Skoltech

Skolkovo Institute of Science and Technology

Lomonosov Moscow State University

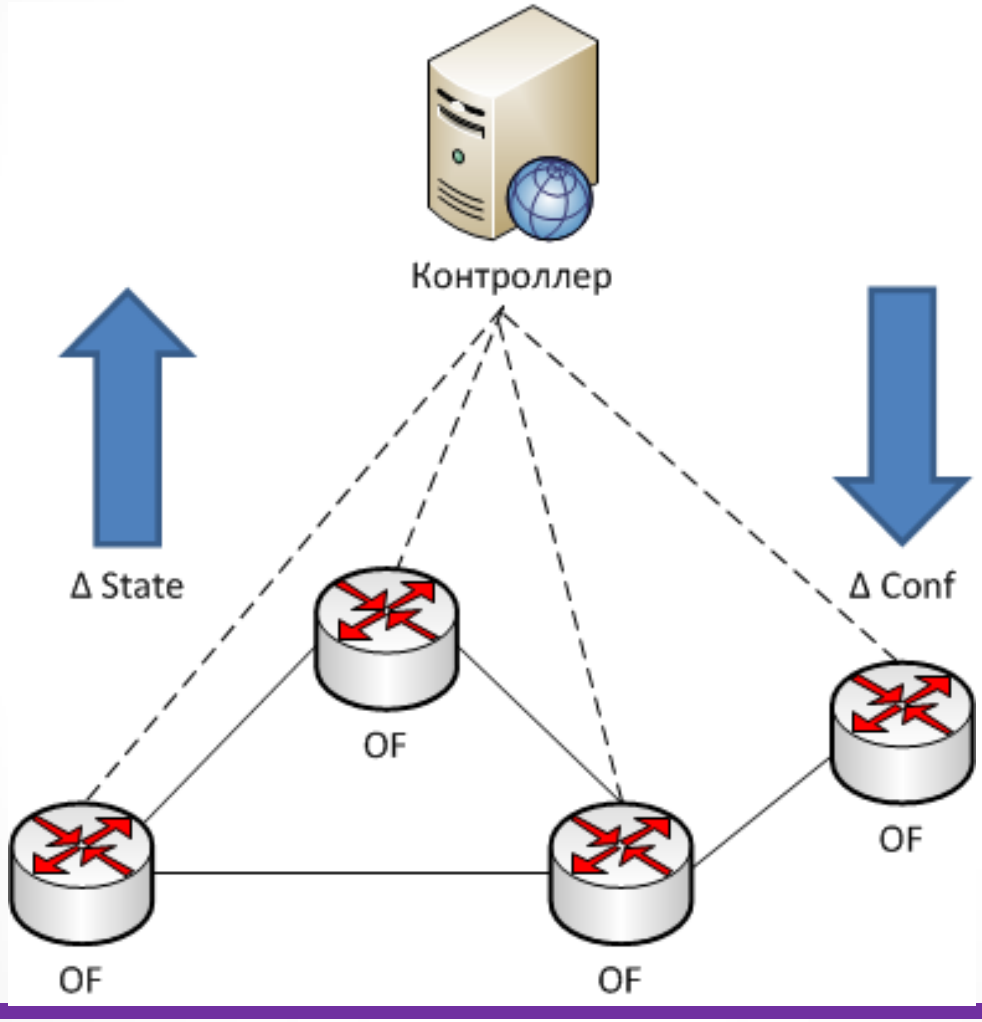
Software-Defined Networks (SDN)

