



ЦЕНТР
ПРИКЛАДНЫХ
ИССЛЕДОВАНИЙ
КОМПЬЮТЕРНЫХ
СЕТЕЙ



Проблемы разработки приложений для SDN сетей и их решение в контроллере Runos

Александр Шалимов

ЦПИКС, МГУ



<http://arccn.ru/>



ashalimov@arccn.ru



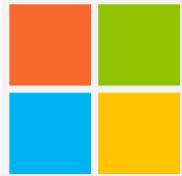
@alex_shali

@arccnnews

Что такое SDN/OpenFlow?

SDN = Software Defined Networking

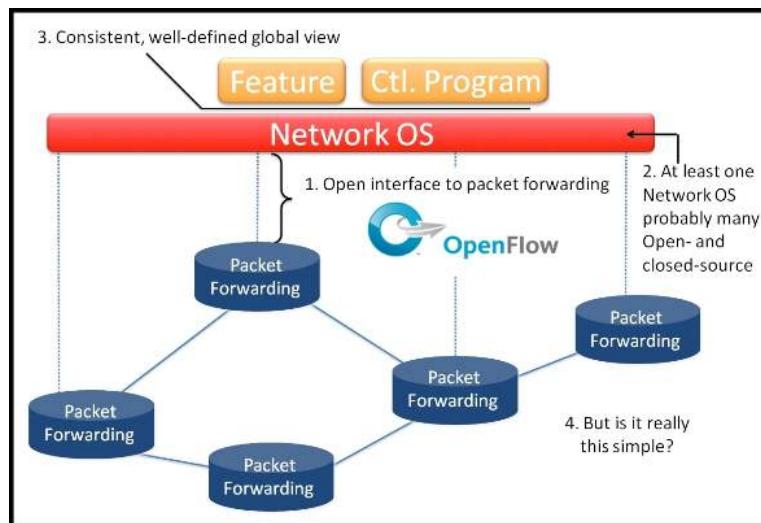
Внедрения



-
-
-

Основные принципы

- Физическое разделение уровня передачи данных от уровня управления сетевых устройств.
- Логически централизованное управление.
- Программируемость.
- Открытый единый интерфейс управления.



Преимущества

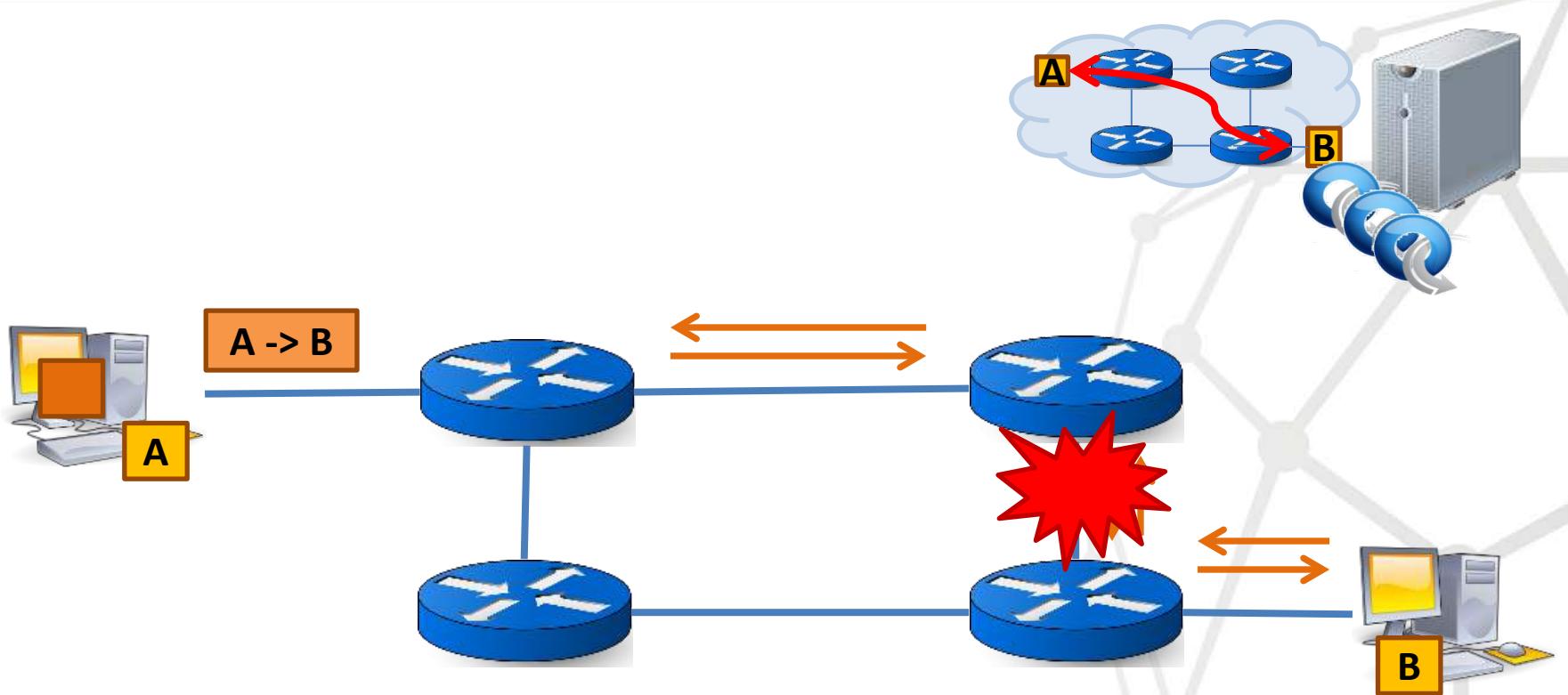
- Упрощение управления сетью (OPEX)
- Удешевление оборудования (CAPEX)
- Разработка ранее недоступных сервисов

