SDN intro + Openstack

Radim Roška / worwan

5.5.2015 @ SH

Obsah

- Intro do SDN
  - Pojmy a myšlenky
  - Různé open source projekty SDN světa
  - OpenFlow
- OpenStack
  - Hlavní myšlenky a popis architektury
  - Detailněji Nova a Neutron
  - cvičení

Wiki intro

- *Software-defined networking (SDN) is an approach to computer networking that allows network administrators to manage network services through abstraction of lower-level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the selected destination (the data plane). The inventors and vendors of these systems claim that this simplifies networking.[1]*

- *SDN requires some method for the control plane to communicate with the data plane. One such mechanism, OpenFlow, is often misunderstood to be equivalent to SDN, but other mechanisms could also fit into the concept.*

SDN architektura

- The OpenFlow protocol is a foundational element for building SDN solutions. The SDN architecture is:
  - *Directly programmable*: Network control is directly programmable because it is decoupled from forwarding functions.
  - *Agile*: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.
  - *Centrally managed*: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
  - *Programmatically configured*: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
  - *Open standards-based and vendor-neutral*: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

Src: wiki

SDN Standards Organization

Several Efforts related to SDN and smooth migration using existing NEs

- Programmatic Interfaces
- SDN Agents and Controllers
- Infrastructure Virtualization

Initiated by IT companies, with some involvement from operator side

- SDN architecture definition
- OpenFlow standard definition
- Not much consideration on existing network infrastructure

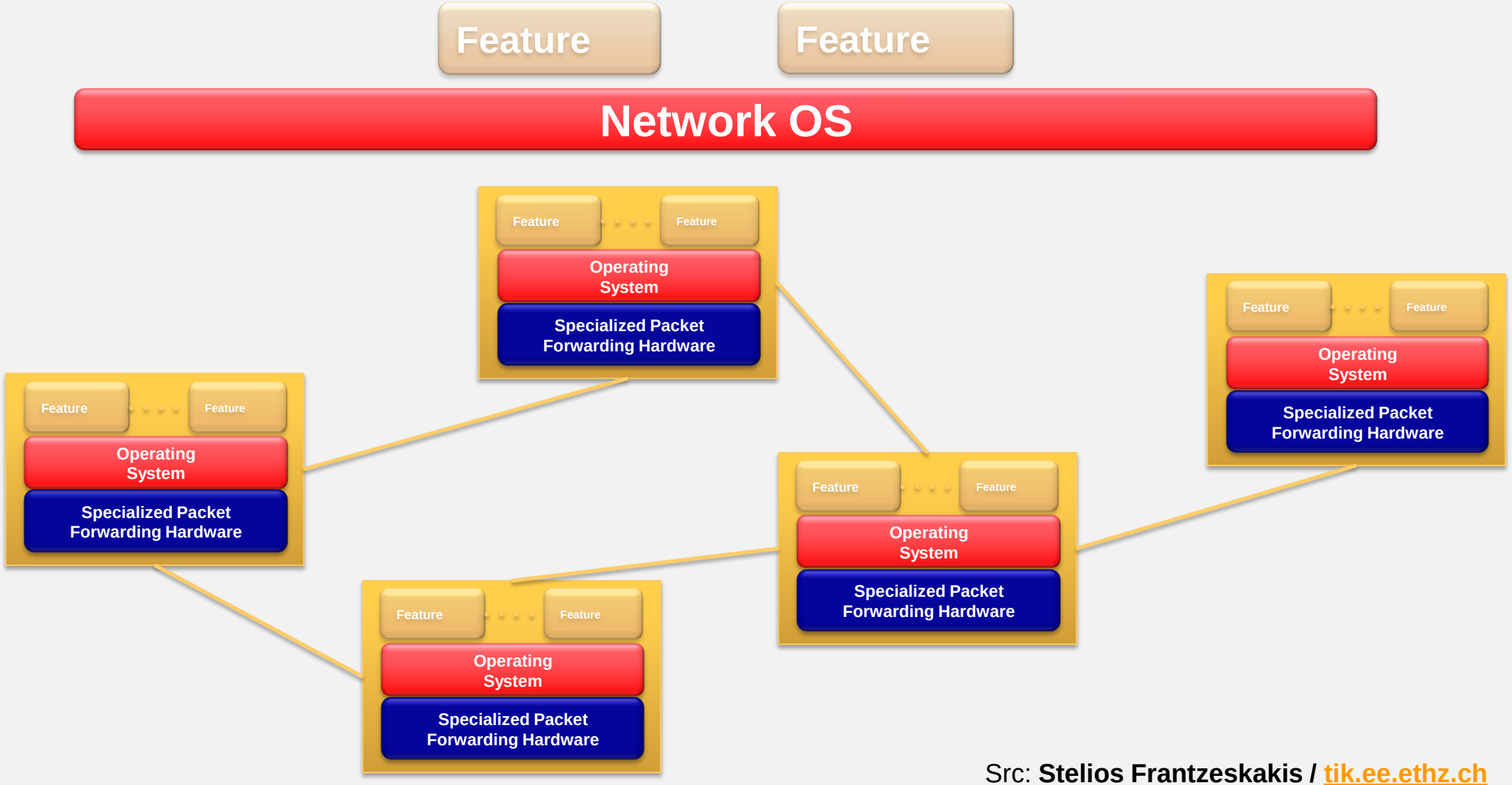
Initiated NfV Industry Standards Group

Network-operator-driven - Started by 13 Operators (VZ, DT, ATT..)
Virtualization of Network Functions
Complimentary to SDN and Open Innovation
IT vendors such as HP extremely active

SDN

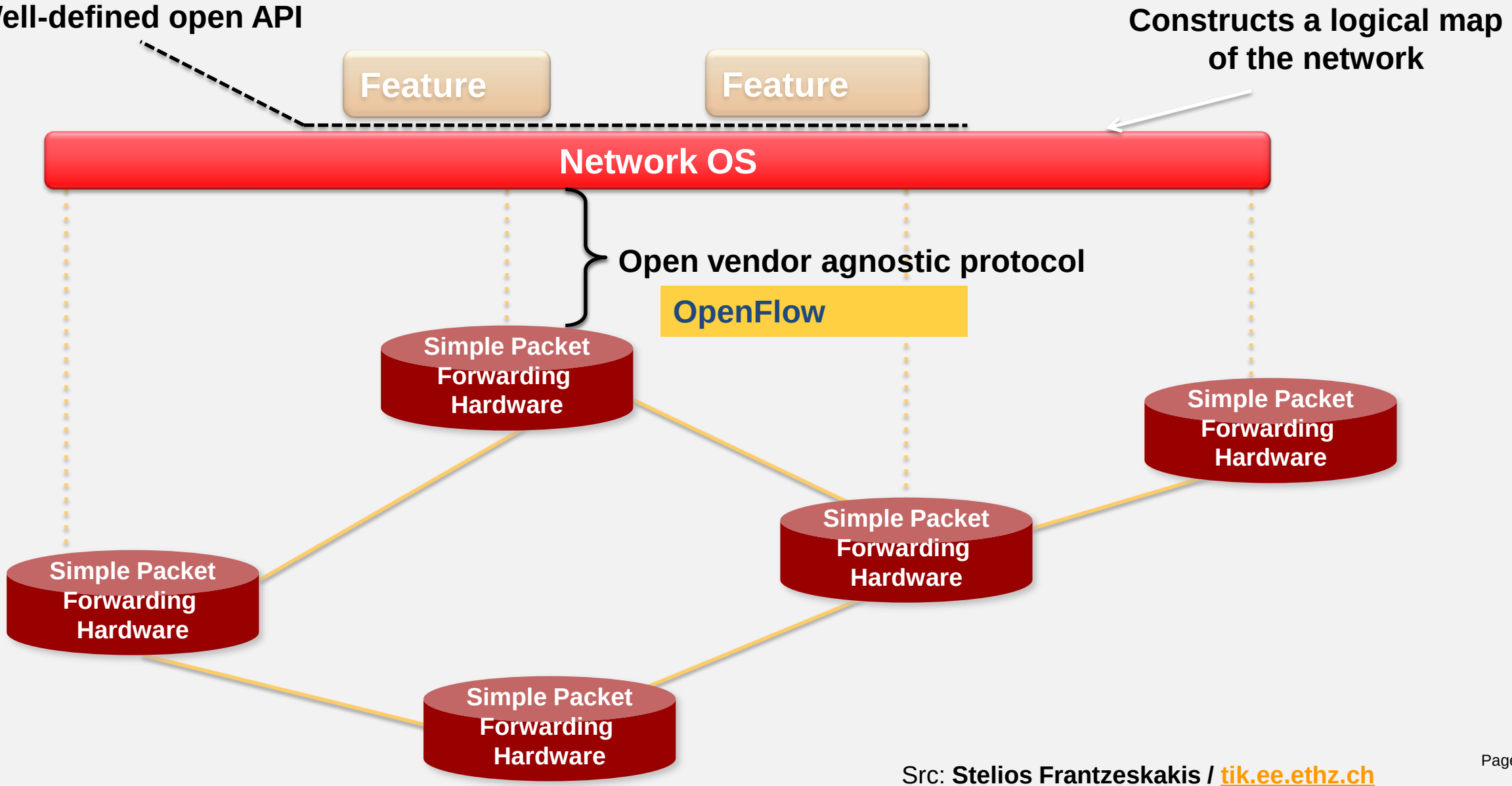
- **Možné definice – stručnější ☺**
  - SDN - nová architektura sítí
    - Usnadňuje programování toho, jak se sítě chovají
    - Hlavní myšlenka SDN je remote control síťového HW
    - …
  - Je to nyní populární téma ☺
    - Probírá se ve škole?

