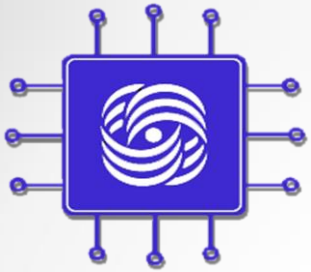


APPLIED
RESEARCH
CENTER FOR
COMPUTER
NETWORKS

SDN&NFV:
Технологии SDN/OpenFlow

Доп. главы Компьютерных сетей и телекоммуникации

к.ф.-м.н., м.н.с., Шалимов А.В.

ashalimov@lvk.cs.msu.su

@alex_shali

@arccnnews

Часть I: SDN

Уникальное время

“Россия и SDN – это идеальный брак, который должен состояться на небесах”

SDN уже здесь

Google перевел сеть между ЦОД на SDN в 2012 году, сейчас анонсирована внутренняя облачная платформа **Andromeda**.

Microsoft перевел сеть между ЦОД на SDN в конце 2013 года, на очереди публичное облако **Azure.**

NTT перевел всю свою сетевую инфраструктуру на SDN в 2013 году.

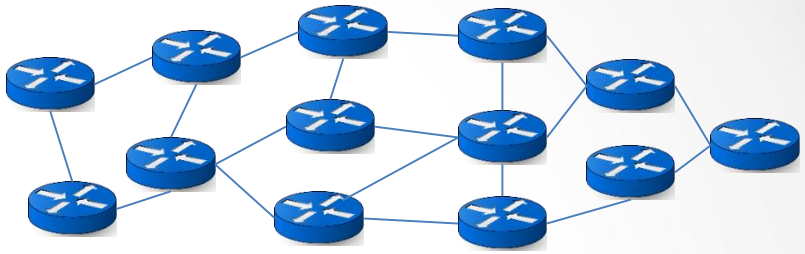
В июне 2015 года **AT&T** объявило SDN своим основным стратегическим направлением развития и переориентацию на разработку ПО.

Gartner: *“Рынок SDN решений к 2018 году достигнет объема $35 млрд”*.

А что с SDN в России?

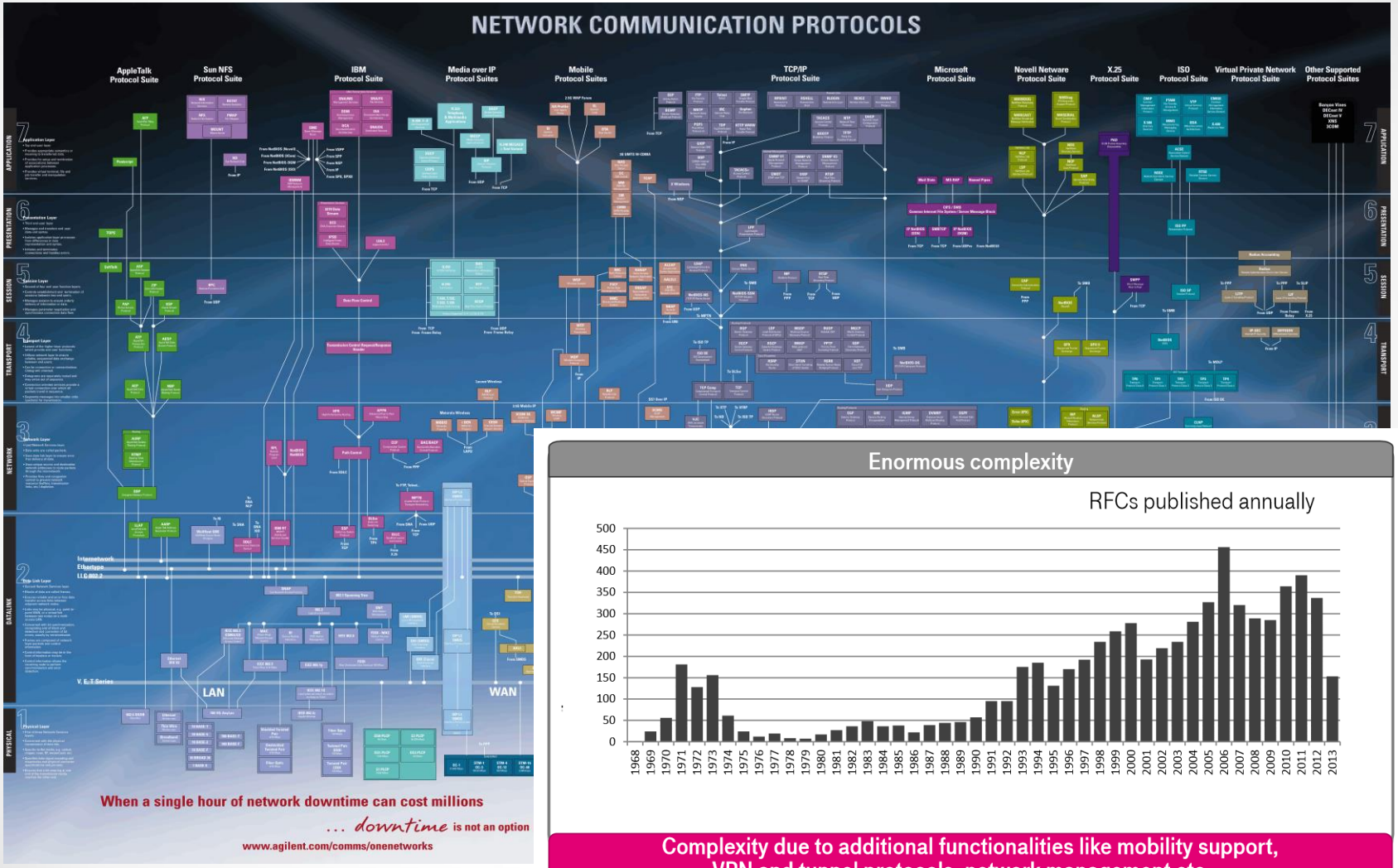
- «Ростелеком» начал работу над внедрением перспективных технологических направлений **«Программно-конфигурируемых сетей» (SDN)** и **«виртуализации сетевых функций» (NFV)**.
- «Ростелеком» и впредь намерен укреплять **технологическое лидерство**. Новые технологии позволят **упростить сетевую инфраструктуру** и **снизить стоимость эксплуатации сети**.
  - старший Вице-Президент по эксплуатации сетей связи «Ростелекома» Александр Цейтлин
- «Считаю, что технологии SDN и NFV позволят существенно сократить капитальные затраты и **ускорить ввод в строй новых сервисов**»
  - исполнительный директор по технической стратегии и архитектуре «Ростелекома» Эдуард Василенко
- «Ростелеком» разыскивает **стартапы**, которые занимаются **разработкой технологий** в области SDN и NFV
  - Руководитель направления Департамента управления венчурными активами компании Сергей Шлыков

Проблемы традиционных сетей



- Зависимость от производителя
- Ошибки в реализациях сетевых протоколов
- Миллионы строк закрытого проприетарного кода (6000+ RFC)
- Высокая стоимость оборудования
- Высокая стоимость эксплуатации
- Сложность управления большими сетями
- Сложность отладки
- “Закрытость” оборудования и программного обеспечения
- Сложность внедрения новых идей
- Неэффективность использования аппаратных ресурсов, энергоэффективность

Постоянный рост сложности



Source: http://www.telegeography.com/products/ip_transit/index.php; http://www.ietf.org/

Основные принципы SDN

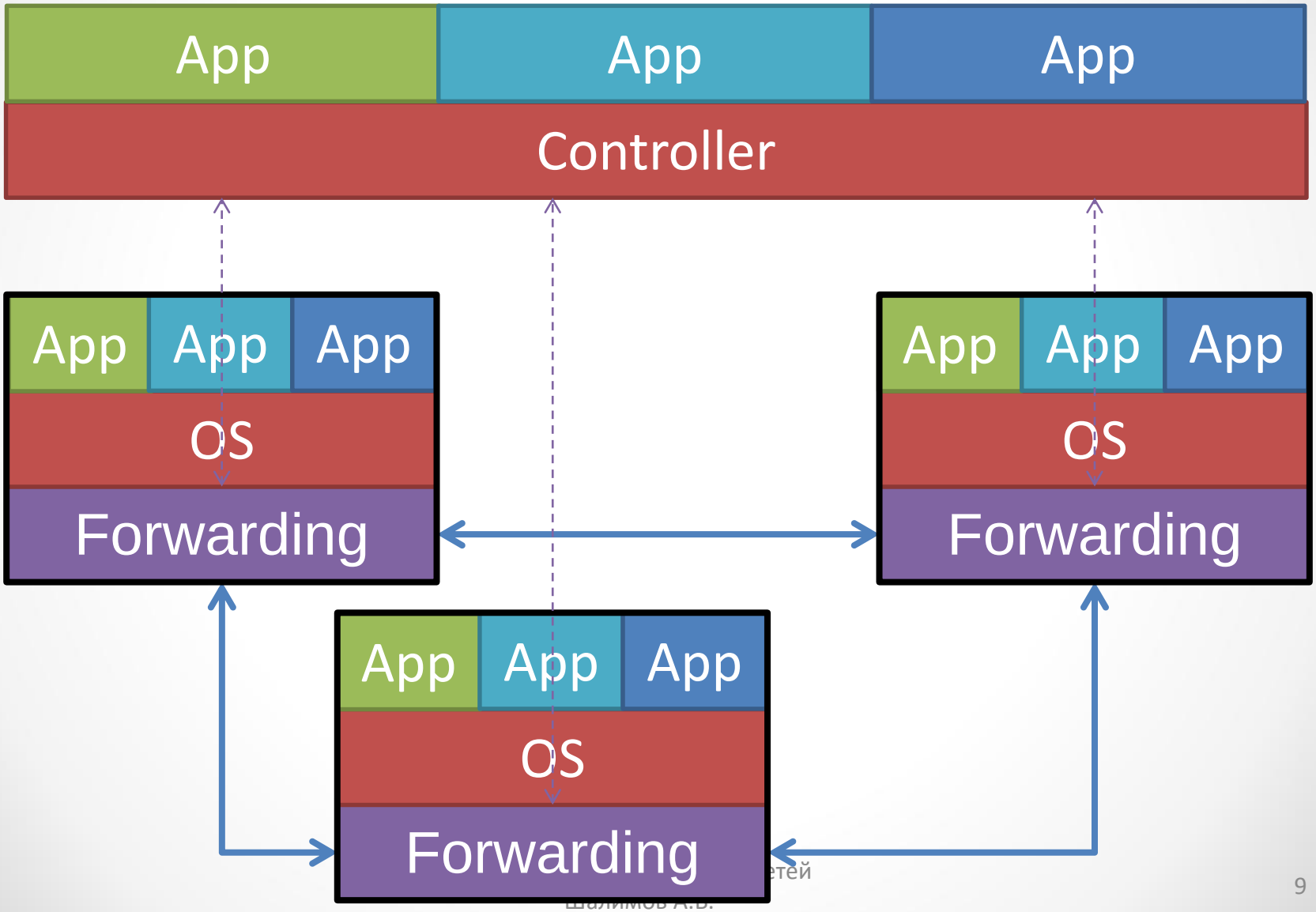
Программно-**К**онфигурируемые **С**ети (**S**oftware **D**efined **N**etworking/**SDN**) – это разделение плоскости передачи и управления данными, позволяющее осуществлять программное управление плоскостью передачи, которое может быть физически или логически отделено от аппаратных коммутаторов и маршрутизаторов

1. **Физически разделить** уровень управления сетевым оборудованием от уровня управления передачей данных.
2. Перейти от управления отдельными экземплярами сетевого оборудования к управлению сетью в целом – **логически централизованное управление**.
3. Создать **открытый программно-управляемый интерфейс** между сетевыми приложениями и транспортной сетью.

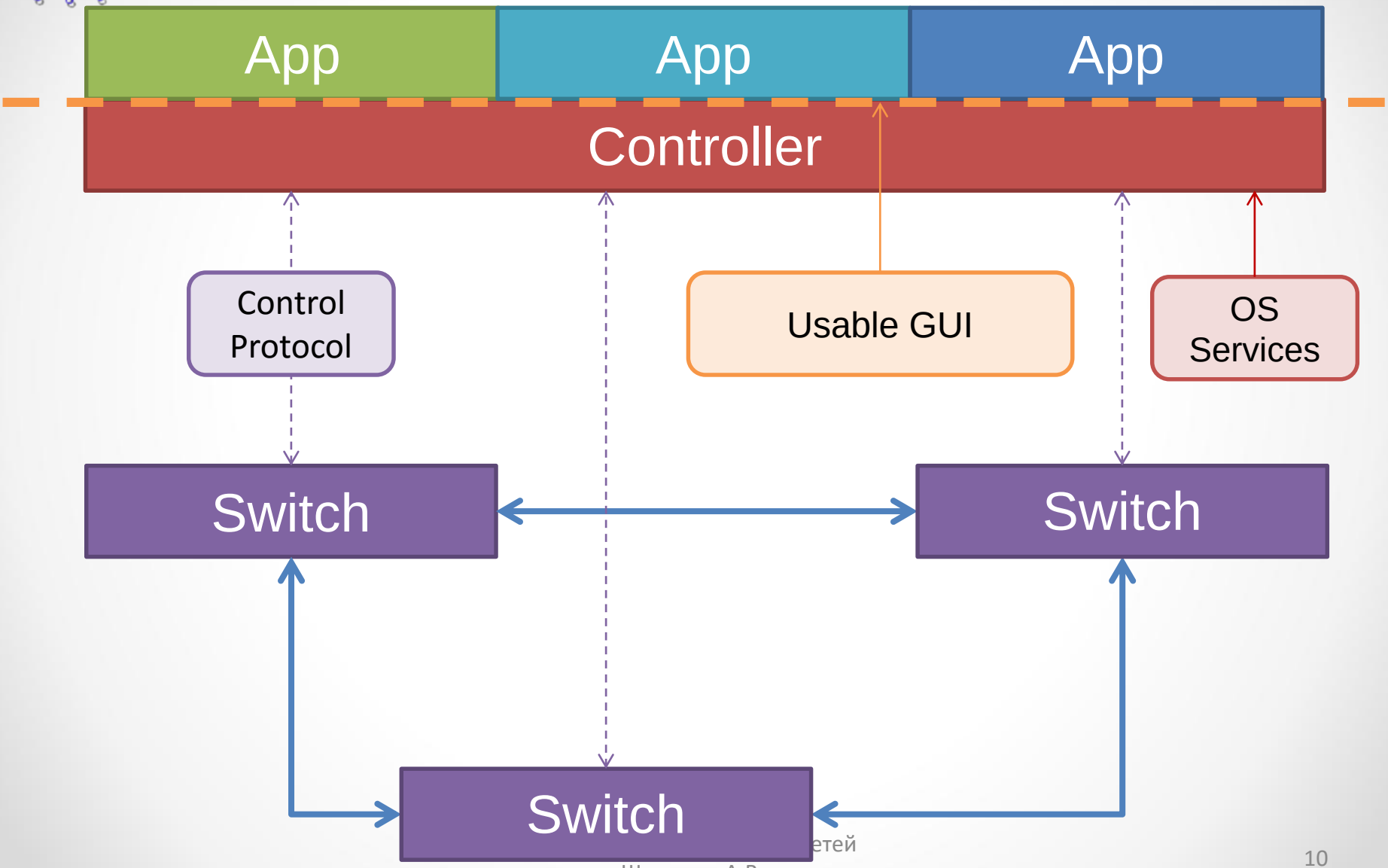
На самом деле ПКС – как четвертое поколение сотовых телефонов, только в сфере сетевых технологий :

- Новые инструменты и функции;
- Простота администрирования;
- Открытость инновациям и экспериментам;
- Революция на ИТ-рынке

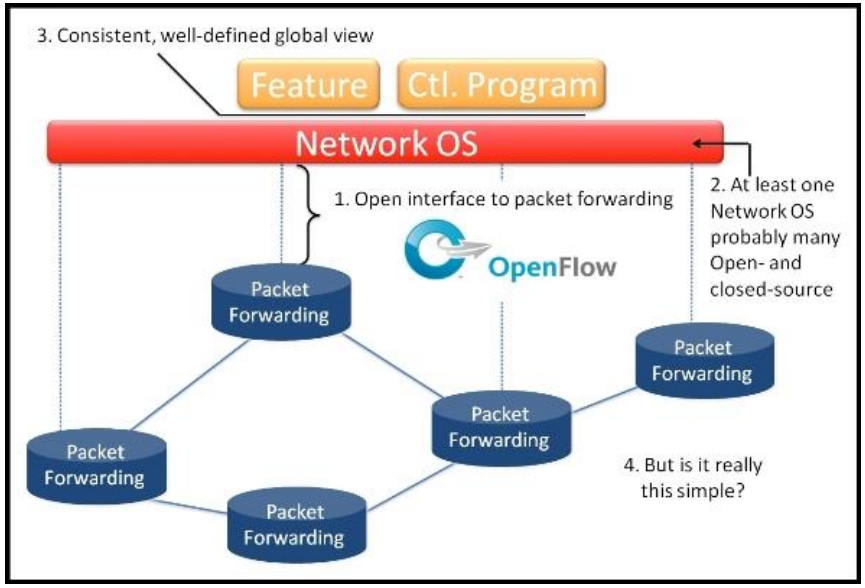
Переход к SDN



Архитектура SDN

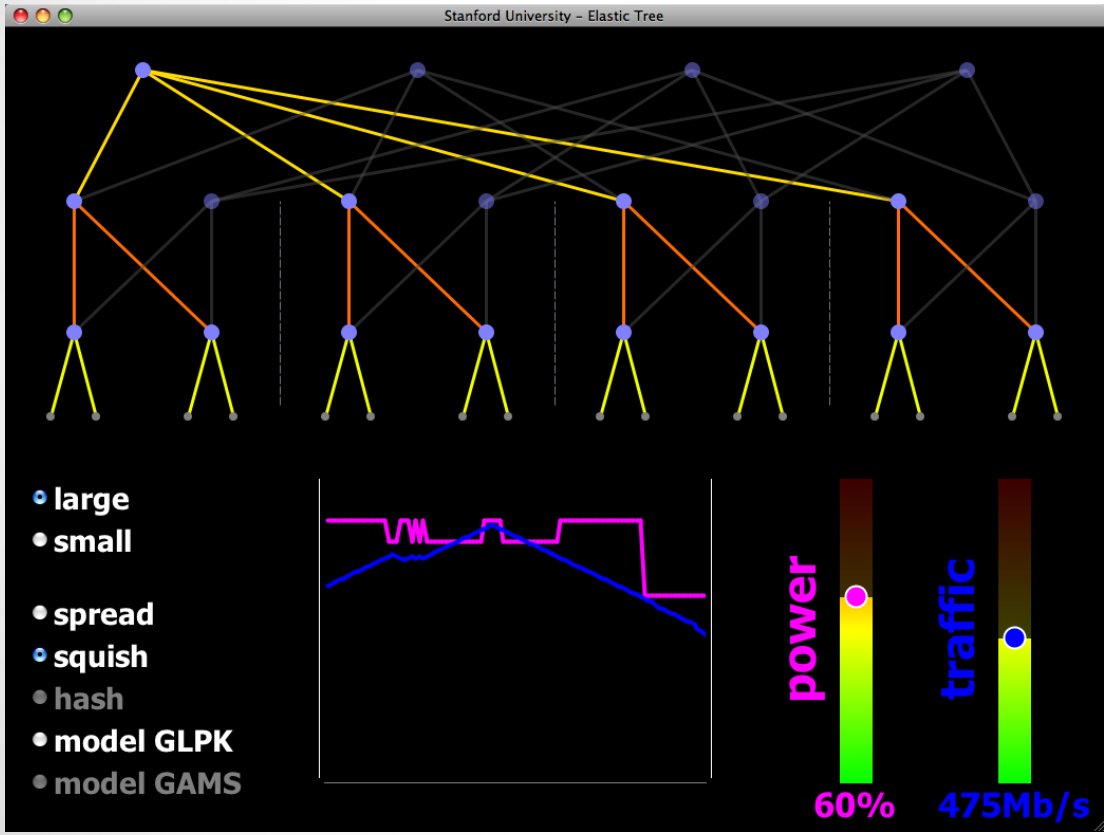


Достоинства



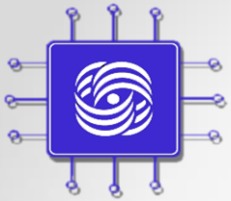
- **Удешевление оборудования (CAPEX)**
- **Облегчение управления сетью (OPEX)**
- **Программируемость, открытость, инновации**

Пример применения



Уменьшение энергопотребления в ЦОД

- Отключение неиспользуемых коммутаторов и каналов на основе собранной информации о сети
- ElasticTree (Stanford): сокращение энергопотребления до 60%
- Применение в Google

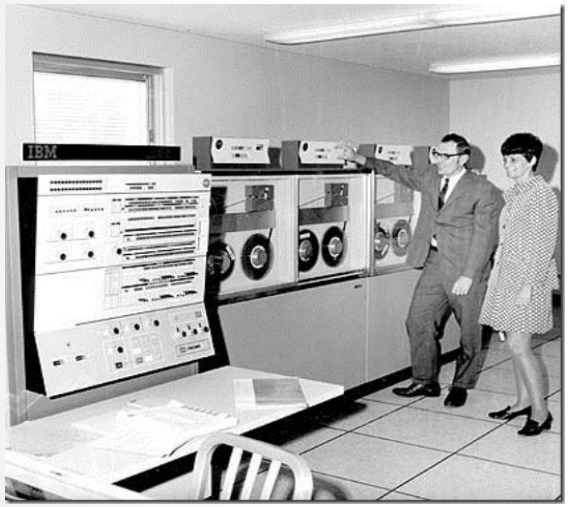


Абстракция

- Курсы по Операционным системам учат фундаментальным принципам:
  - Примитивы синхронизации, потоки, исключения, файловая система и т.д.
  - Новые языки программирования, операционные системы
- Курса по Сетевым технология учат куче протоколов
  - TCP, UDP, ARP, MPLS, GRE, BGP, OSPF, IS-IS, LDP, RSVP, PIM, ….
  - Отсутствие фундаментальных принципов, только руководства по эксплуатации сетей
  - Алгоритмы маршрутизации одни и теже много лет, управление сетью примитивно

