Централизирана маршрутизация

Софтуерно дефинирани мрежи.
Виртуализация на мрежовите функции.

Централизирани адаптивни алгоритми

При централизираните адаптивни алгоритми в мрежата се създава един маршрутен управляващ център.

Той изчислява маршрутните таблици на всички възли и им ги изпраща.

За да се адаптират маршрутните таблици към текущата топология и текущия трафик, всички възли трябва да изпращат информация към маршрутния център.

На базата на получените сведения, маршрутният център изчислява теглата на ребрата и след това пресмята оптималният маршрут между всеки два възела.

Добре е да се поддържат алтернативни пътища между възлите.

Централизирани адаптивни алгоритми

Информацията от по-близките до маршрутния център възли ще пристигне по-бързо отколкото от по-далечните.

Поради това периодът на обновяване на маршрутните таблици трябва да е поне два пъти по-голям от времето за преминаване на пакет от маршрутния център до най-отдалечения от него възел.

Преизчислена маршрутна таблица, получена в един възел, не трябва да се използва веднага, тъй като маршрутните таблици пристигат по различно време в различните възли.

Централизирани адаптивни алгоритми

Ако по някаква причина маршрутният център отпадне, мрежата остава без управление.

За целта може да се дублира маршрутният център, но тогава служебният трафик би се увеличил твърде много.

...отново централизирана. SDN и OpenFlow

Системите преминаха:

- централизирани - големи машини (maiframes);

- разпределени – PCs;

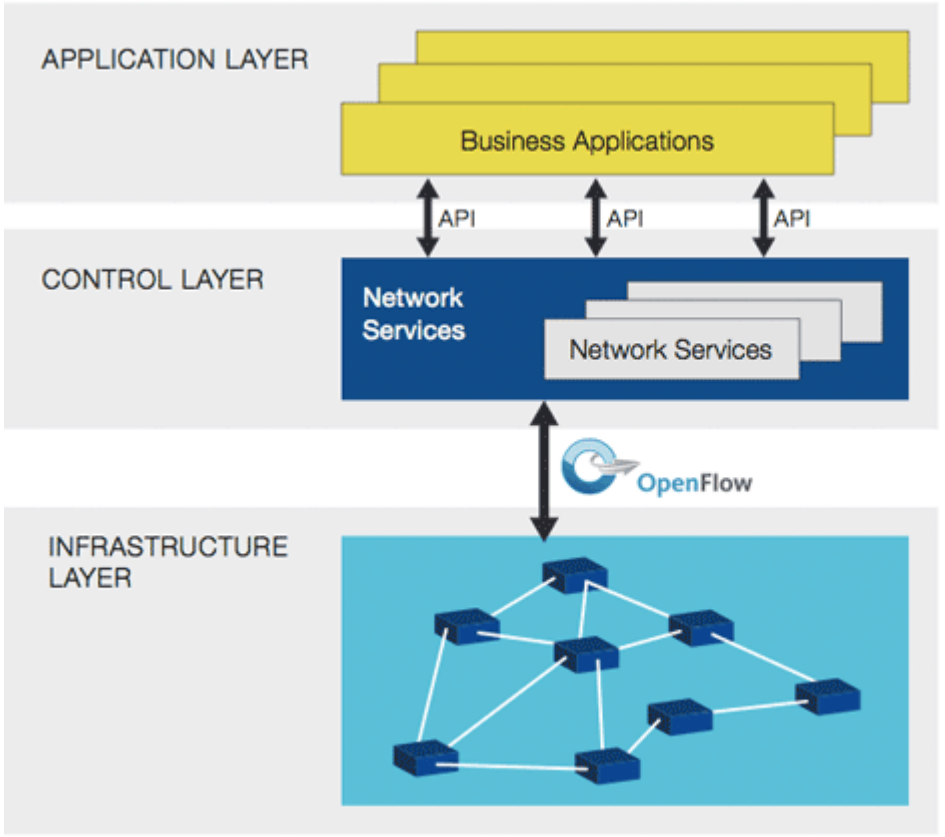
- пак централизирани – VMs, Cloud.

Същото и с мрежите. Днес Cloud Networking Services (CNS):

- SaaS и/или

- IaaS и/или NaaS.