From Vertically Integrated to …



Src: **Stelios Frantzeskakis / tik.ee.ethz.ch**

Software Defined Network



Src: **Stelios Frantzeskakis /** tik.ee.ethz.ch

SDN & NFV: Driving Architectural Transformation

From This:

Traditional networking topology

Monolithic vertical integrated box

TEM proprietary solutions

Firewall | VPN | IDS/IPS

TEM/OEM Proprietary OS

ASIC, DSP, FPGA, ASSP

To This:

Networking within VMs

Standard x86 COTS HW

Open SDN standard solutions

VM: Firewall | VM: VPN | VM: IDS/IPS

SDN/NFV

IA CPU

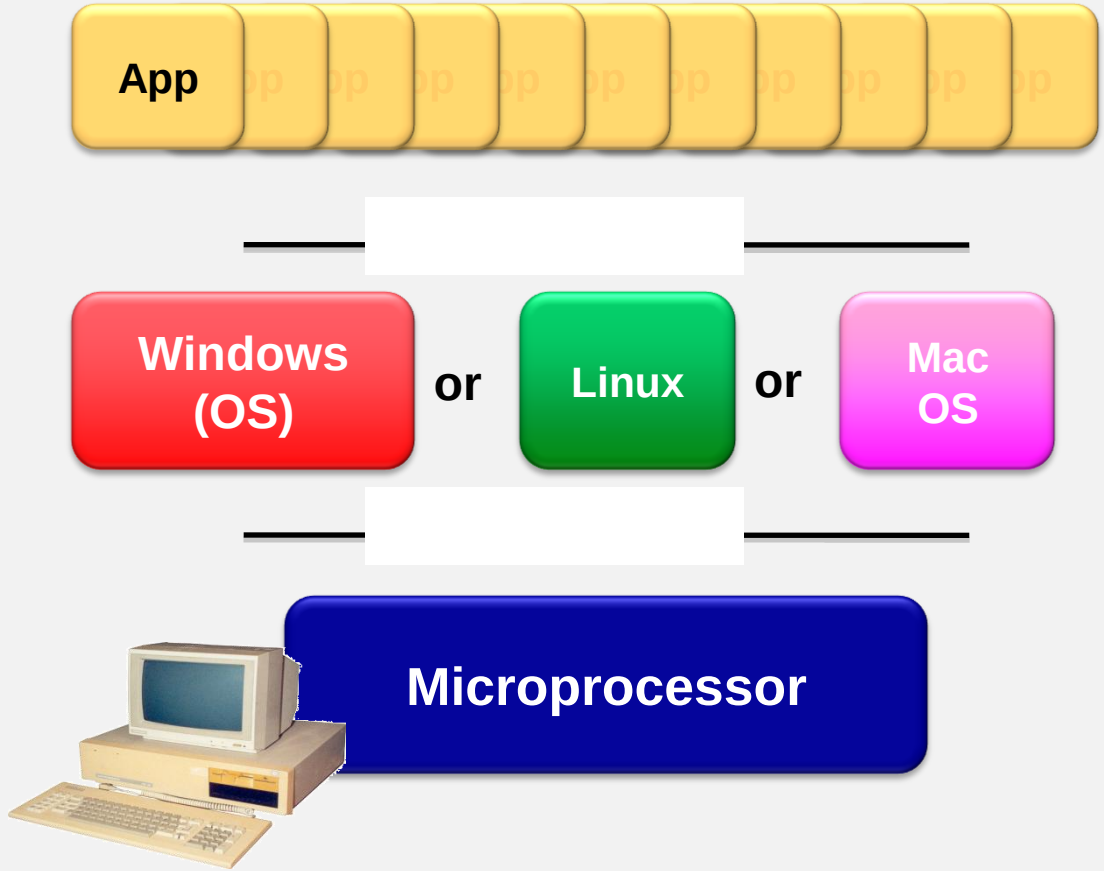
NIC Silicon

Chipset Acceleration

Switch Silicon

Wind River Linux + Apps

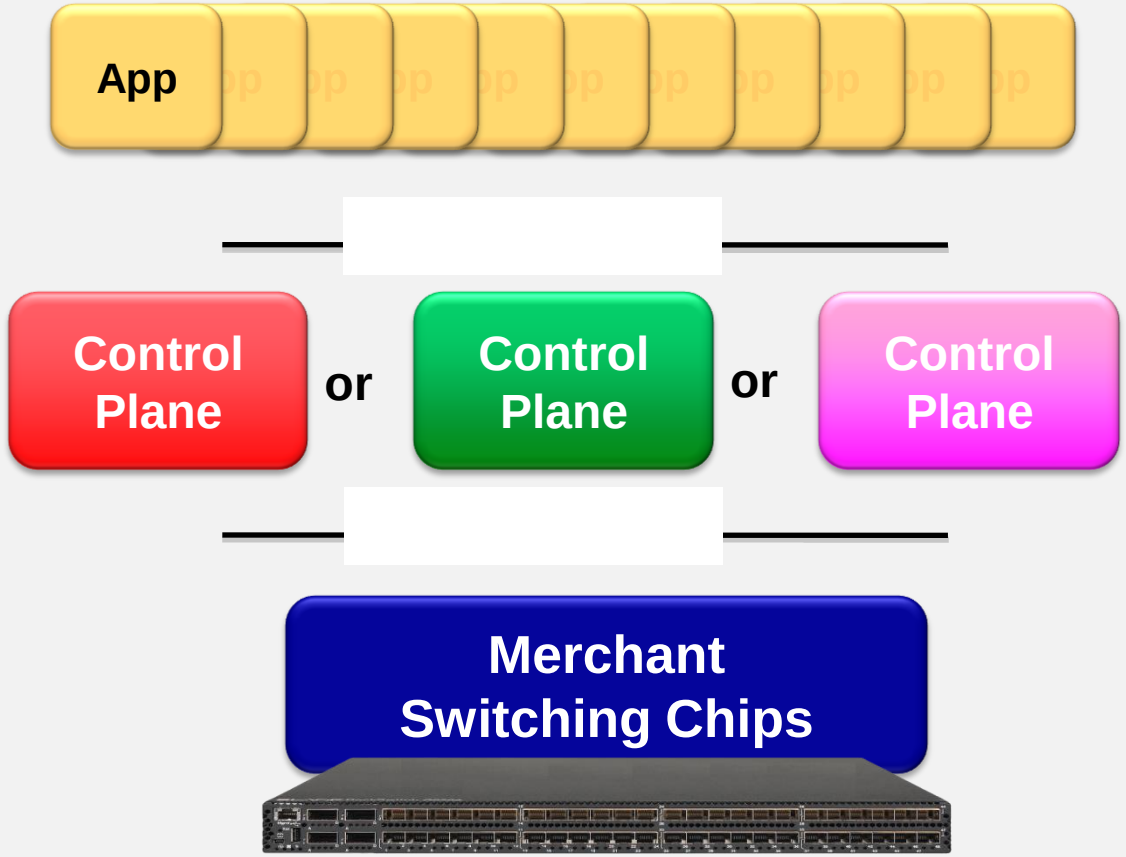
Mainframe industry in the 1980s: Vertically integrated
Closed, proprietary
Slow innovation
Small industry



Horizontal
Open interfaces
Rapid innovation
Huge industry

Src: **Stelios Frantzeskakis /** tik.ee.ethz.ch

Networking industry in 2007: Vertically integrated
Closed, proprietary
Slow innovation