Абстракции в IT

Медленно развивающаяся, закрытая, дорогая система. Малый рынок сбыта



Специализированные программы

Специализированная операционная система

Специализированная аппаратура

Быстрое внедрение инноваций
Открытые интерфейсы
Большой рынок сбыта

Приложения

Открытый интерфейс

Операционные системы

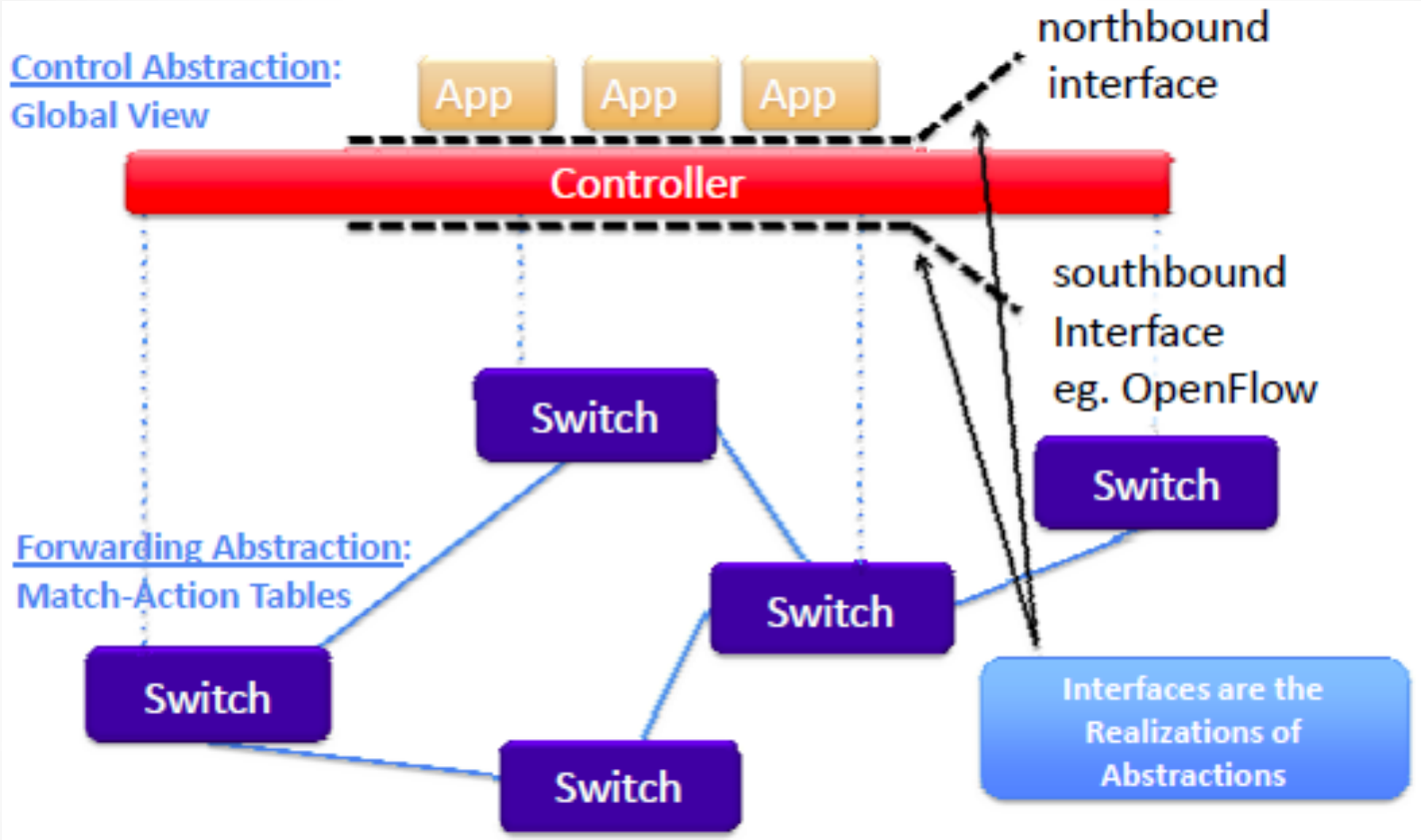
Windows iOS

Открытый интерфейс

Микропроцессоры

intel AMD QUALCOMM

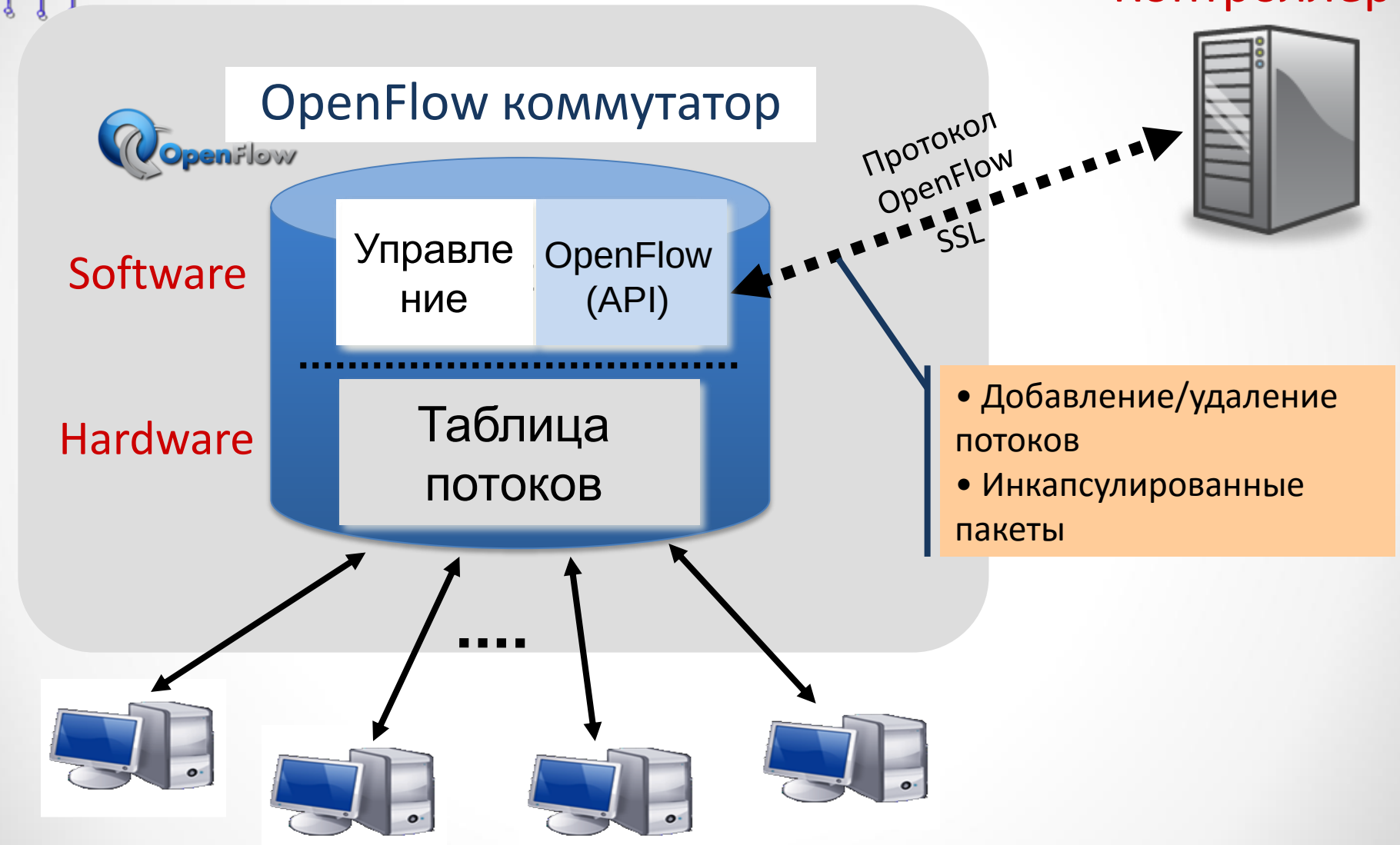
Абстракция в SDN?



- Абстракция уровня управления
- Абстракция уровня передачи данных

Часть II: OpenFlow

OpenFlow


Контроллер
OpenFlow коммутатор
OpenFlow
Software
Hardware
Управление
OpenFlow (API)
Таблица потоков
Протокол OpenFlow SSL
• Добавление/удаление потоков
• Инкапсулированные пакеты
....

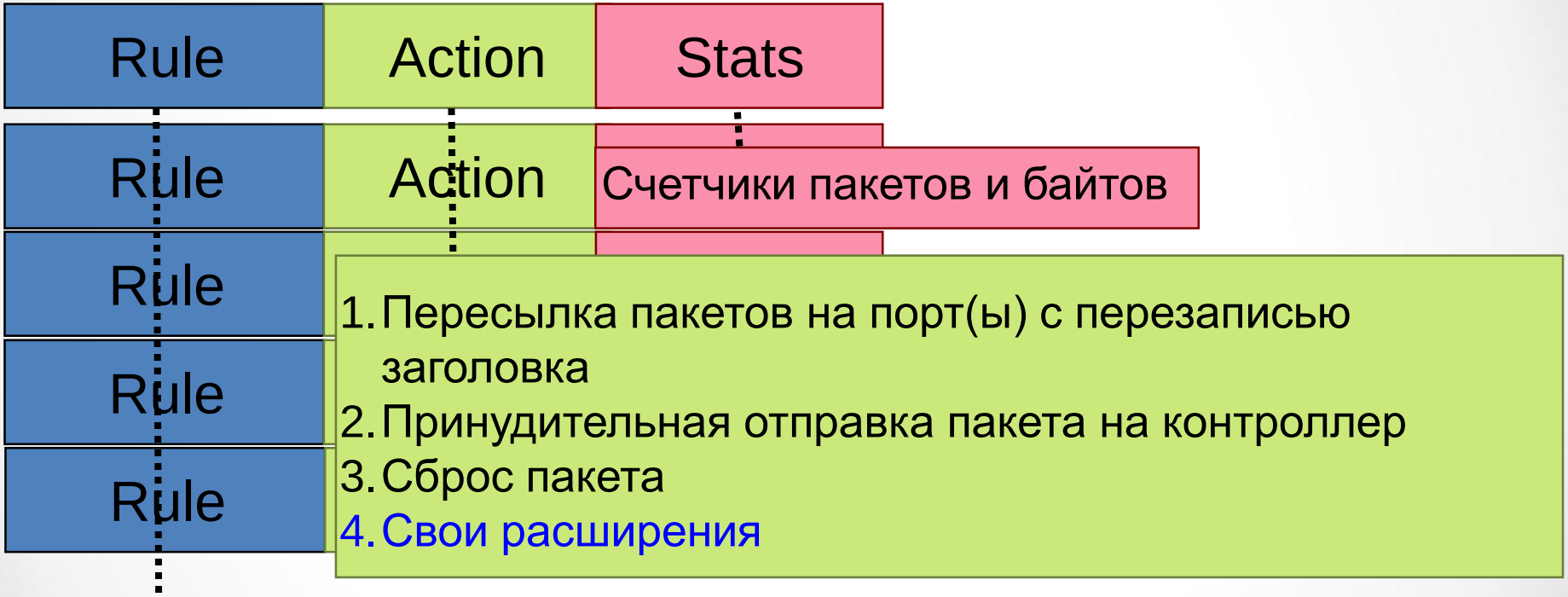
OpenFlow протокол

Поддерживается три типа сообщений:

- Сообщения контроллер-коммутатор
  - Конфигурирование коммутатора
  - Управление и контроль состояния
  - Управление таблицами потоков
  - Features, Configuration, Modify-State (**flow-mod**), Read-State (multipart request), **Packet-out**, Barrier, Role-Request
- Симметричные сообщения
  - Отправка в обоих направлениях
  - Обнаружение проблем соединения контроллера с коммутатором
  - Hello, Echo
- Ассиметричные сообщения
  - Отправка от коммутатора к контроллеру
  - Объявляют об изменении состояния сети, состояния коммутаторов
  - **Packet-in**, flow-removed, port-status, error

OpenFlow 1.0

Flow Table



+ маска по полям

Примеры правил OpenFlow

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

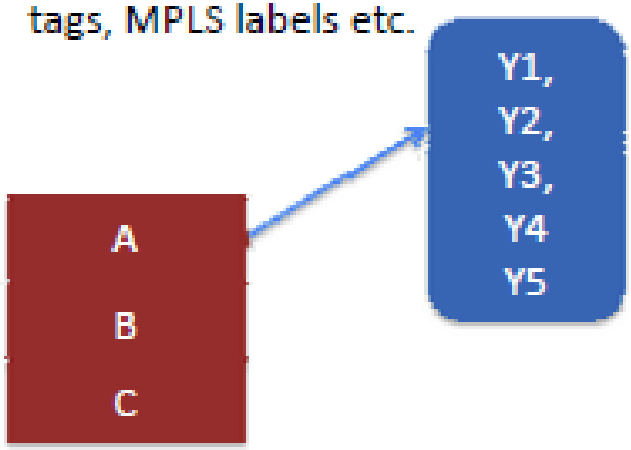
Firewall

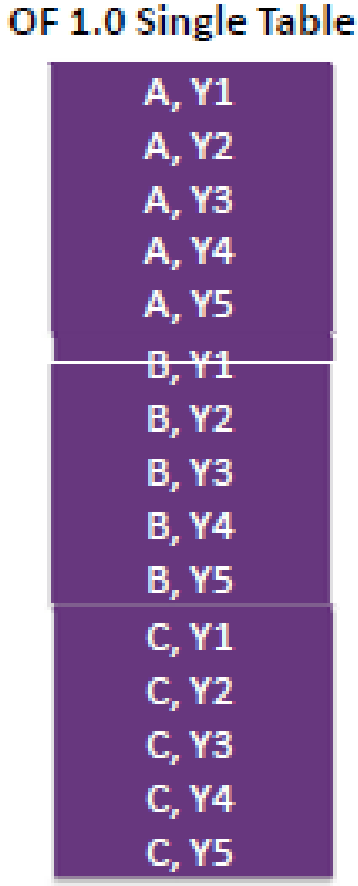
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

Чем плохо одна таблица?

- Table space explosion

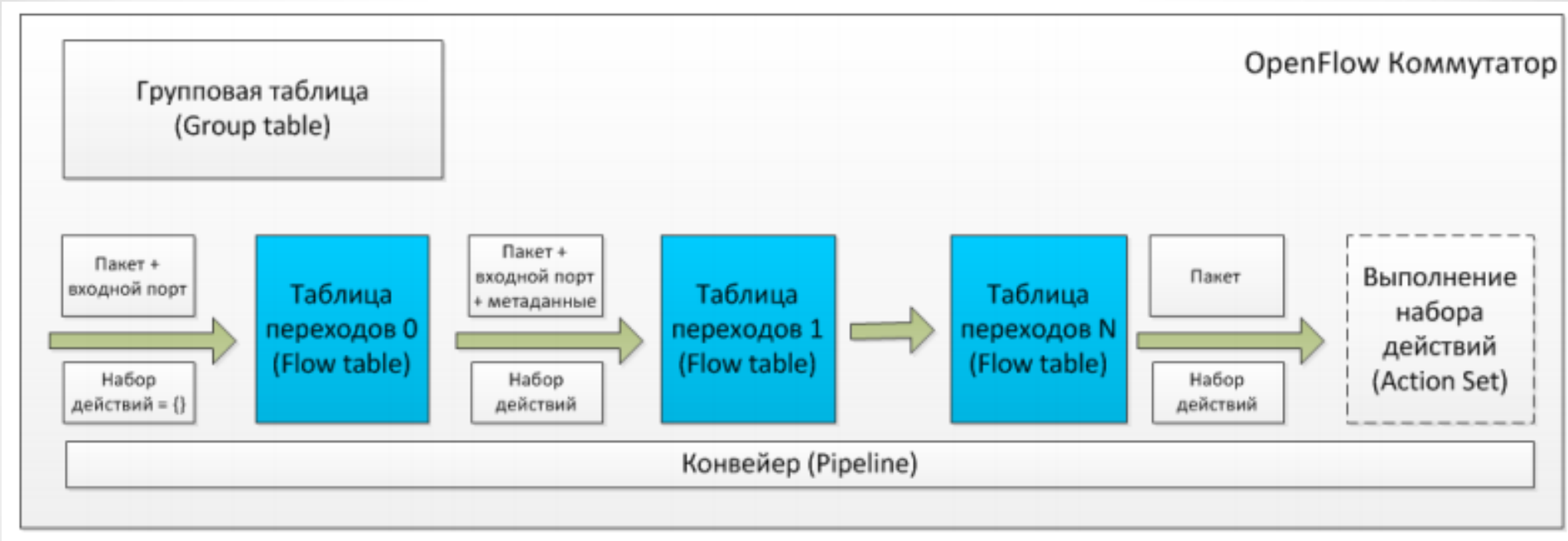
A, B, C, Y could be MAC or IP addresses, VLAN tags, MPLS labels etc.





Single-table abstraction may use table space inefficiently compared to multiple tables

OpenFlow 1.1



- Продвижение пакета только вперед
- Переход: модификация пакета, обновление набора действий, обновление метаданных

Групповые таблицы

Идентификатор группы	Тип группы	Счётчики	Контейнеры действий

Определены следующие типы групп:

All - выполняются все контейнеры действий в группе.

Select - выполняется только один контейнер действий в группе.

Indirect - выполняется один определённый контейнер действий в группе.

Fast failover - выполняется первый существующий (живой) контейнер действий.

- Экономия места для одинаковых действий
- Также для реализации сетевых механизмов:
  - Multicast
  - ECMP
  - Active/Standby маршруты

Meter table

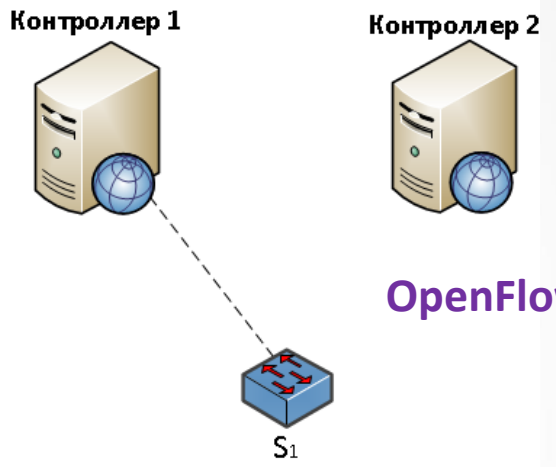
- Для реализации QoS и ограничения скорости
  - Для каждого потока или группы потоков
  - Следит за превышение значений счетчиков
  - Действия: **drop** или **dscp remark**

Meter Identifier	Meter Bands	Counters

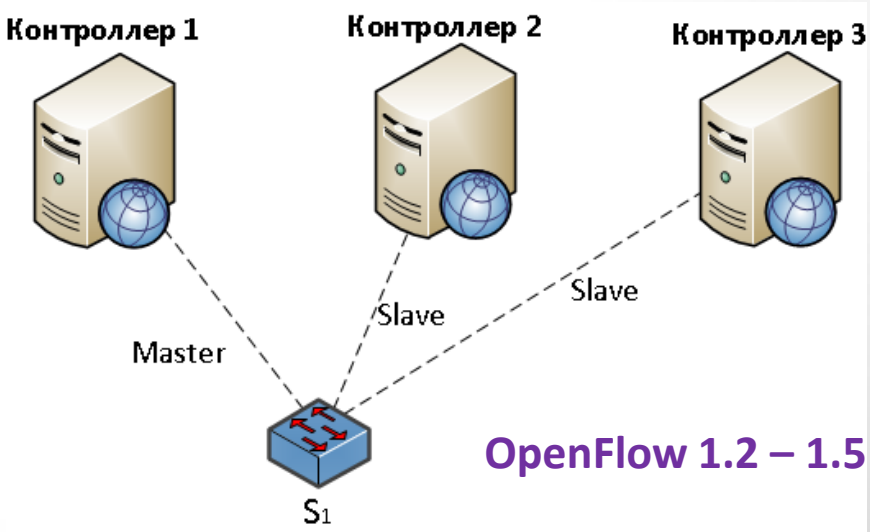
Band Type	Rate	Counters	Type specific arguments

Несколько контроллеров

- Протокол OpenFlow 1.2:
  - Множество контроллеров
  - Механизм ролей
  - **Роли:** Master, Slave, Equal
  - **По умолчанию:** контроллер находится в роли Equal для коммутаторов.
  - **Смена роли:** OFPT_ROLE_REQUEST
  - **Распределение ролей:** возложено на контроллеры.

