Санкт-Петербургский государственный университет
телекоммуникаций им. проф. М.А. Бонч-Бруевича

Особенности эксплуатации программно-конфигурируемых сетей

(Лекция 3)

Елагин В.С.
к.т.н. доцент каф. ИКС

СПб ГУТ)))

Сетевая операционная система

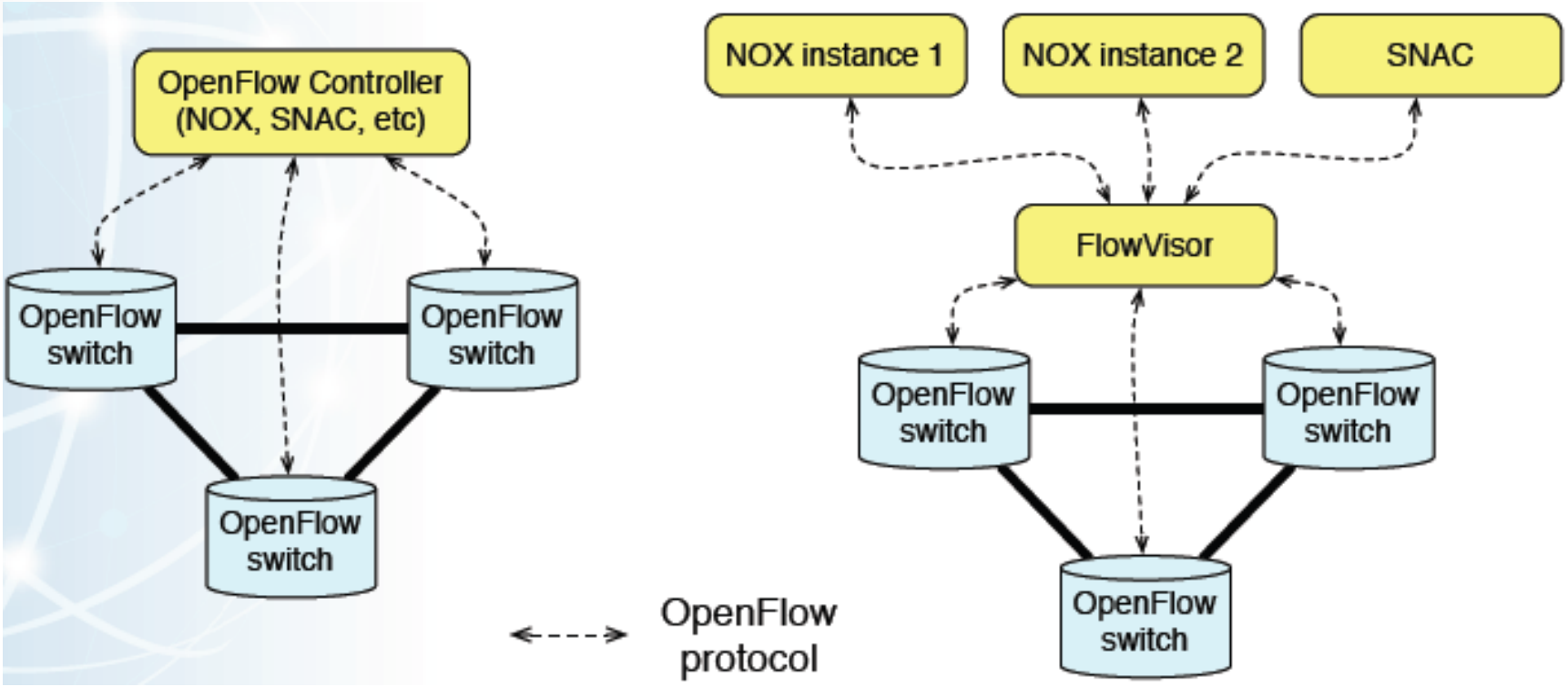
Особенности сетевой ОС

- API сетевой ОС предоставляет возможность создавать приложения на основе ***централизованной модели программирования***

- API сетевой ОС позволяет создавать приложения в терминах ***высокоуровневых абстракций***, а не низкоуровневых параметров конфигурации.

[имя хоста вместо MAC-адреса]

Управление сетью



OpenFlow контроллер

- Программа, TCP/IP сервер, ожидающий подключения коммутаторов
- Отвечает за обеспечение взаимодействие приложения-коммутатор.
- Предоставляет важные сервисы (например, построение топологии, мониторинг хостов)
- API сетевой ОС или контроллер предоставляет возможность создавать приложения на основе **централизованной модели программирования**.

Структура контроллера

Приложения для контроллера, статически/динамически подгружаемые

Ядро контроллера

Библиотека – реализация протокола OpenFlow (C, Java, Python, Ruby, Haskel, Ocaml, Earlang, Javascript)

Список OpenFlow контроллеров

- Их действительно много
  - Nox, Pox, MUL, Ruy, Beacon, OpenDaylight, Floodlight, Maestro, McNettle, Flower, Runos
  - Разные языки программирования от Python до Haskell, Erlang
- Для образования - Pox.
- Два больших комьюнити
  - ONOS (Stanford)
  - OpenDayLight (Cisco)
- В России – наш Runos
  - arccn.github.io/runos

Требования к контроллеру ПКС

- Производительность
  - Пропускная способность
    - events per second
  - Задержка
    - us
- Надежность и безопасность
  - 24/7
- Программируемость
  - Функциональность: приложения и сервисы
  - Интерфейс программирования

- ЦОД требует обработку >10М событий в секунду
- Реактивные контроллеры более “чувствительные”