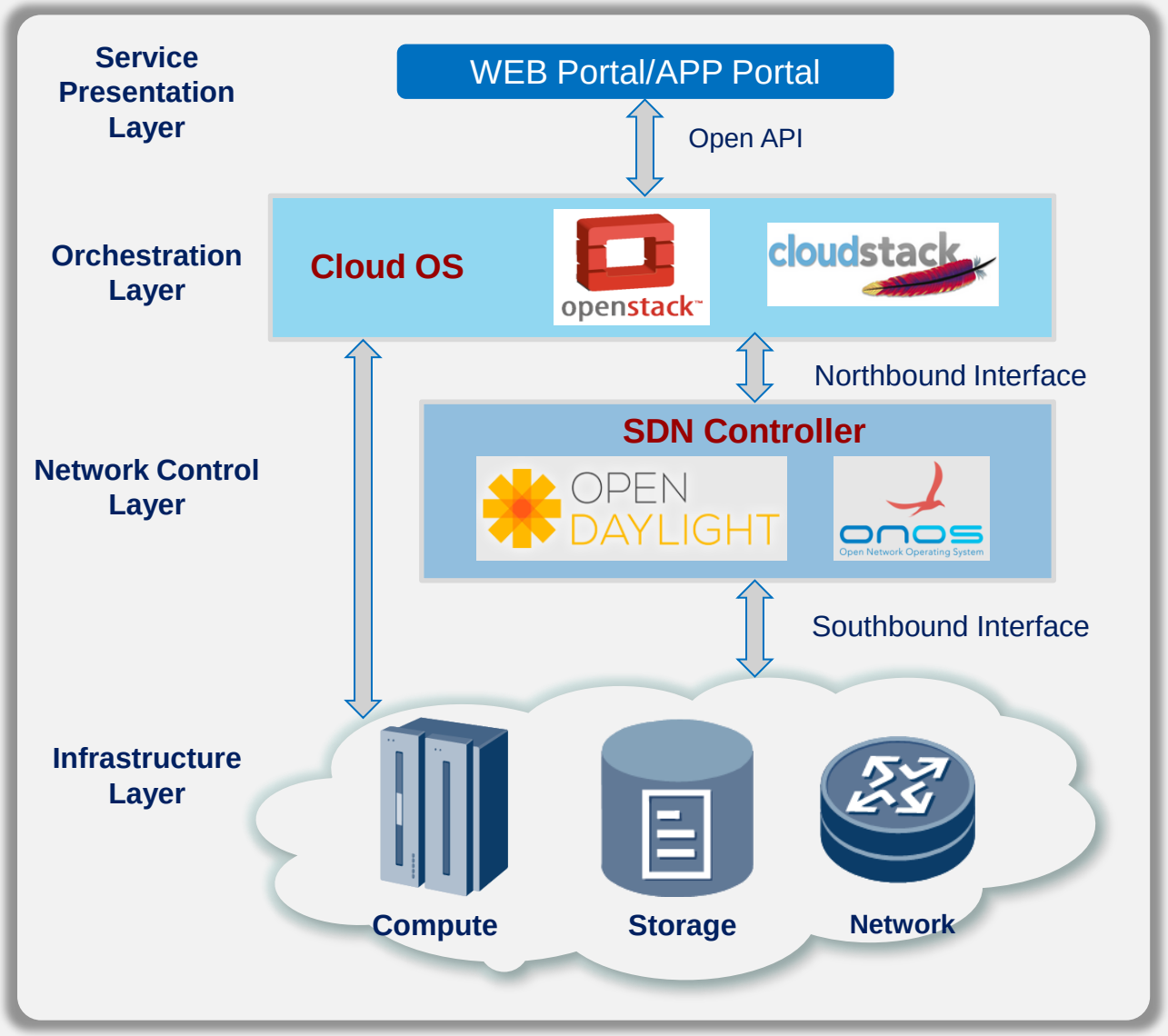


Horizontal
Open interfaces
Rapid innovation

Src: **Stelios Frantzeskakis / tik.ee.ethz.ch**

Základy SDN

Cloud Architecture



- **Service presentation layer**
  - Portals oriented to carriers, enterprises, tenants, and RSPs
  - Flexible service customization interfaces
- **Orchestration layer**
  - Standard and open architecture, compatible with multi-vendors
  - Orchestration of storage, computing, and network resources
- **Network control layer**
  - Implementing network modeling and network instantiation.
  - Northbound open APIs can be deployed for rapid customization and automatic provisioning of services.
  - Southbound interfaces can be deployed for managing physical and virtual networks in a unified manner.
- **Infrastructure layer**
  - The infrastructure layer is composed of computing, storage and network resources
  - Overlay network with physical and virtual networks planned and designed in a unified manner

Mainstream Open-source Cloud OS

Background of OpenStack and CloudStack

OpenStack *http://www.openstack.org/*

- OpenStack is a global collaboration whose aim is to produce the open standard cloud operating system for both public and private clouds.
- OpenStack is a freely available, Apache-licensed software system that can be used to build massively scalable cloud environments.
- OpenStack was initially developed by Rackspace and NASA. Opened source on October 2010 and using Apache 2.0 license
- 2011 Rackspace announced OpenStack Foundation and contributed it to the OpenStack Foundation

CloudStack *http://cloudstack.apache.org/*

- CloudStack is a Cloud Orchestration platform that pools computing resources to build public, private, and hybrid Infrastructure as a Service (IaaS) clouds.
- CloudStack was developed by Cloud.com in 2008. There are two versions of commercial and open source. Open source version uses GPL v2 license
- On July 2011, Cloud.com was acquired by Citrix (Citrix) and CloudStack source was open since then
- In April 2012, Citrix contributed CloudStack to the Apache Foundation

What Can We Do on Cloud OS

Application Owner/User

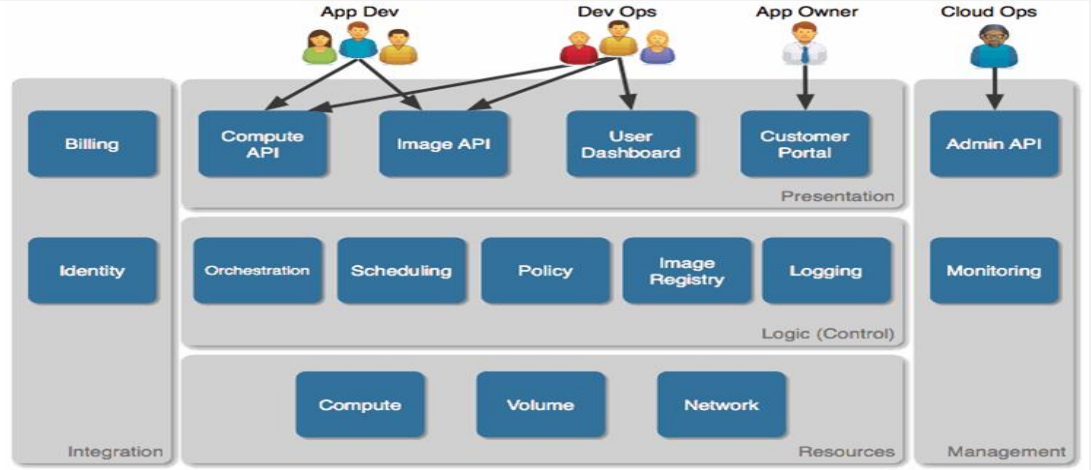
Subscribe to Cloud services, monitor the application operation and billing status

Developer and Operator

Create and Save their customized images. Start, monitor and terminate instances by using customized images

Administrator

Configure and operate the infrastructure includes computing, storage and network resources



Mainstream Open-source SDN Controller

OpenDayLight *http://www.opendaylight.org/*

- On Apr 8, 2013, The Linux Foundation, announced the founding of the OpenDaylight Project as a community-led and industry-supported open source framework to accelerate adoption, foster new innovation and create a more open and transparent approach to Software-Defined Networking (SDN) and Network Functions Virtualization (NFV).
- The project's founding members—Arista Networks, Big Switch Networks, Brocade, Cisco, Citrix, Ericsson, HP, IBM, Juniper Networks, Microsoft, NEC, Nuage Networks, PLUMgrid, Red Hat and VMware—committed to donating software and engineering resources for OpenDaylight's open source framework to help define the future of an open source SDN platform.
- On Feb, 2014 First Release "Hydrogen"
  On Oct,2014. Second Release "Helium"
  Third Release "Lithium" is on roadmap

ONOS *http://onosproject.org/*

- The Open Network Operating System (ONOS) is the first open source SDN network operating system targeted specifically at the Service Provider and mission critical networks.
- ONOS has created useful Northbound abstraction and APIs to enable easier application development and Southbound abstractions and interfaces to allow for control of OpenFlow-ready and legacy devices.
- ONOS has been developed in concert with leading service providers (AT&T, NTT), with demanding network vendors (Ciena, E///, Fujitsu, Huawei, Intel, NEC), R&E network operators (Internet2, CNIT, CREATE-NET), collaborators (SRI, Infoblox), and with ONF to validate its architecture.
- First release "Avocet" was released on Dec, 2014.
  Second release "Blackbird" will be released on Feb, 2015.