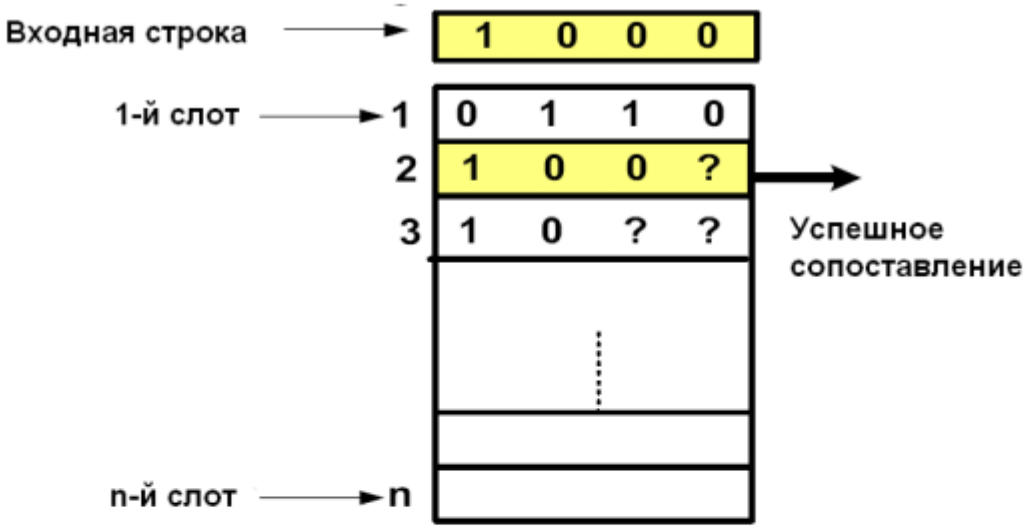


OpenFlow 1.0, 1.1

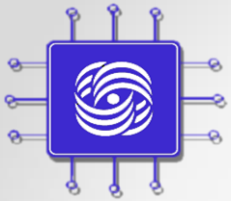


OpenFlow 1.2 – 1.5

TCAM троичная ассоциативная память



- Множество нумерованных слотов
- Три возможных значения каждого бита: “0”, “1” и “?”
- Ширина TCAM (длина слота) – настраиваемый параметр
- На вход подается битовая строка
- TCAM выдает номер первого слота с успешным сопоставлением
- Фиксированное время каждого такта работы TCAM



OpenFlow контроллер

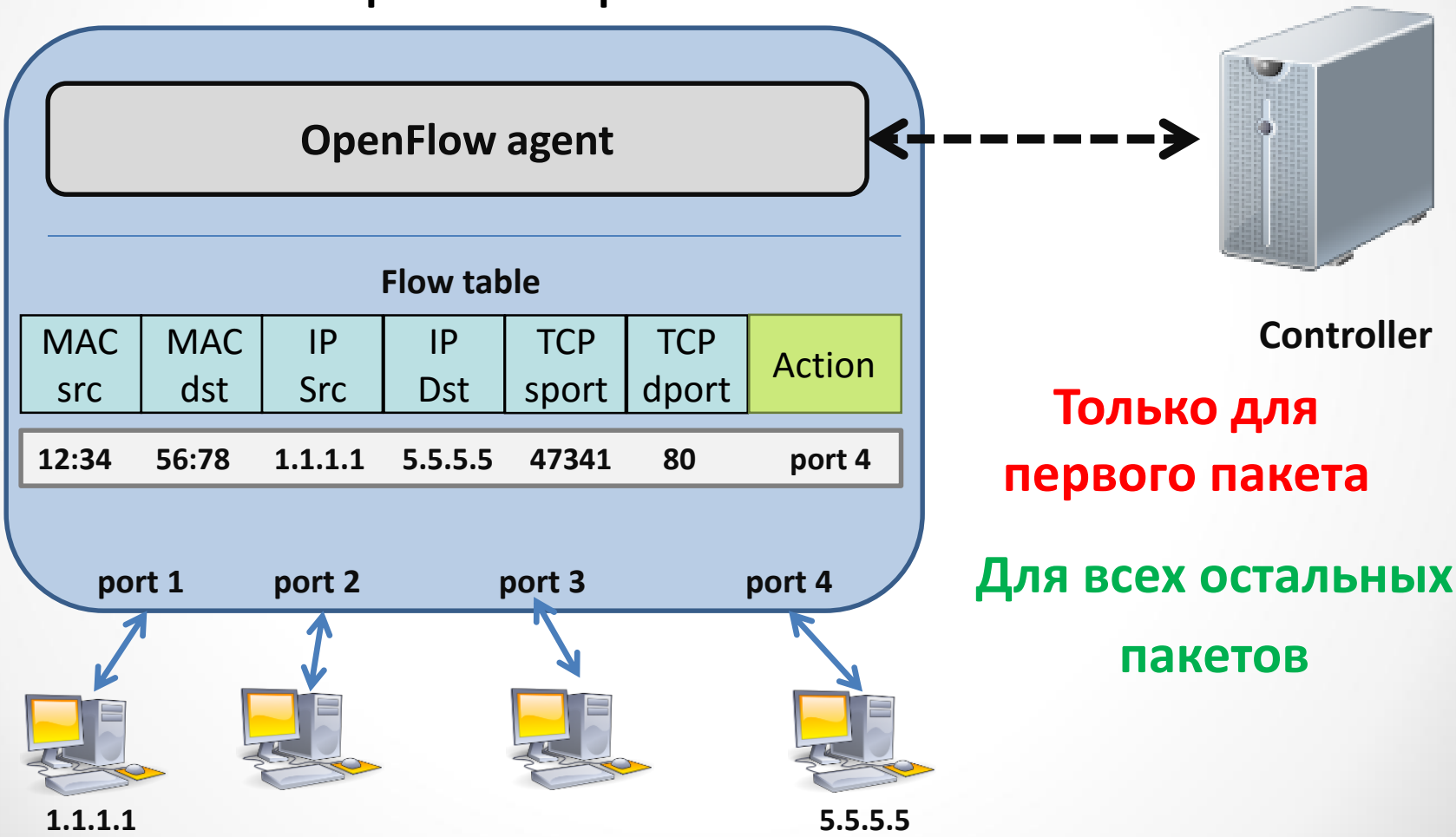
- Программа, TCP/IP сервер, ожидающий подключения коммутаторов
- Отвечает за обеспечение взаимодействие приложения-коммутатор.
- Предоставляет важные сервисы (например, построение топологии, мониторинг хостов)
- API сетевой ОС или контроллер предоставляет возможность создавать приложения на основе **централизованной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

Список OpenFlow контроллеров

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

Схема работы OpenFlow

Реактивный режим работы

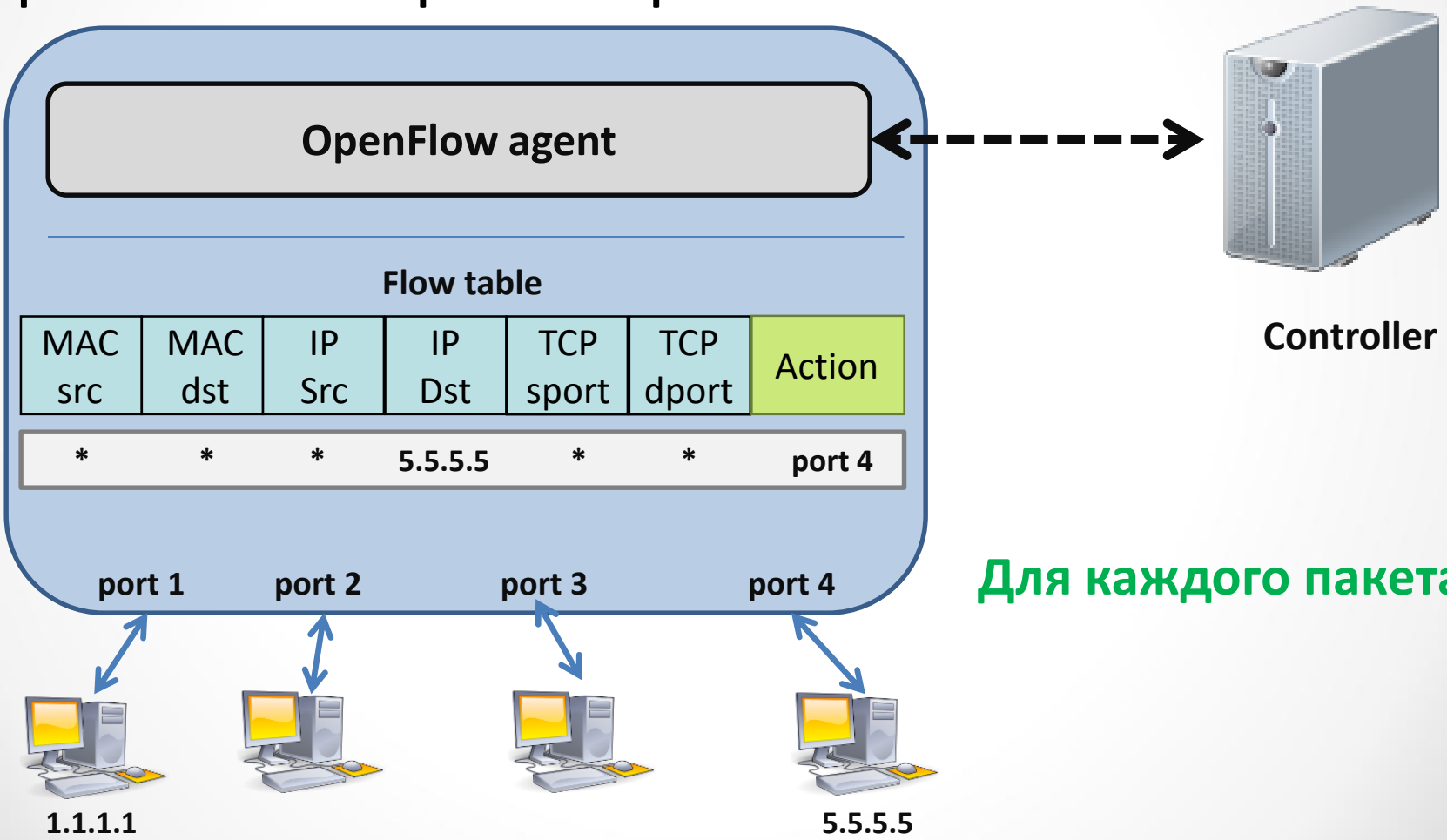


Только для первого пакета

Для всех остальных пакетов

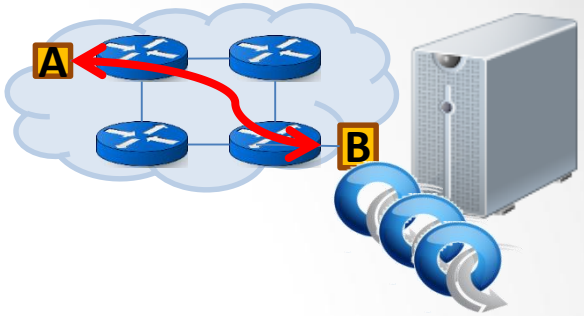
Схема работы OpenFlow

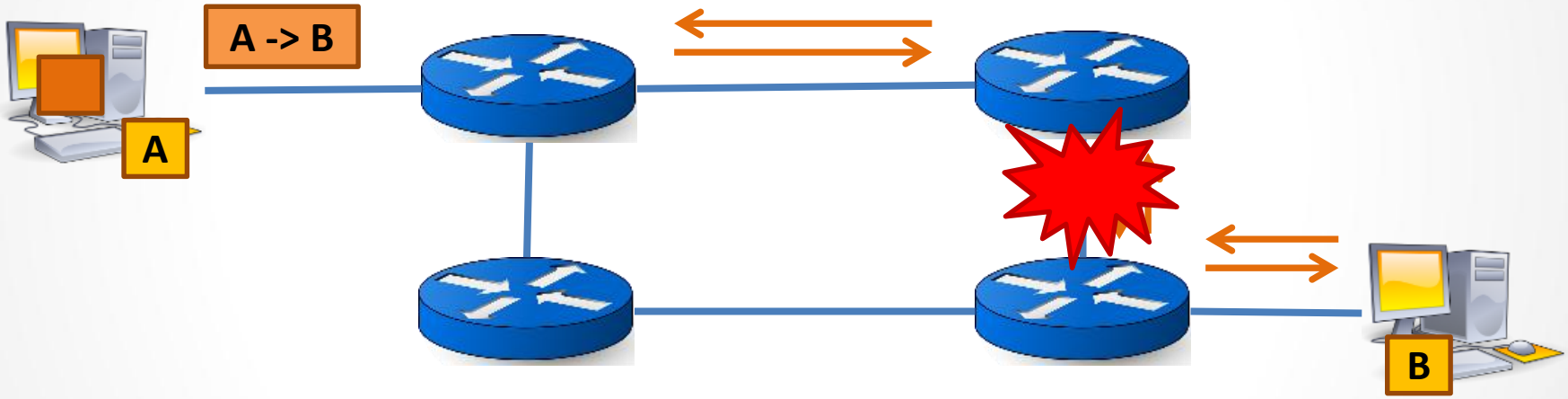
Проактивный режим работы



Для каждого пакета

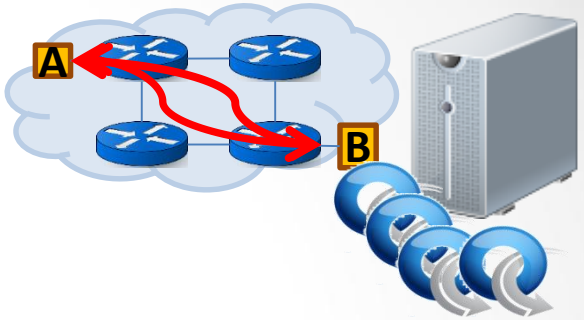
Маршрутизация с SDN/OpenFlow

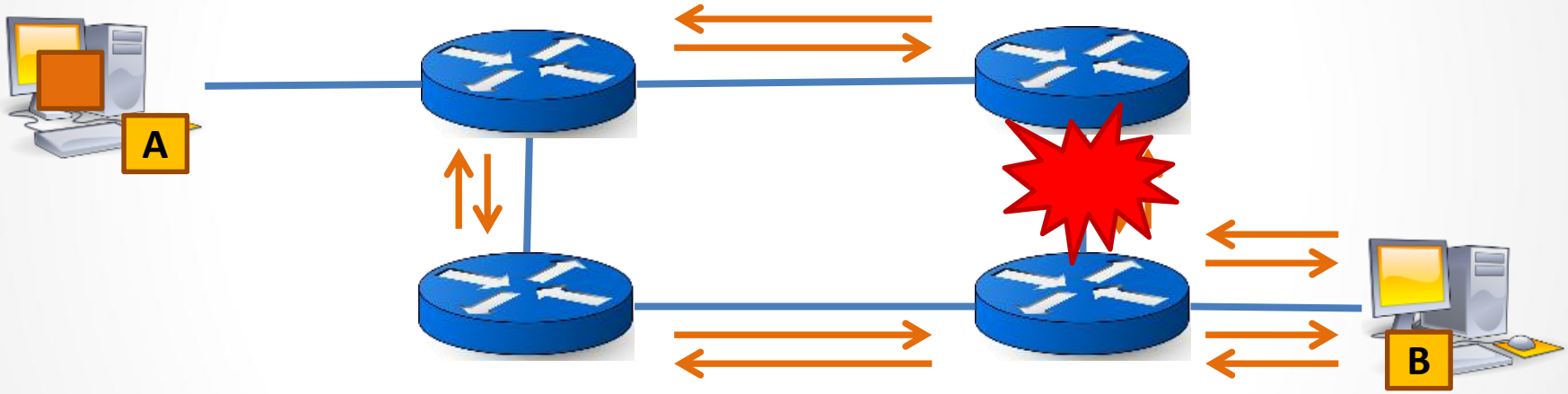




- Неизвестный пакет отправляется на контроллер (OF_PACKET_IN).
- Контроллер вычисляет лучший маршрут через всю сеть (с наименьшей стоимостью и удовлетворяющий политикам маршрутизации).
- Соответствующие правила OpenFlow устанавливаются на коммутаторы + сразу и обратный маршрут (OF_PACKET_OUT/FLOW_MOD).

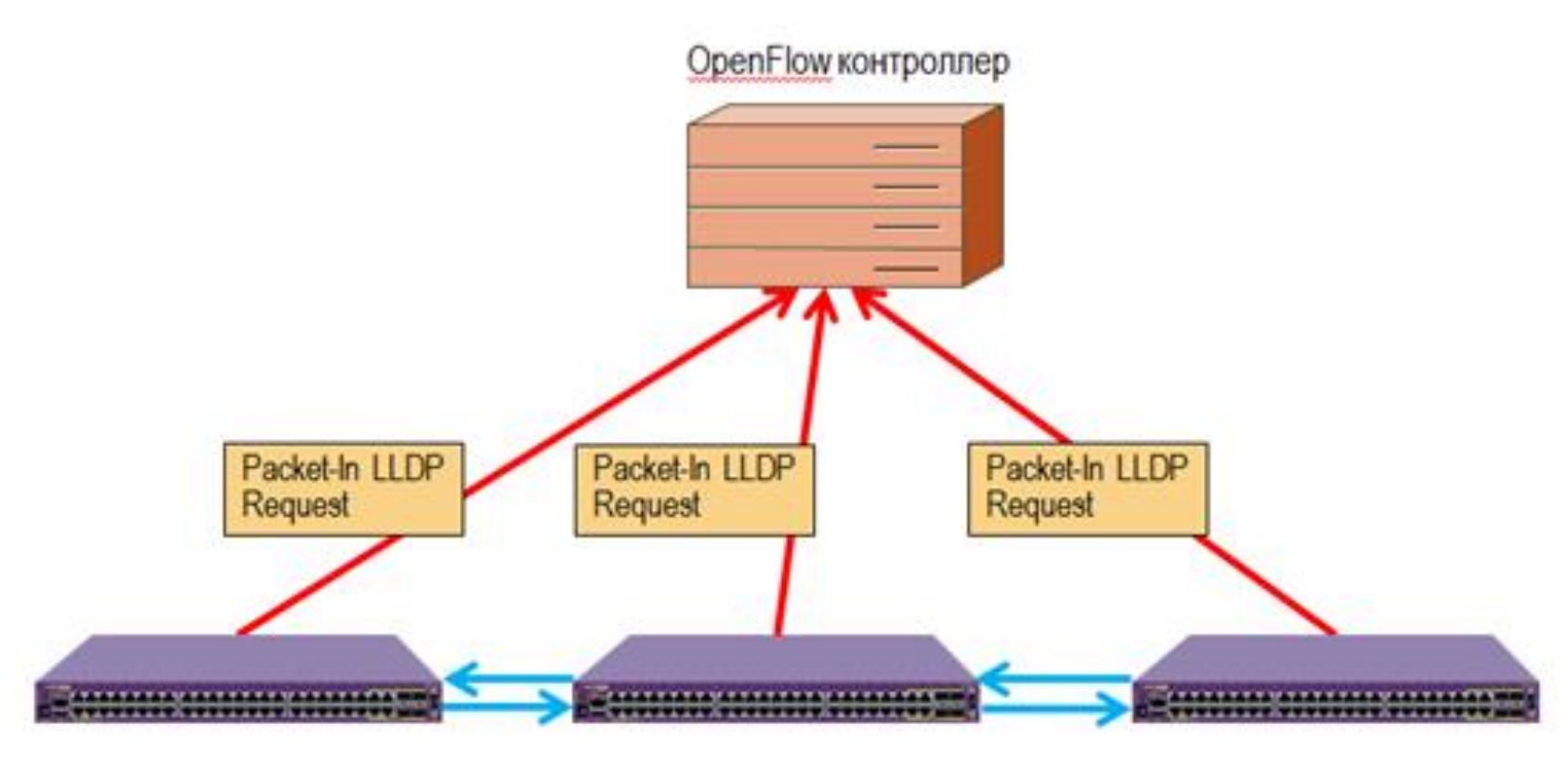
Маршрутизация с SDN/OpenFlow

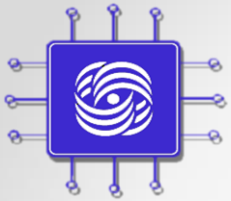




- Неизвестный пакет отправляется на контроллер (OF_PACKET_IN).
- Контроллер вычисляет лучший маршрут через всю сеть (с наименьшей стоимостью и удовлетворяющий политикам маршрутизации).
- Соответствующие правила OpenFlow устанавливаются на коммутаторы + сразу и обратный маршрут (OF_PACKET_OUT/FLOW_MOD).
- **Динамическая переконфигурация в случае ошибки сети.**

Построение топологии?

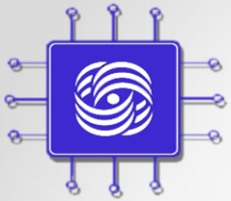




Абстракции OpenFlow

- Уровень управления – forwarding, управление пересылкой пакетов
- Уровень передачи данных – match-action таблицы

Часть III: Варианты применения SDN/OpenFlow

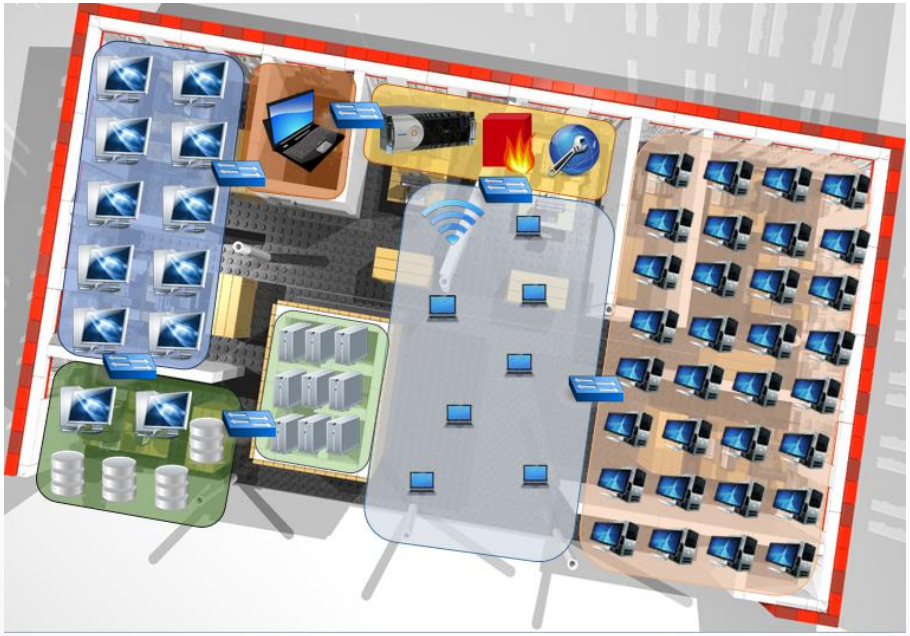


Области применения

- Компании
- Телеком операторы и сервис провайдеры
- ЦОД и облака

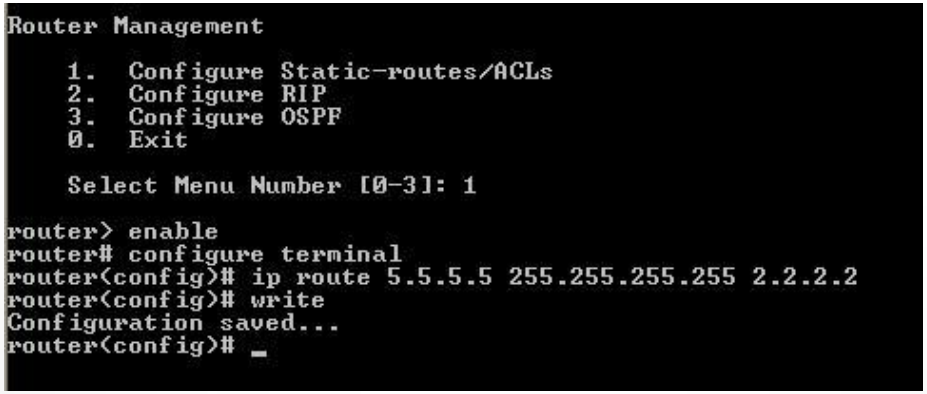
Корпоративная сеть

- Современные компания имеют сложную сетевую инфраструктуру:
  - Большое количество сетевых элементов
  - Разветвленная топология
  - Набор различных политик маршрутизации и безопасности



Трудности администрирования

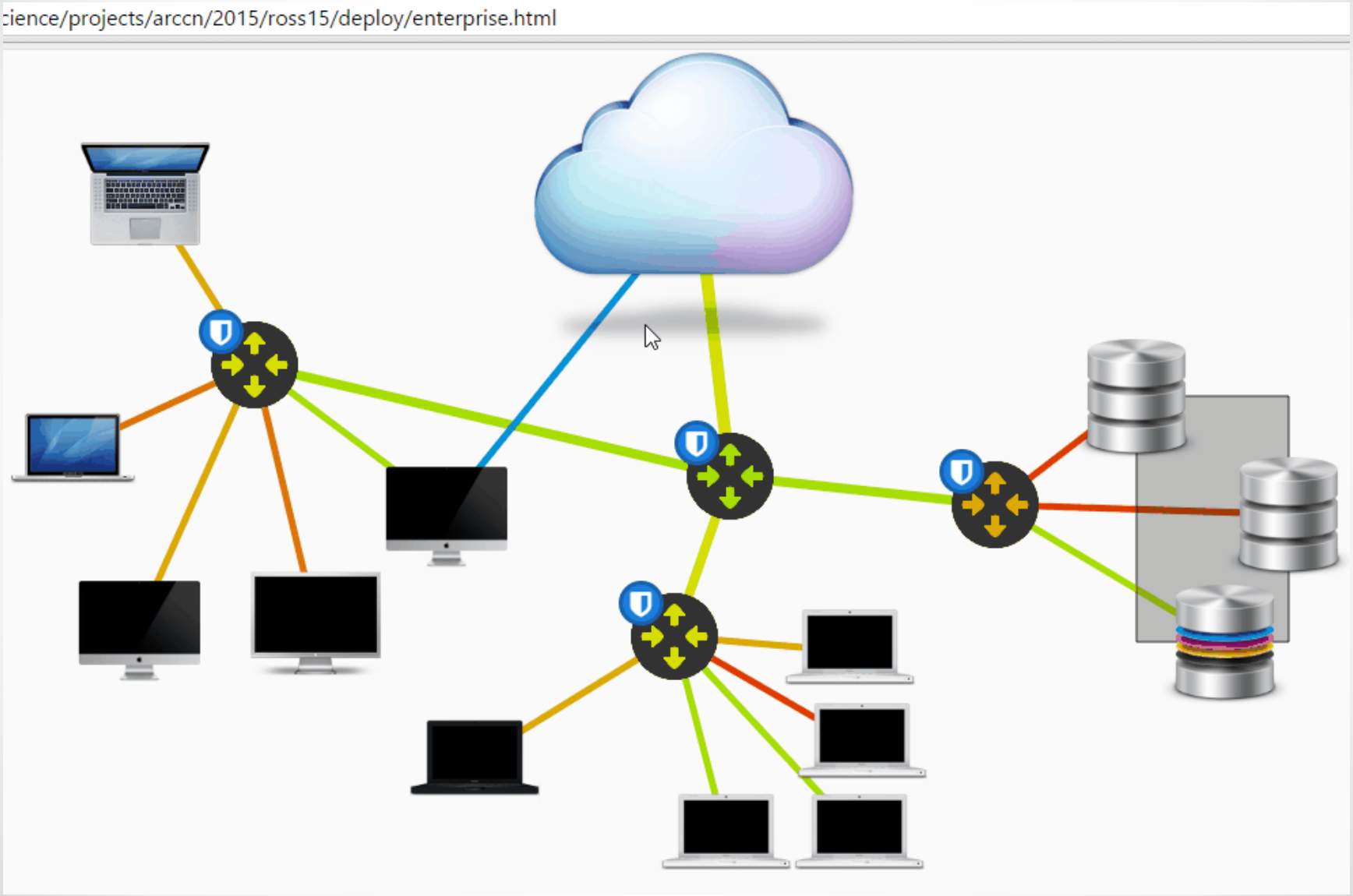
- Сетевые администраторы отвечают за поддержания работы сетевой инфраструктуры:
  - Сетевые инженеры руками переводят высокоуровневые политики в низкоуровневые команды
  - Ручная настройка всех сетевых устройств
  - Ограниченный инструментарий по управлению сетевыми устройствами
  - Переучивание под каждого вендора
- Существуют т.н. системы управления по SNMP, но только мониторинг состоятния, а не управление – настройска все равно в ручном режиме.