"SDN means thinking differently about networking"

Доп.главы Компьютерных сетей

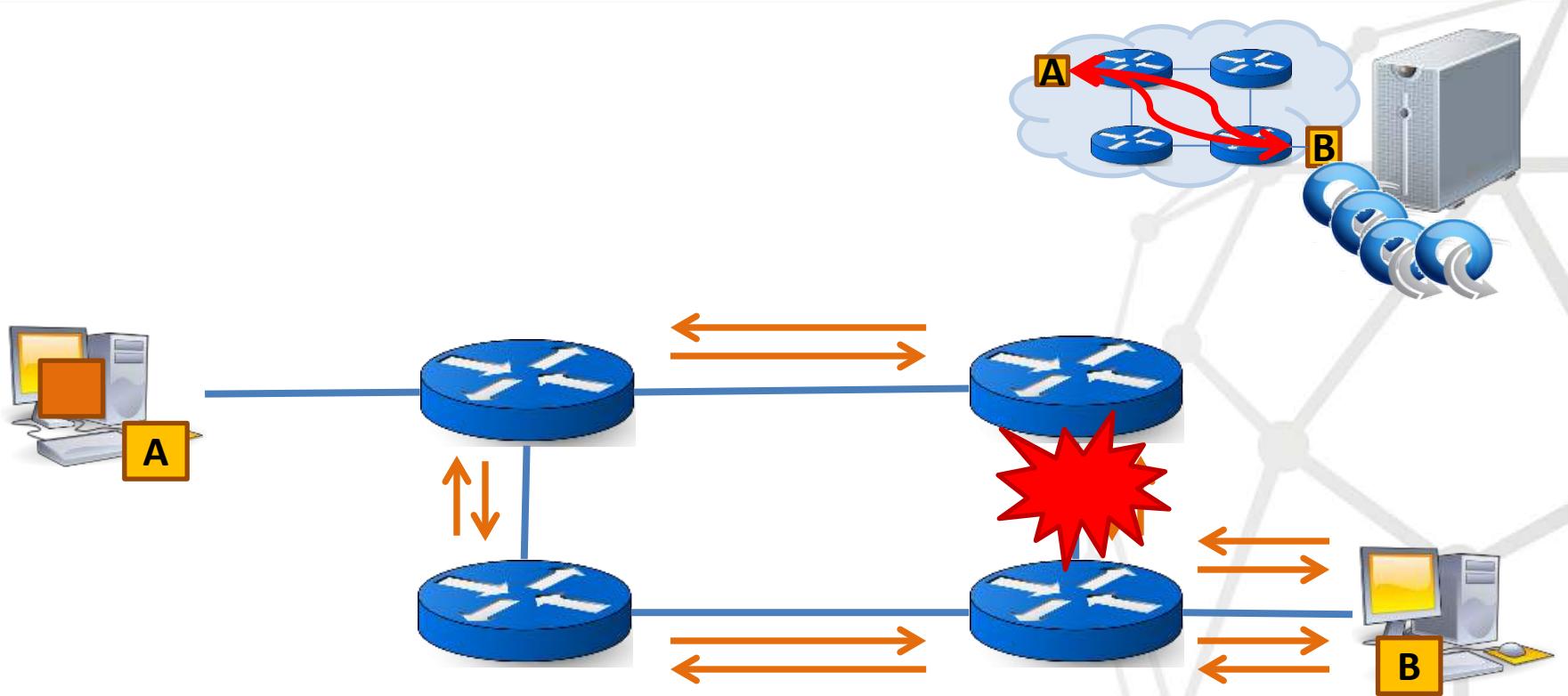
Шалимов А.В.