REST & RESTful API

REST & RESTful

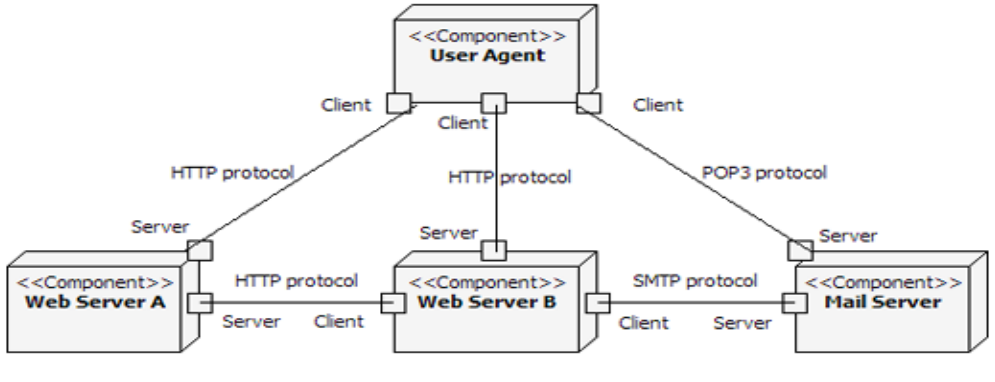
➢ **REST/RESTful**

1. Representational State Transfer (REST) is a style of software architecture. As described in a dissertation by Roy Fielding, REST is an "architectural style" that basically exploits the existing technology and protocols of the Web.
2. Web service APIs that adhere to the REST architectural constraints are called RESTful

➢ **REST Constraints**

1.Everything on the network is abstracted as a resource
2.Each resource corresponds to a unique resource identifier URI
3.Using HTTP as a connector for manipulating resource
4.Any operation does not change the resource identifier URI
5. All server operations are stateless (bezstavová)

REST Architecture

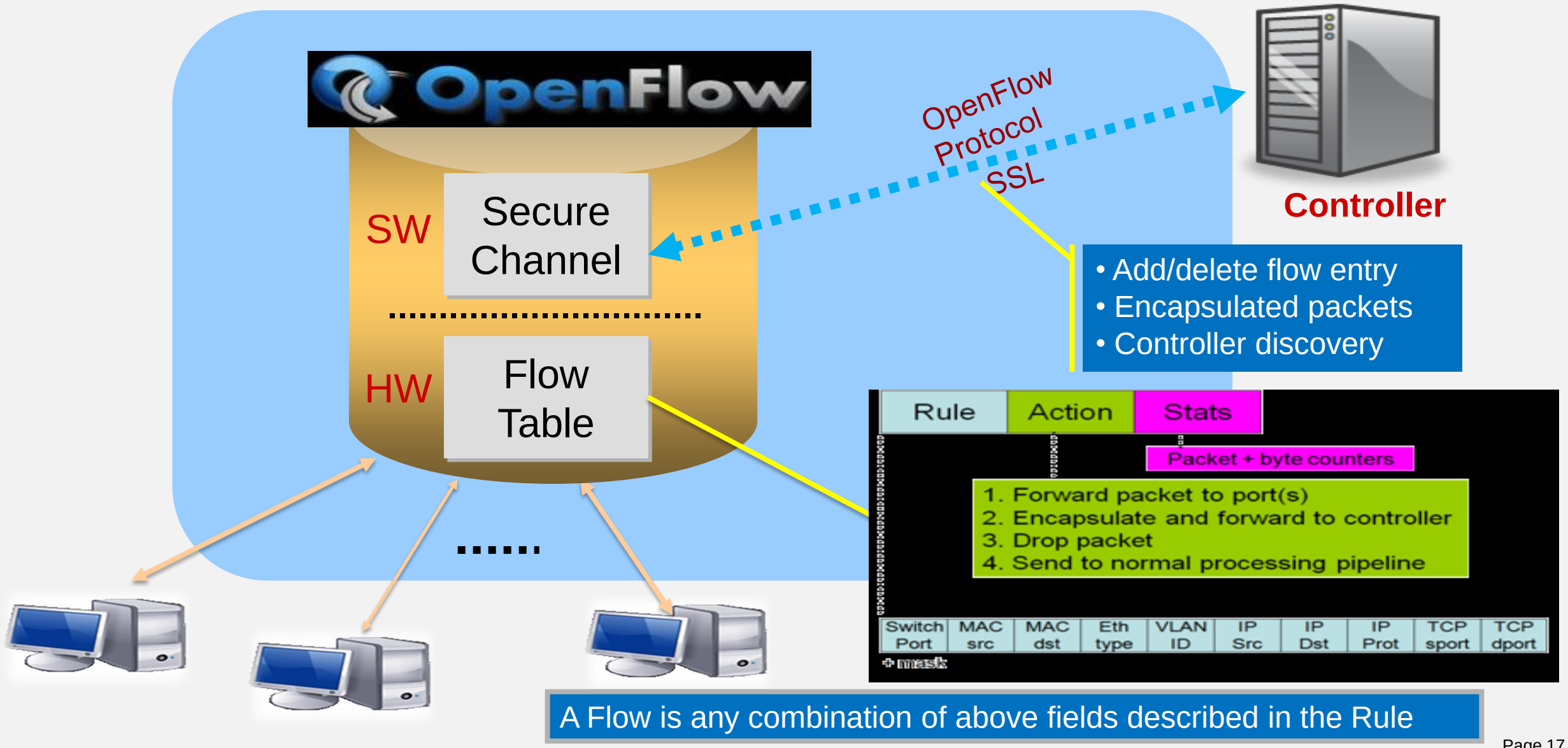


➢ **Typical HTTP methods for RESTful API**

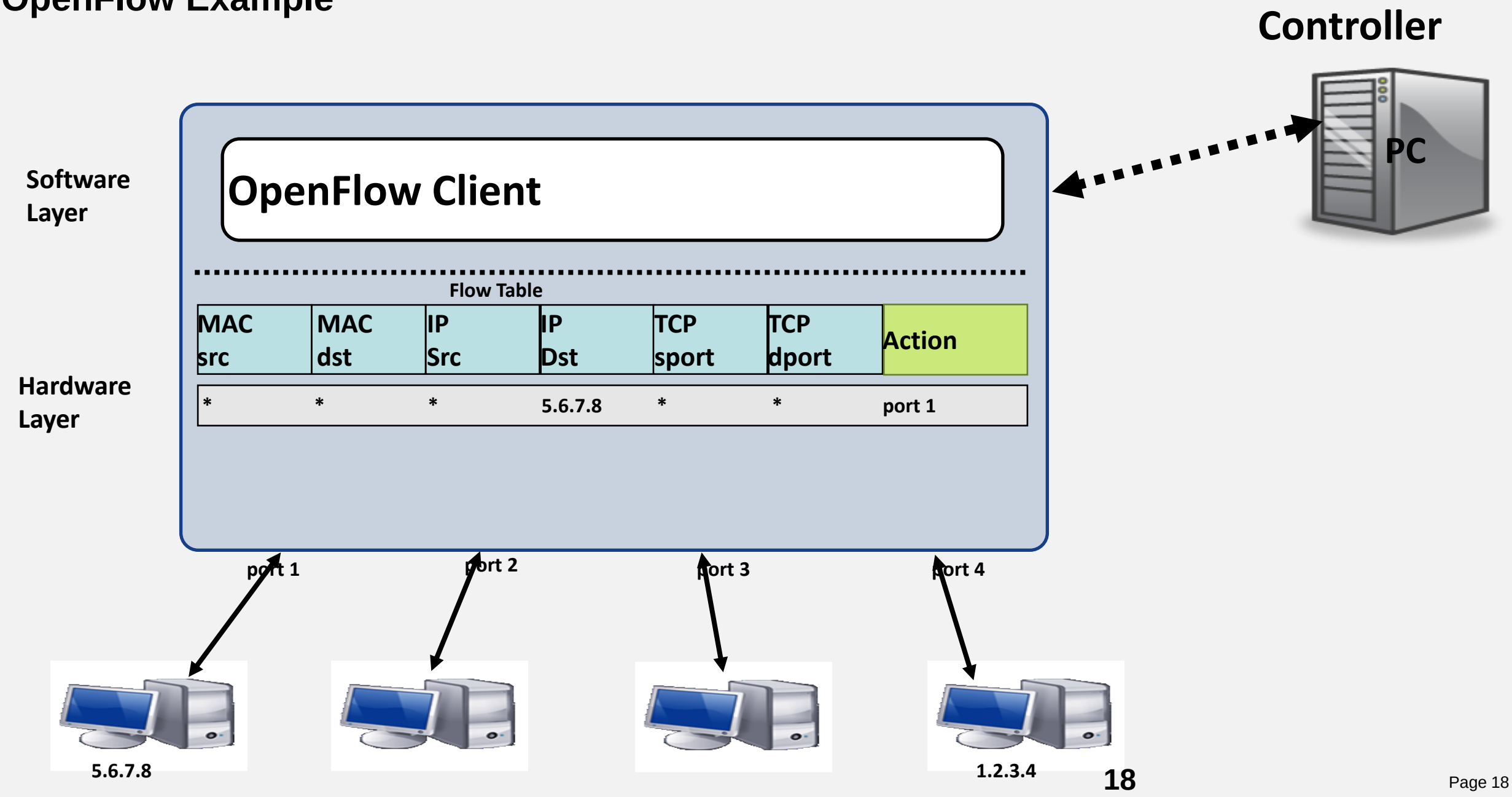
Resource: Collection URI
(such as http://example.com/resources/