Цель

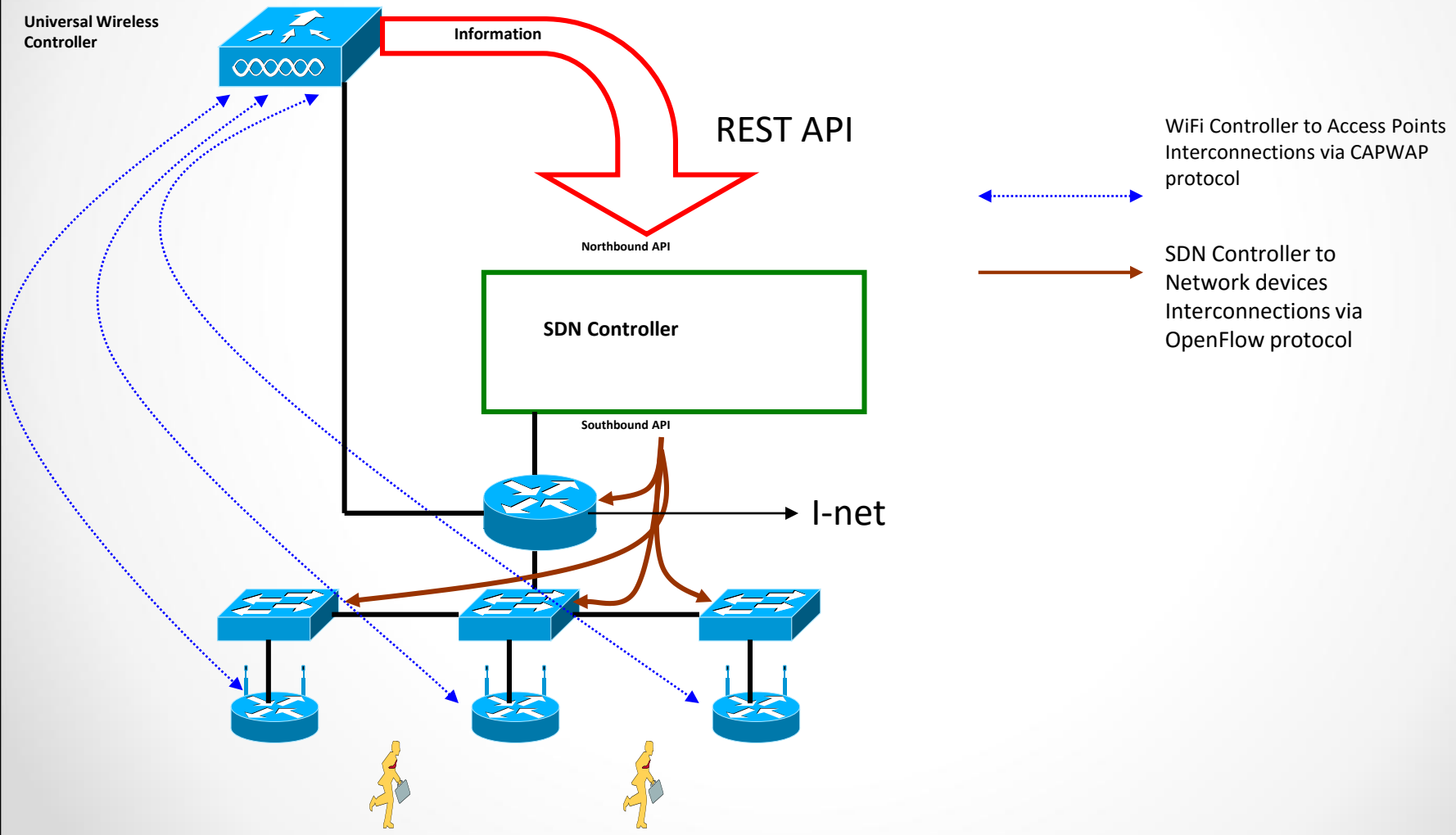
1. Сделать сеть управляемой без ручного доступа к оборудования.
2. Повысить уровень абстракции управления сетью.

Как это выглядит?



WiFi Controller & SDN Networks

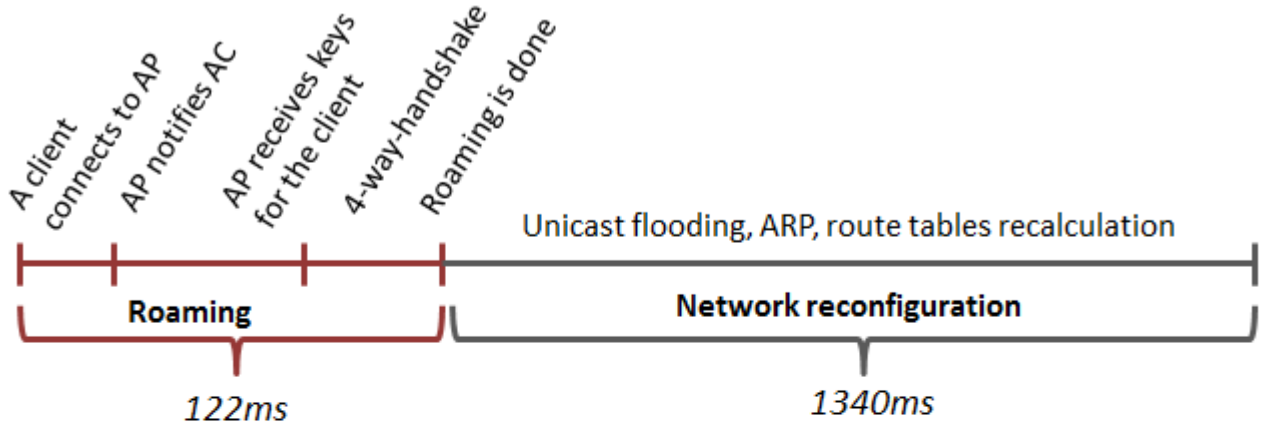
The principle of interaction with SDN controller over Northbound API:

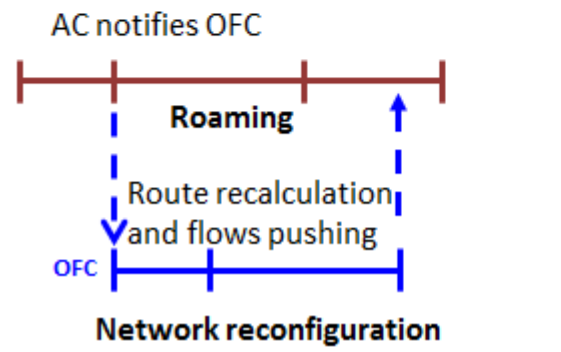


WiFi Controller & SDN Networks

Test bed consists of:

- Three hardware OpenFlow switches Extreme Networks SummitX 460t
- Two TP-LINK access points
- A laptop moving from one AP to another one and running ping command to the outside server
- Both controllers run on the same server



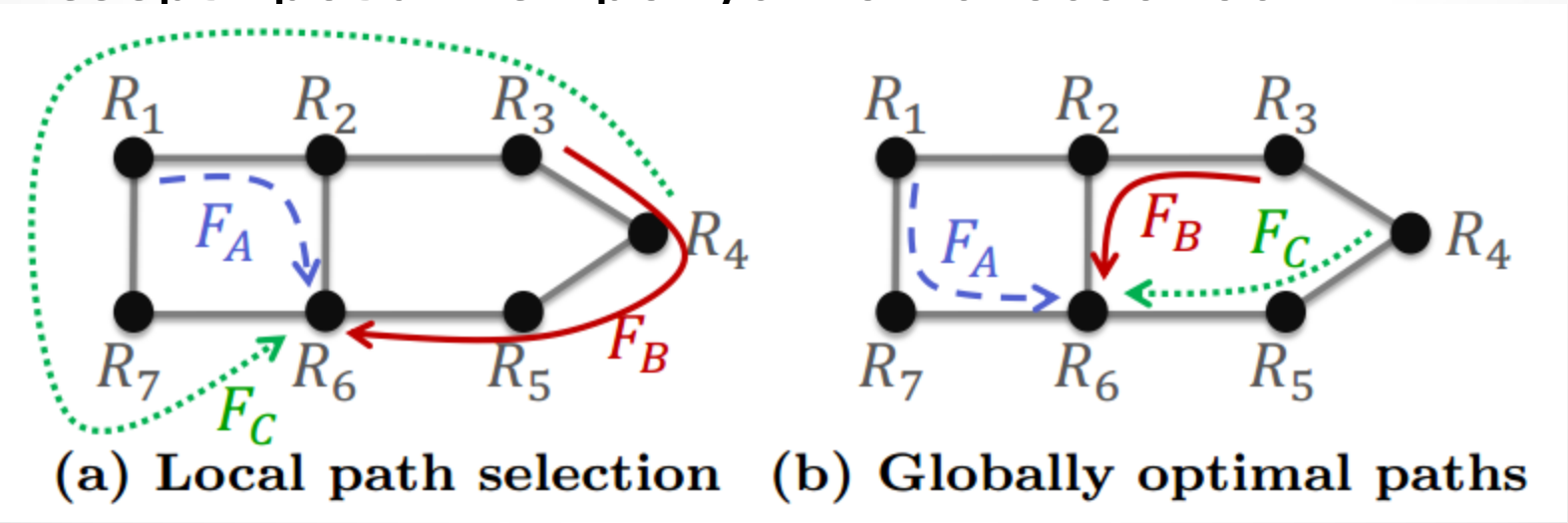


- The legacy network needs in average 1.5 seconds to reconfigure, while the SDN/OpenFlow network doesn't bring additional delay.
- This is because the migration procedure in Chandelle requires less than 80ms and the OpenFlow controller has enough time to reconfigure the switches.
- **Finally, we have more faster roaming with SDN/OpenFlow.**

Телеком

1. Интеллектуальный Traffic Engineering:

- Выбор оптимального пути
- Реакция на отказ канала
- Резервирование пропускной способности



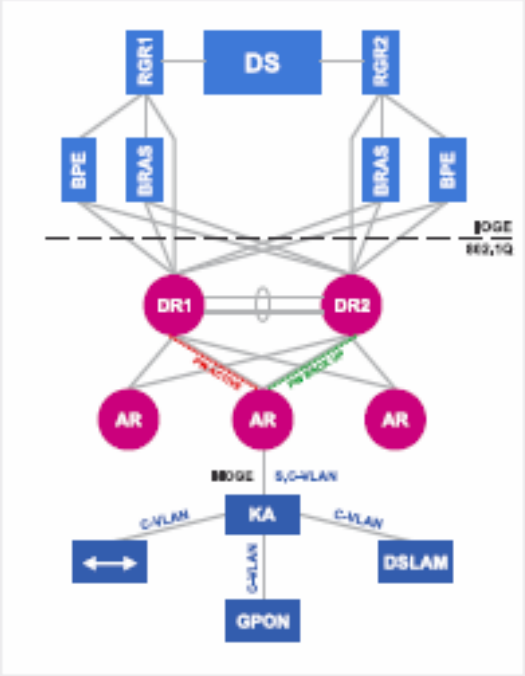
Телеком

- Как применить все это на практике?
  - Greenfield?
  - Проблемы интеграции с традиционной сетью
    - Нужно подыгрывать протоколам традиционной сети, т.е. правильно отвечать на запросы.
    - Чем меньше стыков с традиционной сетью, тем лучше.
  - Проблема интеграции с существующими системами управления

WAN segment (Service Provide)

Services:

- L2 transit for B2C, B2B/G (Internet, VPN)
- Fast backup path
- SLA
- IPTV multicast
- VoIP
- Mobile backhaul



Before:

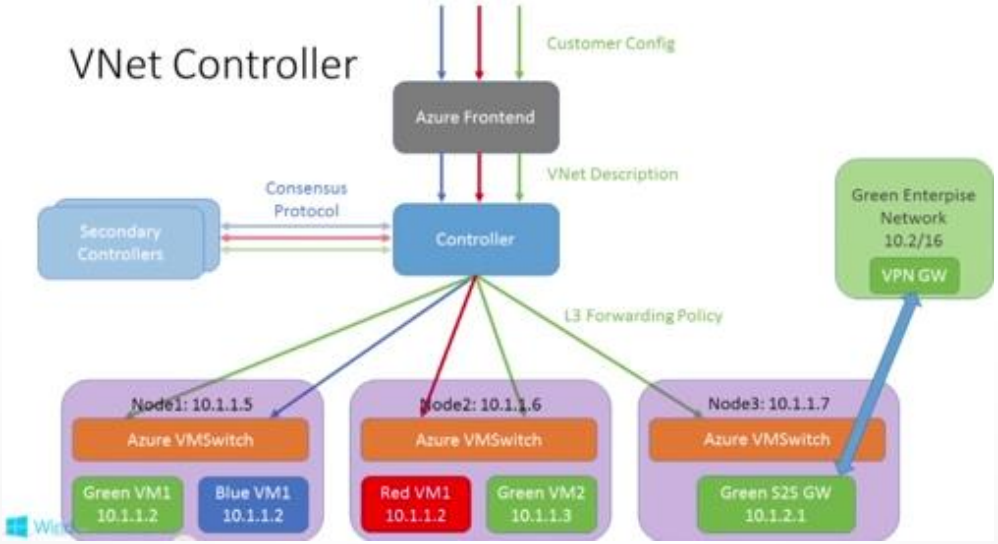
- IS-IS (RFC 1195)
- OSPF
- PIM-SSM
- PIM-SM
- LDP (RFC 3036)
- Targeted LDP
- BGP (PW для AToM)
- RSVP (RFC 2205)
- MPLS PW
- BGP (RFC 4271)
- MP-BGP (RFC 4760)
- MPLS-VPN (RFC 4364)
- MPLS (RFC 3031, 3032)

After:

- L2 PW application + stats (no encap)
- Bridge domain (no learning)
- Multicast (IGMP)
- L2 LAG, L3 ECMP
- H-QoS

ЦОД/Облака

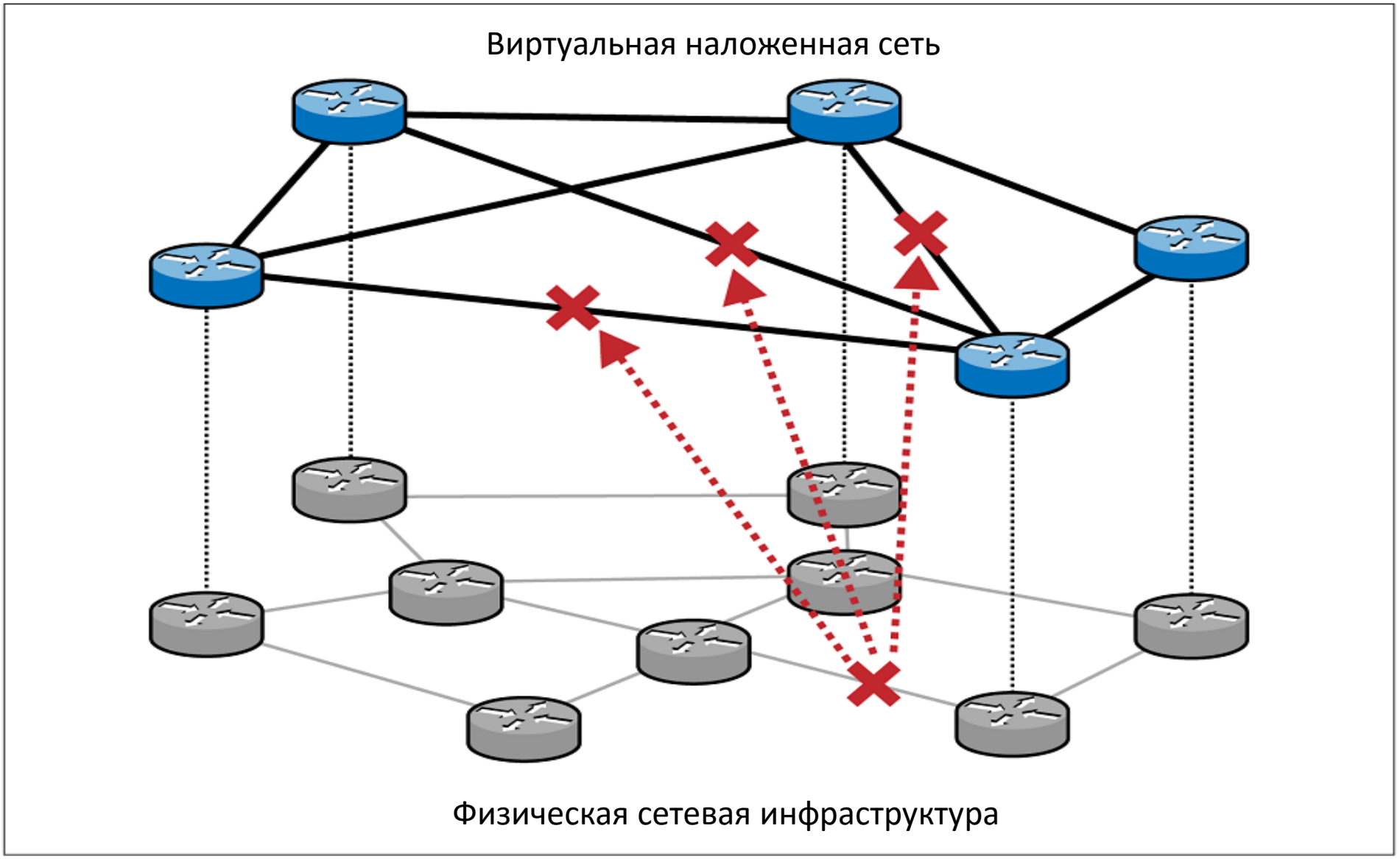
- Повышение утилизации оборудования и каналов
- Мониторинг и оптимизация потоков
- Виртуализация сети пользователей
- Балансировка нагрузки
- Обеспечение качества доступа



ЦОД/Облака

- Как правило есть два SDN
  - Без OpenFlow и так есть в OpenStack
    - ТОЛЬКО виртуальные каналы
    - Туннели, таблицы, новые VM, политики
  - С OpenFlow для управление физическими устройствами
    - Качество канала, определение узких мест