Маршрутизация с SDN/OpenFlow



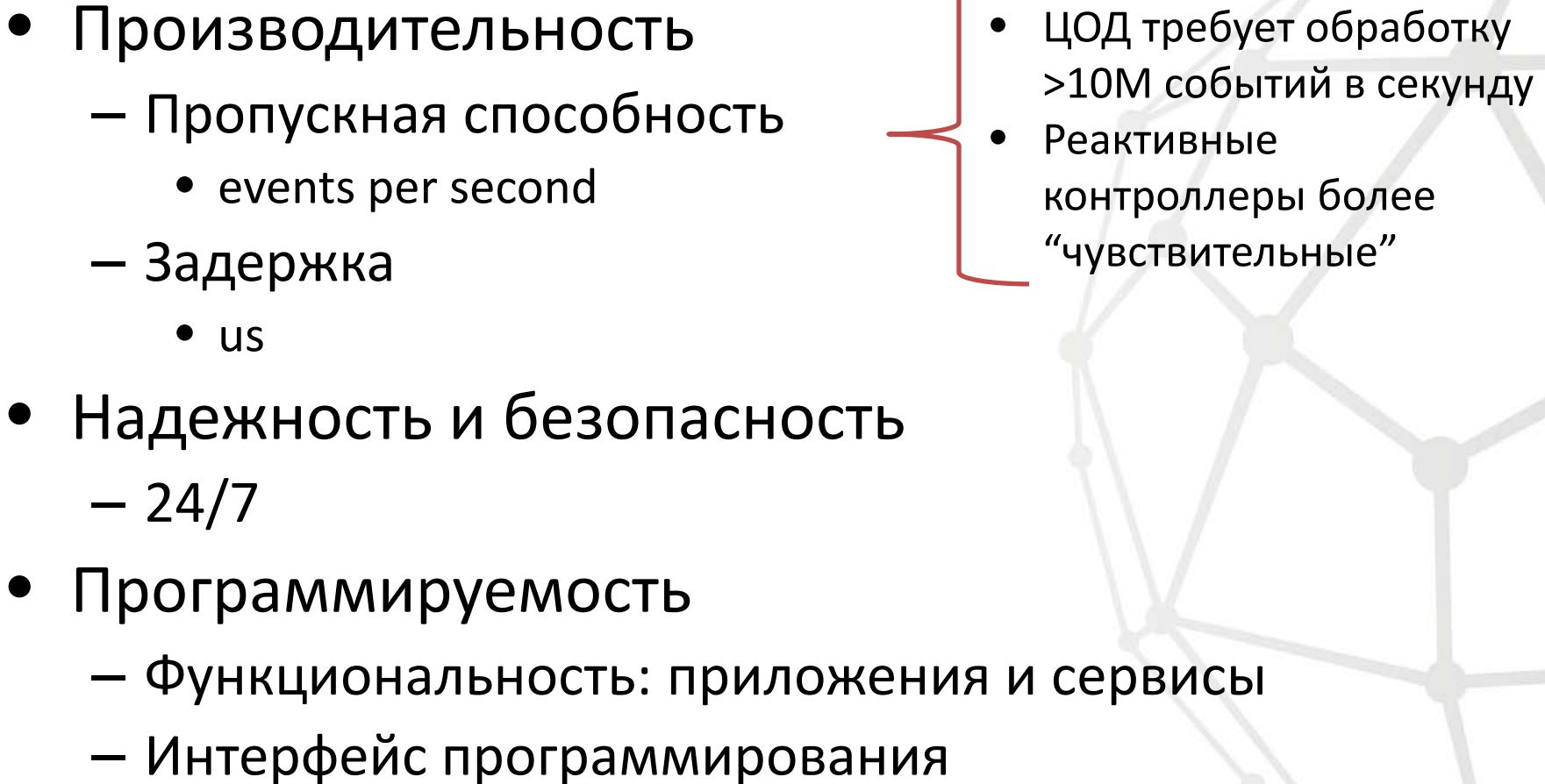
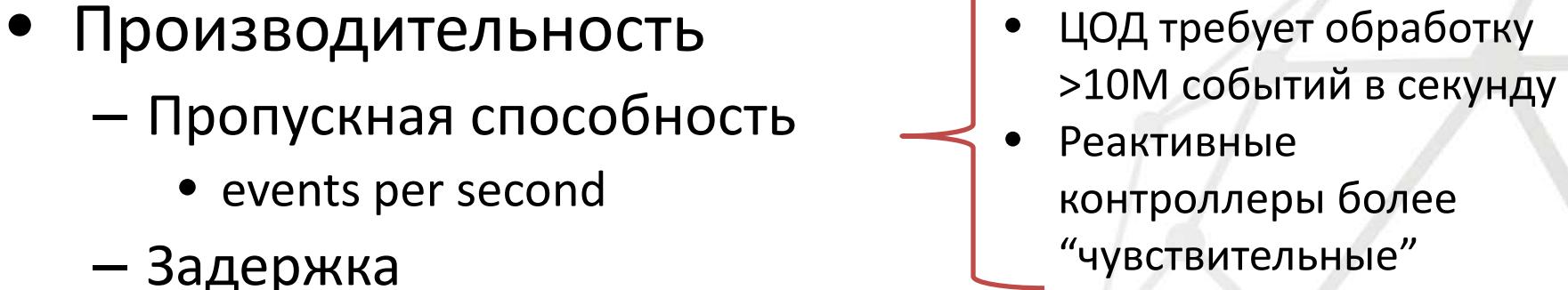
- Неизвестный пакет отправляется на контроллер (OF_PACKET_IN).
- Контроллер вычисляет лучший маршрут через всю сеть (с наименьшей стоимостью и удовлетворяющий политикам маршрутизации).
- Соответствующие правила OpenFlow устанавливаются на коммутаторы + сразу и обратный маршрут (OF_PACKET_OUT/FLOW_MOD).