1. **GET: List** the URIs
2. **PUT: Replace** the entire collection with another collection
3. **POST: Create** a new entry in the collection
4. **DELETE: Delete** the entire collection

OpenFlow: Work Flow ~ "forwarding table management tool"

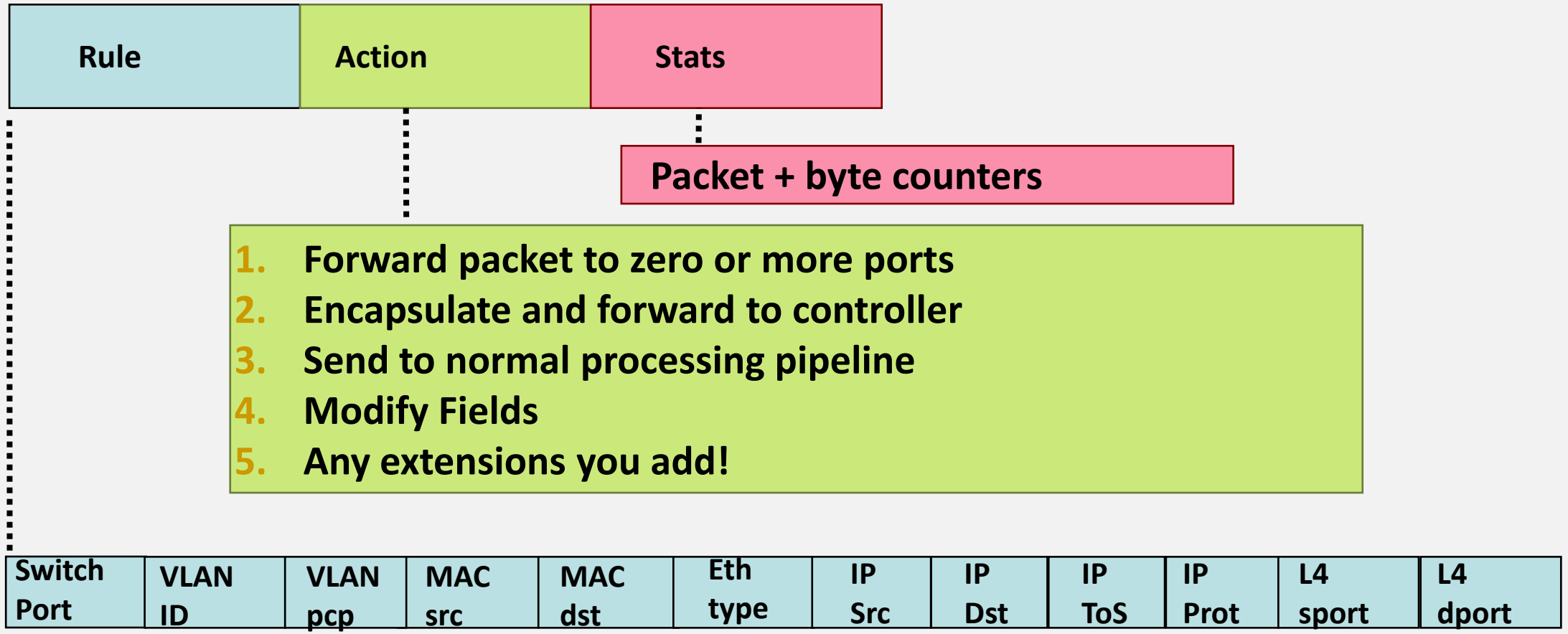


OpenFlow Example



OpenFlow Basics

Flow Table Entries



+ mask what fields to match

Src: **Stelios Frantzeskakis / tik.ee.ethz.ch**

Examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

Src: **Stelios Frantzeskakis / tik.ee.ethz.ch**

Examples

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

Src: **Stelios Frantzeskakis / tik.ee.ethz.ch**

Open Flow tutorial

- http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial

Src: **Stelios Frantzeskakis /** tik.ee.ethz.ch

Configuration/Provisioning Protocol: NETCONF & YANG

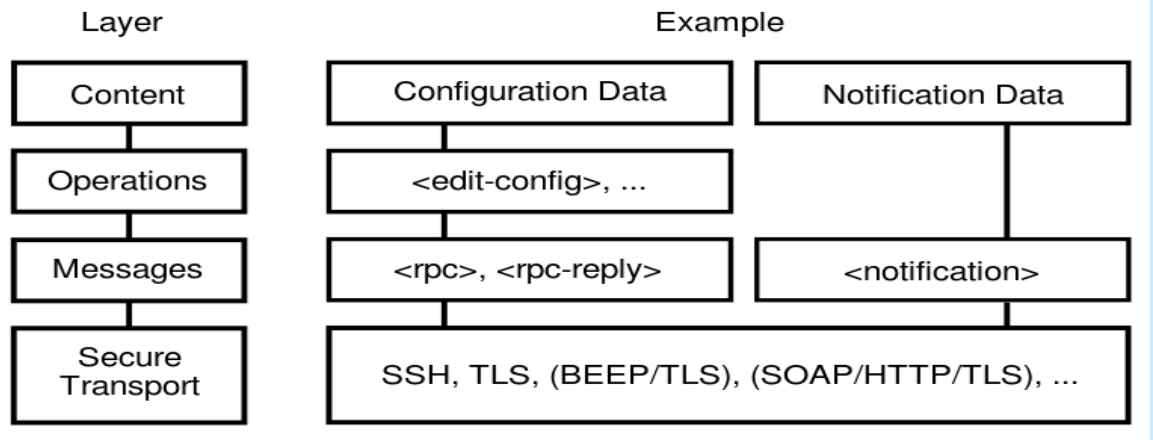
NETCONF & YANG

- NETCONF (RFC4711/6241) provides mechanisms to install, manipulate, and delete the configuration of network devices. Its operations are carried on top of a simple remote procedure call (RPC) layer. The NETCONF protocol uses data encoding based on the Extensible Markup Language [XML] for data as well as protocol messages. This in turn is realized on top of the transport protocol, which can be TCP, HTTP, or HTTPS.

- Basic NETCONF Operations: get, get-config, editconfig, copy-config, delete-config, lock, unlock, close-session, and killsession.

- YANG: The NETMOD working group has completed work to define a "human-friendly" modeling language for defining the semantics of operational data, configuration data, notifications, and operations, called YANG (RFC6020/6021).

NETCONF Architecture



1. The Content layer consists of configuration data and notification data.
2. The Operations layer defines a set of base protocol operations to retrieve and edit the configuration data.
3. The Messages layer provides a mechanism for encoding remote procedure calls (RPCs) and notifications.
4. The Secure Transport layer provides a secure and reliable transport of messages between a client and a server.