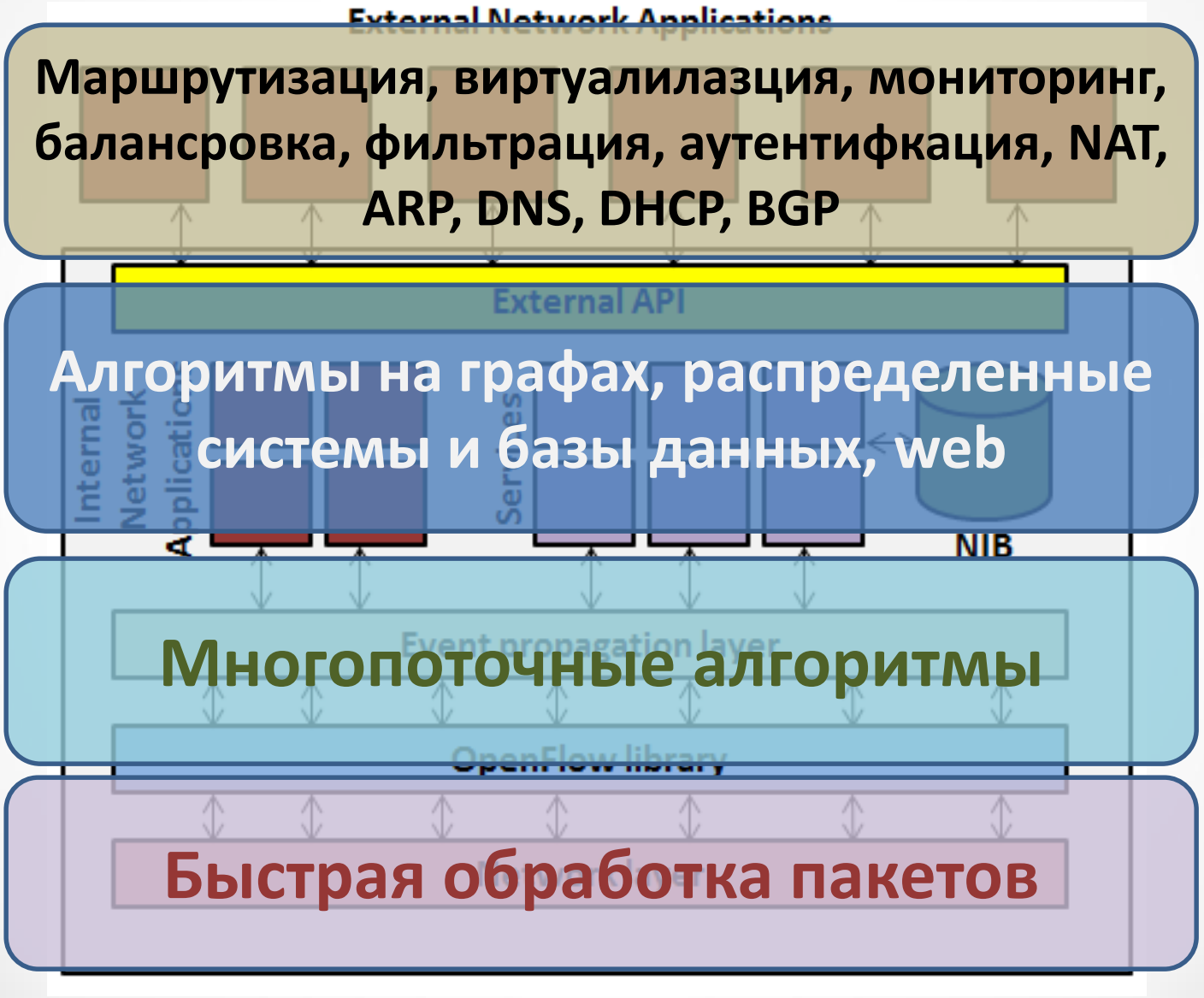
Сетевая виртуализация



Software Defined Data Center

Часть IV: OpenFlow контроллеры

Архитектура OpenFlow контроллера



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”

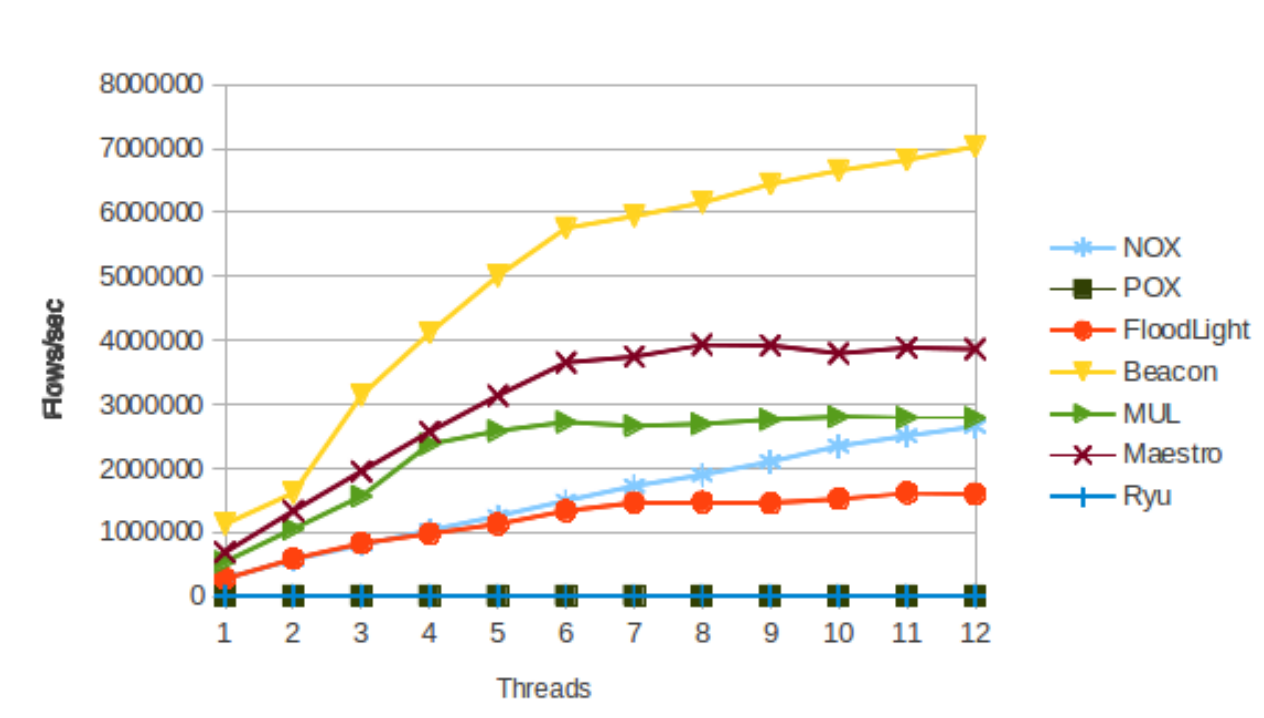
Полный список контроллеров (2013)

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

Экспериментальное исследование

- Производительность
  - максимальное количество запросов на обработку
  - время обработки запроса при заданной нагрузке
- Масштабируемость
  - изменение показателей производительности при увеличении числа соединений с коммутаторами и при увеличении числа ядер процессора
- Надежность
  - количество отказов за время тестирования при заданном профиле нагрузок
- Безопасность
  - устойчивость к некорректно сформированным сообщениям протокола OpenFlow

Результаты сравнения (2013)



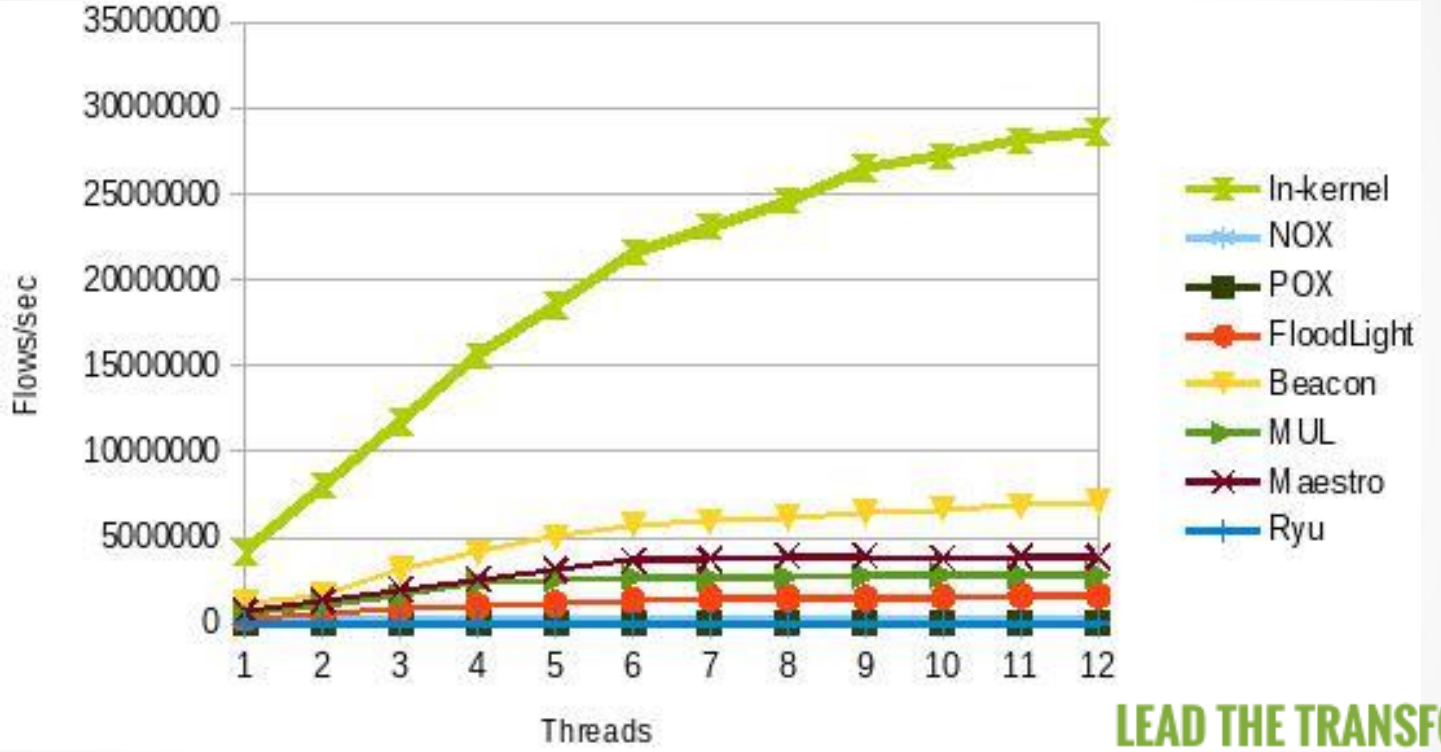
- Максимальная производительность **7 000 000 потоков в секунду**.
- Минимальное время задержки **от 50 до 75 мкс**.
- Недостатки:
  - Надежность контроллеров вызывала вопросы
  - Производительность была не достаточна (DC >10M fps)

Повышение производительности

Самые ресурсоемкие задачи:

- Взаимодействие с OpenFlow коммутаторами:
  - использование многопоточности;
  - учет загрузки нитей и перебалансировка.
- Получение OpenFlow пакетов из канала:
  - чтение пакетов из памяти сетевой карты, минуя сетевой стек OS Linux;
  - переключение контекста;
  - виртуальные адреса.

Производительность (kernel)



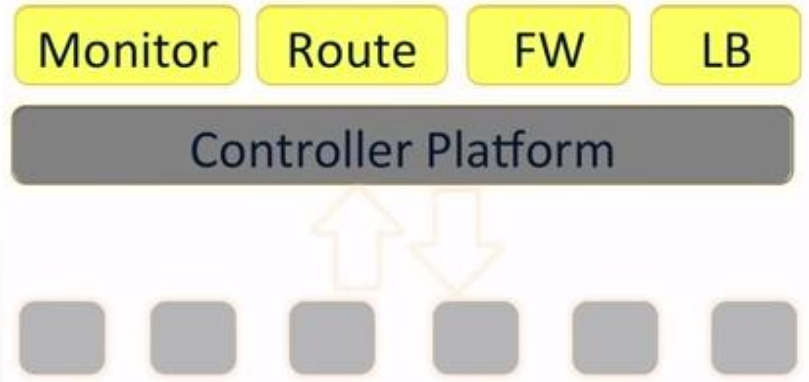
- Производительность равна **30M fps**
- Задержка **45us**

Программируемость

- На языке контроллера [быстро]
- На любом языке через REST интерфейс [медленно]
- Специальные языки программирования с другой абстракцией (например, Pyretic, Maple)

Проблематика NorthBound API

- NorthBound API – интерфейс между контроллером и приложениями
- Программирование с OpenFlow не простая задача!
  - Сложно выполнять независимый задачи (routing, access control)
  - Низкоуровневая абстракция
  - Нужно помнить о правилах на коммутаторах
  - Порядок установки правил на коммутаторах неизвестен
- Переносимость приложений между контроллерами



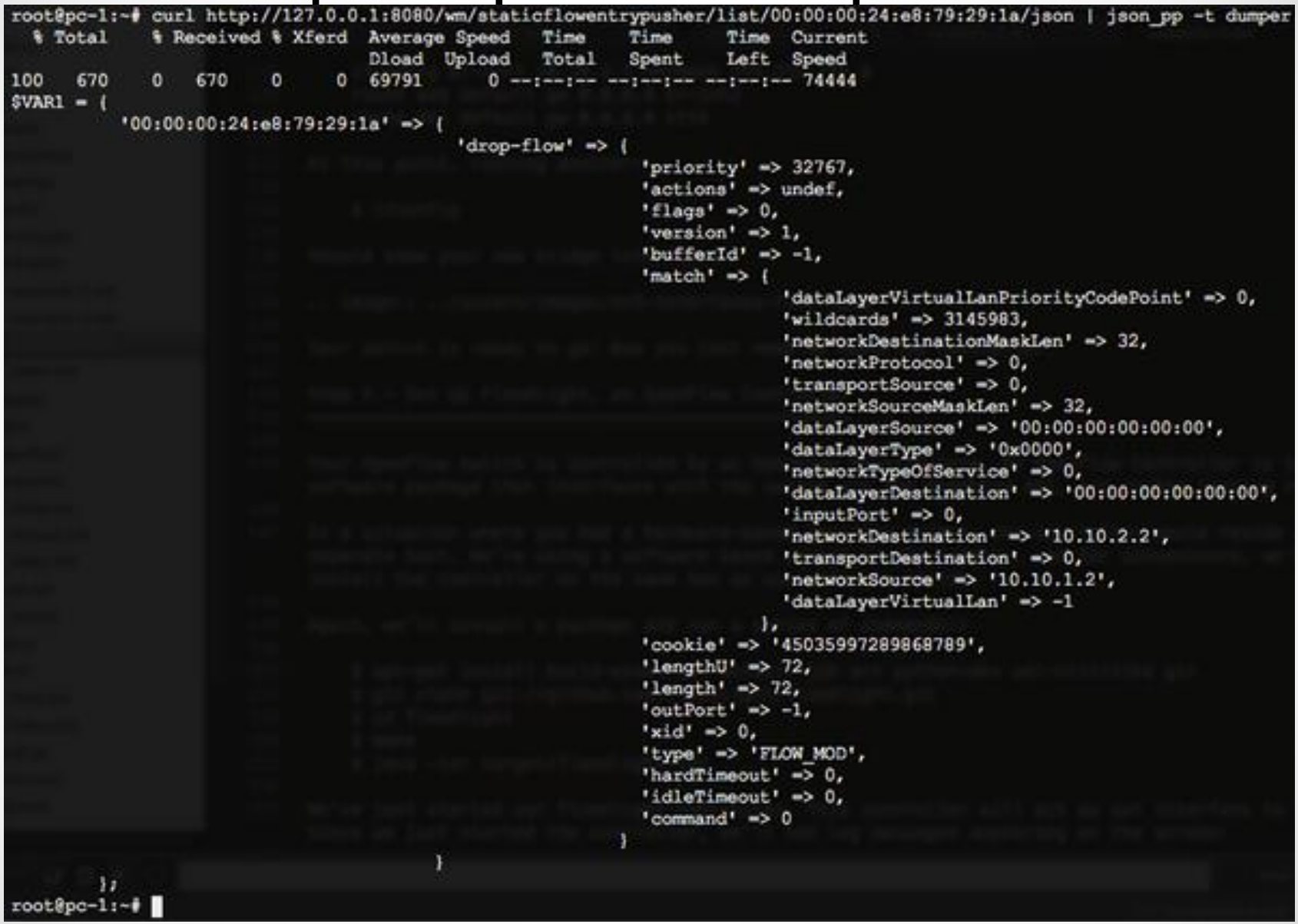
Низкоровневые детали OpenFlow

A.3.4.1 Modify Flow Entry Message

Modifications to a flow table from the controller are done with the OFPT_FLOW_MOD message:

```
/* Flow setup and teardown (controller -> datapath). */
struct ofp_flow_mod {
    struct ofp_header header;
    uint64_t cookie;              /* Opaque controller-issued identifier. */
    uint64_t cookie_mask;         /* Mask used to restrict the cookie bits
                                     that must match when the command is
                                     OFPFC_MODIFY* or OFPFC_DELETE*. A value
                                     of 0 indicates no restriction. */

    /* Flow actions. */
    uint8_t table_id;              /* ID of the table to put the flow in.
                                      For OFPFC_DELETE_* commands, OFPTT_ALL
                                      can also be used to delete matching
                                      flows from all tables. */
    uint8_t command;               /* One of OFPFC_*. */
    uint16_t idle_timeout;         /* Idle time before discarding (seconds). */
    uint16_t hard_timeout;         /* Max time before discarding (seconds). */
    uint16_t priority;             /* Priority level of flow entry. */
    uint32_t buffer_id;            /* Buffered packet to apply to, or
                                      OFP_NO_BUFFER.
                                      Not meaningful for OFPFC_DELETE*. */
    uint32_t out_port;             /* For OFPFC_DELETE* commands, require
                                      matching entries to include this as an
                                      output port.  A value of OFPP_ANY
                                      indicates no restriction. */
    uint32_t out_group;            /* For OFPFC_DELETE* commands, require
                                      matching entries to include this as an
                                      output group.  A value of OFPG_ANY
                                      indicates no restriction. */
    uint16_t flags;                /* One of OFPFF_*. */
    uint8_t pad[2];
    struct ofp_match match;        /* Fields to match. Variable size. */
    //struct ofp_instruction instructions[0]; /* Instruction set */
};
OFP_ASSERT(sizeof(struct ofp_flow_mod) == 56);
```

Пример REST запроса



```
root@pc-1:~# curl http://127.0.0.1:8080/wm/staticflowentrypusher/list/00:00:00:24:e8:79:29:1a/json | json_pp -t dumper
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   670    0   670    0     0  69791      0 --:--:-- --:--:-- --:--:-- 74444
$VAR1 = {
          '00:00:00:24:e8:79:29:1a' => {
                                         'drop-flow' => {
                                                          'priority' => 32767,
                                                          'actions' => undef,
                                                          'flags' => 0,
                                                          'version' => 1,
                                                          'bufferId' => -1,
                                                          'match' => {
                                                                       'dataLayerVirtualLanPriorityCodePoint' => 0,
                                                                       'wildcards' => 3145983,
                                                                       'networkDestinationMaskLen' => 32,
                                                                       'networkProtocol' => 0,
                                                                       'transportSource' => 0,
                                                                       'networkSourceMaskLen' => 32,
                                                                       'dataLayerSource' => '00:00:00:00:00:00',
                                                                       'dataLayerType' => '0x0000',
                                                                       'networkTypeOfService' => 0,
                                                                       'dataLayerDestination' => '00:00:00:00:00:00',
                                                                       'inputPort' => 0,
                                                                       'networkDestination' => '10.10.2.2',
                                                                       'transportDestination' => 0,
                                                                       'networkSource' => '10.10.1.2',
                                                                       'dataLayerVirtualLan' => -1
                                                                     },
                                                          'cookie' => '450359972898688789',
                                                          'lengthU' => 72,
                                                          'length' => 72,
                                                          'outPort' => -1,
                                                          'xid' => 0,
                                                          'type' => 'FLOW_MOD',
                                                          'hardTimeout' => 0,
                                                          'idleTimeout' => 0,
                                                          'command' => 0
                                                        }
                                       }
        };
root@pc-1:~#
```

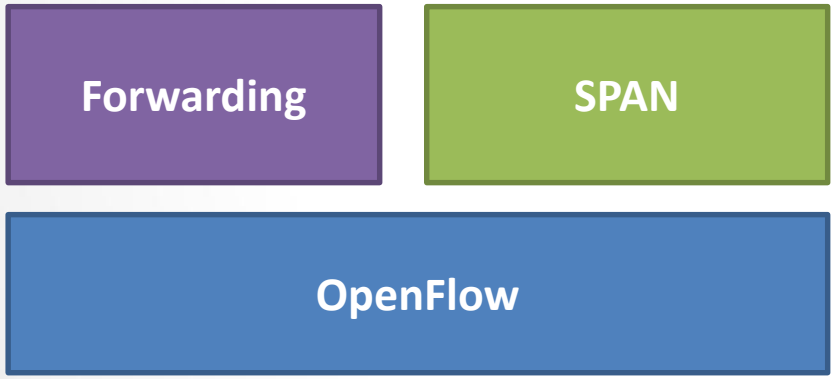
Низкоуровневые детали OpenFlow

```
of13::FlowMod fm2; // Table 0: in_port,VLAN=ar.ep.stag -> output(ar.ep.port)
fm2.table_id(FORWARDING_TABLE);
fm2.priority(100);
fm2.cookie(cookie);
fm2.xid(mgr->impl->xid);
fm2.buffer_id(OFP_NO_BUFFER);
fm2.add_oxm_field(new of13::InPort(main_route[0].port));
fm2.add_oxm_field(new of13::VLANVid(end_switch_list[0].ep_list[0].stag | of13::OFPVID_PRESENT));
of13::ApplyActions acts2;
acts2.add_action(new of13::OutputAction(end_switch_list[0].ep_list[0].port, 0));
fm2.add_instruction(acts2);
mgr->get_connection(sw1_dpid)->send(fm2);

/* DR */
/* rules for the last one switch in the route */
of13::FlowMod fm3; // Table 0: VLAN=ar.ep.stag -> META=bbd_id, GOTO 1
fm3.table_id(FORWARDING_TABLE);
fm3.priority(100);
fm3.cookie(cookie);
fm3.xid(mgr->impl->xid);
fm3.buffer_id(OFP_NO_BUFFER);
fm3.add_oxm_field(new of13::VLANVid(end_switch_list[0].ep_list[0].stag | of13::OFPVID_PRESENT));
fm3.add_instruction(new of13::WriteMetadata(domain_id, 0xFFFF));
fm3.add_instruction(new of13::GoToTable(LEARNING_TABLE));
mgr->get_connection(sw2_dpid)->send(fm3);

// Table 1: META=bbd_id, in_port=dr.ep0.port -> TO_CONTROLLER
for (auto ep : end_switch_list[1].ep_list) {
    of13::FlowMod fm4; // Table 1: META=bbd_id, in_port=dr.ep0.port -> TO_CONTROLLER
    fm4.table_id(LEARNING_TABLE);
    fm4.priority(100);
    fm4.cookie(cookie);
    fm4.xid(mgr->impl->xid);
    fm4.buffer_id(OFP_NO_BUFFER);
    fm4.add_oxm_field(new of13::Metadata(domain_id, 0xFFFF));
    fm4.add_oxm_field(new of13::InPort(ep.port));
    of13::ApplyActions acts4;
```

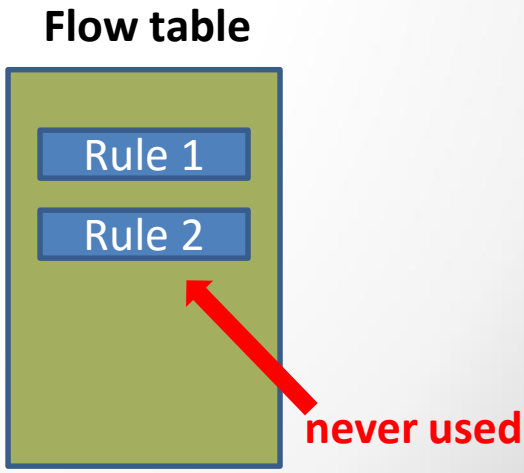
Possible problems in OpenFlow controllers

Example of **the problem** with running several apps independently:

- Forwarding and Span apps. First app sends a flow over port 1, while second ones sends the same flow over port 5. Rules intersect with each other.
- Final rules order in the flow table is unknown.
- Packets will go using only the first rule. Thus, only one app will work. **Conflict!**
- We may to resolve such conflicts and some others. **Just ip_src:10.0.0.1 -> output:1,5!**



Rule 1: ip_src:10.0.0.1 -> output:1
Rule 2: ip_src:10.0.0.1 -> output:5



New packet

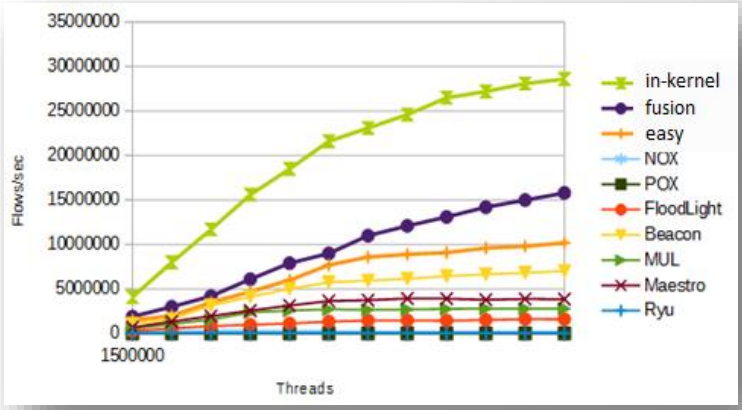
Часть V: Runos контроллер

Сетевая операционная система Runos

Система управления сетью первый российский SDN-контроллер RUNOS

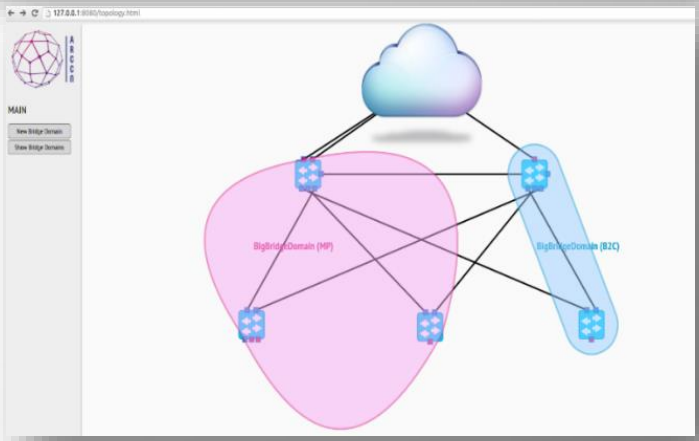
RUssian Network Operation System

Есть разные варианты контроллера с единой базой и различным набором сервисов и приложений



- **Открытая версия на Github** http://arccn.github.io/runos/
  - Своя база на C++11/14, а не Java
  - цель: упростить разработку сетевых приложений и не забывать о производительности
  - приложения: топология, маршрут, перестроение в случае обрыва, REST, WebUI, проактивная загрузка правил, резервирование Active-Passive

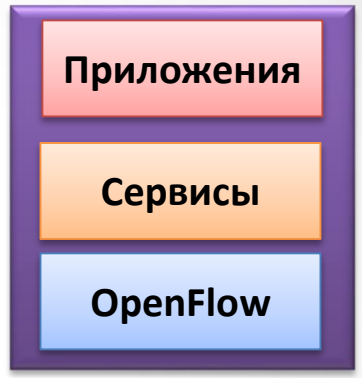
- **Внутренняя ядерная версия**
  - Супер-производительность 30 млн событий в секунду
  - Разработка приложений под заказчика



- **Внутренняя версия с приложениями под оператора связи**
  - База такая же, как и на Github. Заказчики сами могут разрабатывать приложения. Учиться по доступным материалам
  - Сервисы B2C, B2B (p2p, mp2mp, multicast, и т.п.)
  - Active-Standby режим

RUNOS: особенности

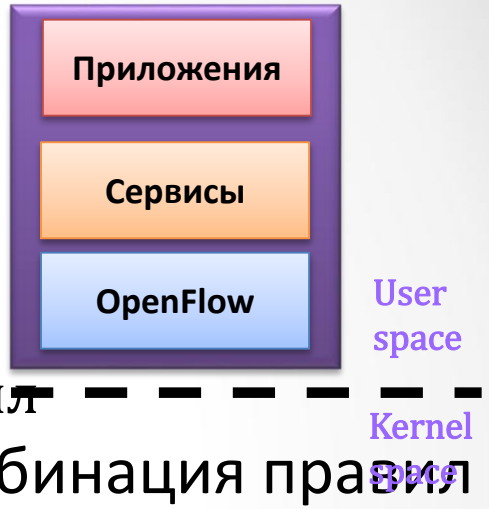
- Проблема запуска нескольких приложений, интеграция с приложениями других разработчиков
  - требуется статическая подстройка приложений под себя, порядок и способ передачи информации между ними.
  - нет механизма контроля и разрешения конфликтов между приложениями (генерация пересекающихся правил).
- 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)

Описания релизов



Сейчас версия **0.5**

- ядро контроллера
- построение топологии
- построение маршрута через всю сеть
- первая версия системы генерации правил
  - Распределение приоритетов, комбинация правил
  - LOAD, MATCH, READ абстракции
  - На основе MAPLE
- Rest API (совместимый с Floodlight)
- WebUI (мониторинг загрузки, просмотр таблиц, удаление и добавление правил)
- Проактивная загрузка правил
- Холодное резервирование
- ARP кеширование

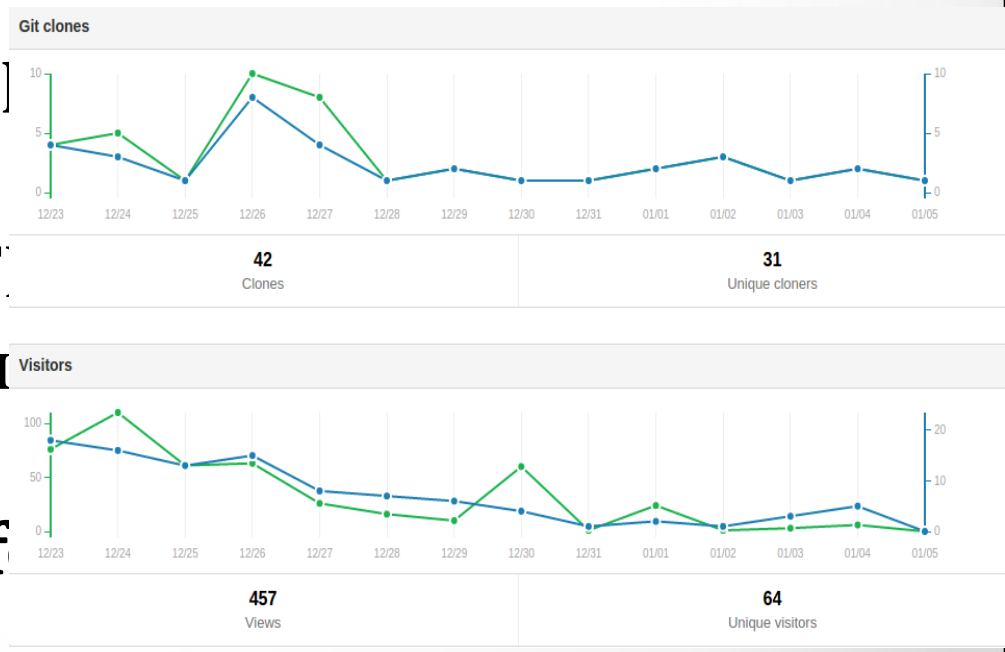
Описания релизов

Версия **0.6** – новый большой релиз

- **Полное обновление структуры ядра контроллера**. Нет привязка к конкретной версии протокола OpenFlow. Своя модель, расширяемая под любые новые поля, в том числе и специфические для оборудования.
- **Пакетная грамматика для сетевых приложений**. Упрощает разработку новых приложений.
  - “pkt[eth src] == eth addr”
  - “if (ethsrc == A || ethdst == B) doA else doB”
  - “test((eth_src & “ff0…..0”) == “….”)”
  - *“modify(ip_dst >> “10.0.0.1”)”*
  - decision are “unicast()”, “broadcast()”, “drop()”
- **Обновление системы генерации правил** — повышена скорость работы и улучшена генерация правил (по количеству правил и числу приоритетов).
- **Система тестов**.
- **Runos-book** подробная документация и инструкция по разработке новых приложений.
- **Приложения**: stp, arp, flow-manager

Проект с открытым исходным кодом

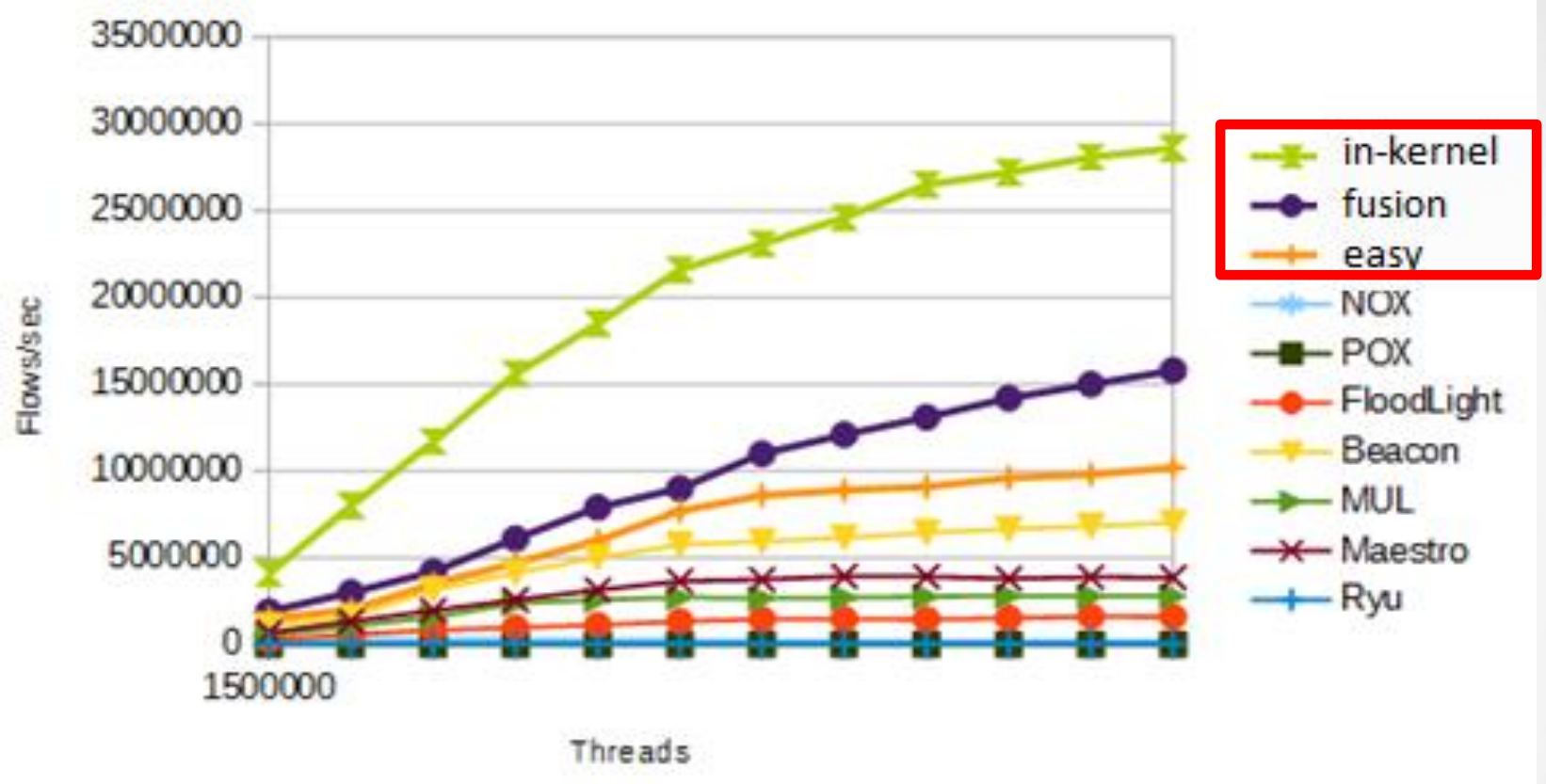
- Исходный код http://arccn.github.io/runos/
  - Apache, version 2.0
- Tutorial (Readme.md + Runos-book)
  - Как установить, запустить, написать свое первое п
- Виртуальная машина
  - Уже собранный конт
  - Средства для работы
- Список рассылки
  - Google group **runos-of**



Новые релизы Runos (открытая версия)

- v0.6.1
  - Чистый OpenFlow интерфейс для программирования коммутаторов (возможность самим формировать правила)
  - Обновленный REST: совместимость с Ryu, библиотека для Postman
- v0.7
  - Оптимизации системы генерации правил:
    - Глобальное видение сети
    - Оптимизация работы – по кол-ву FlowMod
  - Новые приложения: корпоративная сеть
  - Улучшение Web интерфейса (перенос часть функционала из коммерческой версии)

Производительность



- Производительность : **10 млн. потоков в секунду**
- Задержка: **55 мкс**

Реализация

Ключевые слова: C++11/14/17, QT, Boost (asio, proto, graph)

Основные сторонние компоненты:

- **libfluid project** (_base, _msg)
  - для взаимодействия со свитчами и разбор OpenFlow 1.3 сообщений
- **libtins**
  - разбор пакетов внутри OpenFlow сообщений
- **glog** (google log)
  - логирование, многопоточное
- **tcmalloc** (google performance tools)
  - альтернативная более быстрая реализация malloc/free
- **json11**
  - разбор конфигурационного файла
- **boost graph**
  - Хранение топологии, поиск маршрута

Параметры запуска

Config (json):

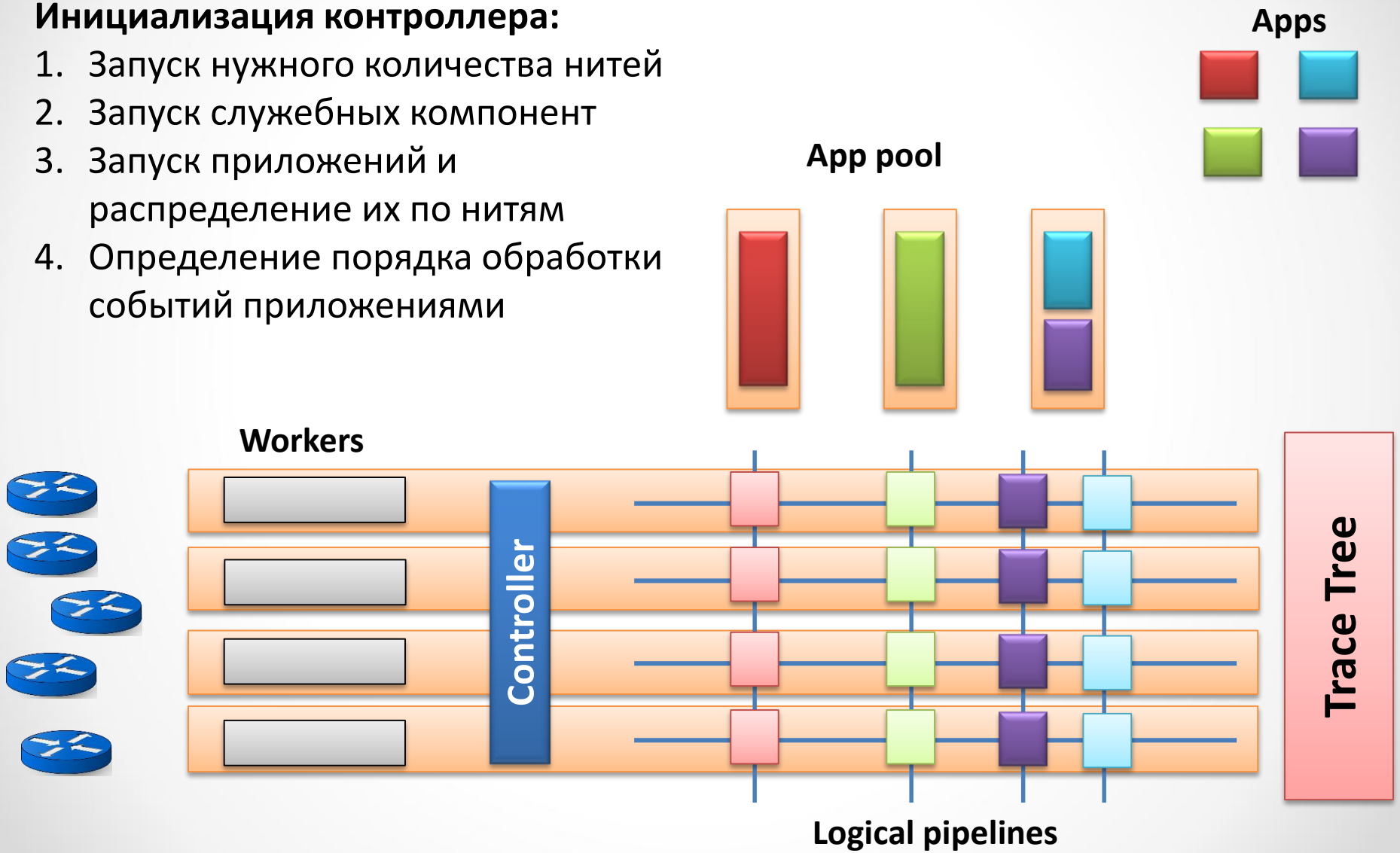
```
“controller”: {
  “threads”: 4
},
“loader”: {
  “threads”: 3
},
“link discovery”: {
   “poll-interval” : 10,
   “pin-to-thread” : 2
},
“learning switch”: {
}
…
```

- Задается количество нитей контроллера
  - для взаимодействия со свитчами
  - для работы приложений
- Список приложений
  - их параметры (poll-interval)
  - зафиксировать нить выполнения или выделить в монопольное пользование (pin-to-thread, own-thread)

Архитектура

Инициализация контроллера:

1. Запуск нужного количества нитей
2. Запуск служебных компонент
3. Запуск приложений и распределение их по нитям
4. Определение порядка обработки событий приложениями



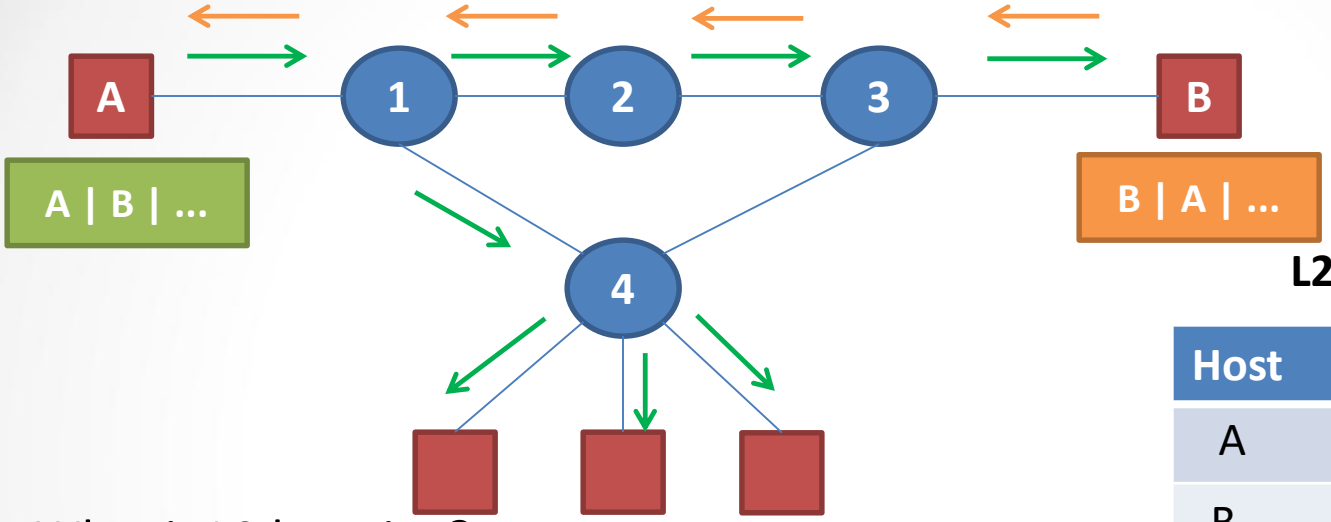
Open Source

- Два типа OpenSource проектов:
  - ради Идеи: “Free as in Freedom”
  - продаем свои компетенции, а не продукт
    - доработка под нужды заказчика,
    - продавать advanced версии и плагины (eg, приложения для Runos)
    - community вокруг (семинары, обучение)
- Важна лицензия (*): Apache, BSD или GPL, Eclipse, проприетарные лицензии, перелицензирование за деньги
- Угрозы:
  - Скопируют и будут продавать под другим названием (eg, runos-ng)
  - Будут дорабатывать своими силами (для компаний с большим R&D)

* http://www.slideshare.net/gerasiov/license-44646637

Часть VI: Разработка приложений для Runos контроллер

First application – L2 learning



L2 learning table

Host	Switch:port
A	1:1
B	3:2

- What is L2 learning?
  - L2 table – where particularly host resides (host <->sw:port)
- A->B. What should we do on sw1?
  - Learn and broadcast
- B->A. What should we do on sw3?
  - Learn and unicast
- **Advanced question: will it work for ping utilities? Ping 10.0.0.2 (assuming B has this IP)**
  - Yes, arp (broadcast), ip (icmp)

Host Databases

```
class HostsDatabase {
    boost::shared_mutex mutex;
    std::unordered_map<ethaddr, switch_and_port> db;

public:
    void learn(uint64_t dpid, uint32_t in_port, ethaddr mac)
    {
        LOG(INFO) << mac << " seen at " << dpid << ':' << in_port;
        {
            boost::unique_lock< boost::shared_mutex > lock(mutex);
            db[mac] = switch_and_port{dpid, in_port};
        }
    }

    boost::optional<switch_and_port> query(ethaddr mac)
    {
        boost::shared_lock< boost::shared_mutex > lock(mutex);

        auto it = db.find(mac);
        if (it != db.end())
            return it->second;
        else
            return boost::none;
    }
};
```

.insert(...)