Open standards-based and vendor-neutral

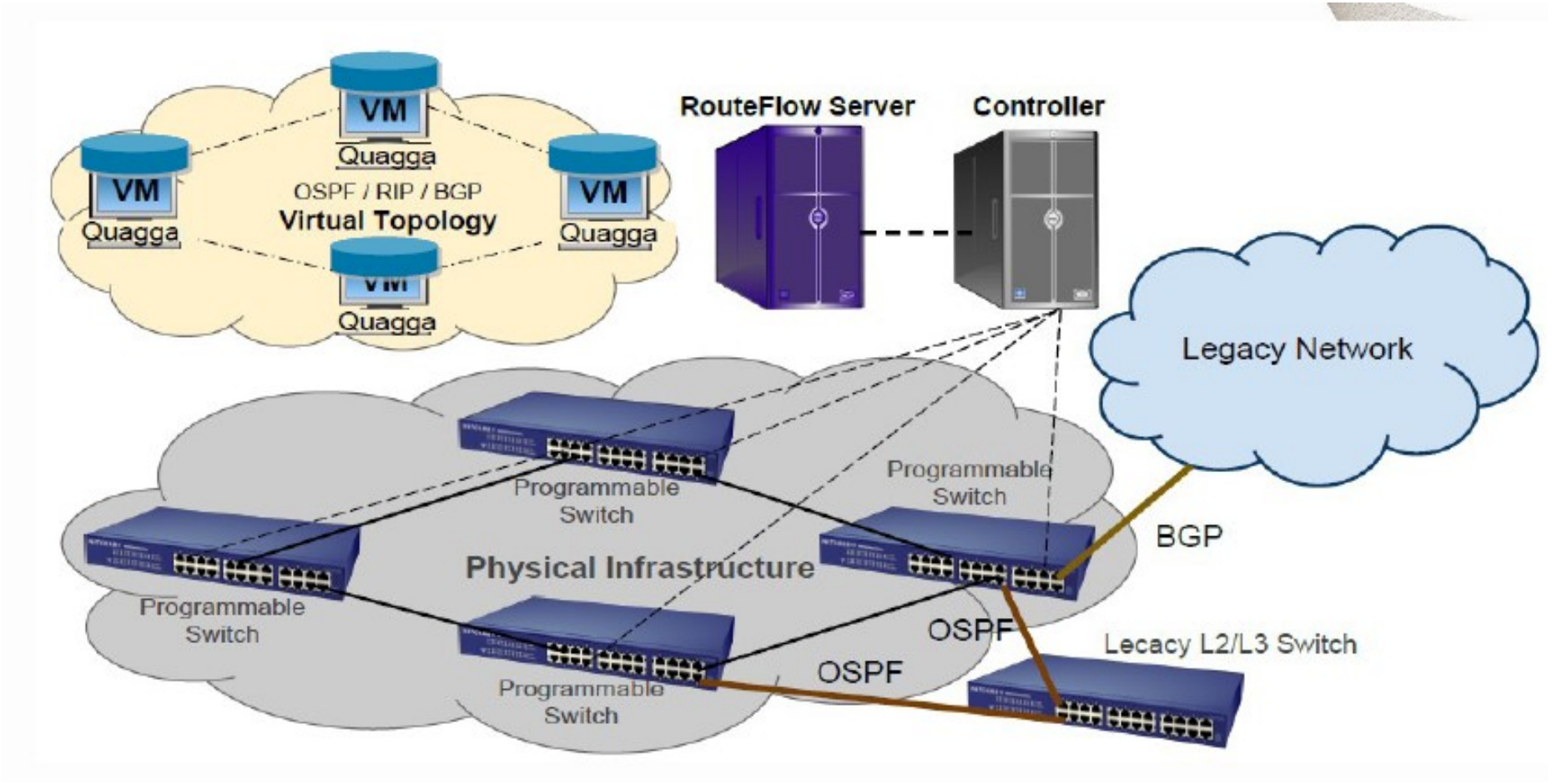


Проектът OpenFlow (www.openflow.org)

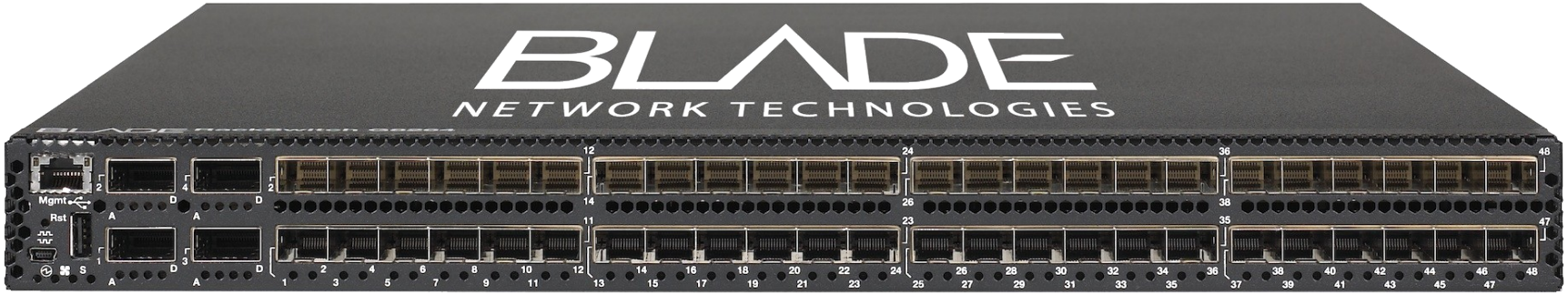
OpenFlow е в основата на software-defined networking (**SDN**). Потребителите дефинират потоците данни и пътищата им независимо от инфраструктурата под тях – маршрутизатори и комутатори.

Проект с отворен код – сътрудничество между Stanford University и University of California at Berkeley.

RouteFlow (QuagFlow)



OpenFlow суичове. IBM G8264.



IBM OpenFlow суич G8264:

48 × 10 GbE SFP+ порта и

4 × 40 GbE QSFP+.

OpenFlow суичове. HP 8200 zl.

Освен това:

HP 6600, HP 6200-24G-mGBIC, HP 5400 и HP 3500

RouteFlow (QuagFlow)

Параметри:

- open-source протоколни стекове (напр. Quagga/FRR)

- комерсиален мрежов хардуер с отворени API-та (напр. OpenFlow суичове от HP, IBM)

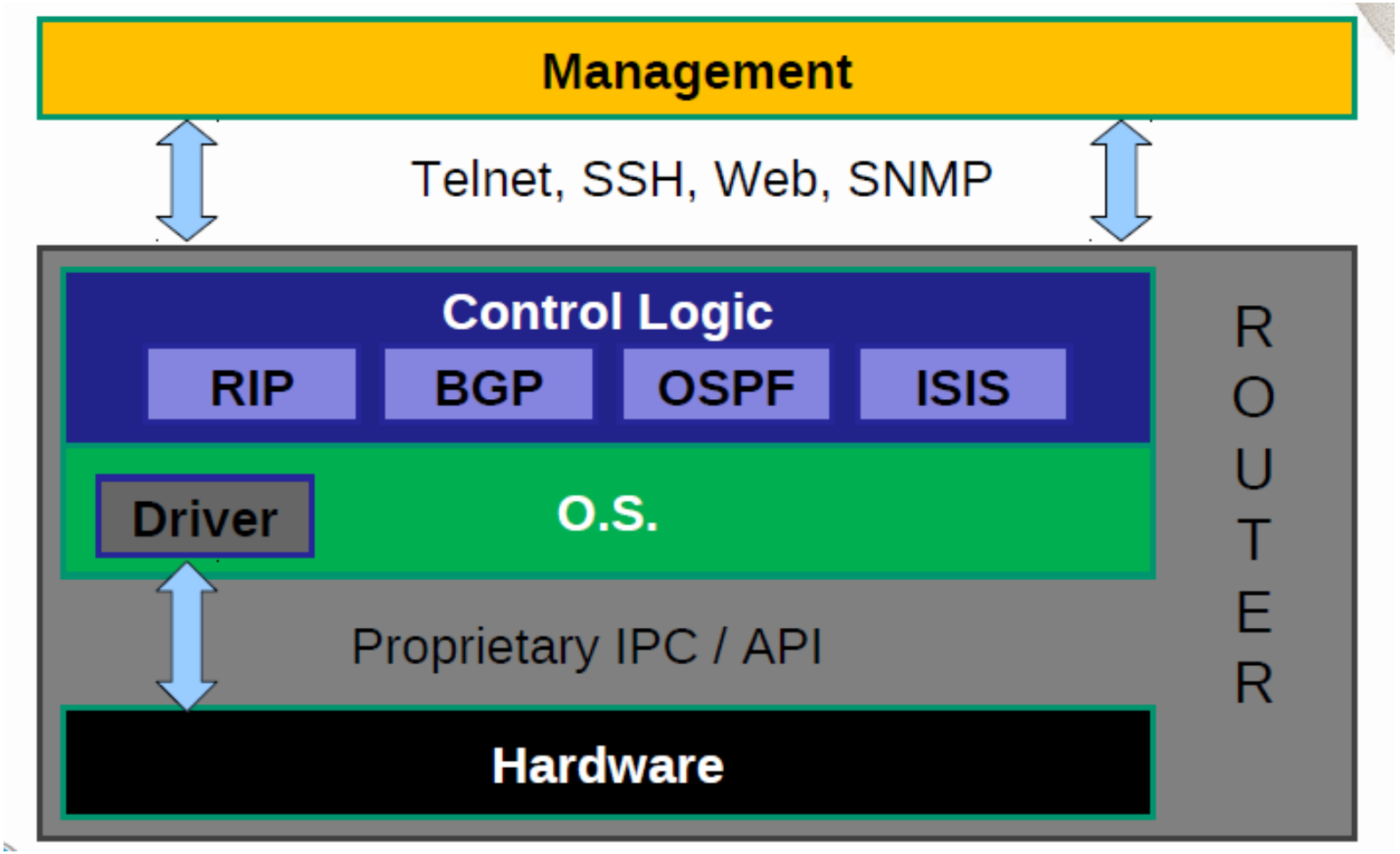
Постига се:

- производителност, съизмерима със скоростта на линията;

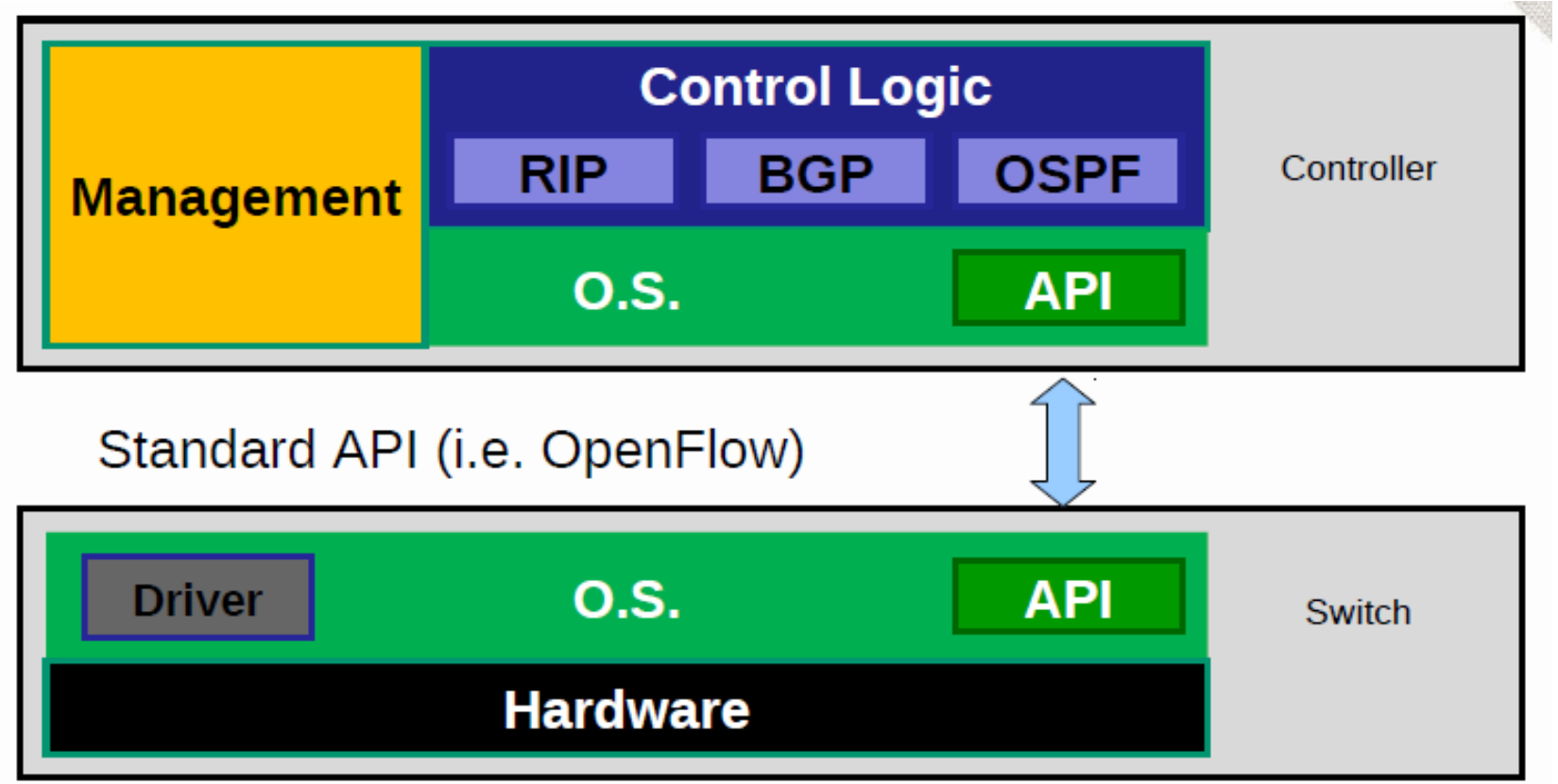
- оптимално съотношение цена/производителност;

- гъвкавост.

Архитектура на маршрутизатор (сега)



OpenFlow модел. Разделяне на Control (Management) Plane от Data Plane



Open Networking Foundation

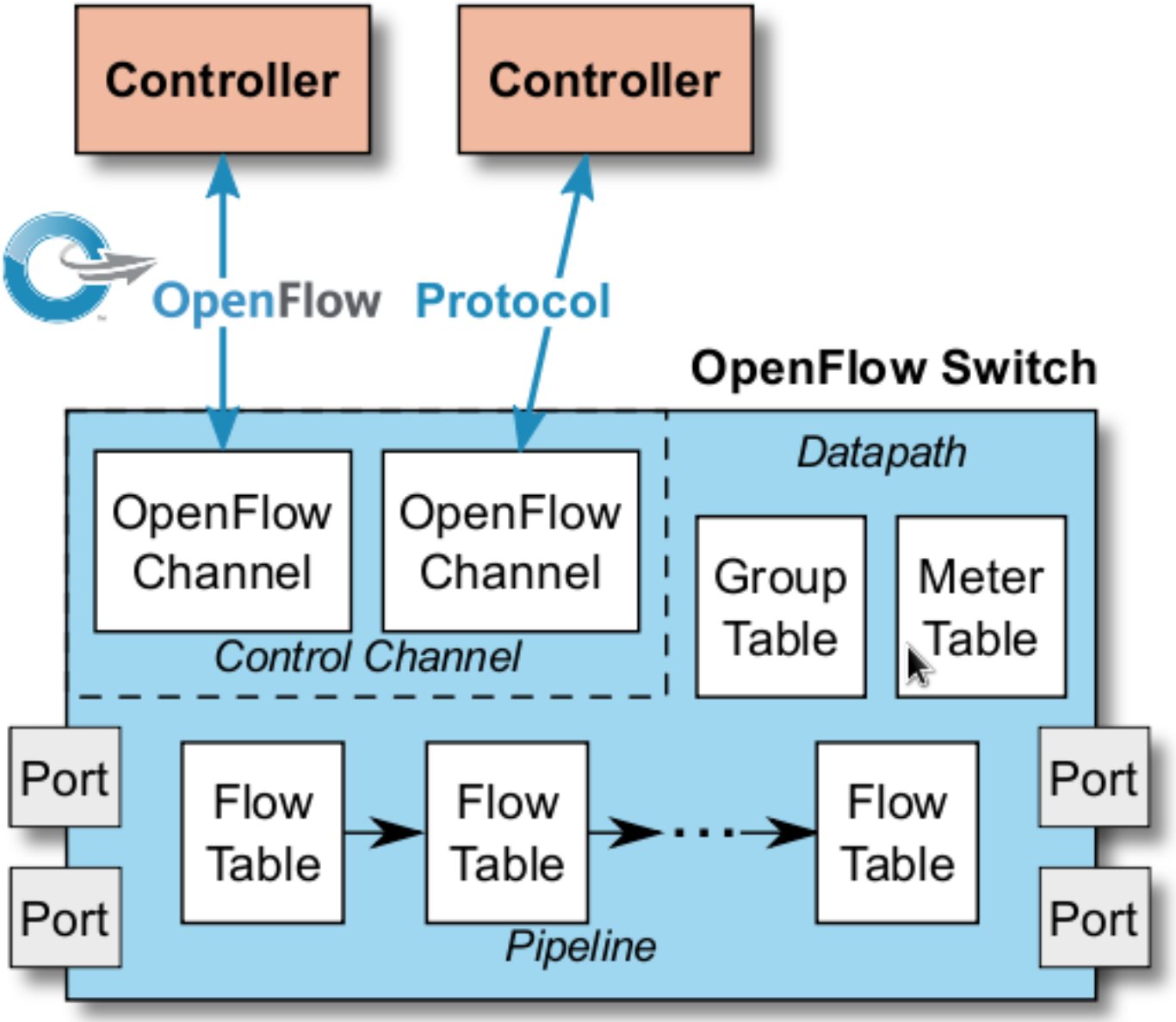
Версия 1.1 на OpenFlow - на 28.02.2011, поддържана от openflow.org.