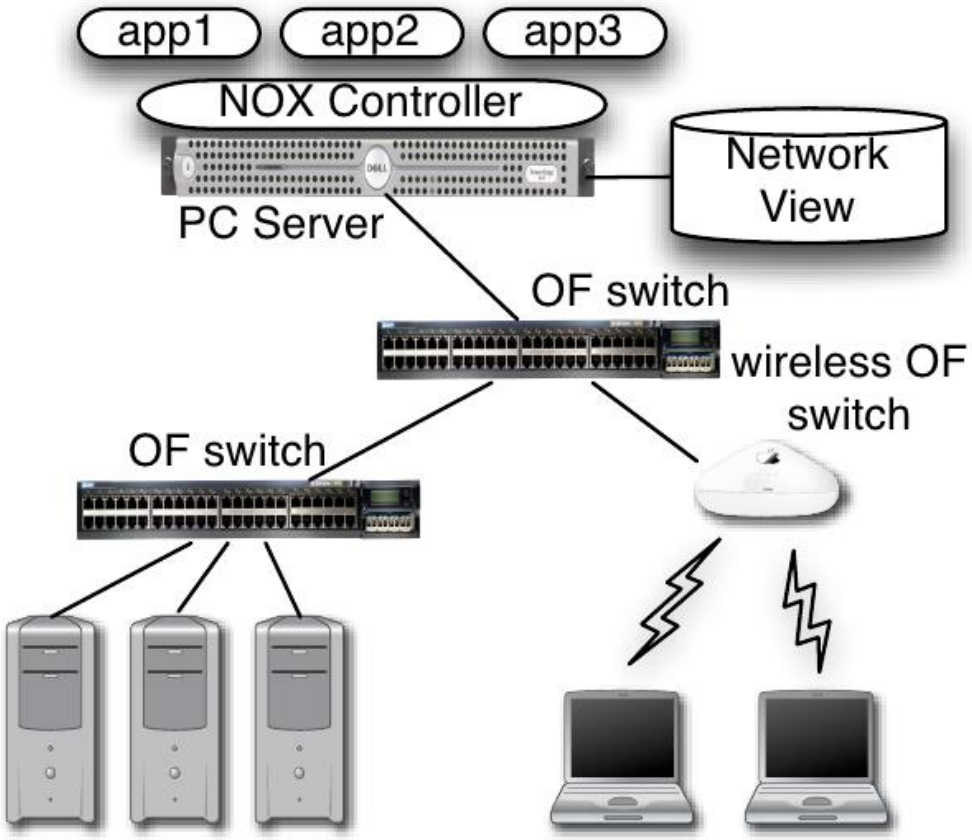
Существующие сетевые ОС

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, RouteFlow

NOX

- Первый контроллер NOX-Classic
- Разработчик: 2008-2009 - Nicira Networks, 2009-2010 - Stanford University, 2009-2013 - ICSI UC Berkeley
- Лицензия: GPL v3
- ЯП: C++
- Протокол OF 1.0, 1.1, 1.2, 1.3
- Платформа: Linux (Ubuntu 11.10, 12.04, Debian, RHEL 6)
- Компоненты: topology discovery, learning switch, network-wide switch.
- Система генерации/подписки на событий (OF и приложений)
- Не имеет GUI

NOX (2)