Lecture 3: SDN/OpenFlow Controllers

Vasily Pashkov

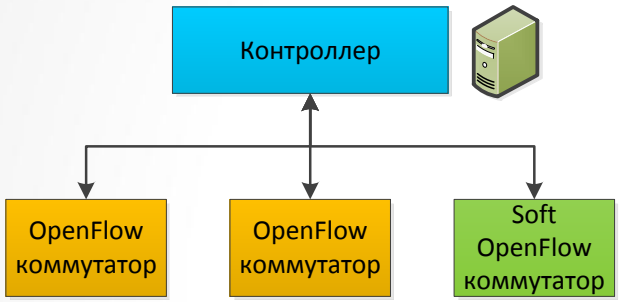
pashkov@lvk.cs.msu.su

SDN Controller

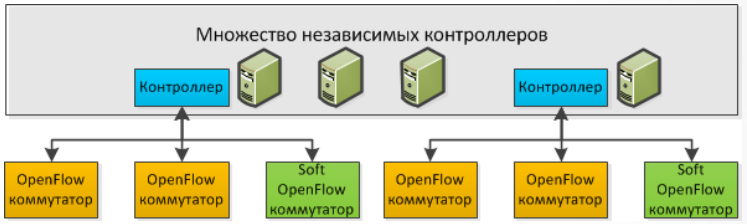


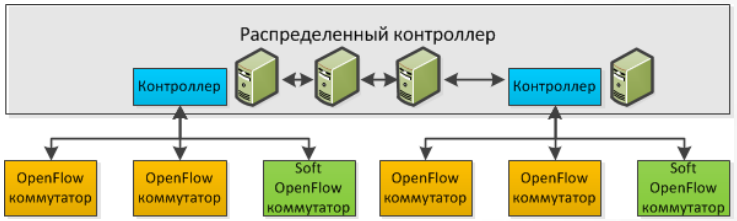
SDN Controller Architecture

1. Cetralized

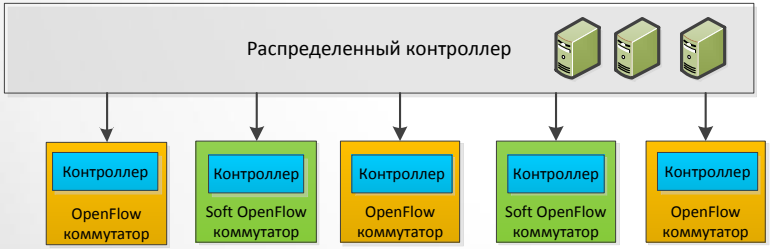


2. Distributed

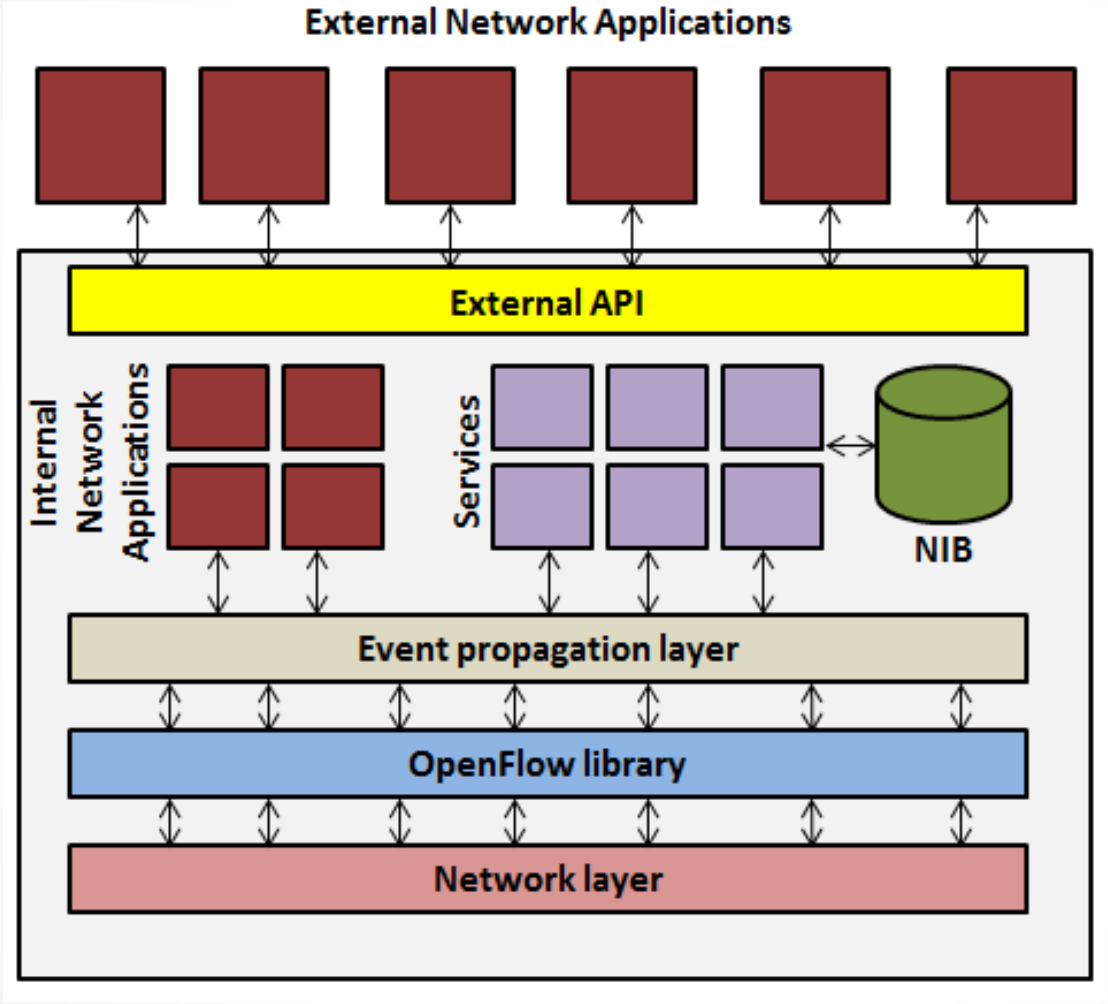




3. Hierarhical



SDN Controller Architecture



SDN Controller Implementations

- NOX-Classic
- NOX
- RUNOS
- Beacon
- Floodlight
- SNAC
- Ryu
- POX
- Maestro
- Trema
- FlowER
- MUL
- McNettle
- NodeFlow
- Onix
- SOX
- Kandoo
- Jaxon
- ONOS
- OpenDaylight
- Cisco ONE controller
- Nicira NVP Controller
- Big Network Controller
- IBM Programmable Network Controller
- HP SDN Controller
- NEC Programmable Flow

SDN Controller Implementations

Название	Язык Прогр.	Цель создания
NOX-Classic	C++, Python	Первый контроллер для опытной аппробации подхода ПКС. В настоящее время данная версия не развивается. Проект NOX-Classic был разделен на два самостоятельных проекта POX – контроллер на Python и NOX – контроллер на C++ с с целью повышения производительности последнего.
NOX	C++	NOX был создан с целью повышения производительности контроллера для ПКС сети за счет его написания на языке C++ и использования многопоточности. Появился из NOX-Classic.
POX	Python	POX создавался для исследований и обучения с возможностью быстрой разработки и прототипирования приложений для управления сетями на основе технологии OpenFlow. Появился из NOX-Classic.
SNAC	C++	SNAC - Simple Network Access Control – OpenFlow контроллер, разработанный для обеспечения гибкого простого способа определения политик (контроля доступа) в корпоративных сетях. SNAC позволяет настраивать сеть с помощью формального языка моделирования - formal modelling language (FML).