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.**

Требования к контроллеру ПКС

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

Программируемость

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

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   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' => '45035997289868789',
            'length0' => 72,
            'length' => 72,
            'outPort' => -1,
            'xid' => 0,
            'type' => 'FLOW_MOD',
            'hardTimeout' => 0,
            'idleTimeout' => 0,
            'command' => 0
        }
    }
};

root@pc-1:~#
```

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



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);
```

Possible problems in OpenFlow controllers

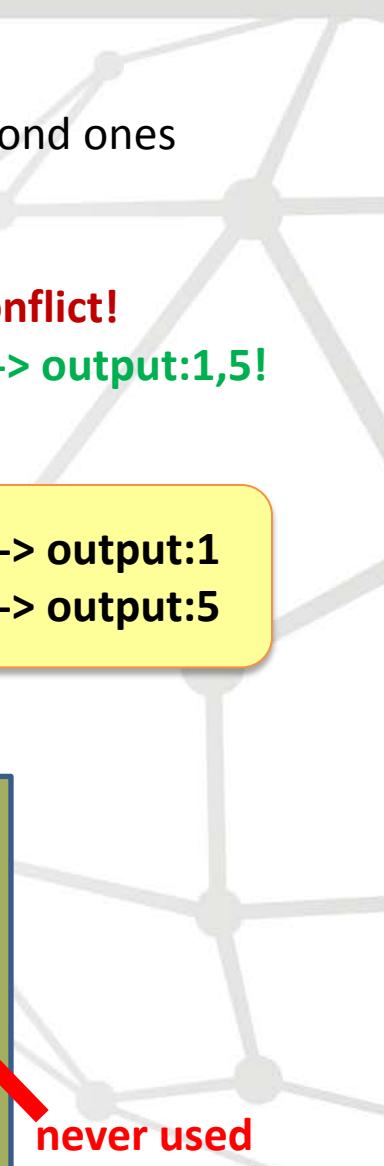
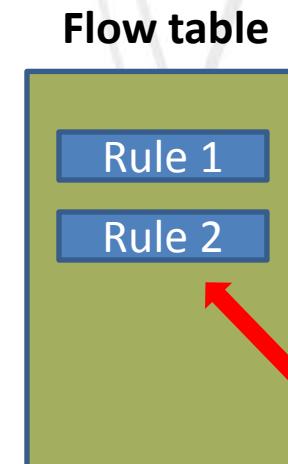
Example of **the problem** with running several apps independently:

- 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!**



New packet
ip_dst:10.0.0.1

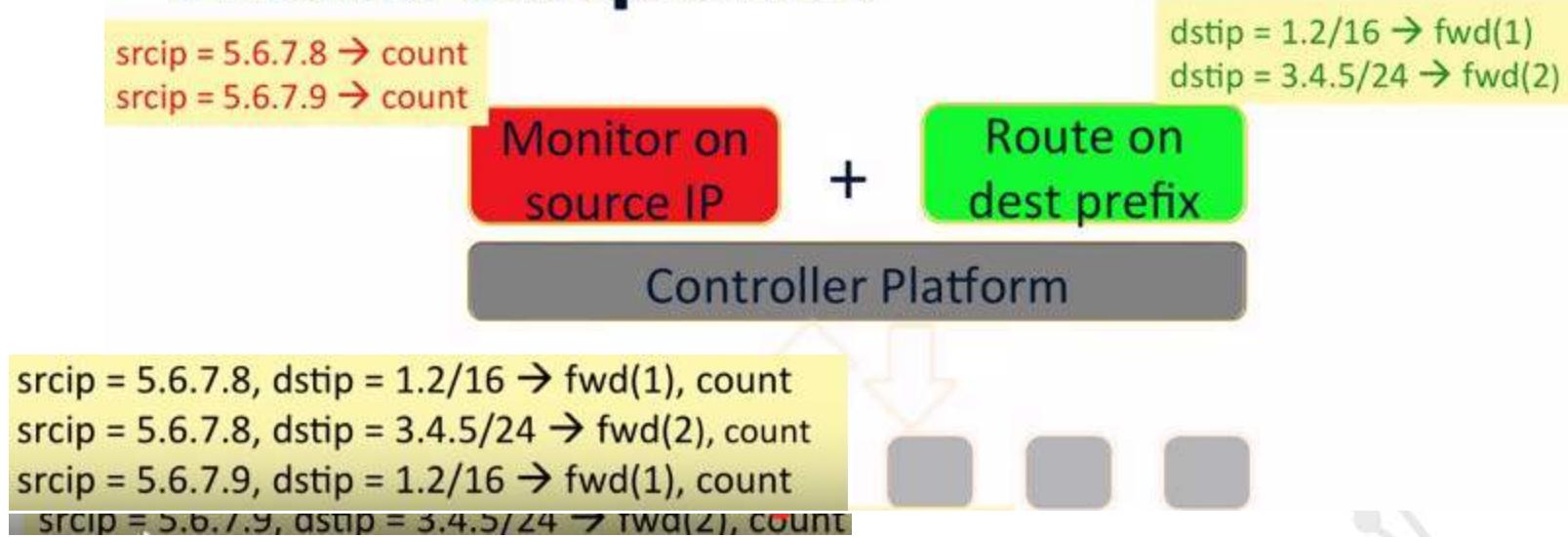
Rule 1: ip_src:10.0.0.1 -> output:1
Rule 2: ip_src:10.0.0.1 -> output:5