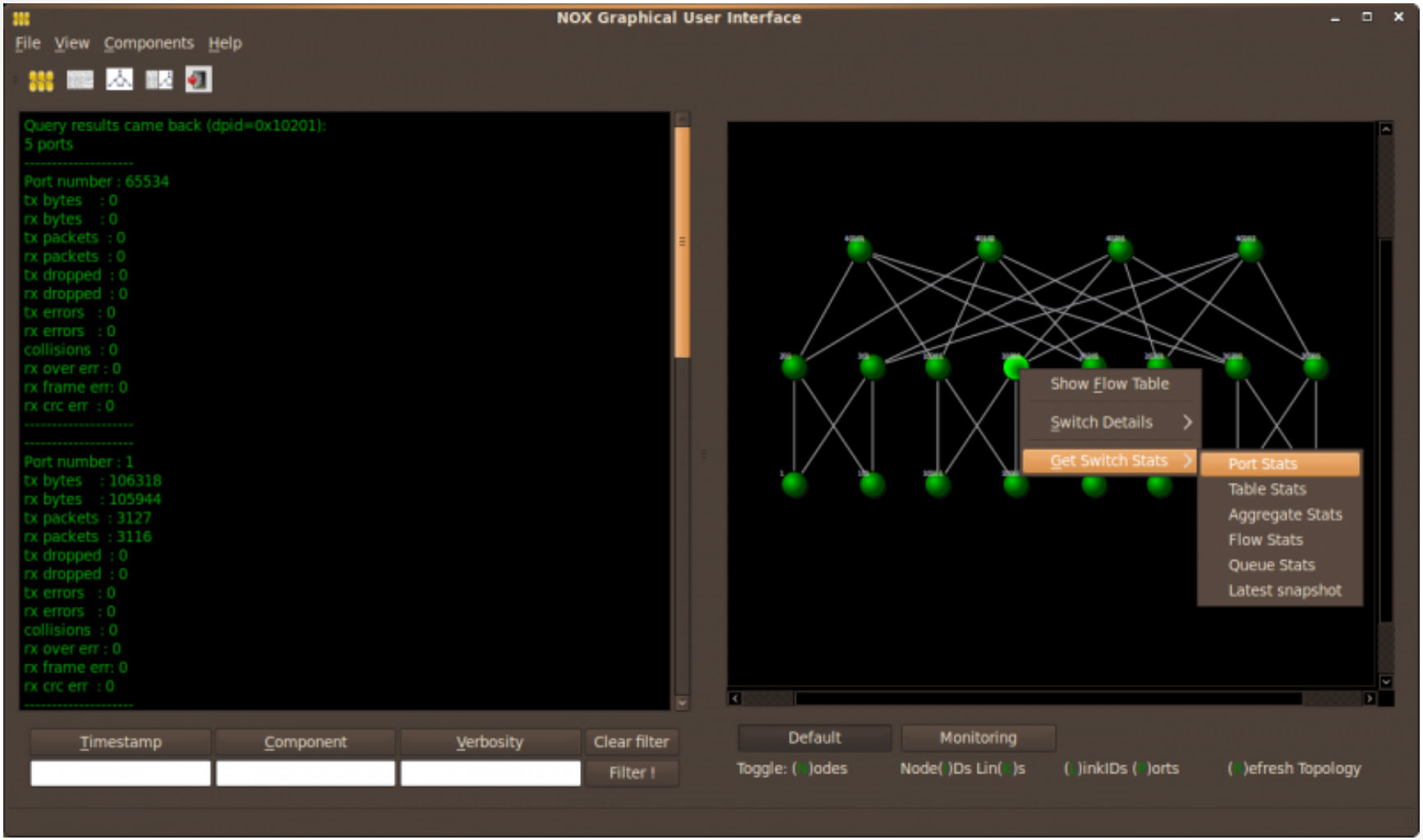


Компоненты сети с сетевой ОС NOX

NOX (3)

- Событийное управление
- Приложения являются генераторами событий
  - Приложения уровня ядра системы
  - Встроенные приложения:
    - Discovery
    - Topology
    - Authenticator
    - Routing
  - Пользовательские приложения

GUI NOX

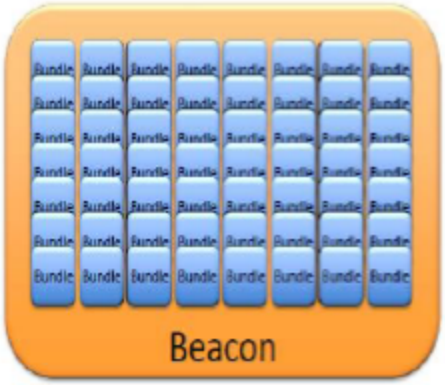

NOX Graphical User Interface
File View Components Help
Query results came back (dpid=0x10201):
5 ports
Port number : 65534
tx bytes : 0
rx bytes : 0
tx packets : 0
rx packets : 0
tx dropped : 0
rx dropped : 0
tx errors : 0
rx errors : 0
collisions : 0
rx over err : 0
rx frame err: 0
rx crc err : 0
Port number : 1
tx bytes : 106318
rx bytes : 105944
tx packets : 3127
rx packets : 3116
tx dropped : 0
rx dropped : 0
tx errors : 0
rx errors : 0
collisions : 0
rx over err : 0
rx frame err: 0
rx crc err : 0
Show Flow Table
Switch Details
Get Switch Stats
Port Stats
Table Stats
Aggregate Stats
Flow Stats
Queue Stats
Latest snapshot
Timestamp
Component
Verbosity
Clear filter
Filter !
Default
Monitoring
Toggle: (N)odes
Node( )Ds Lin(K)s
( )inkIDs ( )orts
(R)efresh Topology

POX

- Разработчик: James McCauley, ICSI UC Berkeley
- Лицензия: GPL v3
- ЯП: Python
- Протокол OF 1.0
- Платформы Linux, Mac OS и Windows
- Предназначен для быстрого прототипирования приложений, создавался для исследований
- Компоненты: L2/L3 forwarding, Topology discovery, keepalive, of_01, debug, spanning tree и др.
- GUI