Skoltech
Skolkovo Institute of Science and Technology

SDN Controller Implementations

Название	Язык Прогр.	Цель создания
Beacon	Java	Первый контроллер, написанный на Java. Основные цели проекта: • повышение производительности разработчика, • обеспечение возможности запуска и остановки существующих или новых сетевых приложений в процессе работы контроллера, • достижение высокой производительности контроллера
Maestro	Java	Maestro – кросс-платформенный многопоточный контроллер на языке Java.
FloodLight	Java	Контроллер с открытыми исходными кодами, поддерживаемый открытым сообществом разработчиков. Написан на основе контроллера Beacon языке Java. Разрабатывается по лицензии Apache, т.е. может использоваться для любых целей.
Ryu	Python	Первый контроллер, интегрированный с OpenStack.

SDN Controller Implementations

Название	Язык Прогр.	Цель создания
Trema	C, Rybu	Trema разрабатывалась как платформа для разработки OpenFlow контроллеров на языках C/Ruby. Trema разрабатывалась для исследовательских и учебных учреждений, поэтому включает в себя интегрированную среду тестирования и отладки приложений.
Mul	C	Контроллер MūL разрабатывался для обеспечения производительности и надежности, которые необходимы для развертывания сетей, выполняющих критически-важные задачи (с повышенными требованиями к надежности).
NodeFlow	JavaScript для Node.JS	Первый OpenFlow контроллер, написанный на чистом JavaScript для Node.JS.
FlowER	Erlang	Первый OpenFlow контроллер, написанный на функциональном языке программирования Erlang. Цель - обеспечить простейшую платформу для написания сетевых управляющих приложений на Erlang.

	Характеристика	NOX-Classic	NOX	SNAC	POX	Beacon
1	Язык програм-мирования	C++, Python	C++	C++, Python	Python (2.7)	Java
2	Версия OpenFlow протокола	OF 1.0	OF 1.0, 1.2, 1.3	OF v0.8.9, v1.0	OF 1.0	OF 1.0
3	Платформа	Linux-платформы	Последние версии Ubuntu 11.10 и 12.04, Debian и RHEL 6	Linux-платформы	Linux, MacOS и Windows Android, BeOS-подобная система Haiku OS	Windows, Linux, Mac Android (64-х битные)
4	Многопоточ-ность	нет	да	нет	нет	да
5	Документирова нность	плохая	хорошая	плохая	хорошая	хорошая
6	Связь с другими СОС	Нет данных	На основе NOX Classic	на основе NOX Classic	На основе NOX Classic	Нет данных

	Характеристика	Maestro	FloodLight	Trema	MuL	FlowER
1	Язык програм-мирования	Java	Java	C, Ruby	C	Erlang
2	Версия OpenFlow протокола	OF 1.0	OF 1.0	OF 1.0 (OF 1.3 в разработке)	OF 1.0 (OF 1.2 планируется)	OF 1.1 (OF 1.2 в разработке)
3	Платформа	Linux	Windows, Linux, Mac Android	Ubuntu 12.10, 12.04, 11.10, and 10.04 (i386/amd64, Desktop Edition) Debian GNU/Linux 6.0 (i386/amd64) Fedora 16 (i386/x86_64)	Ubuntu 10.04 LTS	Linux
4	Многопоточ-ность	да	да	да	да	да
5	Документирова нность	хорошая	хорошая	хорошая	хорошая	плохая
6	Связь с другими СОС	Нет данных	На основе Beacon	Нет данных	Нет данных	Нет данных

