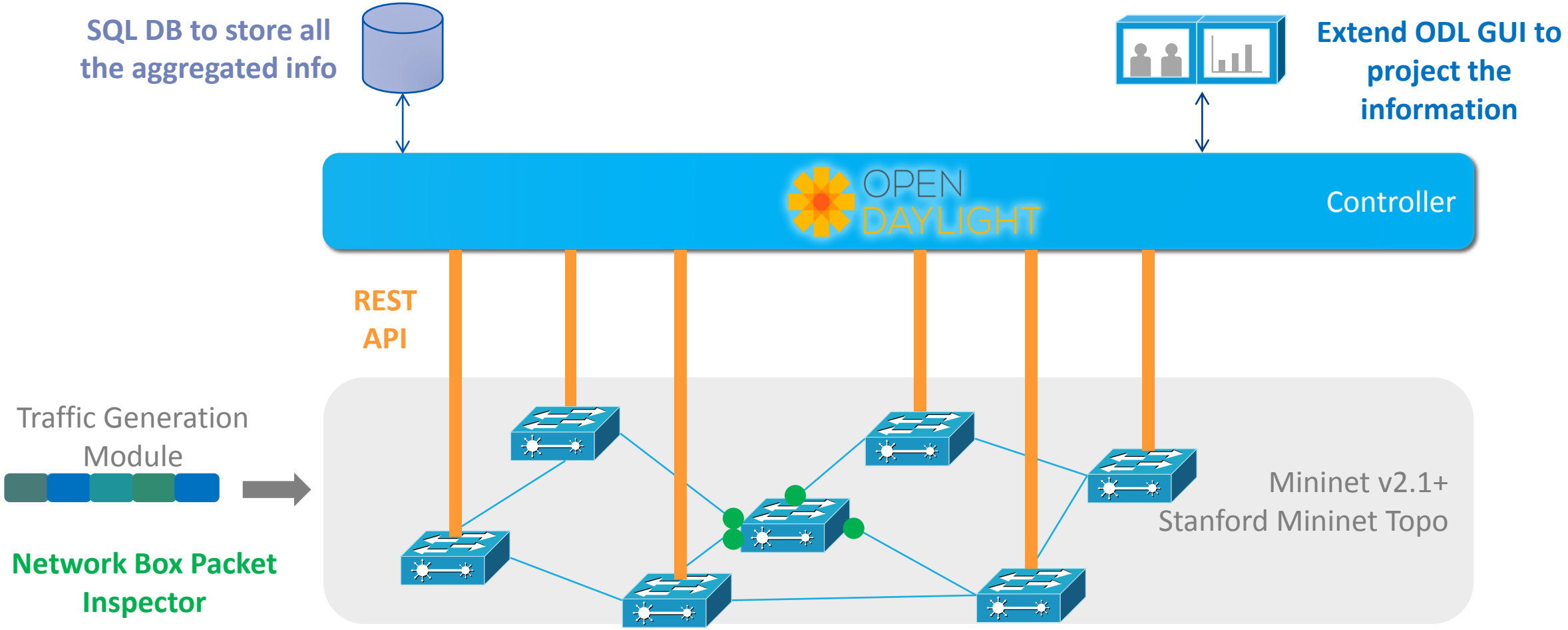
Distributed Monitoring Event Aggregation & Management for SDN

- **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 from slices, expired TTLs,
    - logged data to OS from protocols/switches/routers,
    - CPU/Memory/TCAM/cache/NIC and per port usage statistics for switches.

Distributed Monitoring Event Aggregation & Management for SDN

- **Task**
  - 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 [3] will be favored as the basic controller.
  - Develop side tools on the controller side (Extending ODL GUI) to effectively project the information.
- **Required skills**
  - C++, Java, Advanced Networking, Linux, Web Services, Web GUI, SQL, mininet network emulator, Wireshark, OpenDaylight, SDN principles.