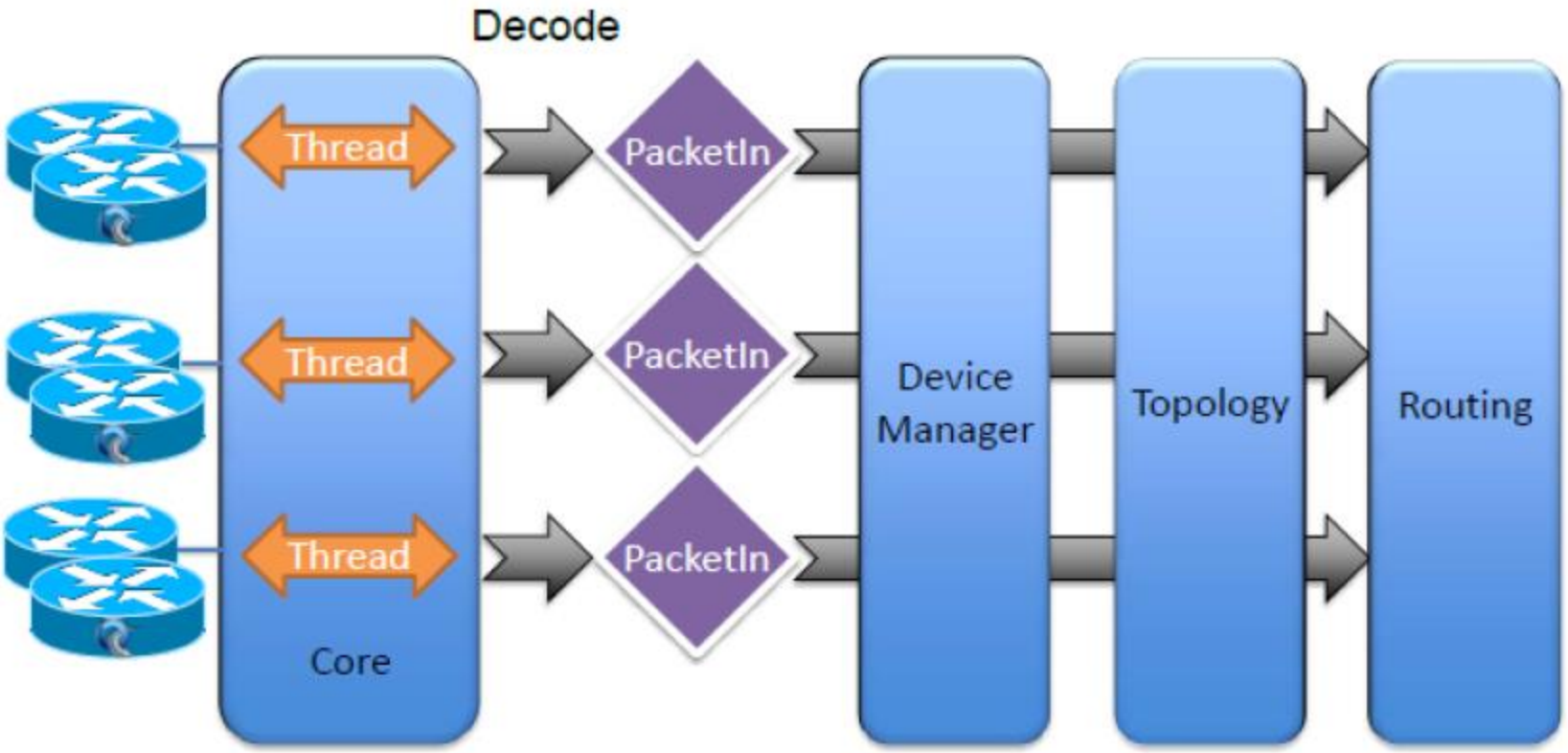
Beacon

- Разработчик: David Erickson, Stanford University
- Лицензия: GPL v2 license and the Stanford University FOSS License Exception v1.0
- ЯП: Java
- Протокол OF 1.0
- Многопоточный
- Web-интерфейс пользователя



- Стабильность, кросс-платформенность, открытость, динамичность, быстрая разработка приложений, скорость работы
- Базовые компоненты: OpenFlowJ, Шифратор/дешифратор пакетов, Core, Learning Switch, Hub, Device Manager, Topology, Layer 2 Shortest Path Routing

Beacon



Многопоточный конвейер для обработки пакетов

Maestro

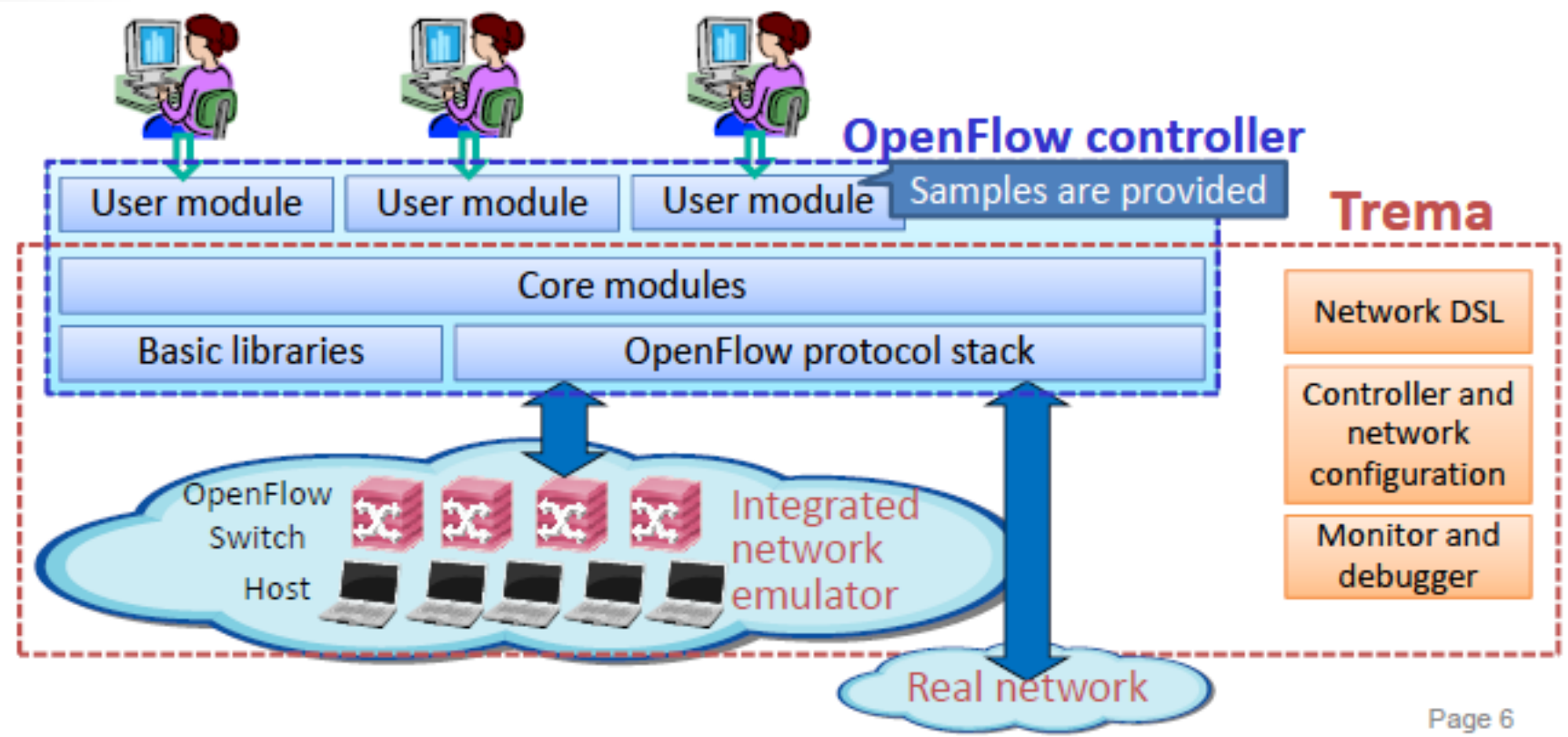
- Разработчик: Rice University
- Лицензия: LGPL v2
- ЯП: Java
- Протокол OF 1.0
- Многопоточный
- Компоненты: Discovery, Route Flow, Periodic Timer, Input stage и др.
- Максимальная пропускная способность: 600000 запросов в секунду