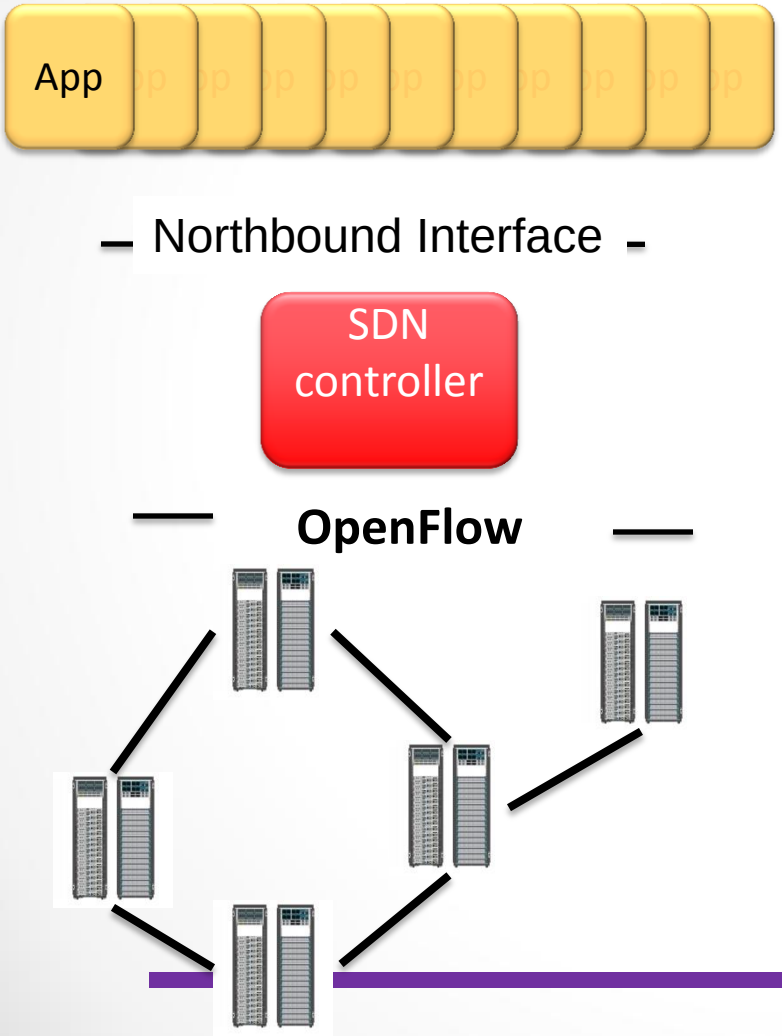
SDN Controller Implementations

Язык программирования	Сетевые ОС
C	MUL
Python	POX, Ryu
C++	NOX
Java	Beacon, Maestro, FloodLight, Jaxon
JavaScript	NodeFlow
Erlang	FlowER
Haskell	McNettle
C, Ruby	Trema
C++, Python	NOX Classic, SNAC

OpenFlow Controllers

Skoltech
Skolkovo Institute of Science and Technology

Openflow controllers



- Network elements has two components: OpenFlow client, forwarding hardware with flow tables.
- The SDN controller must implement the network OS functionality
  - Provide abstraction to the upper layer
  - Provide control to the underlying hardware
  - Managing the resources

SDN controllers (NOS) .vs. OS

OS

- Resources managed
  - CPU, memory, disk, IO devices, etc
- Applications:
  - User programs that use the resources
- OS functionality (abstraction):
  - CPU virtualization
  - Memory virtualization
  - IO virtualization
  - File systems

Network OS

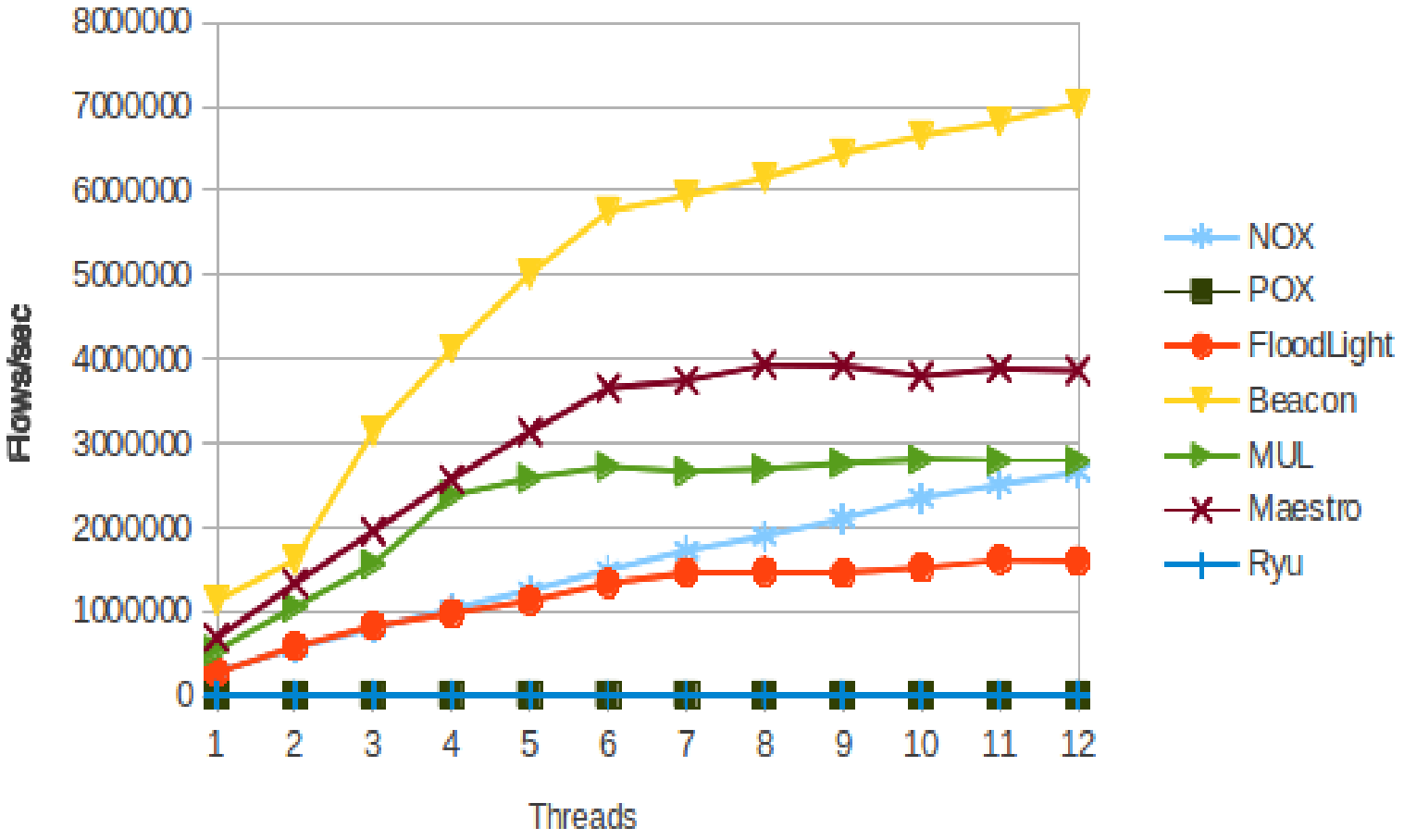
- Resources managed
  - Connected switches/routers/NICs
- Applications
  - Firewall, migration, network virtualization, NAT, TE, etc
- NOS functionality?
  - Network abstraction – this is a new thing that is not well understood.