Развитието на стандарта се поема от Open Networking Foundation (ONF).

www.opennetworking.org

В момента версията е: OpenFlow v. 1.5.1 .

OpenFlow v.1.5.1 суич



OpenFlow суич. Параметри и протоколи.

OpenFlow Logical Switch съдържа една или повече потокови таблици (**flow tables**) и една групова таблица (**group table**), които идентифицират и пренасочват пакетите. Суичът е свързан към външен контролер, който го управлява посредством **OpenFlow switch protocol**.

Контролерът добавя, обновява и изтрива записи в потоковите таблици при промени в поведението на даден пакет/и или по алгоритъм.

Всяка потокова таблица съдържа потокови записи, които се състоят от полета на съвпадението (match fields), броячи и набор от инструкции, които се прилагат към съвпадащите пакети.

Записите в таблицата с метрика (**meter table**) дефинират метрики на даден поток. С помощта на метриките OpenFlow прилага ограничения на скоростта, QoS, приоритети на трафиците.

Вендори на SDN контролери

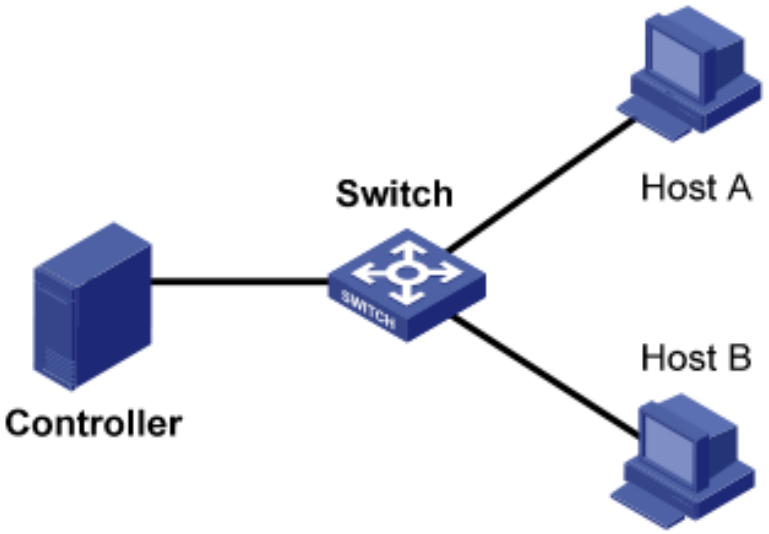
SDN контролерът е софтуер, за Windows и/или Linux.

По-известни вендори са Cisco, Cumulus Networks, Hewlett Packard Enterprise, Juniper Networks, Vmware и др.

Известни open source продукти са:

- Floodlight (https://floodlight.atlassian.net/)
- OpenDaylight (https://www.opendaylight.org/)
- OpenContrail (http://www.opencontrail.org/)
- ONOS (Open Network Operating System) https://www.opennetworking.org/onos/
- RYU (https://ryu-sdn.org/)

Активиране на OpenFlow пример на суич HP 59XX



Създавате VLANs 4092 и 4094.

<Switch> system-view

[Switch] vlan 4092

[Switch-vlan4092] quit

[Switch] vlan 4094

[Switch-vlan4092] quit

Създавате OpenFlow instance (пример) 1 и го обвързвате с VLANs.

[Switch] openflow instance 1

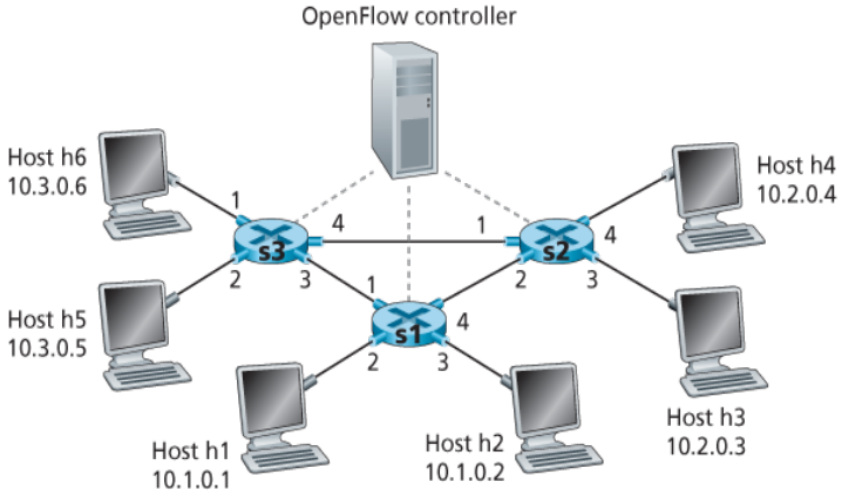
[Switch-of-inst-1] classification vlan 4092 mask 4093

Определяте кой е контролера за OpenFlow instance и активирате instance.

[Switch-of-inst-1] controller 1 address ip 192.168.49.49

[Switch-of-inst-1] active instance

Определяне на път на пакети с OpenFlow - пример



Имаме мрежа с 6 хоста (h1 - h6) и три пакетни комутатота (суичове) (s1, s2 и s3), всеки с по 4 интерфейса (1 - 4). Ще определим поведението на мрежата, като зададем съответните редове в поточните таблици (flow table) за суичовете s1, s2 и s3.