- Разработчик: Big Switch
- Лицензия Apache 2.0
- ЯП: Java
- Протокол OF 1.0
- Появился на основе Beacon
- Компоненты: Module Manager, Packet Streamer, Link Discovery, Device Manager, Topology Manager, Routing, OpwnFlow Services и др.
- Web UI

Trema

- Разработчик: NEC (для исследований)
- Лицензия: GPL v2
- ЯП: C и Ruby
- Протокол OF 1.0
- Модульная архитектура
- Содержит TremaShark – строенный сетевой симулятор и отладчик контроллера
- Основные компоненты: Openflow interface, Openflow message ditributer, Path manager, Topology manager, Learning L2 Switch.
- Пользовательские приложения на C или Ruby

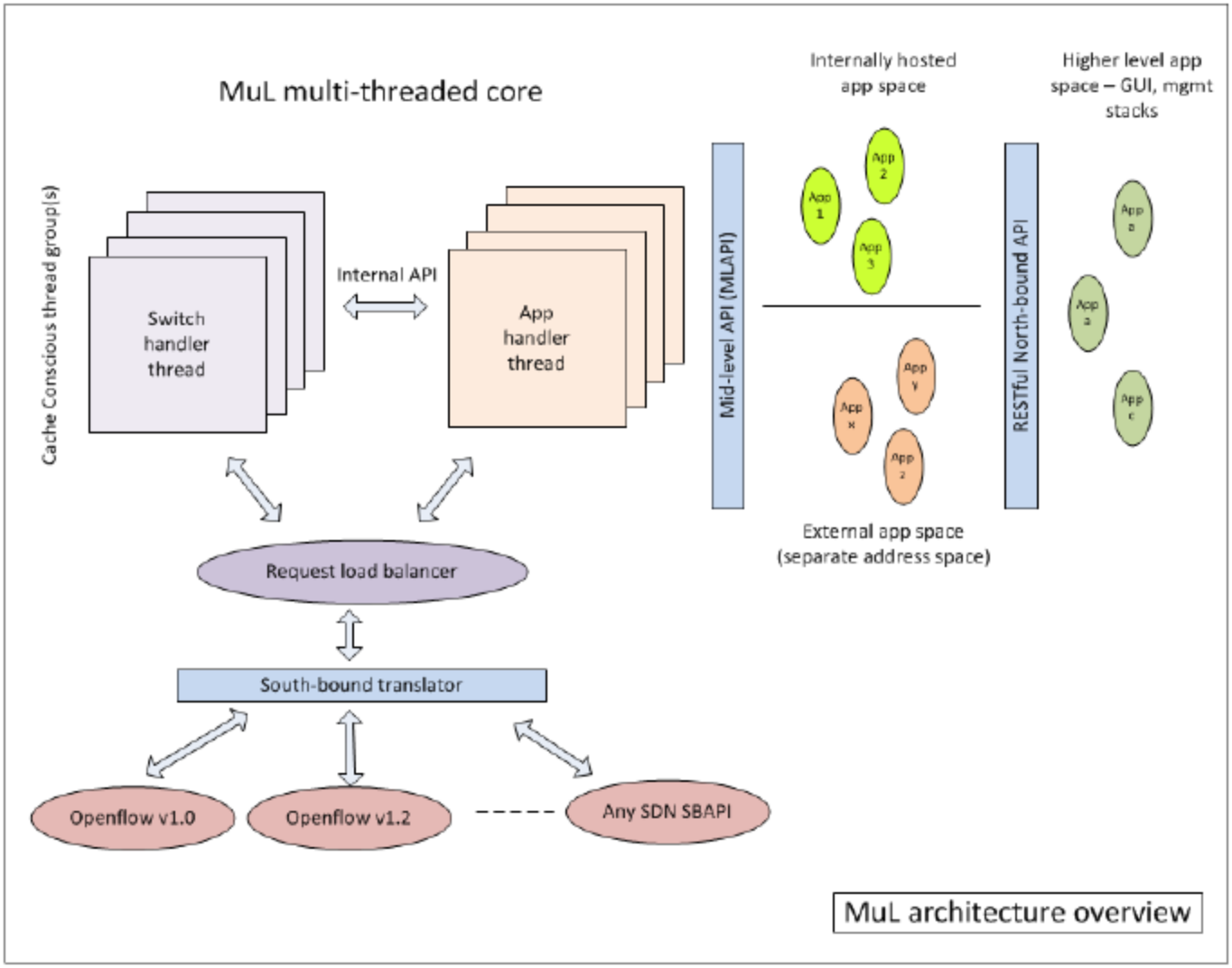
Trema (2)