Skoltech
Skolkovo Institute of Science and Technology

NOS functionality

- From: “NOX: towards an Operating System to Networks”
  - NOS should present application programs with a centralized programming model
  - Programs should be written in terms of high level abstractions, not low-level parameters

SDN/OpenFlow Controller Performance



Skoltech
Skolkovo Institute of Science and Technology

Openflow controller: NOX/POX

- Originally developed by Nirica
- NOX: C++ version; POX: python version
  - Nox for performance; Pox for rapid prototyping.
- POX comes with Mininet – the simulation infrastructure
- OpenFlow v.1.0
- Programming model:
  - Controller registers for events (PacketIn, ConnectionUP, etc).
  - Programmer write event handler

Skoltech
Skolkovo Institute of Science and Technology

NOX/POX Events

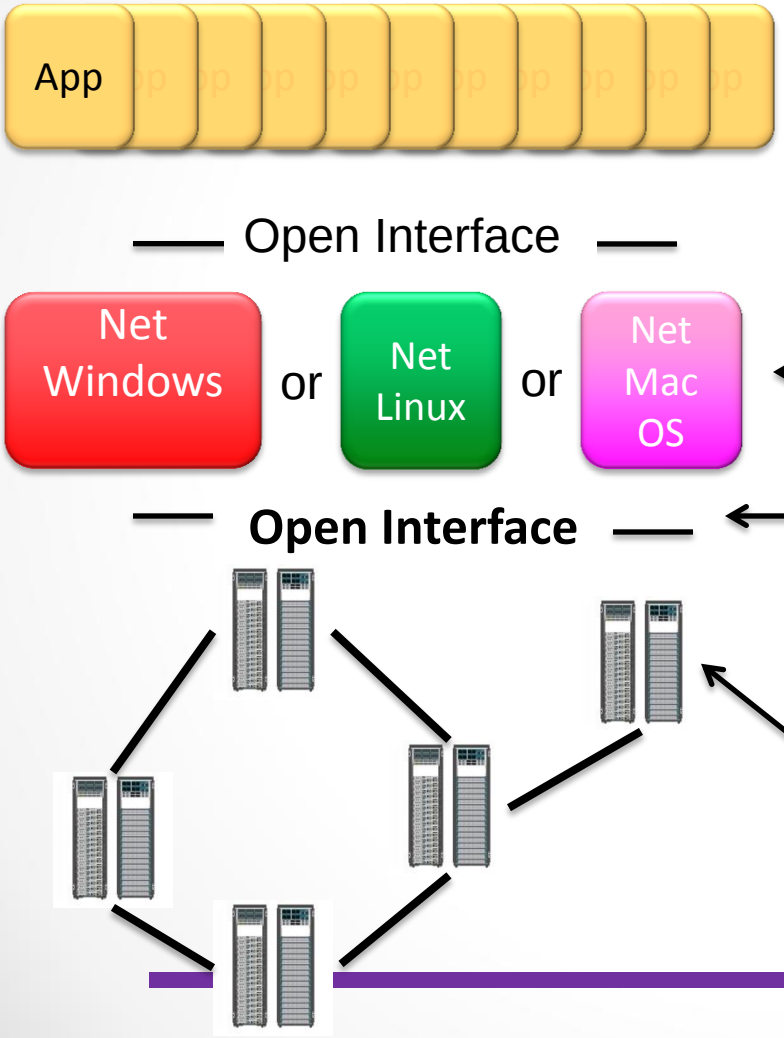
- FlowRemoved
- ConnectionUP
- PacketIn
- etc

- User write event handlers
  - E.g. ConnectionUp: record in the database, PacketIn: compute the route, setup flow table along the path, etc

Abstraction? Global view build from control program, fairly low level.