Ред в потоковата таблица на s3

Задаваме пакетите от h5 или h6, които са предназначени за h3 или h4, да бъдат пренасочвани от s3 към s1, а след това - от s1 към s2 (така заобикалят връзката между s3 и s2).

Редът в **потоковата таблица на s3** показва, че пакетите, които са изпратени от h5 или h6 се пренасочват към s1 през изходен интерфейс 3:

Съвпадение (Match)	**Действие (Action)**
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)
...	...

Ред в потоковата таблица на s1

Редът в **потоковата таблица на s1** показва, че пакетите, които са изпратени от h5 или h6 се пренасочват към s2 през изходен интерфейс 4:

Съвпадение (Match)	Действие (Action)
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)
...	...

Ред в потоковата таблица на s2

И накрая ще ни трябват и **редове в потоковата таблица на s2**, които да насочват пакетите, идващи от s1, към крайната им дестинация - h3 или h4:

Съвпадение (Match)	Действие (Action)
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)
...	...

SDN контролер RYU

Ryu е SDN контролер с отворен код под лицензa на Apache 2.0. Осигурява лесна за работа развойна среда, в която програмистите създават нови програми за мрежово управление.

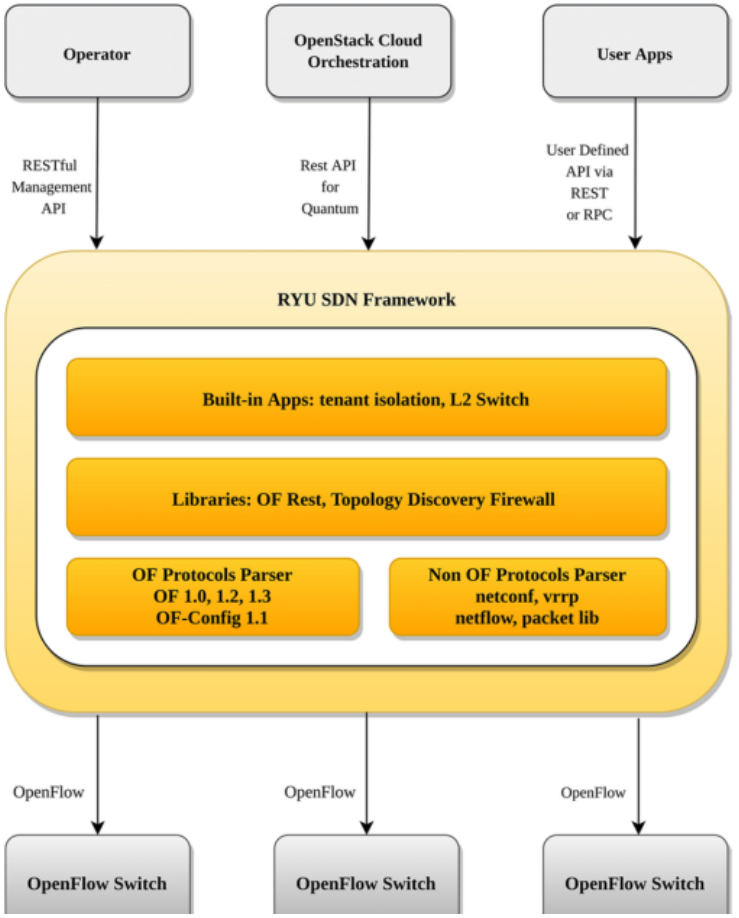
Разработен е от Nippon Telegraph and Telephone (NTT). На японски език **Ryu** означава **поток**.

RYU е програмиран на **Python** и поддържа протоколи като OpenFlow, NETCONF, OF-config и др.

Идеята е Ryu да управлява потока от трафик и да изгражда интелигентни мрежи.

Ryu предоставя софтуерни компоненти с ясно дефинирани приложно-програмни интерфейси, които улесняват създаването на нови мрежови програми за управление и контрол.

Архитектура на RYU



Архитектура на RYU

RYU е с 3-слойна архитектура.

В най-горният, приложен слой, са програмите, реализиращи бизнес и мрежова логика.

На средният слой, за управление, са мрежовите услуги. Включва "**northbound**" (нагоре) APIs (Restful management API, REST, API for Quantum, User Defined API via REST or RPC) и "**southbound**" (надолу) APIs, които поддържат протоколи като OpenFlow, OF-config, NETCONF и др.

Ryu осигурява и компоненти като OpenStack Quantum, Firewall, OFREST и др.

В програмите се използват алгоритми, извършващи анализ на мрежата и имплементация на нови правила за работа ѝ на ниво контролер.

Ryu работи с няколко вида OpenFlow комутатори като OpenvSwitch и други, произведени от Centec, HP, IBM и NEC.

Най-долният слой се състои от физически и виртуални устройства и е познат като инфраструктурен слой.

Виртуализация на мрежовите функции. NFV vs. VNF

или **NFV** – Network Functions Virtualization

Според European Telecommunications Standards Institute (**ETSI**):

... прилага стандартни технологии за виртуализация,

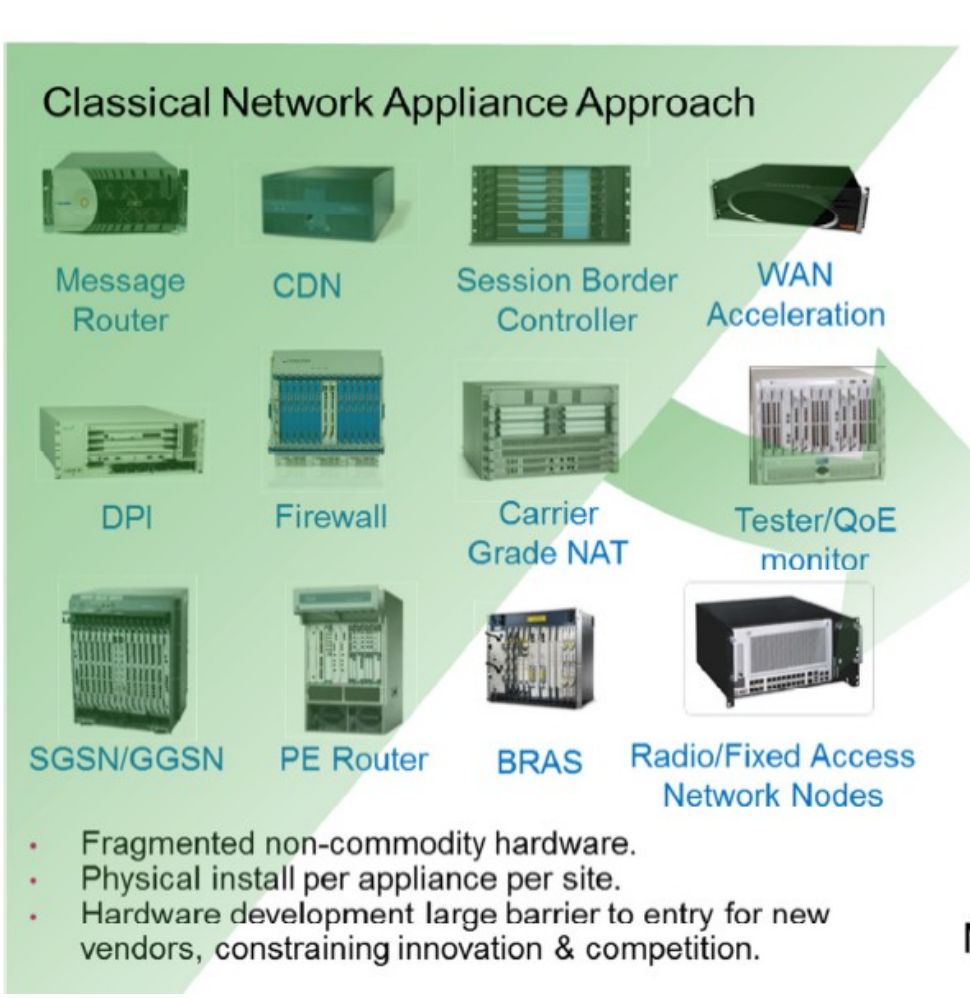
Консолидира различни видове мрежово оборудване върху стандартни и мощни сървъри, комутатори и устройства за съхранение.