Více: http://www.slideshare.net/tailfsystems/netconf-yang-tutorial

Src: **Huawei slidy**

Netconf example

Dotaz

```
<?xml version="1.0"?>
<rpc
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:cpi="http://www.cisco.com/cpi_10/schema"
message-id="101">
   <get-config>
        <source>
            <running/>
         </source>
          <filter>
              <config-format-text-cmd>
               <text-filter-spec>
        interface Loopback113
                  </text-filter-spec>
              </config-format-text-cmd>
          </filter>
      </get-config>
</rpc>
```

Odpověď

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-
id="101"xmlns=\"urn:ietf:params:netconf:base:1.0\"
>
   <data>
      <cli-config-data>
interface Loopback113
description test456
no ip address
load-interval 30
end
       </cli-config-data>
    </data>
</rpc-reply>
```

Porovnání NetConf a OpenFlow

…..?

Porovnání NetConf a OpenFlow

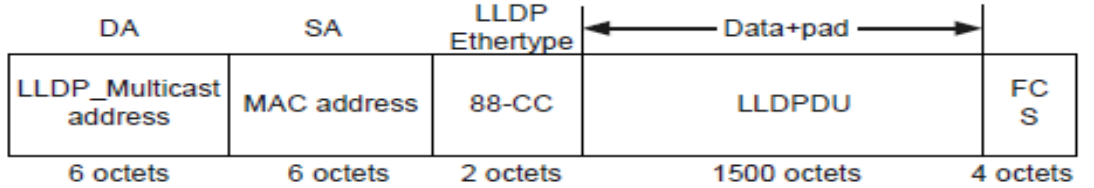
- **Netconf**
  - upravuje konfiguraci zařízení
  - Konfigurace je uložená na zařízení
  - Nové featury musí podporovat zařízení samo o sobě (netconf není separace ctrl a dt plane)
- **OpenFlow**
  - upravuje forwarding table
  - Konfigurace není uložená na zařízení
  - Cokoliv jde implementovat v controlleru a pak jako flow vložit do switche/routeru
- **Pravděpodobně tedy – obojí bude „ ve hře “**
  - OpenFlow – nové featury
  - Netconf – konfigurace stávajicího SW na prvcích (IOS, JunOS,VRP, Comware…)

Topological Information Abstraction: LLDP

What's LLDP

- **The Link Layer Discovery Protocol (LLDP)** is an industry-standard protocol defined in IEEE 802.1AB that allows networked devices to discover and advertise capabilities and identity information onto a layer 2 LAN.
- The layer 2 protocol that was standardized by the IEEE1 replaces several proprietary protocols implemented by individual vendors for their equipment including the Cisco Discovery Protocol (CDP).
- LLDP allows network devices that operate at the lower layers of a protocol stack (e.g., layer 2 bridges and switches) to learn some of the capabilities and characteristics of LAN devices available to higher layer protocols (e.g., IP addresses).
- The information gathered through the LLDP operation is stored in a network device and can be queried using the SNMP protocol, the CLI, or NETCONF. A device's neighbor topology and associated information can also be gathered from this database.

LLDP Frame Architecture



Some of the information that can be gathered by LLDP includes the following:

- System name and description
- Port name and description
- VLAN name and identifier
- IP network management address
- Capabilities of the device (e.g., switch, router, or server)
- MAC address and physical layer information
- Power information

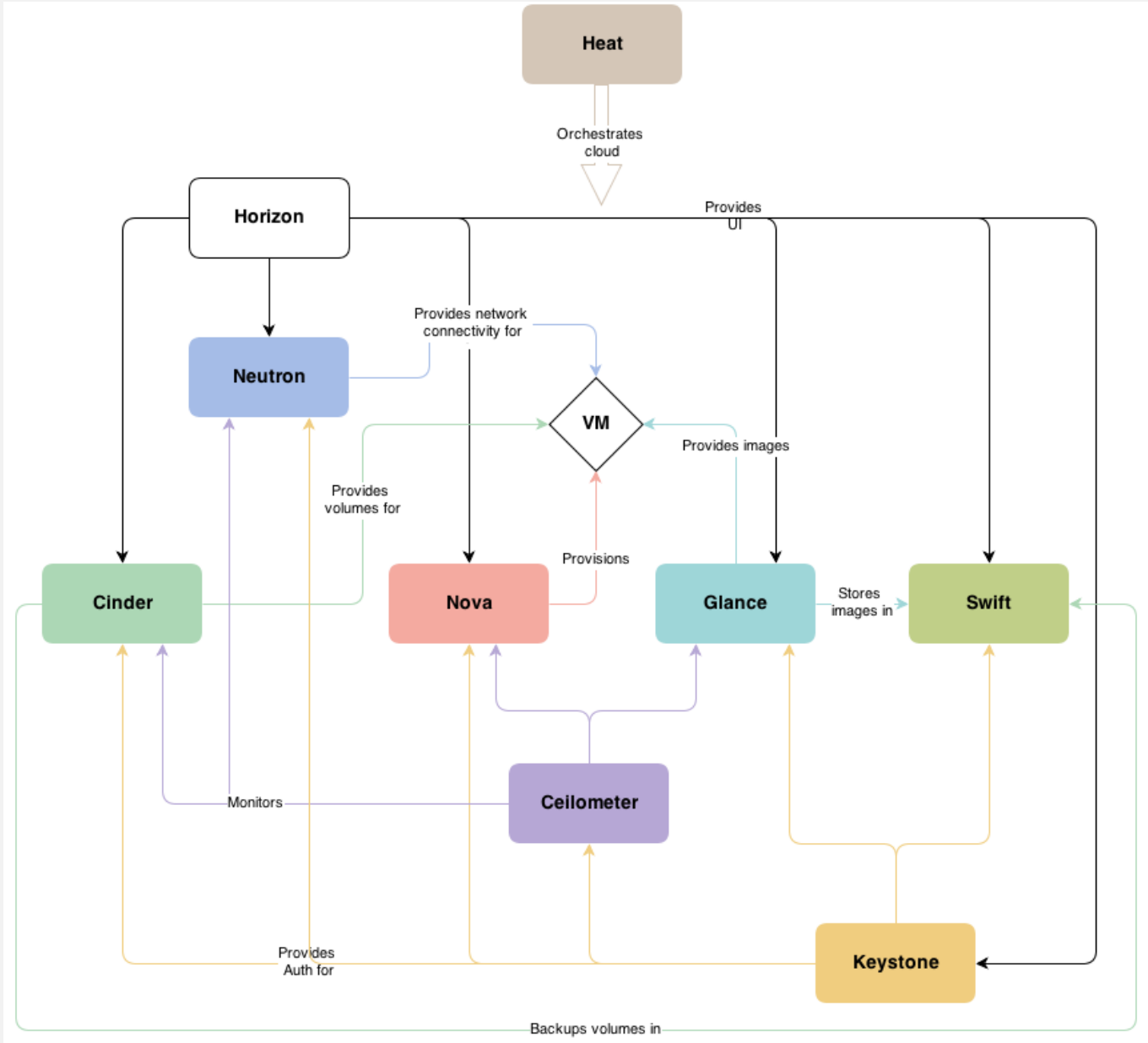