Skoltech
Skolkovo Institute of Science and Technology

Openflow SDN current state



4. Firewall, virtual network, TE, IDS, etc

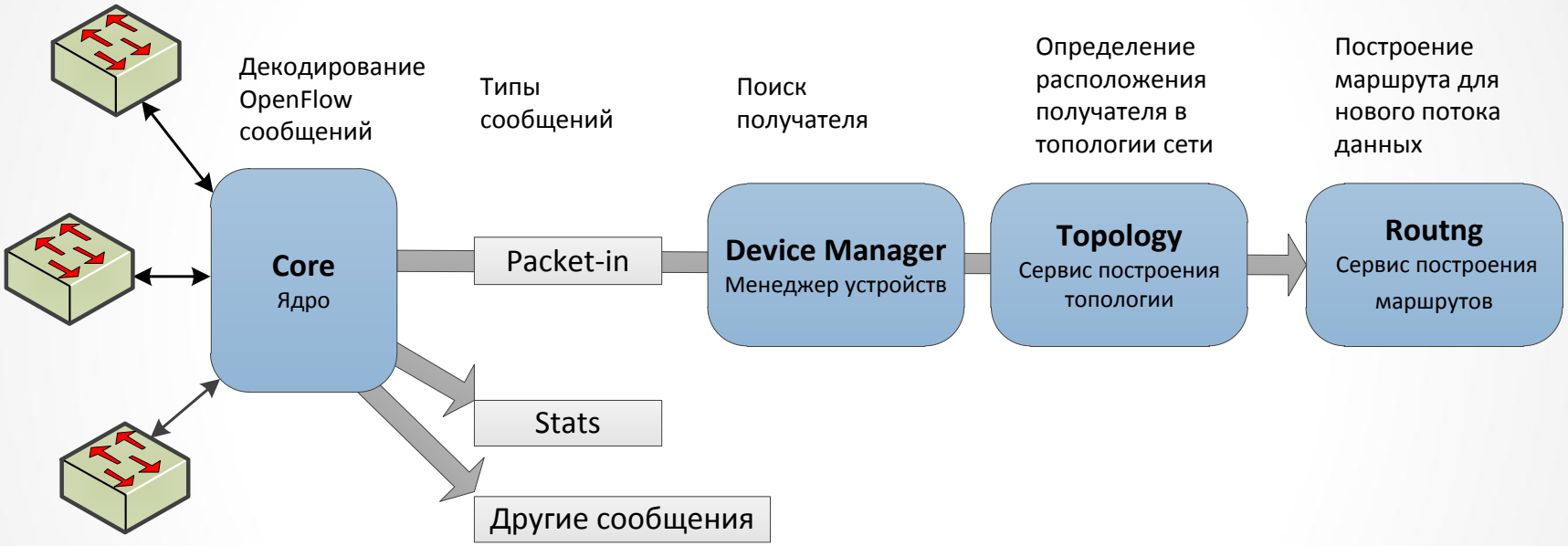
Northbound API, not standardized yet

3. SDN controllers (floodlight, nox, etc)

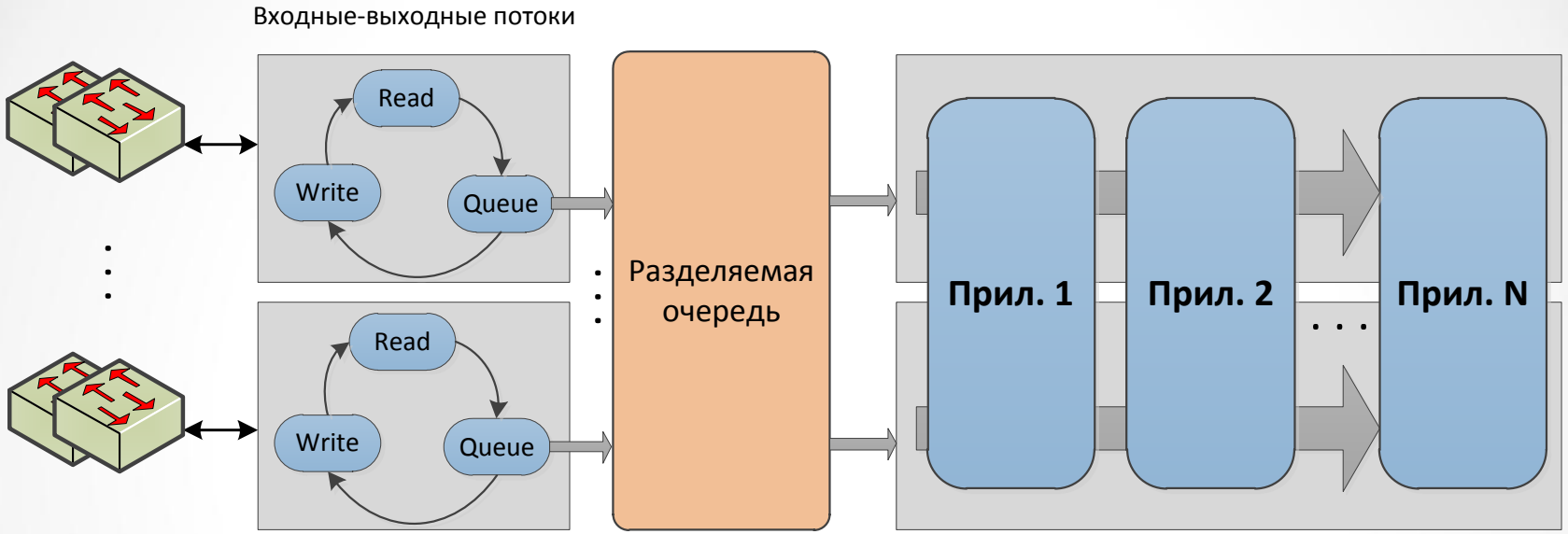
1. OpenFlow: standardized for Ethernet/IP/TCP