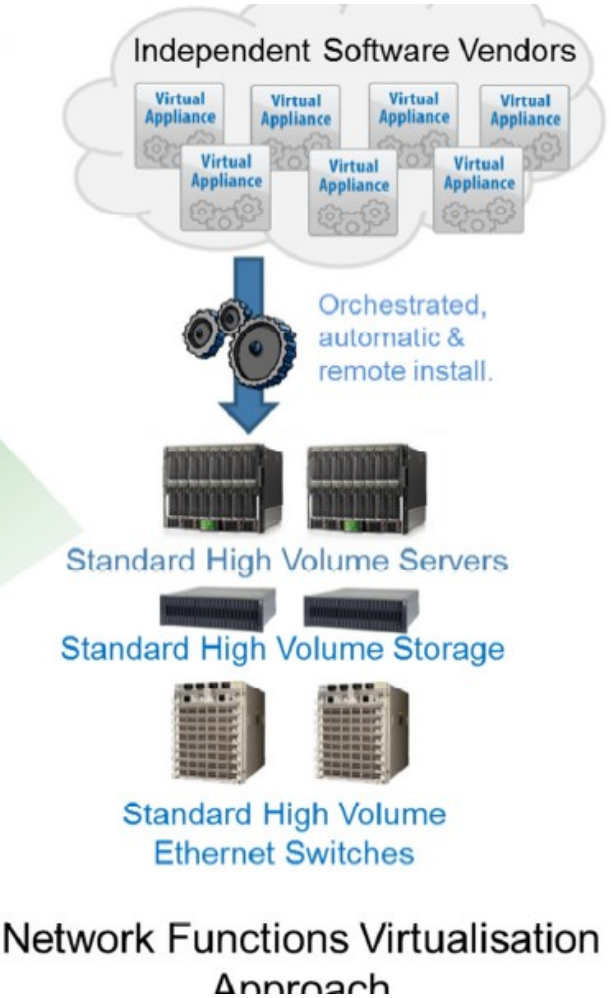
NFV реализира мрежовите функции в софтуера, работещ на стандартни сървъри

Този софтуер може “динамично” да се “премества” или да се репликира в различни примери на различни места в мрежата, без да се налага инсталиране на ново оборудване.

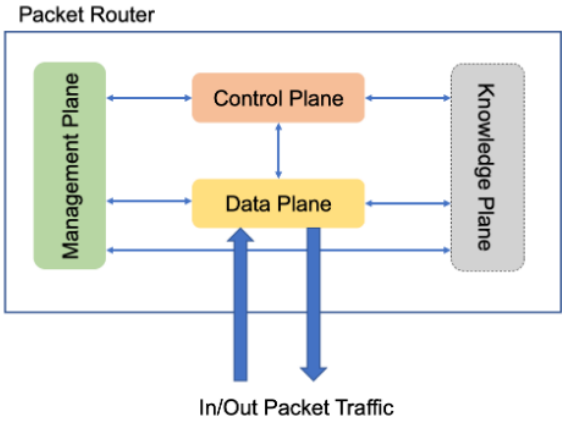
Virtual network functions (**VNFs**) са елементи от NFV пъзела, включително технологии за оркестрация и мениджмънт.

Идеята за NFV





Добавяне на равнина на знанието в концептуалния модел на рутера



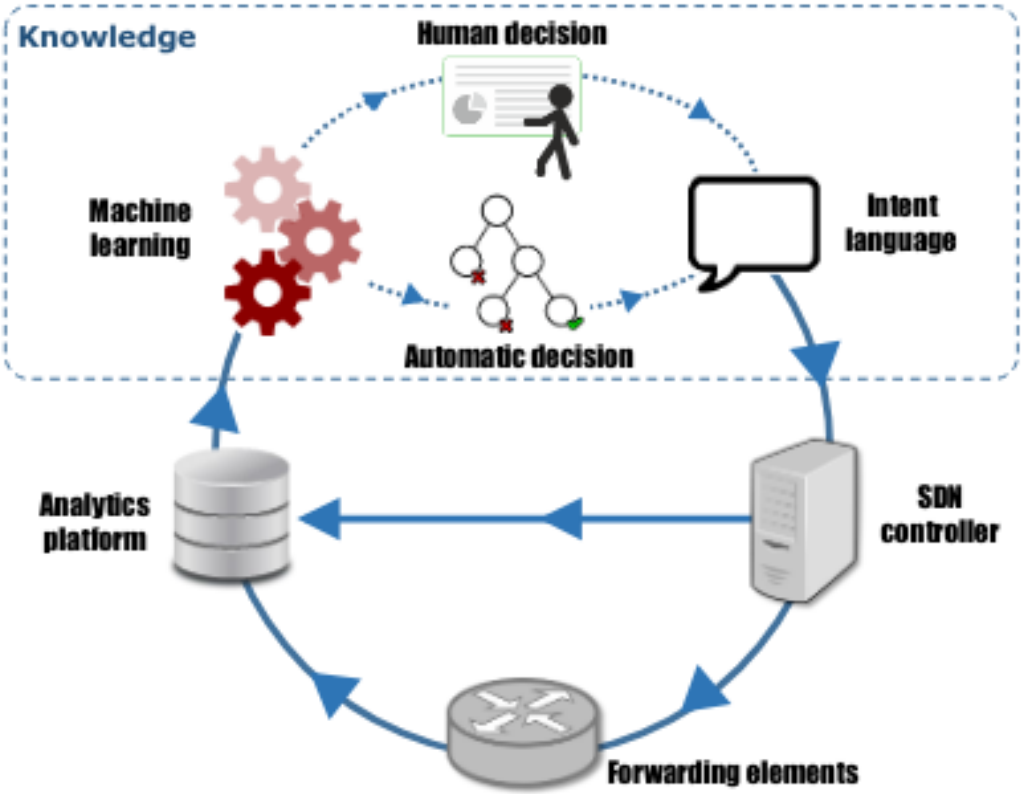
С развитието на машинното обучение и анализите в областта на компютърни мрежи беше предложена четвърта равнина - равнината на знанието.