A device that is configured for LLDP operation sends PDUs on each of their interfaces where the protocol is enabled. The PDUs are sent at a fixed interval and are sent in the form of an Ethernet Frame or PDU.

Open stack

Úvod + Koncepční architektura

- Open stack =
  - Open source cloud platforma
  - On demand Self-service (multitenancy)
  - **Masivní** škálování, resource pooling
  - Přístup k síti pro cokoliv
  - Měřitelná služba – lze účtovat
  - Funkce rozděleny do mnoha služeb
  - Služby jsou propojené přes API (REST)
  - Pay as you grow (nekonečná kapacita)
  - (cca 1M Python řádků kódu)

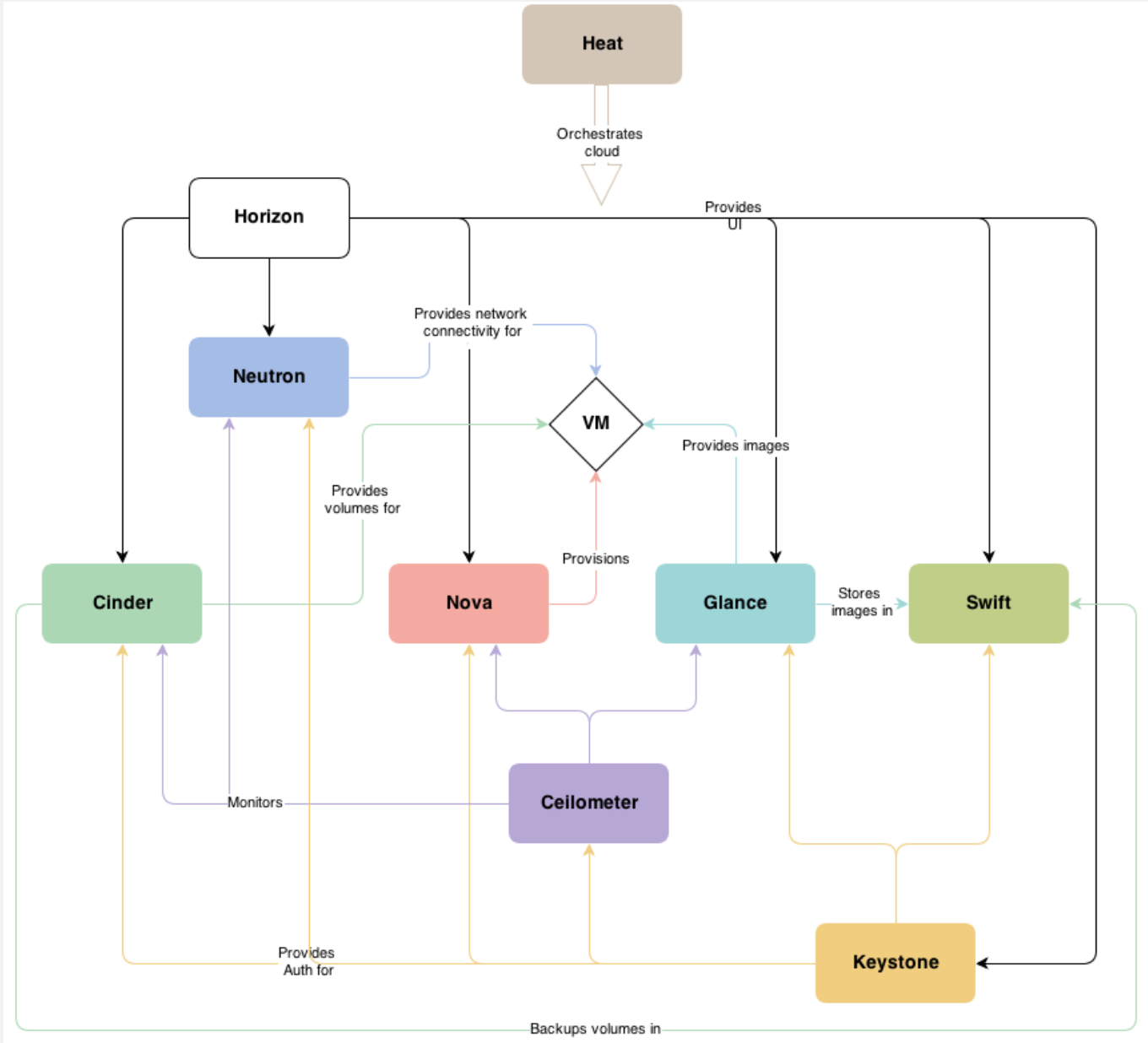
- Služby mají svá projektová jména
  - Příklad: Networking => Neutron



Úvod + Koncepční architektura

- Cloud poskytuje různé service modely:
  - SaaS – uživatel má přistup k softwaru v cloudu (web-based email, dropbox etc)
  - PaaS – uživatel může do cloudu nahrávat své vyvíjené aplikace, např platforma jako Java EE prostředí
  - IaaS – uživatel získává VMs, síťové propoje, storage atd – celou infrastrukturu ve větší či menší míry virtualizovanou

- Open Stack pracuje s 3 pilíři ICT:
  - Výpočetní výkon
  - Storage
  - Networking

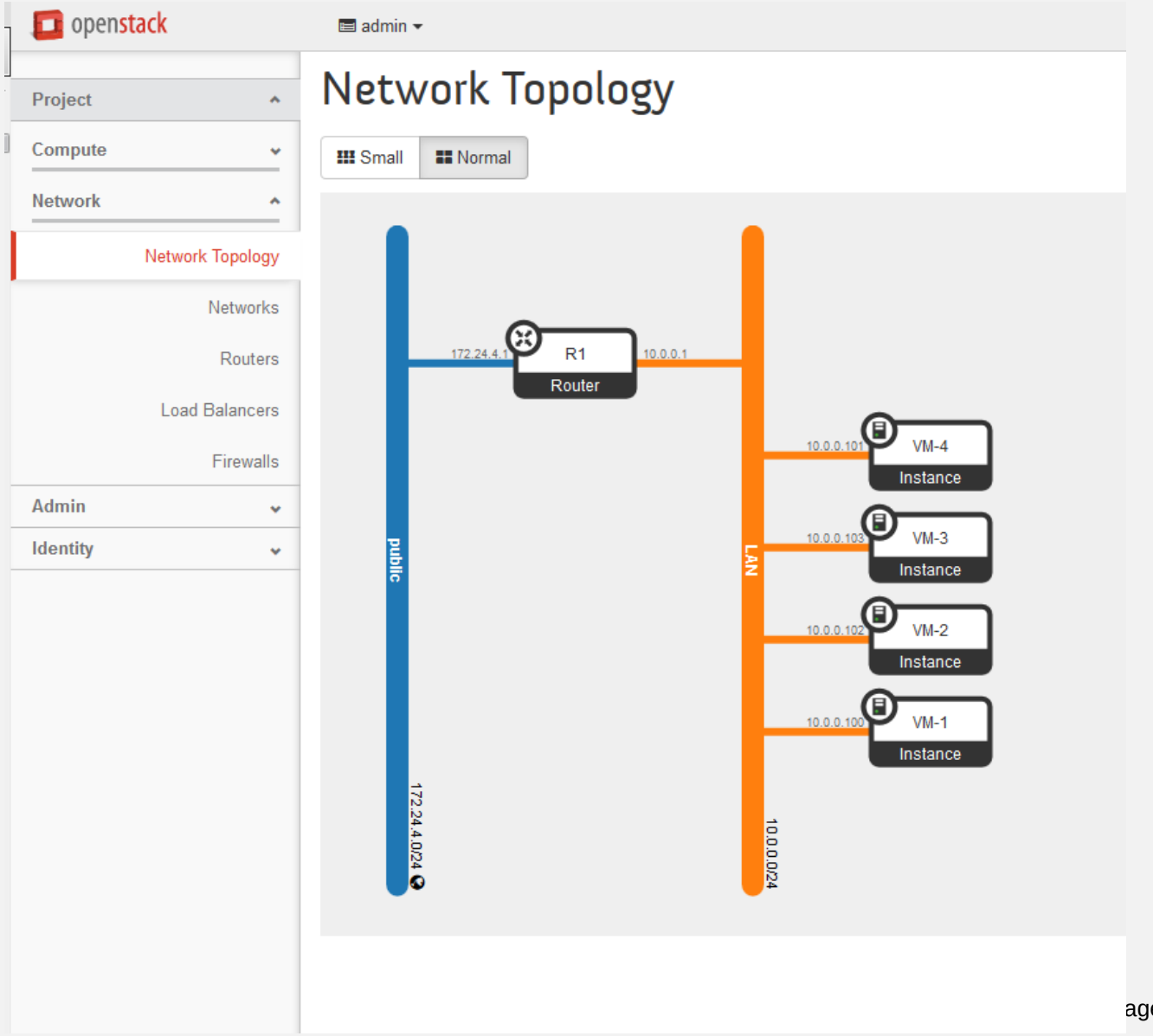


Uživatelé OpenStacku

- **NASA**
- **CERN**
- **NSA**
- **PayPal**
- **SUSE, RedHat..**
- **DT, AT&T…**
- **…..**

Horizon ~ dashboard

- Webové rozhraní
- Self-service portál
  - › Uživatel si může sám definovat
    - » Síť
    - » VM
    - » Pravidla
    - » …



Nova ~ Compute (hlavní součást IaaS systému)

- Nova-API – uživatelské akce
- Spravuje životní cyklus výpočetních instancí (VM) v OpenStack prostředí
- Pracuje (nova-compute) s hypervizory (KVM, VMWARE,Hyper-V, XEN,..dokonce i s barematel..)
  - Přijme pokyn z fronty akcí, zavolá systémové příkazy na tvorbu VM atd a u toho aktualizuje stavovou databázi

Nova ~ Compute (hlavní součást IaaS systému)

- Nova-scheduler – z fronty vezme požadavek a vybere na kterém HW má VM spustit
- Compute Node – na každém nodu běžící agent
- Různé VM příchutě
- Zodpovídá za:
  - Vytváření VM – kolik zdrojů (CPU, memory…)
  - Spoštění
  - Vypínání
  - …

Neutron ~ Networking

- Vše potřebuje IP adresu (staticky, DHCP,..)
- Routing, switching, security pravidla, VPN služby, firewall as a service, loadbalancer as a service,…
- Neutron – škálovatelný systém, veřejné API, self-service
- Vendor pluginy – neutron umí pracovat s infrastrukturou na od různých výrobců
- Agenti
- Díky API a pluginům lze neutron napojit na:
  - Běžný x86 HW
  - Switche, routery, firewally, loadbalancery

Neutron ~ Networking

- Plugin – open vSwitch
  - Bridge a porty
  - Br-int – připojuje VMs a příslušné služby
  - Br-ex – poskytuje připojení do externí sítě
  - Podporuje OpenFlow
- `Ovs-vsctl show`
- `Ovs-ofctl dump-flows br-tun`

Swift ~ Object Storage

- Ukládá nestrukturované datové objekty skrz RESTful API ~ soubory ☺
- Tyto DB používá např Facebook pro fotky, Dropbox apod.