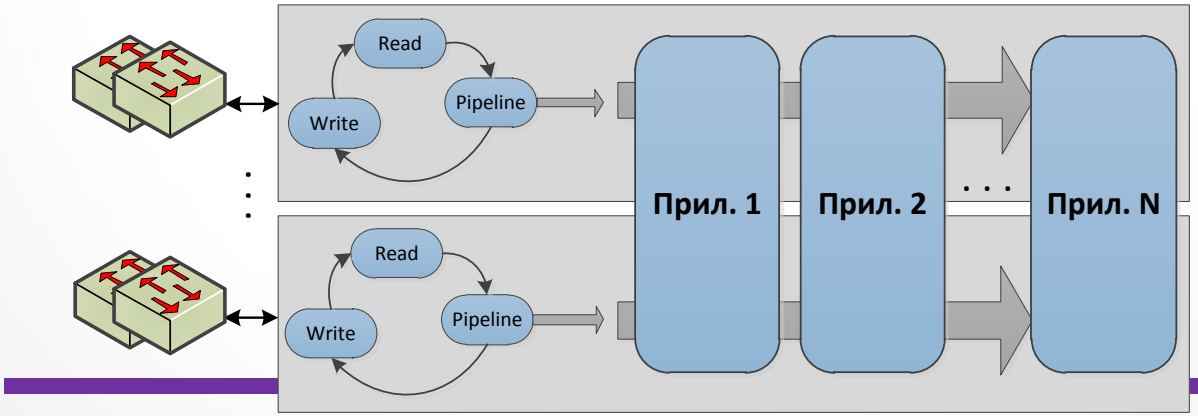
2. OpenFlow enabled switches/routers
simple hardware doing forwarding only
forwarding table can be set by other entity through OpenFlow