Distributed Monitoring Event Aggregation & Management for SDN



Distributed Monitoring Event Aggregation & Management for SDN

- **Specific details**
  - OpenDaylight Hydrogen [3] should be downloaded and installed.
  - Mininet [1] will emulate the network devices. Stanford backbone network will be the testbed [7].
  - Restful API (WS) will be used to push info from switches to the controller.
  - Linux /proc, networking, process and HW management will be used to gather the statistics.
  - SQL (MySQL) will be used to store the aggregated information (controller side) for being pre-processed by other modules later on.
  - A packet inspection module must also be implemented. The module should be instructed to capture the packets of a 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.
  - Packet generation modules will be implemented so as to stress the network under different conditions and gather information from all the devices for an extended period. Standard mininet tools (iperf, netcat, ping) can be used and D-ITG [8] traffic generator.
  - ODL web-based GUI will be enhanced to project the gathered information. 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.

Project #2 – V-NM

NETWORK MANAGEMENT USING NFV

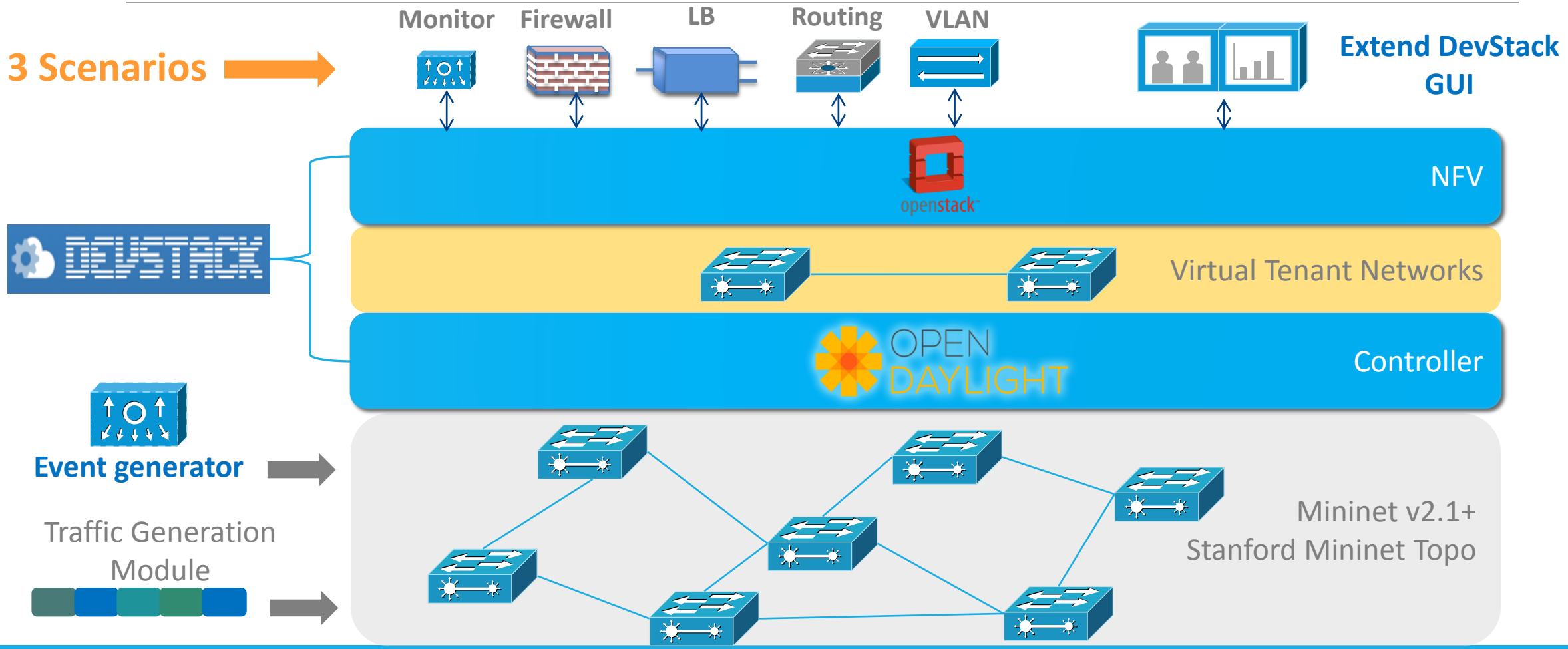
Network Management Using NFV

- **Problem**
  - To date, when a new service enters the market, operators have to install and deploy new infrastructure across the network. The majority of this infrastructure is manually customized to fit the purpose and can be hardly reconfigured with the danger to harm the service/network consistency/availability.
- **Motivation**
  - Eliminate box-by-box configuration to reduce network complexity, OPEX/CAPEX costs, increase network programmability, accelerate time to market for new services.
- **Approach**
  - Implement several, critical network functions in a virtualized environment (NFV/Network as a Service).
  - Abstract a physical topology using Virtual Tenant Networks to meet your clients' needs and develop failover mechanisms to enhance network resiliency.
  - Run various scenarios on this virtual topology to highlight the importance of your functions.

Network Management Using NFV

- **Task**
  - You are a network operator and you have 3 big companies as clients that want to run services on your network (mininet).
  - Use DevStack framework to provide SDN and NFV capabilities.
  - Implement 5 basic network functions (Firewall, Load Balancer, Tunneling/Slicing with security flavors, L3 Routing and NAT, Monitoring/Metering) as virtual OpenStack Neutron applications that can be dynamically instantiated and chained in a virtual network topology.
  - Investigate several event families that are present in today's networks and can be addressed with the above NFs. The purpose is to re-use NFs across all your clients.