OpenFlow controller
User module
User module
User module
Samples are provided
Trema
Core modules
Basic libraries
OpenFlow protocol stack
Network DSL
Controller and network configuration
Monitor and debugger
OpenFlow Switch
Host
Integrated network emulator
Real network
Page 6

MUL контроллер

- Разработчик: KulCloud Networks (Dipjyoti Saikia)
- Версия: 2.0.0 (8 февраля), Лицензия: GPL v2
- ЯП: C
- Протокол OF 1.0 (планируется OF 1.2)
- Масштабируется до 512 коммутаторов
- Компоненты: определение топологии, вычисление кратчайшего пути (Floyd-Warshall алгоритм), обработчик Openflow сообщений, поддержка интеграции с OpenStack, проактивная установка потоков.

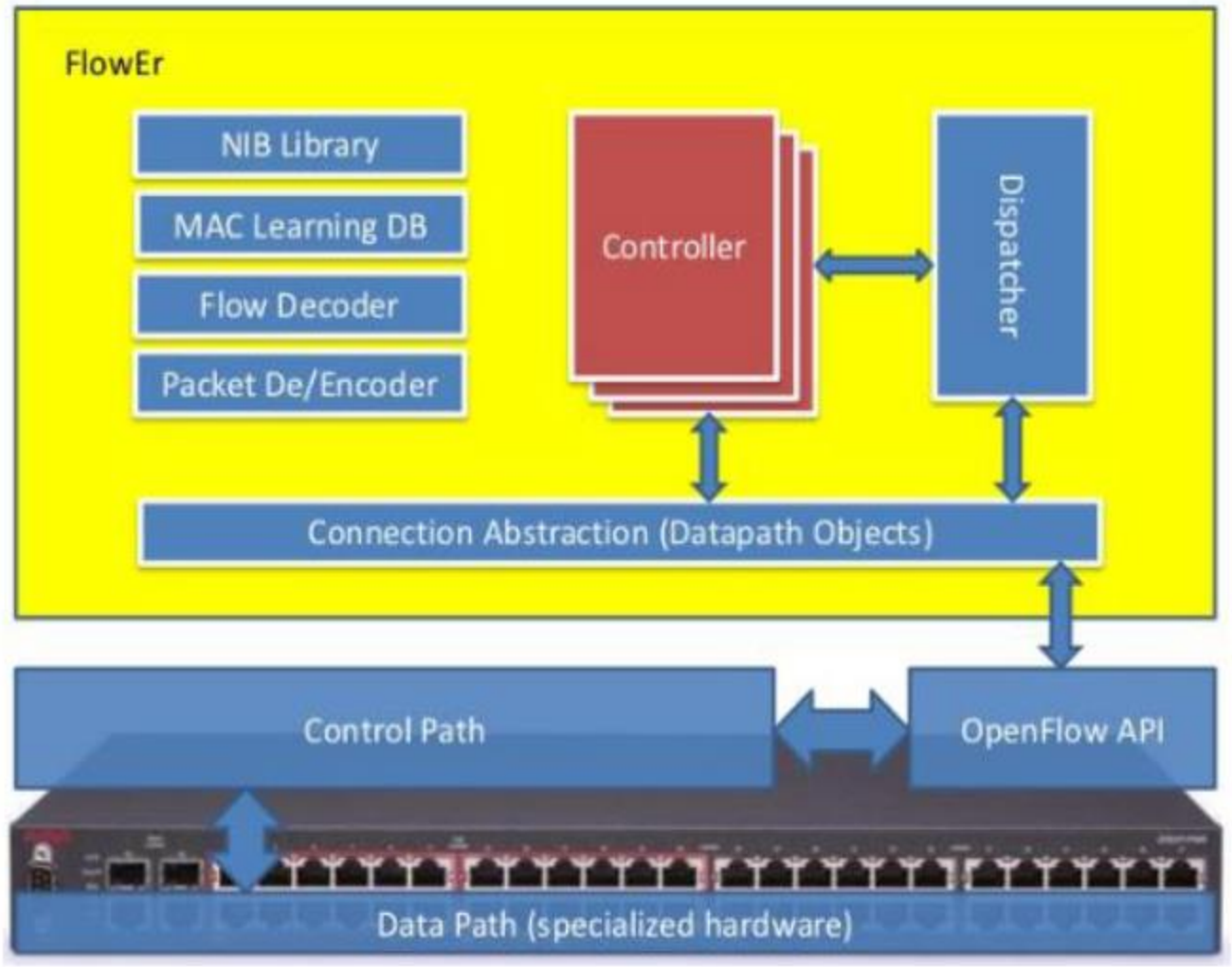
Архитектура Mul



FlowER

- Разработчик: Travelping GmbH
- Лицензия: BSD
- ЯП: Erlang (быстрая скорость разработки; легко читать, модифицировать и расширять код; хорошо подходит для реализации бинарных протоколов (таких как OpenFlow) (гибкий бинарный поиск); поведение и обратные вызовы для обеспечения модульности; изоляция ошибок (fault isolation), поддержка параллелизма)
- Протокол OF 1.2
- Основные компоненты: Flower_datapath, Flower_dispatcher, NIB Library, MAC Learning DB, Flow Decoder, Packet De/Encoder.