Потенциалните приложения: откриване и справяне с неизправности, динамично и автоматично (пре)конфигуриране на мрежови устройства и откриване на злонамерени проникванията (intrusion detection).

SDN → KDN

Базирана на знания мрежа (Knowledge-Defined Networking - KDN) включва архитектури, които съчетават принципите на софтуерно дефинираните мрежи - SDN, машинно обучение и аналитични платформи за реализиране на затворен и отворен цикъл на контрол на компютърните мрежи.

KDN като кръгов процес



Литературни източници

Applications of Artificial Intelligence, Machine Learning and related techniques for Computer Networking Systems

https://arxiv.org/pdf/2105.15103.pdf

Knowledge-Defined Networking

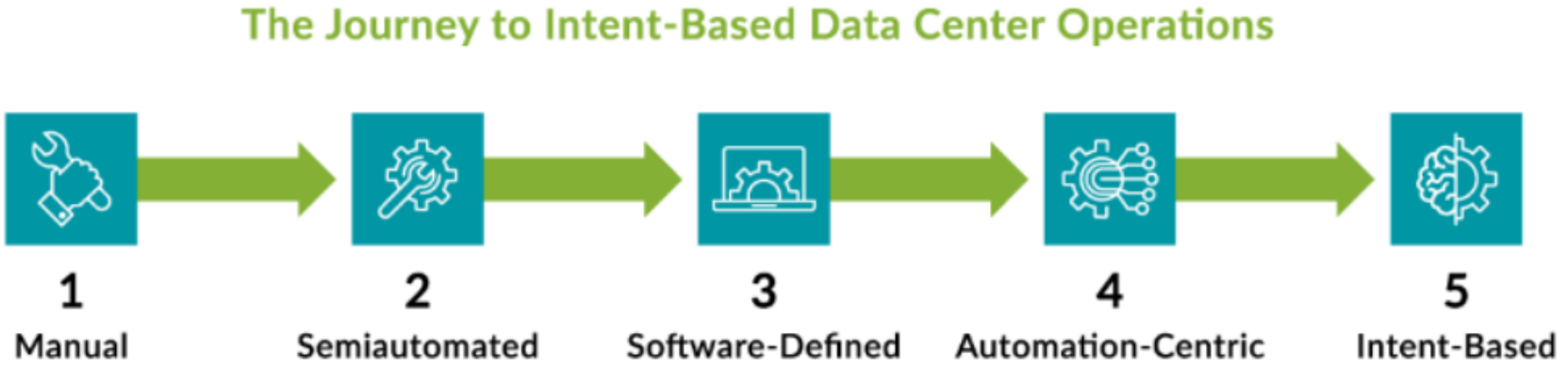
https://arxiv.org/pdf/1606.06222.pdf

A Knowledge Plane for the Internet

https://groups.csail.mit.edu/ana/Publications/PubPDFs/A%20knowlege%20plane%20for%20the%20internet.pdf

Manual --> SDN --> Intent-Based Networking

https://www.juniper.net/us/en/research-topics/what-is-intent-based-networking.html



Ръчно – Администраторите менижират мрежовите устройства чрез CLI, SNMP, Web и др. инструменти.

Полуавтоматизирано – Създават се скриптове, използващи традиционните средства, изброени по-горе.

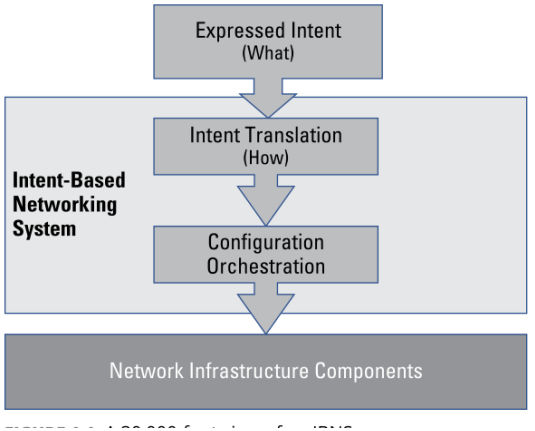
SDN – Софтуерна абстракция на мрежата.

Напълно автоматизирано управление – Изгражда се върху SDN, като автоматизира процесите на доставяне, конфигуриране, внедряване и оркестрация.

Intent-Based Networking (**IBN**)- Мрежа, базирана на намеренията) – Непрекъснато събира и преобразува всички подходящи данни, необходими за предприемане на автоматизирани действия, които поддържат работата на мрежата съгласно приетите политики.

IBN, по-подробно

https://www.juniper.net/content/dam/www/assets/ebooks/us/en/intent-based-networking-for-dummies.pdf



IBN е метод за автоматизиране на администрирането на мрежата, което включва изкуствен интелект (**AI**), **оркестрация** и машинно обучение (**ML**).

То силно се различава от декларативното програмиране, където казваме на системата какво искаме да прави (или в какво състояние искаме да бъде), т.е каква е политиката. Но не казваме на системата как да го направи.

Класически пример на декларативното програмиране е конфигурация на мрежово устройство като рутер, работещ с протокол OSPF:

Какво е оркестрация
(за сведение)

В системната администрация под **оркестрация** разбираме автоматизиране на конфигурирането, координацията и управлението на компютърни системи и софтуер.

Съществуват редица инструменти за автоматизиране на конфигурирането и управлението на сървъри и др. системни и мрежови устройства като Ansible, Kickstart, Puppet, Salt, Terraform] и AWS CloudFormation.

От конфигурация…

https://www.sinog.si/wp-content/uploads/2019/05/Intent-Based-Networking-SINOG6.pdf

```
router ospf 1
router-id 192.168.0.2
!
interface Loopback0
ip address 192.168.0.2 255.255.255.255
ip ospf 1 area 0
```

С това показваме намеренията си, **какво да прави** рутера, но **не и - как** да го направи.

Всяка конфигурация е **структура от данни**, описваща вашето **намерение**, в повечето случаи маскирана от съответния синтаксис.

Абстракция на модела на данни

Всичко трябва да се направи възможно най-просто:

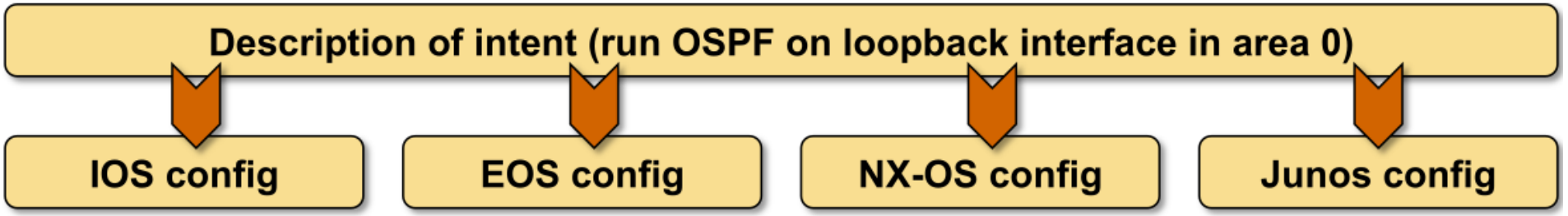
- Структуриране на данните в обекти, вместо да се използват твърде много скаларни променливи;
- Премахване на излишъци в модела на данни;
- Използване на прости термини в моделите на данни и бизнес логиката.