L2 forwarding application

```
// Get required fields
ethaddr dst_mac = pkt.load(ofb_eth_dst);

db->learn(connection->dpid(),
          pkt.load(ofb_in_port),
          packet_cast<TraceablePacket>(pkt).watch(ofb_eth_src));

auto target = db->query(dst_mac);
// Forward
if (target) {
    flow->idle_timeout(60.0);
    flow->hard_timeout(30 * 60.0);

    auto route = topology->computeRoute(connection->dpid(),
                                        target->dpid);

    if (route.size() > 0) {
        flow->unicast(route[0].port);
    } else {
        flow->idle_timeout(0.0);
        LOG(WARNING) << "Path from " << connection->dpid()
            << " to " << target->dpid << " not found";
    }

    return PacketMissAction::Continue;
} else {
    flow->broadcast();
    return PacketMissAction::Continue;
}
```

Часть VII: Распределенный уровень управления

Высокая доступность (High Availability, HA)

ЦЕНТР ПРИКЛАДНЫХ ИССЛЕДОВАНИЙ КОМПЬЮТЕРНЫХ СЕТЕЙ

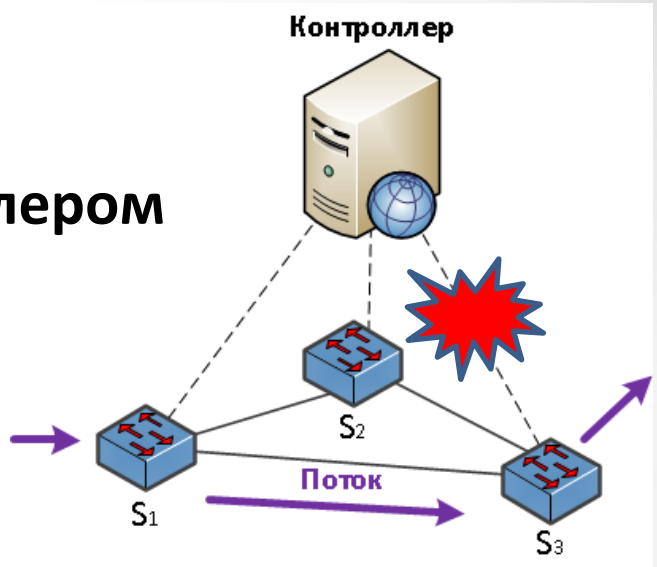
- Сеть работает в режиме 365/24/7.
- Платформа управления ПКС должна работать непрерывно.
- Цель обеспечения высокой готовности – поддержание непрерывной работоспособности платформы управления, сетевых приложений.
- Причины простоя: обслуживание, программные и аппаратные ошибки, отказы оборудования, атаки, отключение электроэнергии, аварии.

Коэффициент готовности, %	Время простоя за год
99,999	5 минут
99,99	52 минуты
99,9	8,7 часов
99	3,7 дней

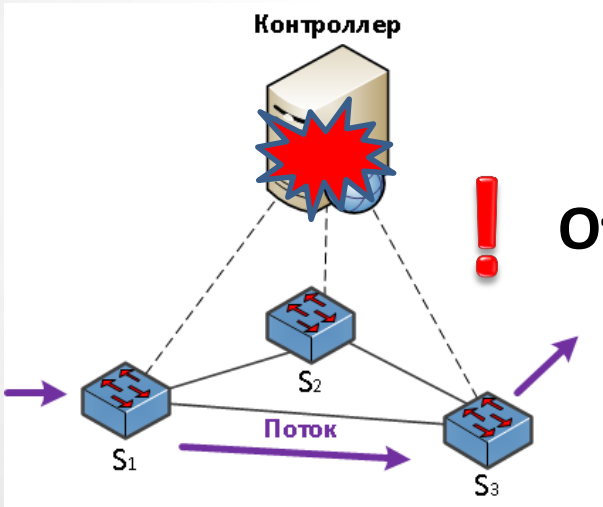
Угрозы

ЦЕНТР ПРИКЛАДНЫХ ИССЛЕДОВАНИЙ КОМПЬЮТЕРНЫХ СЕТЕЙ

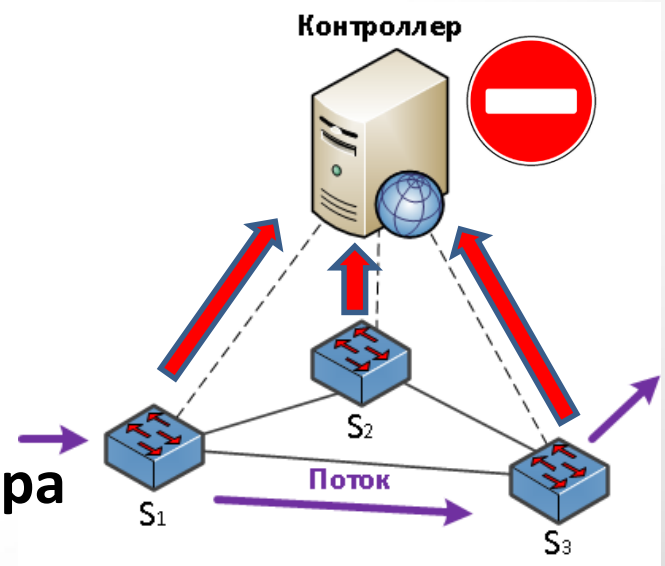
Потеря соединения коммутатора с контроллером



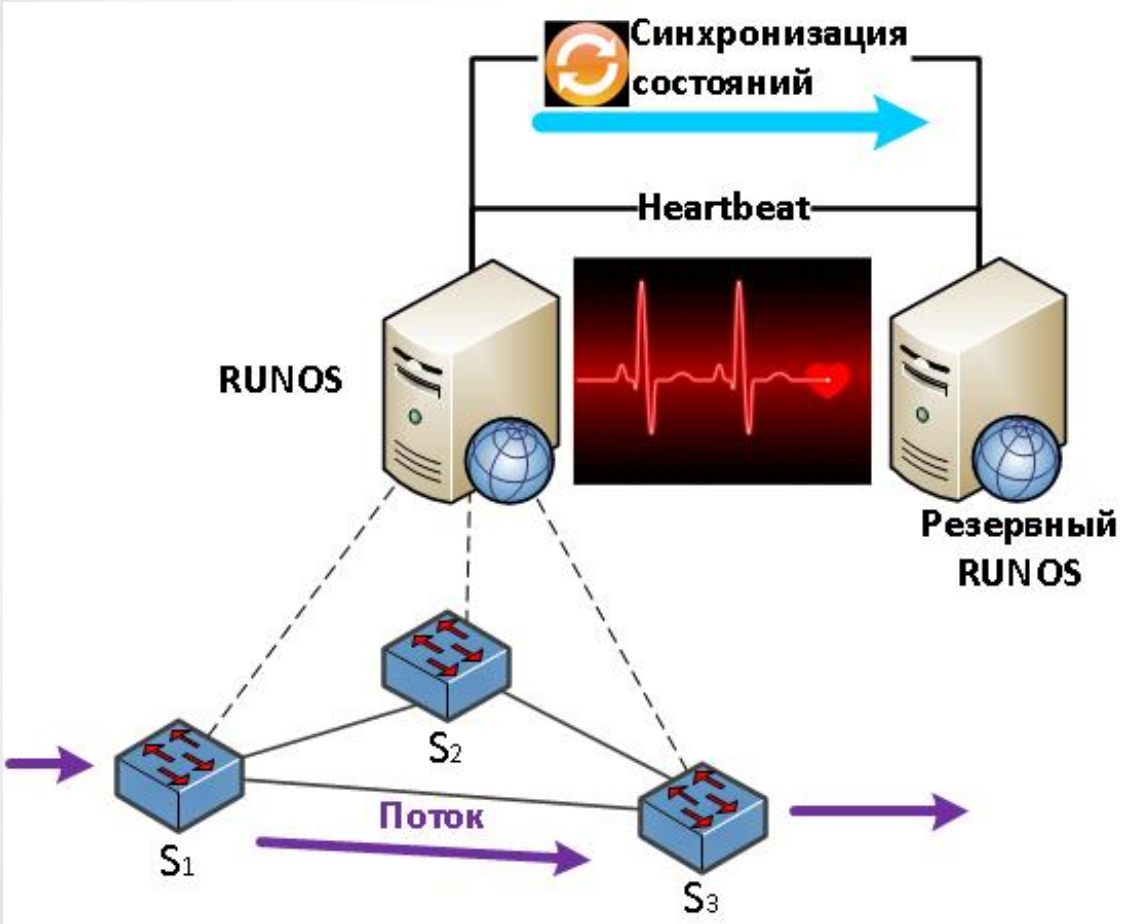
Отказ контроллера



Перегрузка контроллера



Стратегии резервирования /1



Active/Standby (Passive) **стратегии**:

- **Холодное**

[без синхронизации]

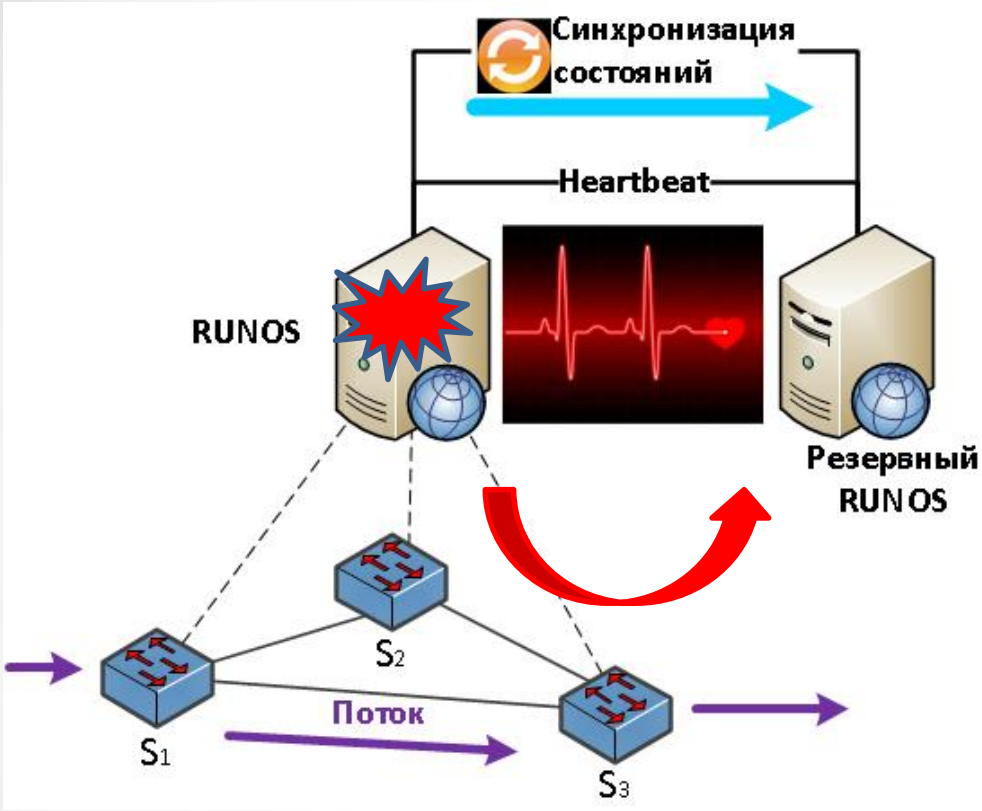
- **Теплое**

[периодическая синхронизация]

- **Горячее**

[постоянная синхронизация]

Восстановление после отказа контроллера для случая горячего резервирования RUNOS



Особенности решения:

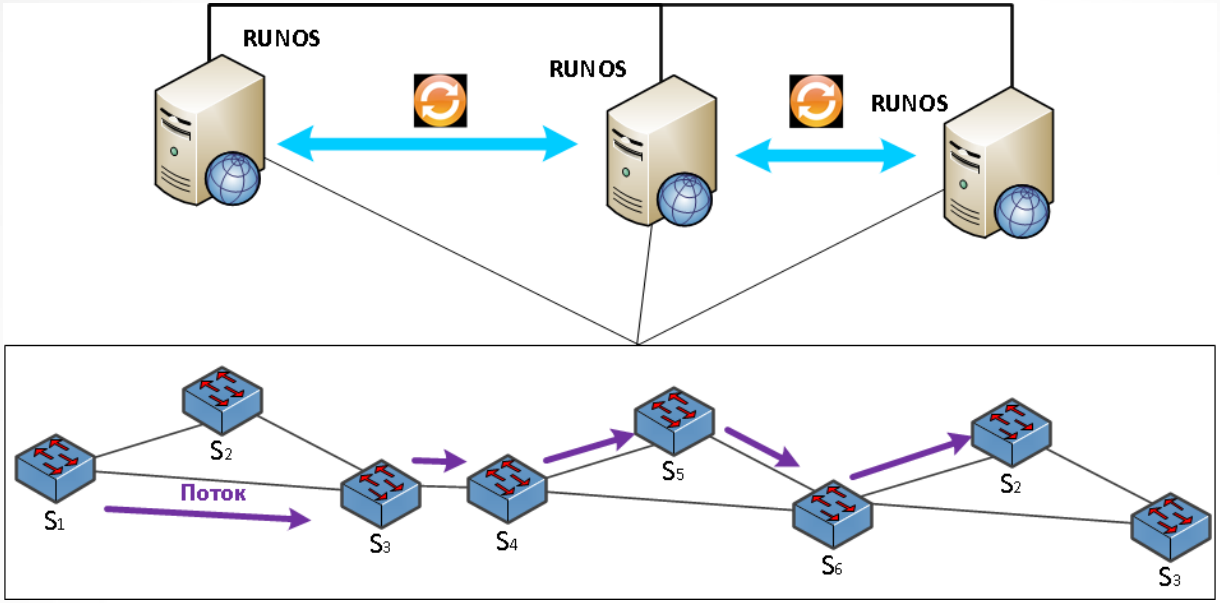
- OpenFlow ≥ 1.0
- Корпоративные сети.
- Не масштабируется
- Не полная утилизация вычислительных ресурсов

\+ Одиночный отказ контроллера

\- Потеря соединения коммутатора с контроллером

\- Перегрузка контроллера

Стратегии резервирования /2

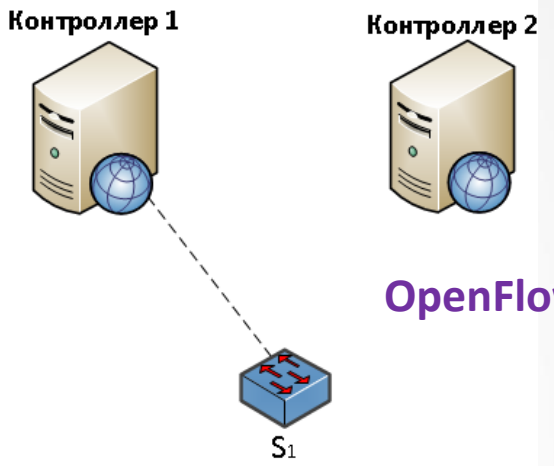


- Стратегии резервирования **Active/Active**
  - Ассиметричная
  - **Симметричная**
- Высокая сложность [Требуется координация контроллеров, поддержка глобального состояния]
- Высокая доступность [минимальное время простоя]
- Высокая утилизация вычислительных ресурсов

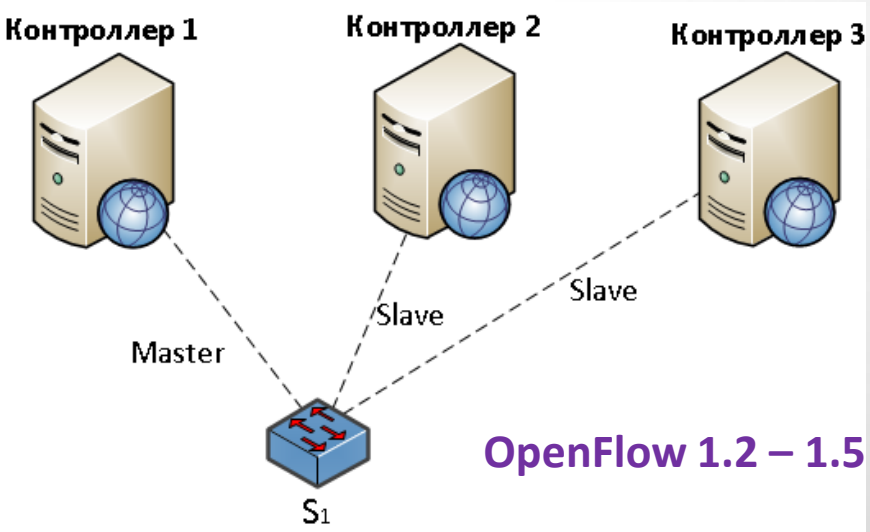
Возможности протокола OpenFlow

ЦЕНТР ПРИКЛАДНЫХ ИССЛЕДОВАНИЙ КОМПЬЮТЕРНЫХ СЕТЕЙ

- Протокол OpenFlow 1.2:
  - Множество контроллеров
  - Механизм ролей
  - **Роли:** Master, Slave, Equal
  - **По умолчанию:** контроллер находится в роли Equal для коммутаторов.
  - **Смена роли:** OFPT_ROLE_REQUEST
  - **Распределение ролей:** возложено на контроллеры.

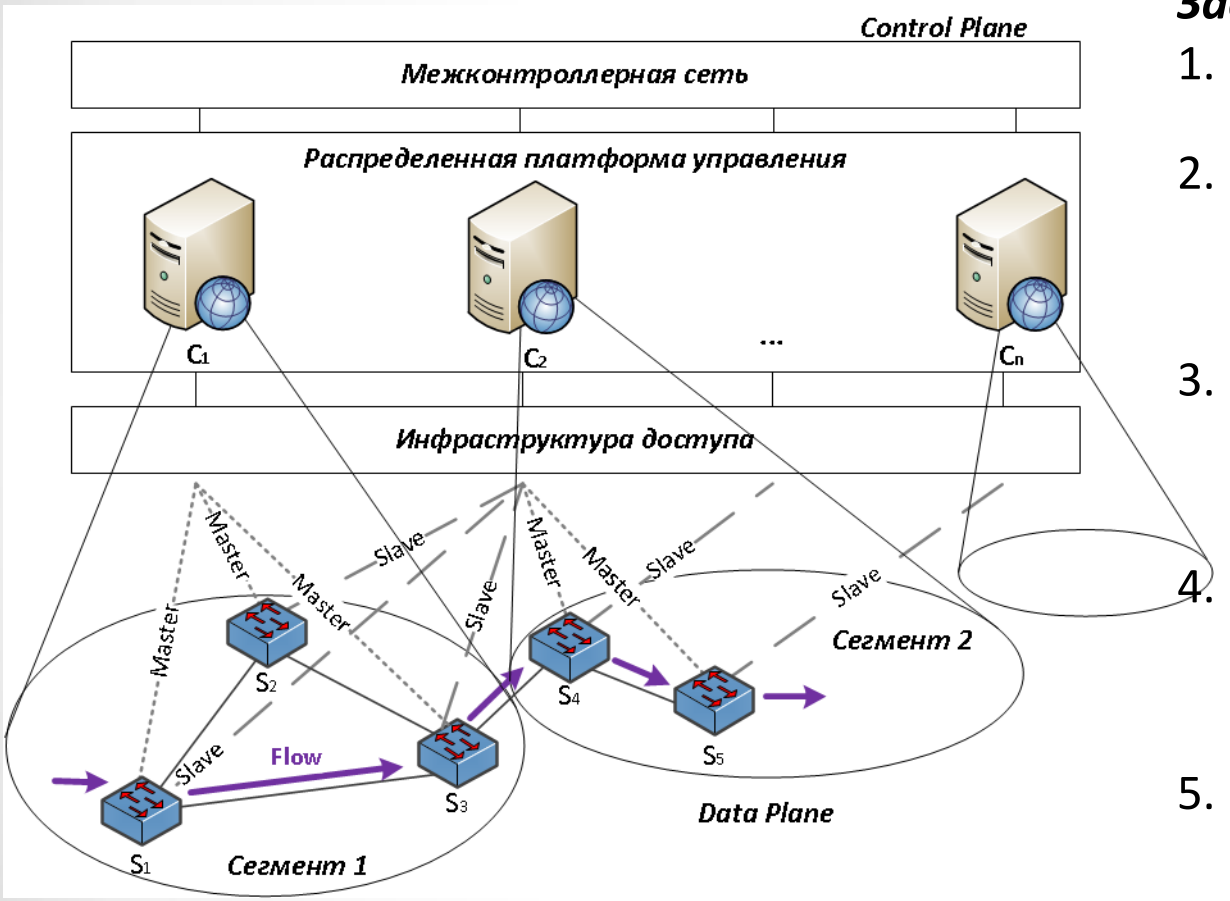


OpenFlow 1.0, 1.1



OpenFlow 1.2 – 1.5

Модель платформы управления



Задачи:

1. Как синхронизовать контроллеры?
2. Как определить начальное распределение коммутаторов по контроллерам?
3. Как перераспределить управление коммутаторами при одиночном отказе контроллера?
4. Как восстановить управление коммутатором при потере соединения с контроллером?
5. Как предотвратить перегрузку контроллера в платформе управления?

Предположения модели

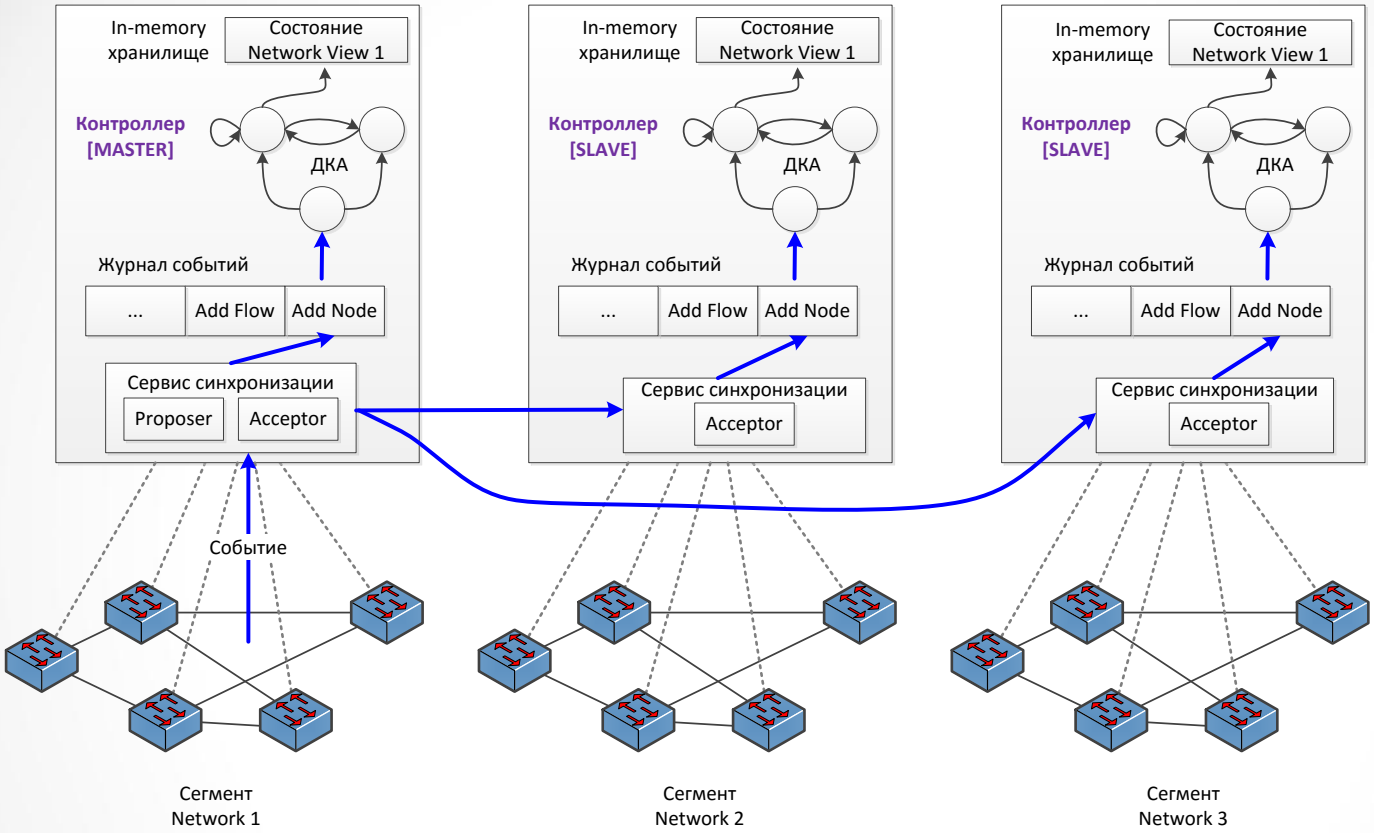
Предположения:

- Все контроллеры имеют одинаковую производительность.
- Контроллер устанавливает правила только на те коммутаторы, для которых он является Master контроллером.
- Инфраструктура доступа обеспечивает связь коммутаторов с каждым контроллером.