FlowER - архитектура



Сетевая операционная система RUNOS

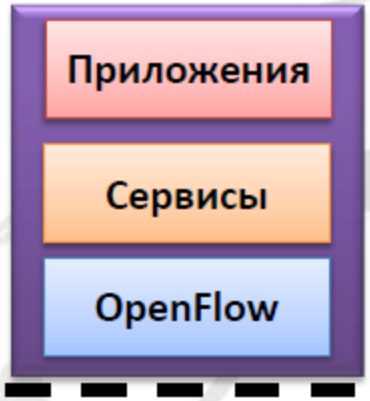
RUNOS = **RU**ssian **N**etwork **O**perating **S**ystem

- Цель проекта
  - *“Could an OpenFlow controller be both easy to develop applications for and also high performance?”*
  - Разработать систему, которая будет удобна для разработки новых сетевых приложений
  - При этом помнить о быстроте

Коммутатор	Версия OpenFlow
Arista 7050T-52	OF1.0
NEC PF5240F	OF1.0, OF1.3
Extreme X460-24p	OF1.0, OF1.3
OpenvSwitch 1.6 и выше	OF1.0, OF1.3
Brocade	OF 1.3
Huawei	OF 1.3

RUNOS: особенности

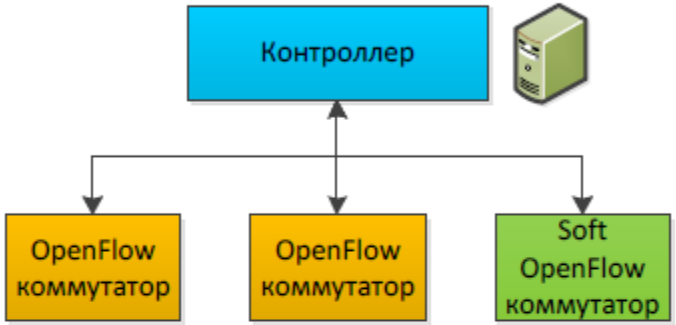
- Проблема запуска нескольких приложений, интеграция с приложениями других разработчиков
  - требуется статическая подстройка приложений под себя, порядок и способ передачи информации между ними.
  - нет механизма контроля и разрешения конфликтов между приложениями (генерация пересекающихся правил).
- В RUNOS стоит задача решить указанные выше проблемы:
  - часть настройки происходит автоматически по мета информации, связывание происходит динамически
  - разработана система разрешения конфликтов
  - Широкий набор сервисов для упрощения разработки новых приложений



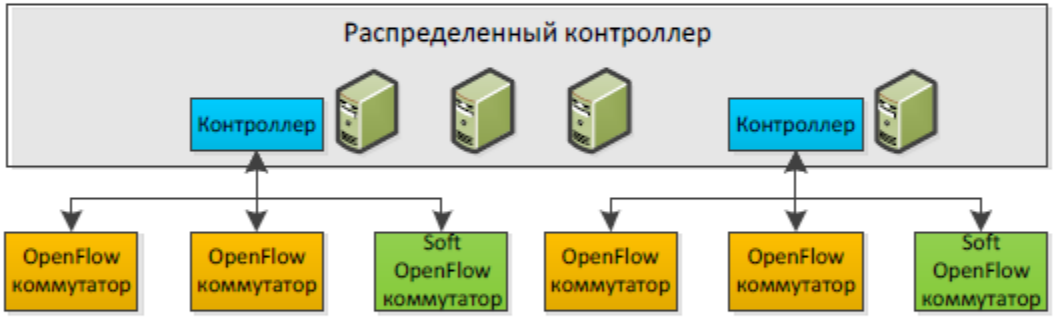
Features:

- Algorithmic policies (rule generation)
- Client-friendly API using EDSL grammar (low level details are hidden inside the runtime – overloading, templates)
- Modules composition (parallel and sequential composition)

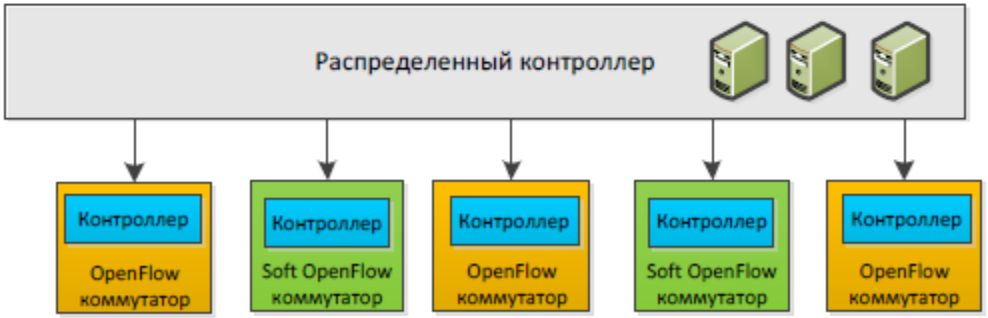
Развитие архитектуры контроллера/сетевой ОС для ПКС



NOX, FloodLight, Beacon, Maestro и др.