  - Run 3 big scenarios that showcase how you isolate your clients (and guarantee certain QoS and security for each), how you handle traffic in different load scenarios and how you deal with errors (propose redundant solutions).
  - Implement a side module able to inject traffic in the network and stress your Network Functions as well.
- **Requirements**
  - Java, Python, Bash, Advanced Networking, Fedora Linux, Web Services, Web GUI, mininet network emulator, Wireshark, OpenDaylight, OpenStack, SDN/NFV principles.

Network Management Using NFV



Network Management Using NFV

- **Specific details**
  - Mininet [1] will emulate the network devices. Stanford topology will be the testbed [7].
  - Fedora 20 based all-in-one VM comes with OpenDaylight Virtualization edition [3], OpenStack Neutron [4], OpenVSwitch. You need at least three nodes, one for control (runs both ODL and OpenStack controllers) and another two for compute (host VMs).
  - Five Network Functions [5] [6]:
    - Load Balancer as a Service. Ability to run per flow or per packet hashing.
    - Firewall as a Service module with certain functionality (rules) to be proposed. Each VTN will have its own Firewall rules (in/out).
    - Virtual Private Network as a service for slicing resources (QoS) and security (IPSec).
    - L3 networking and NAT capabilities.
  - A packet generation module will be implemented so as to stress the network under different conditions. Mininet tools (iperf, netcat, ping) can be used and D-ITG [8] traffic generator.

Network Management Using NFV

- **Specific details**
  - Hint: The nature of the requested network functions will lead you design the scenarios.
  - ODL and OpenStack web-based GUIs will be used (and enhanced when needed) to visualize the results.

Big Idea

Big Idea

- **The two projects use ODL as basic framework. Coincidence??**
- **Project 1** ultimately inspects and projects every single detail of the physical topology. This information can be used to assess the network state.
- **Project 2** can use this information to detect problems and decide how to fine tune the network policy by instantiating/migrating remedy actions (e.g. when part of the network is suddenly congested a LB service can be deployed, QoS guarantees for certain clients).
- More details about each project will follow soon…

References

References

1) Mininet Network Emulator: http://mininet.org/
2) OpenFlow: https://www.opennetworking.org/sdn-resources/onf-specifications/openflow/
3) OpenDaylight Hydrogen: http://www.opendaylight.org/
4) OpenStack Cloud Software: http://www.openstack.org/
5) OpenStack Networking API: http://developer.openstack.org/api-ref-networking-v2.html
6) OpenStack Networking API extensions: http://docs.openstack.org/api/openstack-network/2.0/content/API_extensions.html
7) Mininet Stanford Backbone: http://reproducingnetworkresearch.wordpress.com/2012/07/11/atpg/
8) Distributed Internet Traffic Generator: http://traffic.comics.unina.it/software/ITG/


Q&A