Условие корректности решения:

- В каждый момент времени для каждого коммутатора в сети должен быть определен только один Master контроллер.

Задача 1: Синхронизация контроллеров

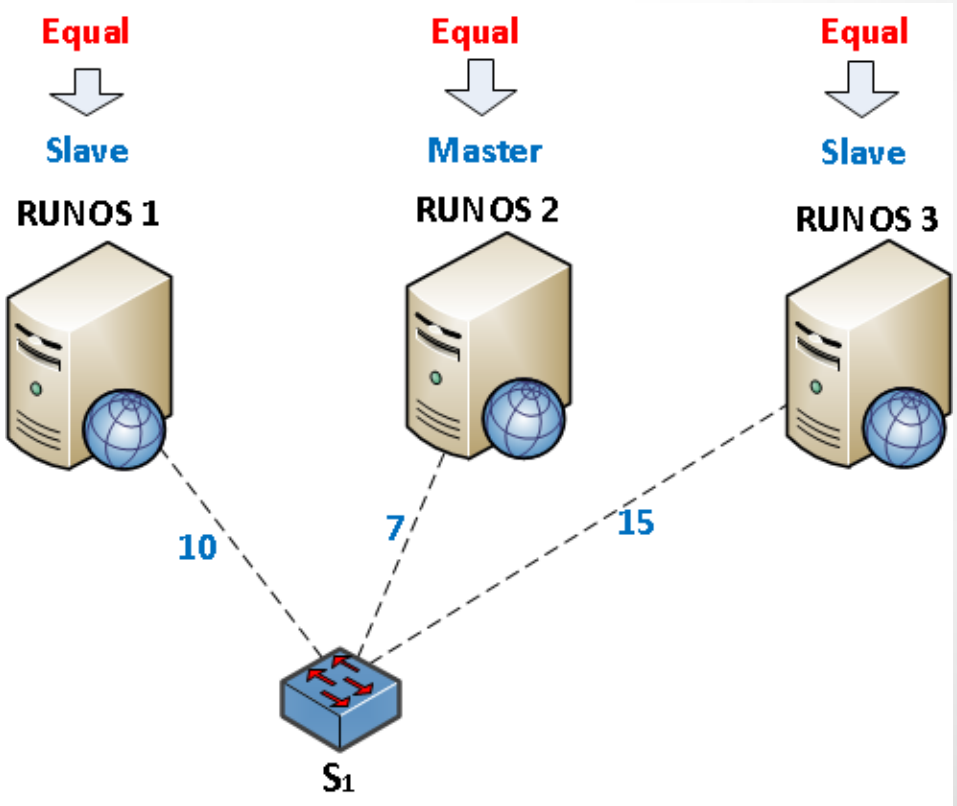


- На основе Multi-Paxos
- Обеспечивает одновременное одинаковое выполнение команд контроллеров
- Команды: изменение Network View, установление новых потоков (чтение NV), изменение ролей контроллеров.

Задача 2: Начальное распределение коммутаторов по контроллерам

ЦЕНТР ПРИКЛАДНЫХ ИССЛЕДОВАНИЙ КОМПЬЮТЕРНЫХ СЕТЕЙ

- **Критерий определения Master-контроллера:**
  - Задержка от коммутатора до контроллера, как мера близости.
- **Ограничение:**
  на количество коммутаторов в сегменте контроллера.
- **Решение:**
  Жадный алгоритм по значению задержки от коммутатора до контроллеров.

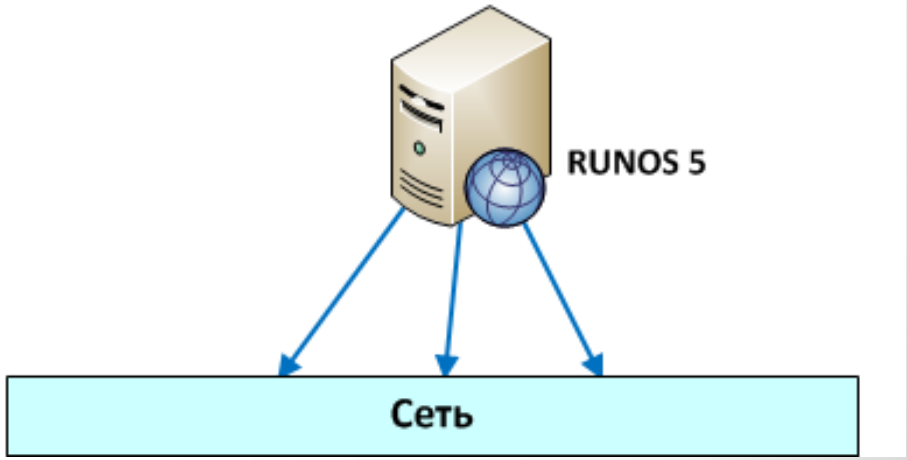


Задача 3: Выработка и представление сценариев восстановления

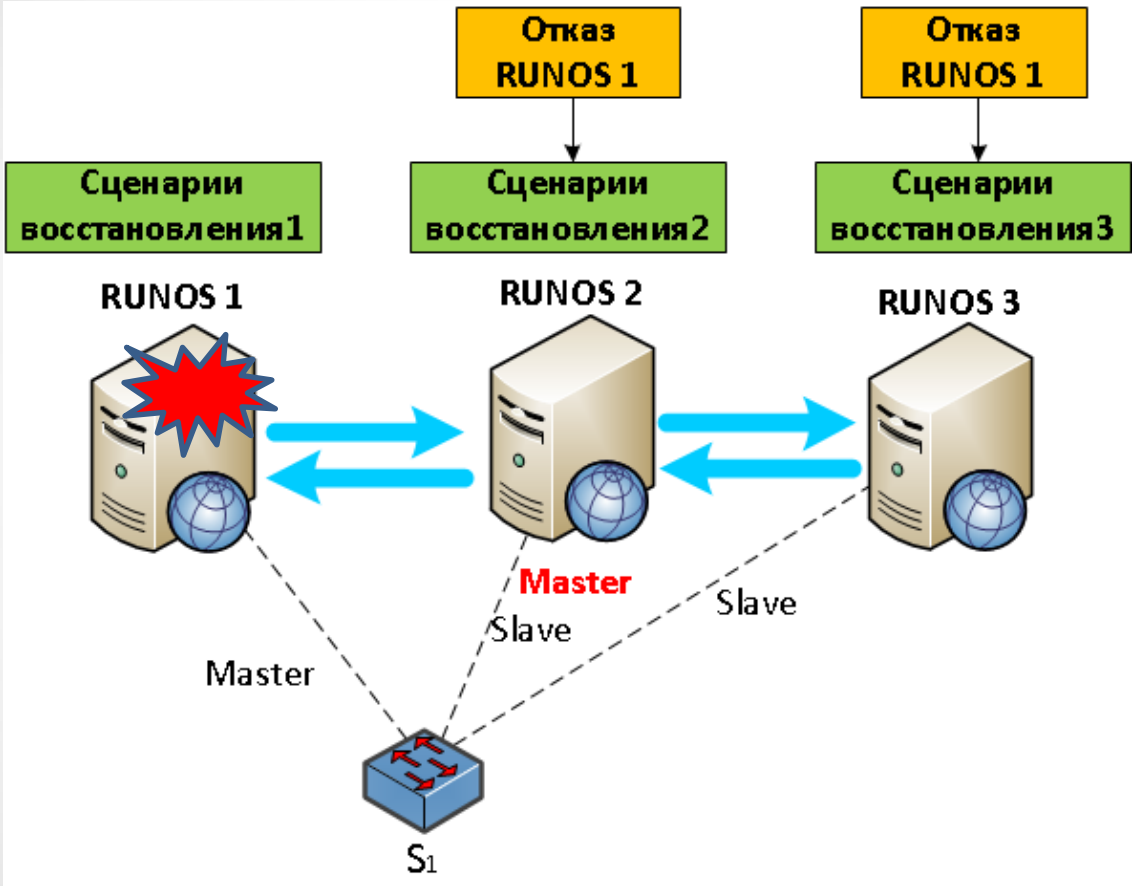
- Проактивная разработка сценариев восстановления
- Критерий выбора нового распределения: задержка от коммутатора до контроллера
- Алгоритм Балаша.
- **Результат:** перечень сценариев для каждого контроллера платформы управления.

Сценарии восстановления

Отказ контроллера	Перечень коммутаторов, для которых контроллер должен стать Master-ом
RUNOS 01	S3,S4
RUNOS 02	S1, S5
…	…
RUNOS N	S k

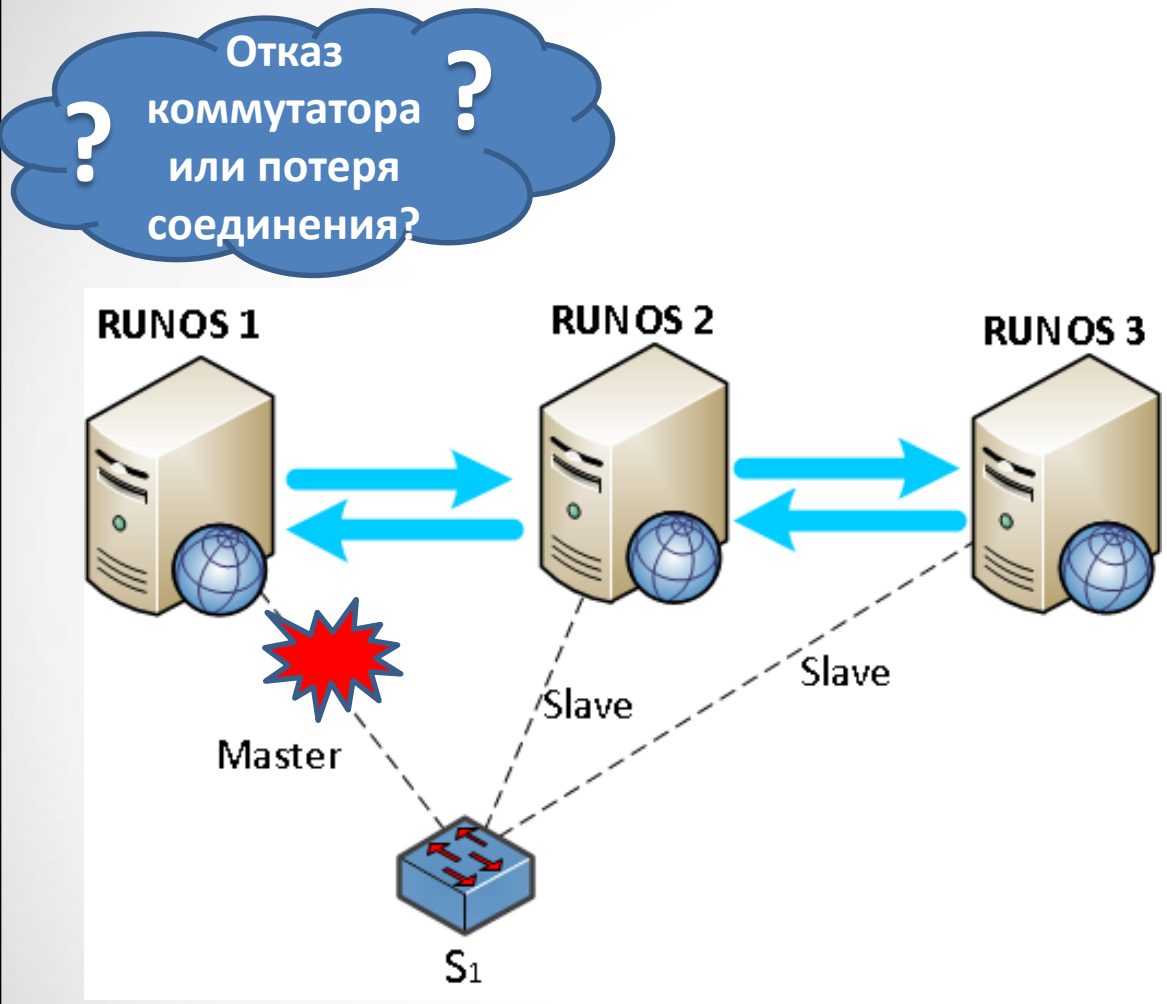


Задача 3: Использование сценария для восстановления ПУ после отказа контроллера



1. Оставшимися контроллерами фиксируется факт отказа контроллера и CID.
2. Каждый контроллер выбирает сценарий восстановления, соответствующий отказу контроллера CID.
3. В соответствии со сценарием каждый контроллер изменяет свою роль на коммутаторах.
4. Каждый контроллера информирует остальные контроллеры о завершении выполнения сценария восстановления.

Задача 4: Восстановление при потере соединения коммутатора с контроллером



1. Фиксируется отключение коммутатора Master-контроллером.
2. Инициируется проверка присутствия коммутатора в сети.
3. Осуществляется измерении задержки контроллерами, поддерживающими связь с коммутатором.
4. Master-контроллером становится контроллер с минимальной задержкой до коммутатора.
5. Новый Master-контроллер устанавливает свою роль на коммутаторе.

Задача 5: Предотвращение перегрузки контроллера платформы управления

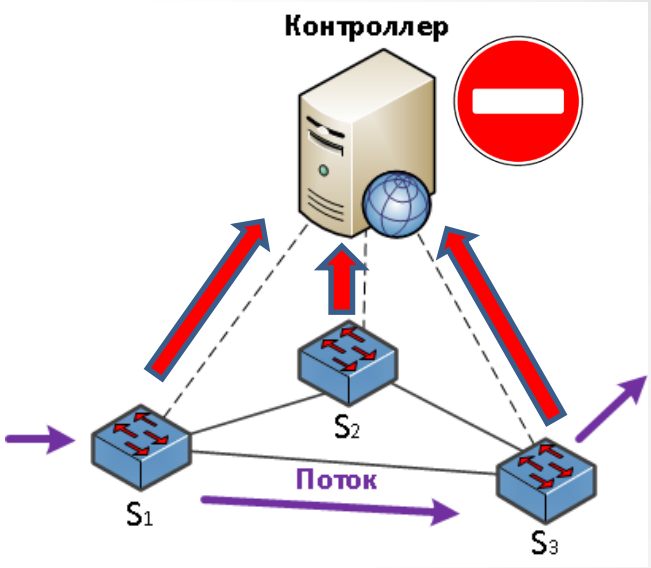
ЦЕНТР ПРИКЛАДНЫХ ИССЛЕДОВАНИЙ КОМПЬЮТЕРНЫХ СЕТЕЙ

Для обнаружения перегрузки:

- Устанавливаются пороговые значения параметров загрузки контроллеров.
- Осуществляется постоянный мониторинг загрузки контроллеров

[количество коммутаторов, количество packet-in запросов в секунду, CPU, память]

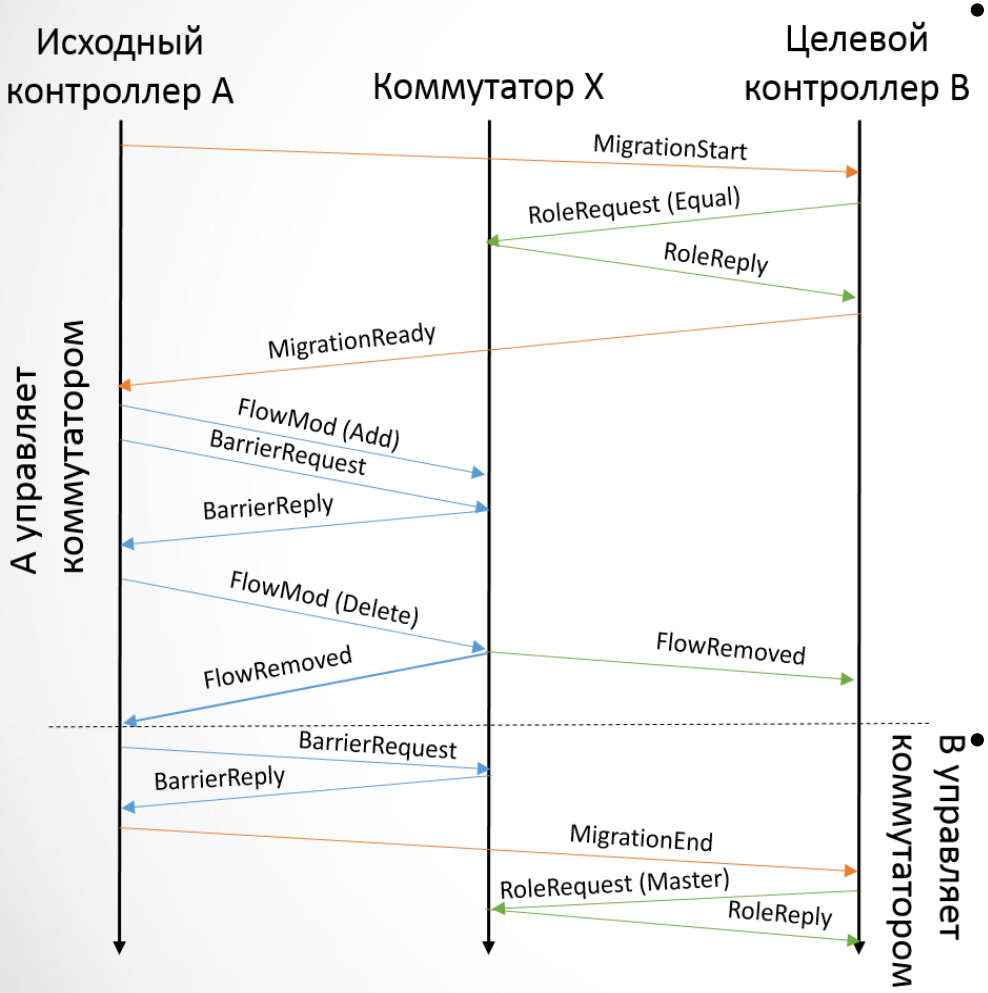
- Факт перегрузки фиксируется при обнаружении превышения порогового значения.



Задача перераспределения коммутаторов по контроллерам: Необходимо перераспределить коммутаторы так, чтобы:

1. Нагрузка на каждый из контроллеров не превышала его производительности
2. Использовалось минимальное количество контроллеров.
3. При перестроении управления сетью было проведено минимальное количество передач управления коммутаторами.

Задача 5: Передача управления коммутатором



- **Свойство живучести:** В любой момент времени для коммутатора существует контроллер, работающий в режиме *Master.* Для любой асинхронной команды, контроллер, который ее отправил, остается активным до тех пор, пока коммутатор не закончит ее обработку.

- **Свойство безопасности:** Ровно один контроллер обрабатывает каждое асинхронное сообщение от коммутатора.

Задача 5: Распределение коммутаторов по группам

ЦЕНТР ПРИКЛАДНЫХ ИССЛЕДОВАНИЙ КОМПЬЮТЕРНЫХ СЕТЕЙ

- **Входные данные:**
  - Топология сети
  - Количество новых потоков с каждого коммутатора
  - Максимально допустимая нагрузка на контроллер *P*
- **Алгоритм** разбиения графа на компоненты связности, чтобы суммарная нагрузка компоненты связности не превышала *P.*
- **Выходные данные:**
  - Матрица, определяющая Master-контроллеры для коммутаторов и сегменты управления для каждого контроллера.

Задача 5: Распределение групп коммутаторов по контроллерам

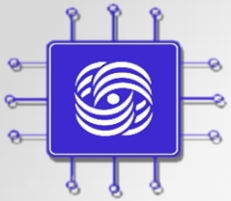
- **Входные данные:**
  - Текущая матрица распределения управления коммутаторами по контроллерам.
  - Желаемая матрица распределения управления коммутаторами.
- **Алгоритм:** жадный алгоритм, минимизирующий количество передач управления коммутаторами между контроллерами платформы управления.
- **Выходные данные:**
  - Список операций по передаче управления отдельными коммутаторами.

Задание по курсу

- Разработка приложений для контроллера Runos
  - Само приложение
  - Скрипты на Mininet для эмулируемой топологии
  - Сдавать все архивом.
- Использовать контроллер версии 0.6 с github
  - https://github.com/ARCCN/runos
- Все вопросы по заданиям и по Runos в список рассылки
  - http://groups.google.com/d/forum/runos-ofc
- Если нашли какой-то баг ☺, то официально описываем их на github issues
  - https://github.com/ARCCN/runos/issues

Пример задания курсу SDN/NFV

- Балансировка нагрузки
  - Взвешанный round robin
- Топология
  - Запрограммировать ее на mininet




Заключение

- SDN уже активно используется в промышленности и является основным трендом в развитии телеком индустрии.
- SDN != OpenFlow
  - SDN – подход к разделению уровня данных и уровня управления
  - OpenFlow – одна из реализаций. Другие, XMPP, SNMP, overlay.

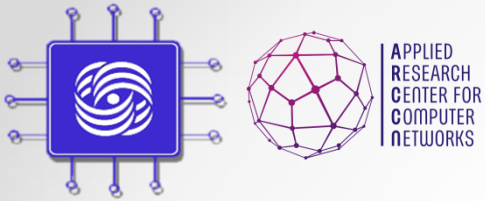
"SDN means thinking differently about networking"

http://arccn.ru/

ashalimov@lvk.cs.msu.su

@alex_shali

@arccnnews



Видео об SDN

- Немного юмора
  - SDN с разных точек зрения
  - http://www.youtube.com/watch?v=GRVygzcXrM0