Pipeline

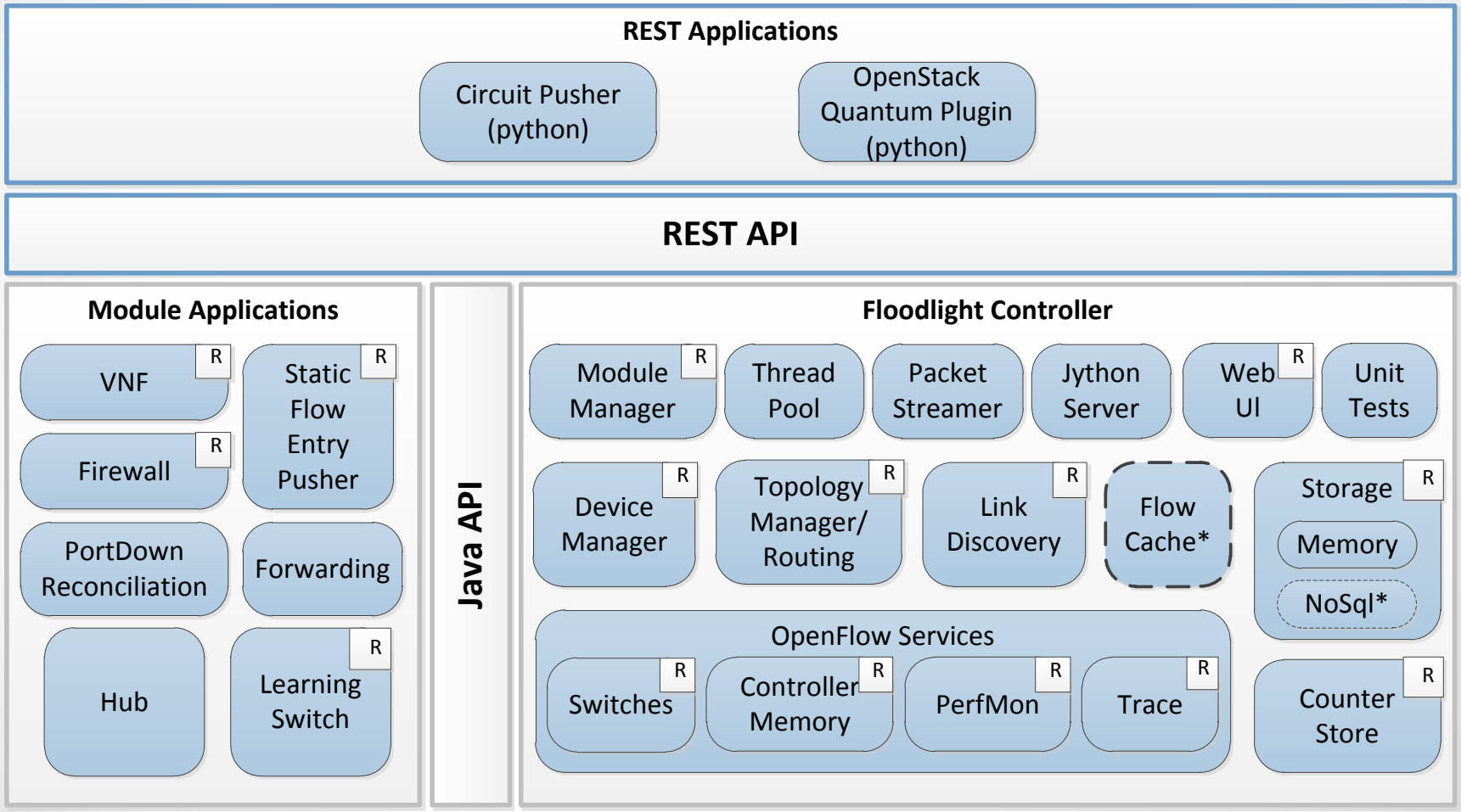

Декодирование OpenFlow сообщений
Типы сообщений
Поиск получателя
Определение расположения получателя в топологии сети
Построение маршрута для нового потока данных
Core
Ядро
Packet-in
Device Manager
Менеджер устройств
Topology
Сервис построения топологии
Routng
Сервис построения маршрутов
Stats
Другие сообщения

Multithreaded controller



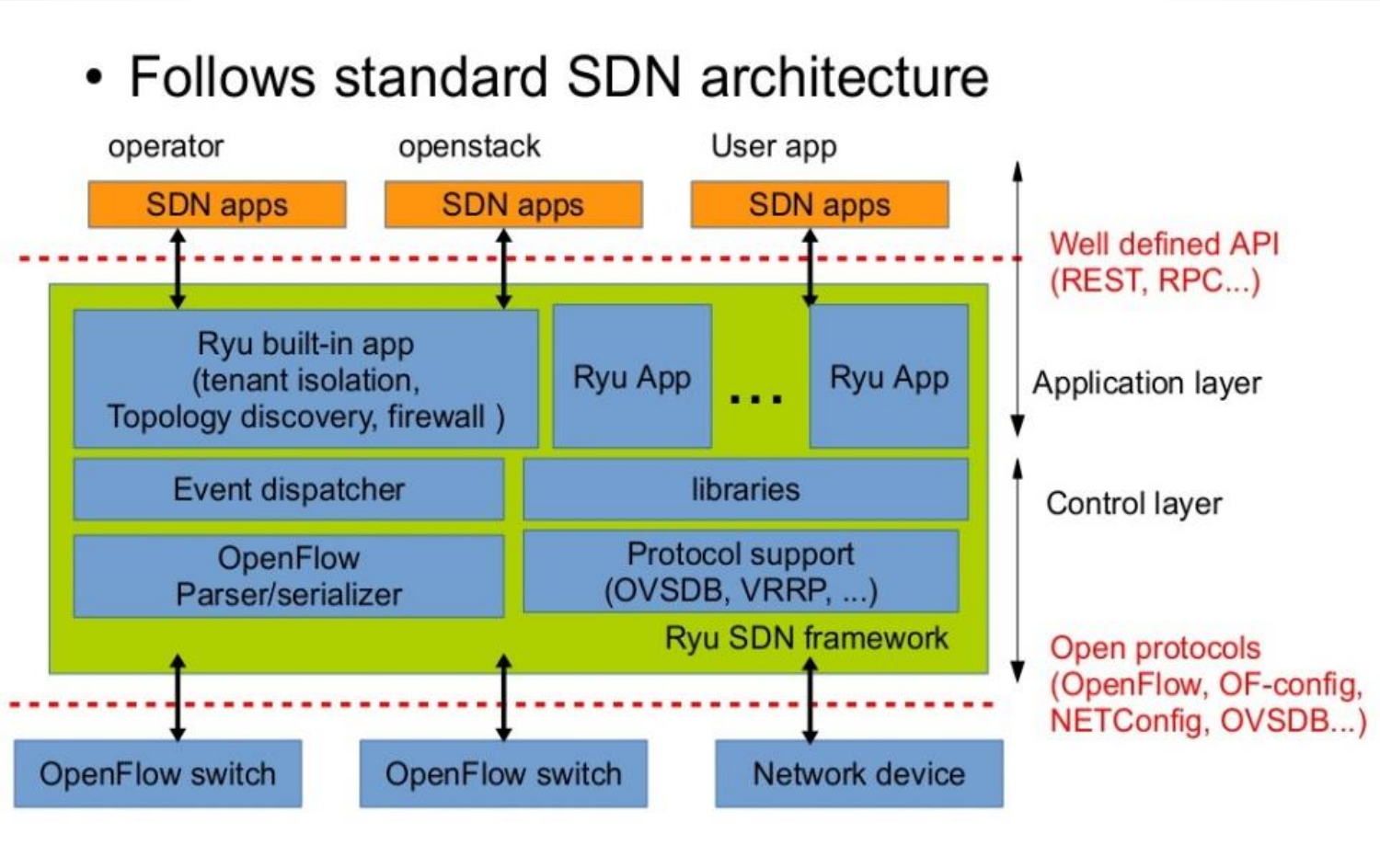


Floodlight Architecture



* Interfaces defined only & not implemented: FlowCache, NoSql

Ryu Architecture



Skoltech
Skolkovo Institute of Science and Technology

Thanks for your attention!

Vasily Pashkov
pashkov@lvk.cs.msu.su