Примерен абстрактен модел на данни за устройство

```
hostname: E1
ospf:
  process_id: 1
  router_id: 192.168.0.2
  interfaces:
   "Loopback 0":
     area: 0
     Cost: 10
```

Описва конфигурацията на устройство в по-абстрактни термини и се използва за генериране на конфигурация без претрупан синтаксис.

Използване на такъв унифициран модел на данни:



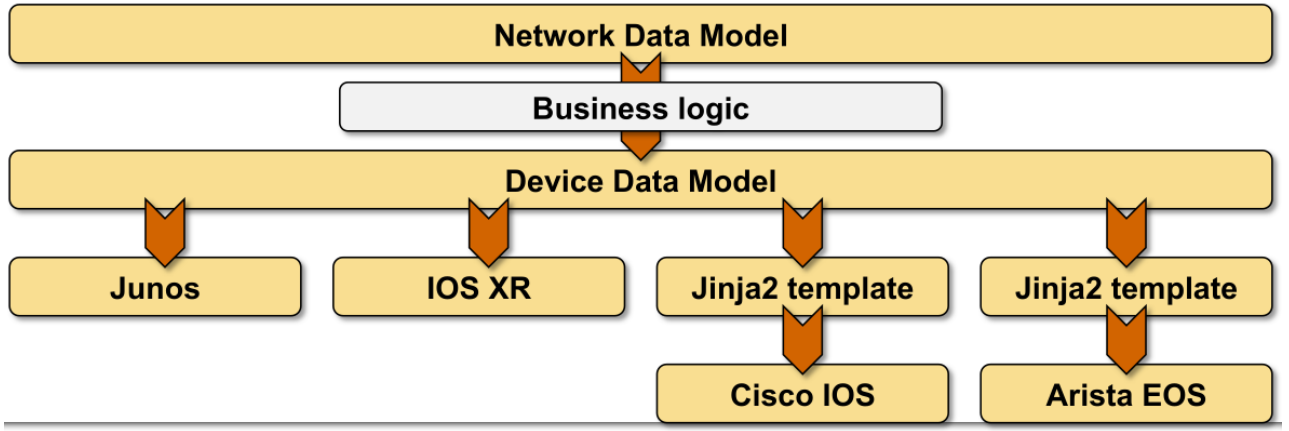
Модел на данни за цяла мрежа

```
fabric:
  - {left: E1, left_ip: 10.0.0.21, left_port: GigabitEthernet0/2,
    right: E2, right_ip: 10.0.0.22, right_port: GigabitEthernet0/2,
    cost: 5 }
  - {left: E1, left_ip: 10.0.0.13, left_port: GigabitEthernet0/1,
    right: PE1, right_ip: 10.0.0.14, right_port:
GigabitEthernet0/1,
    cost: 10 }
```

Описание на желаното състояние в мрежата, а не в отделни устройства:

- Възли и връзки
- Опорни и периферни връзки
- Подмрежи вместо IP адреси

Приложение на модела на данни за цяла мрежа. Бизнес логика.



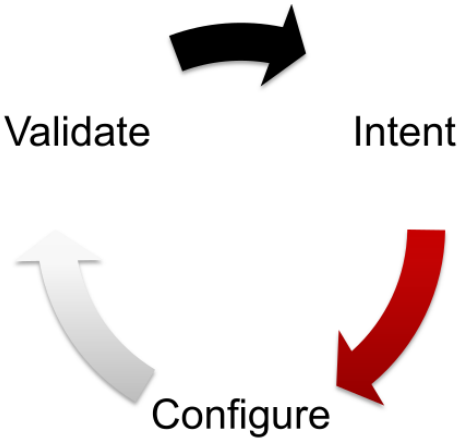
- Автоматично разпределение на IPv4 подмрежи
- Изчисляване на IPv4 адреси на интерфейси
- Избор на протокол за маршрутизация и задаване на параметрите му
- Създаване на списък с BGP съседи

Опростяване на модела на данни на мрежата

```
num_spines: 2
num_leaves: 4
hosts_per_leaf: 1
spine_to_leaf_ports: "swp[1-4]“
leaf_to_spine_ports: "swp51 swp52"
leaf_to_server_ports: "swp[1-2]"
```

- Определяне на броя на суичовете в мрежата
- Определяне на свързаността (обхвати на портове)
- Бизнес логиката създава всички данни за конфигурация на мрежата (**spine-leaf architecture** – дървовидна структура)

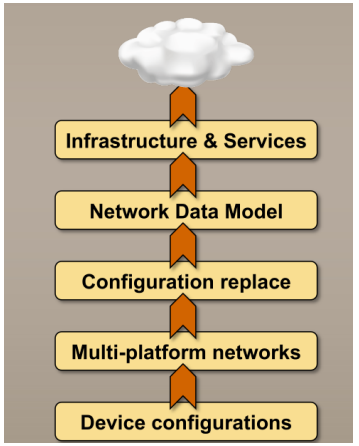
Използване на модела на данни на мрежата за валидиране



- Всички интерфейси работят ли?
- Правилна ли е L2 топологията на мрежата
- Установени ли са OSPF и/или BGP съседства?
- Правилните маршрути разпространяват ли се в мрежата?
- Успешни ли са тестовете за свързване от край до край?

Всичко това по-лесно и по-предвидимо се установява чрез използване на мрежов модел

Разделяне на модела на данни на инфраструктурен и за услуги



Инфраструктурният модел на данни дефинира стабилна транспортна мрежа: топология, рядко случващи се промени, рядко се нуждае от обслужване и промени.

Моделът на данни за услугите е насочен към клиента:

- Дефинира услугите, доставяни по транспортната мрежа, която приемемаме, че е стабилна;
- Услугите непрекъснато се променят, като се нуждаят от автоматизирано обслужване и внедряване.

Автоматизирано възстановяване на мрежата. Затваряне на цикъла.

Мрежите сами се поправят и/или адаптират към промените.

Автоматизиране на управлението на събития:

Откриване на промени в мрежата, като се събира информация от логове (Syslog, телеметрия), валидиране на данните, откриване на корелации в телеметричните данни с цел констатиране на проблеми.

Но да не се очакват чудеса от машинното обучение.

Необходимо е постоянно наблюдение и събиране на данни: поточна телеметрия (Streaming telemetry).

Сами решавате при кои събития си струва да действате.

Един добър пример: Facebook-Defined Networking