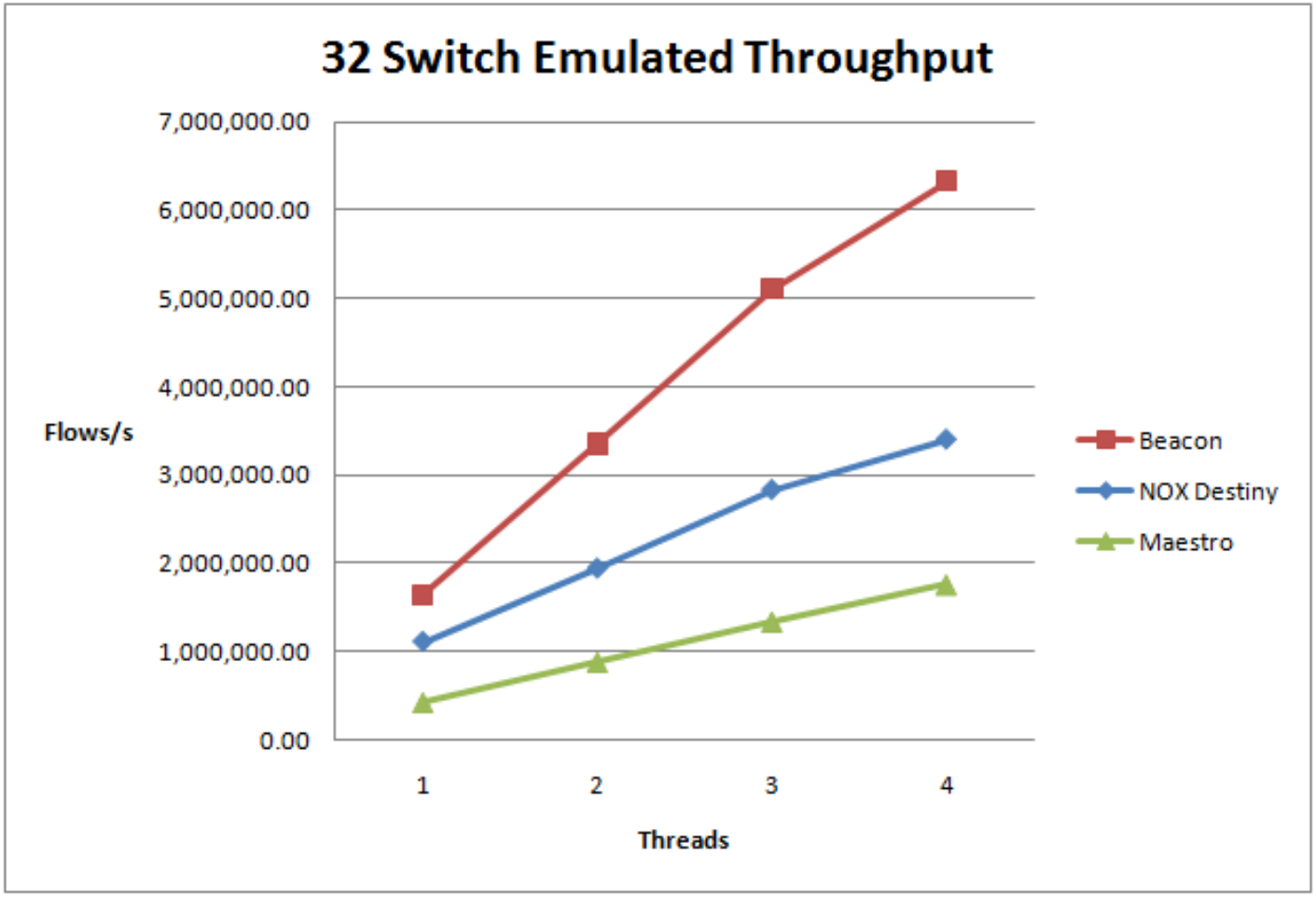


HyperFlow, SOX, ONIX



Kandoo

Сравнение производительности контроллеров



Платформа управления

- Распределенная система, функционирующая на одном или нескольких серверах в сети

Функции:

- Сбор информации от коммутаторов.
- Распространение управляющих команд.
- Координация согласованности состояний серверов платформы управления.
- Программный интерфейс для создания сетевых приложений управления.

Платформа управления сетью Onix

- Распределенная управляющая платформа, работающая на кластере физических серверов, на каждом из которых может быть запущено несколько экземпляров Onix
- Более общий API для управления распределенными состояниями сети по сравнению с другими контроллерами
- Обеспечивает гибкие (распределенные) примитивы
- Масштабируемость
- Надежность
- 150000 строк кода на C++

Основные компоненты SDN-сети под управлением Onix

- Физическая инфраструктура
- Инфраструктура подключения
- Платформа Onix
- Логика управления сетью

ПЕРСПЕКТИВЫ ИСПОЛЬЗОВАНИЯ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

Перспективы использования

[удешевление оборудования]

- Универсализация сетевого оборудования
- Удаление функций управления из коммутационного оборудования
- Комбинирование оборудования и программного обеспечения разных производителей
- Обновление программного обеспечения

Перспективы использования

[упрощение администрирования]

- Централизованное управление оборудованием сегмента
- Управление сегментом сети по аналогии с управление домашним маршрутизатором
- Автоматическая настройка оборудования
- Гранулярный контроль траффика

Перспективы использования

[концепция приложений]

- Упрощение приложений с помощью стандартного интерфейса контроллера
- Расширяемая функциональность сети
- Динамическое взаимодействие пользовательских приложений и сети передачи данных
- Изоляция функциональности

Перспективы использования

[исследование сети]

- Упрощение разработки новых протоколов из-за централизации сети
- Простота установки и конфигурирования необходимых программ
- Проведение экспериментов на реальной инфраструктуре сети
- Простота сбора статистики

Примеры перспективных направлений для исследований

- Разработка алгебры пространства потоков

  [разрешение противоречий между правилами]

- Доработка контроллера

  [масштабируемость, надёжность, безопасность]

- Создание средств администрирования

  [язык описания сети, его транслятор]

- Расширение области применения

  [беспроводные сети, центры обработки данных]

Спасибо за внимание!

СПб ГУТ)))