- Swift řeší redundantní zápis na více míst v clusteru, v případě výpadku části řeší replikaci atd. - na běžném HW
- Poskytuje i přístup k datům přes API – pro aplikaci, pro backup, atd.
  - Object store API nebo HTTP

Cinder ~ Block Storage

- *Dotaz – Rozdíl NAS vs SAN? (block storage vs file storage)*
- Cinder-api – příjímá API volání a ty pak pustí na příslušný cinder-volume
- Cinder-volume
  - zapisuje do statovové databáze
  - Pracuje přes drivery s uložištěm (IBM, Solid Fire, Linux iSCSI,…)
- Cinder-scheduler – daemon vybírá optimální storage pro vytvoření volume.
- Bloková uložiště, lze přidělit VMs – nova
- Součásti:
  - Volumes
  - Snapshots
  - Storage backend

Keystone ~ Identity

- Poskytuje authentizační a autorizační službu pro ostatní služby
- Klíčové pojmy
  - Tenants („nájemníci “) – skupina uživatelů
  - Uživatelé
  - Role
  - Tokens a services

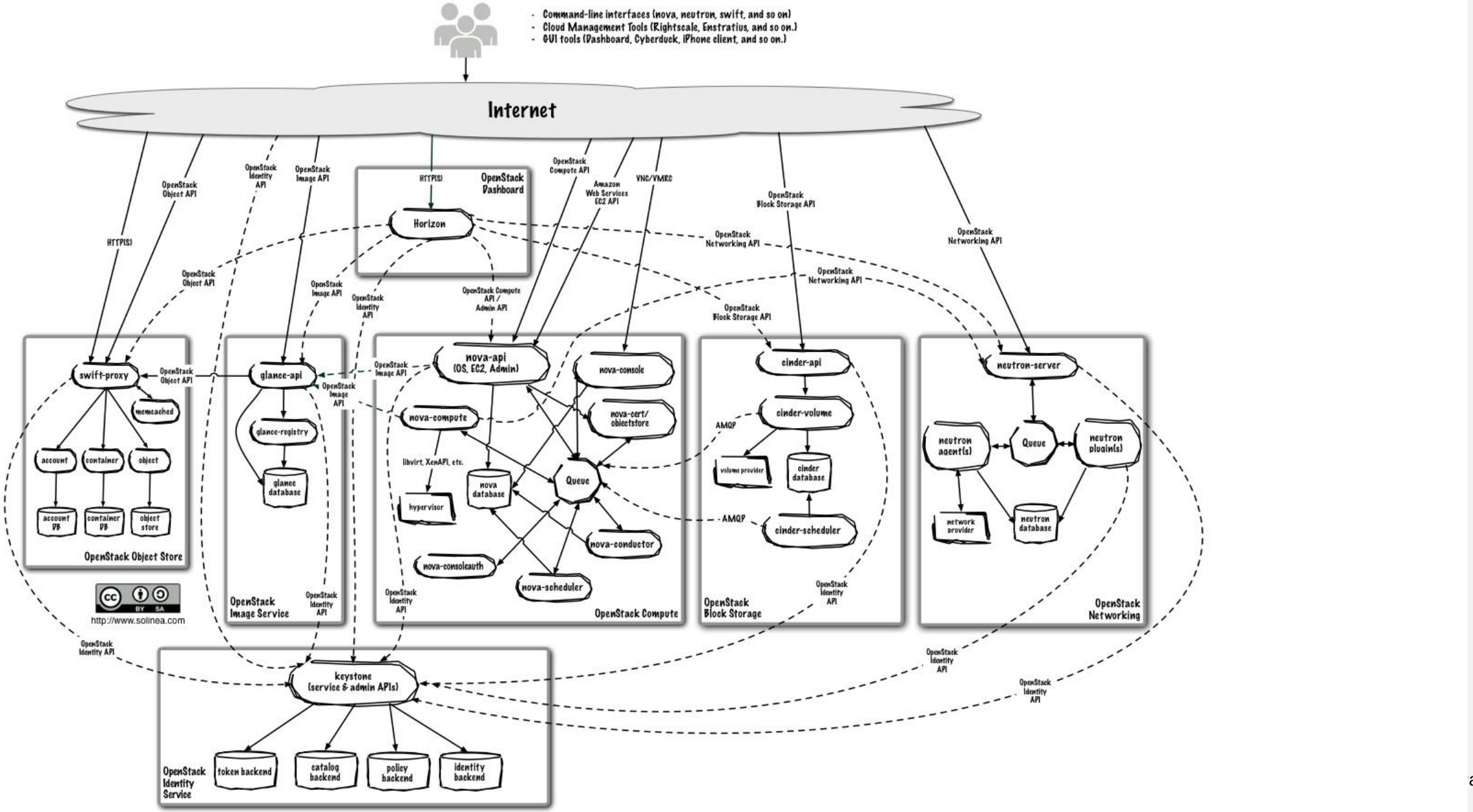
Glance ~ Image Service

- Služba, která shromažďuje image
- Různé formáty (Raw, VDI, QCOW,…)
- Součásti:
  - Image
  - Metadata
  - Storage backend (běžně swift)

Ceilometer ~ Metering

- Základ pro účtování za služby
- Součásti:
  - „měřáky “
  - Notifikace
  - „sběrače “ notifikací…
  - …
- A další a služby – Heat, Trove, ….

Logická architektura



Openstack na hraní

- **Devstack**
  - Z git repozitoře si stáhnete skripty a zdrojové soubory OpenStacku ve formátu připraveném pro jednoduché nainstalování
  - Konfigurační soubor local.conf – specifikace toho, jaké moduly se mají připravit
  - ./stack.sh – nainstaluje OpenStack - vše na jeden stroj – ideální na hraní a případně na vývoj

Příklad

- **Horizon**
  - Vytvořit LAN1 a LAN2
  - Tvorba routeru, který je propojí
  - Zkouška FW
  - Vyrobit si VMs – ping, vyrobit dirs, snapshot, obnovit…
    - CirrOS (malinky linux image – testovací účely ~ ping)
- **CLI**
  - Každá služba má své CLI příkazy (neutron, nova,…)

CLI

- Horizon – Project – Access & Security ..stahnout RC config
- Source RC
- Ukázky příkazů:
  - `glance image-list, glance image-show …`
  - `nova list`
  - `nova flavor-list`
  - `neutron subnet-list`
  - `nova boot --image a8168934-3162-416e-9dc5-219d99d22425 --flavor 42 --nic net-id=637fd788-198f-4e25-8084-5d2d1c928424 testserver`
  - `Nova reboot <server> => nova list`

CLI

```
› neutron subnet-list
› neutron net-create LAN3
› neutron subnet-create LAN3 13.0.0.0/24  --name LAN3
› neutron port-list
› neutron port-create net1 --fixed-ip ip_address=192.168.2.40
```

….dotazy diskuze …

Odkazy

- http://ilearnstack.com/2013/04/23/introduction-to-openstack-2/
- http://docs.openstack.org/admin-guide-cloud/content/ch_preface.html
- http://docs.openstack.org/icehouse/training-guides/content/
- http://mininet.org/
- https://www.openstack.org/assets/presentation-media/openvswitch-in-neutron-1.pdf
- https://youtu.be/hWWSaBOMTNo

- **…google : SDN, Openstack…**