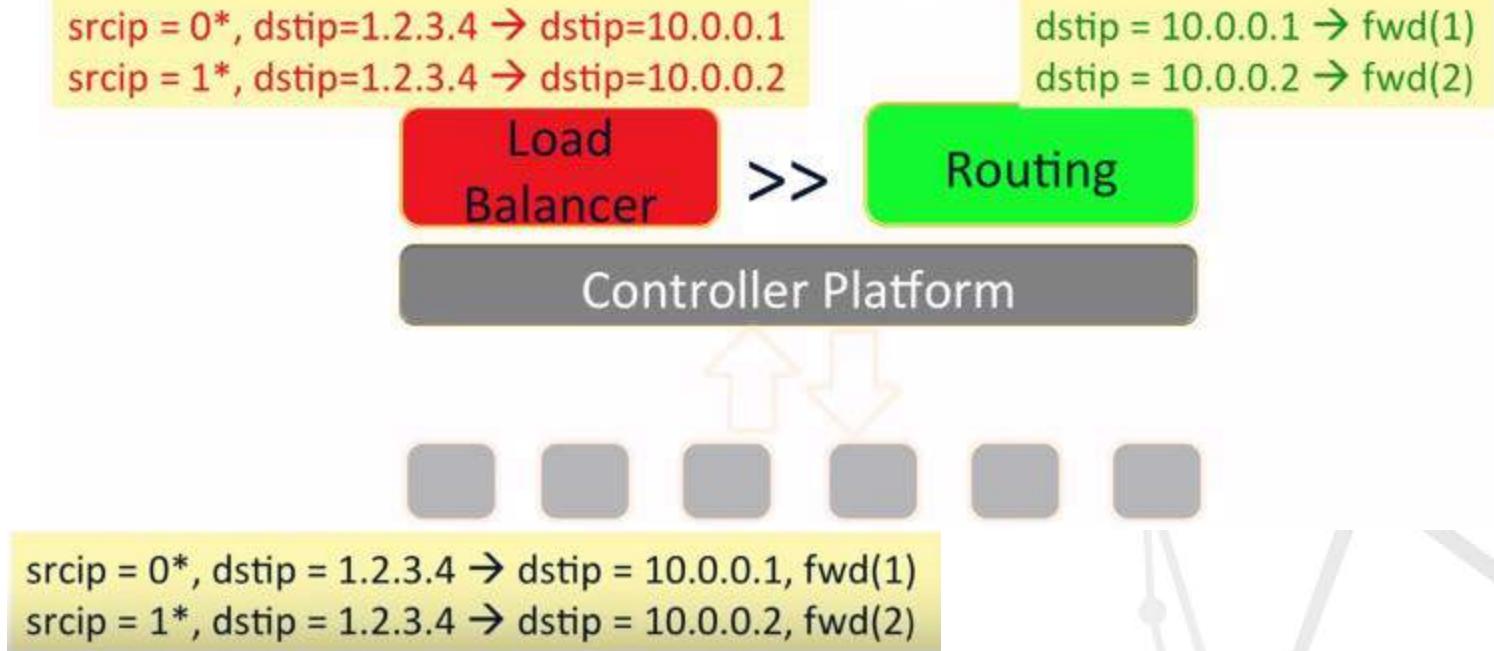
Композиция приложений

- Два типа композиции (на примере, Pyretic)
 - Параллельная – выполняет оба действия одновременно (forwarding и couting)
 - Последовательная композиция – выполняет одно действие за другим (firewall, затем switch)

Parallel Composition



Sequential Composition



- **Pyretic**
 - $(\text{match}(\text{dstip}=2.2.2.8) \gg \text{fwd}(1)) + (\text{match}(\text{dstip}=2.2.2.9) \gg \text{fwd}(2))$

Сетевая операционная система RUNOS

RUNOS = RUssian Network Operating System

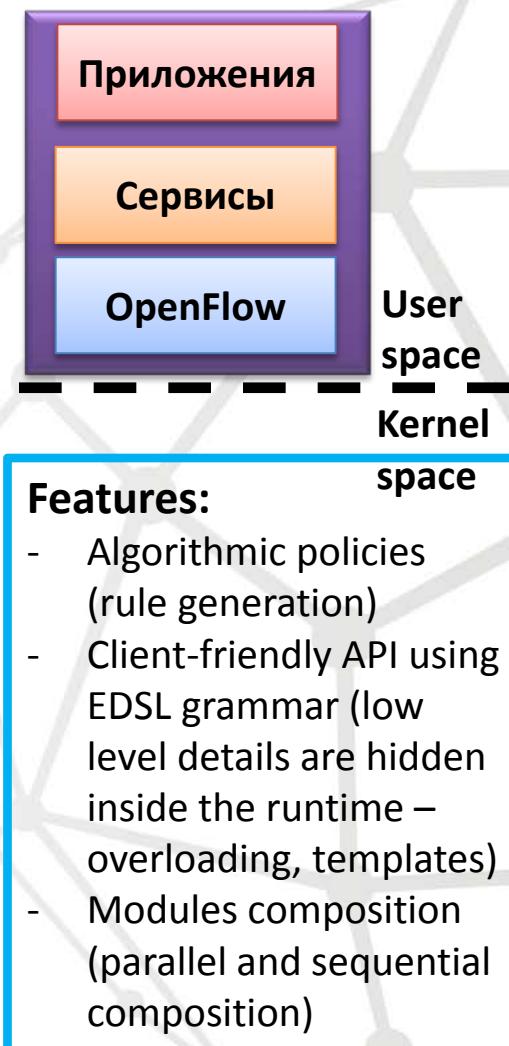
- Цель проекта
 - “*Could an OpenFlow controller be both easy to develop applications for and also high performance?*”
 - Разработать систему, которая будет удобна для разработки новых сетевых приложений
 - При этом помнить о быстроте

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



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)

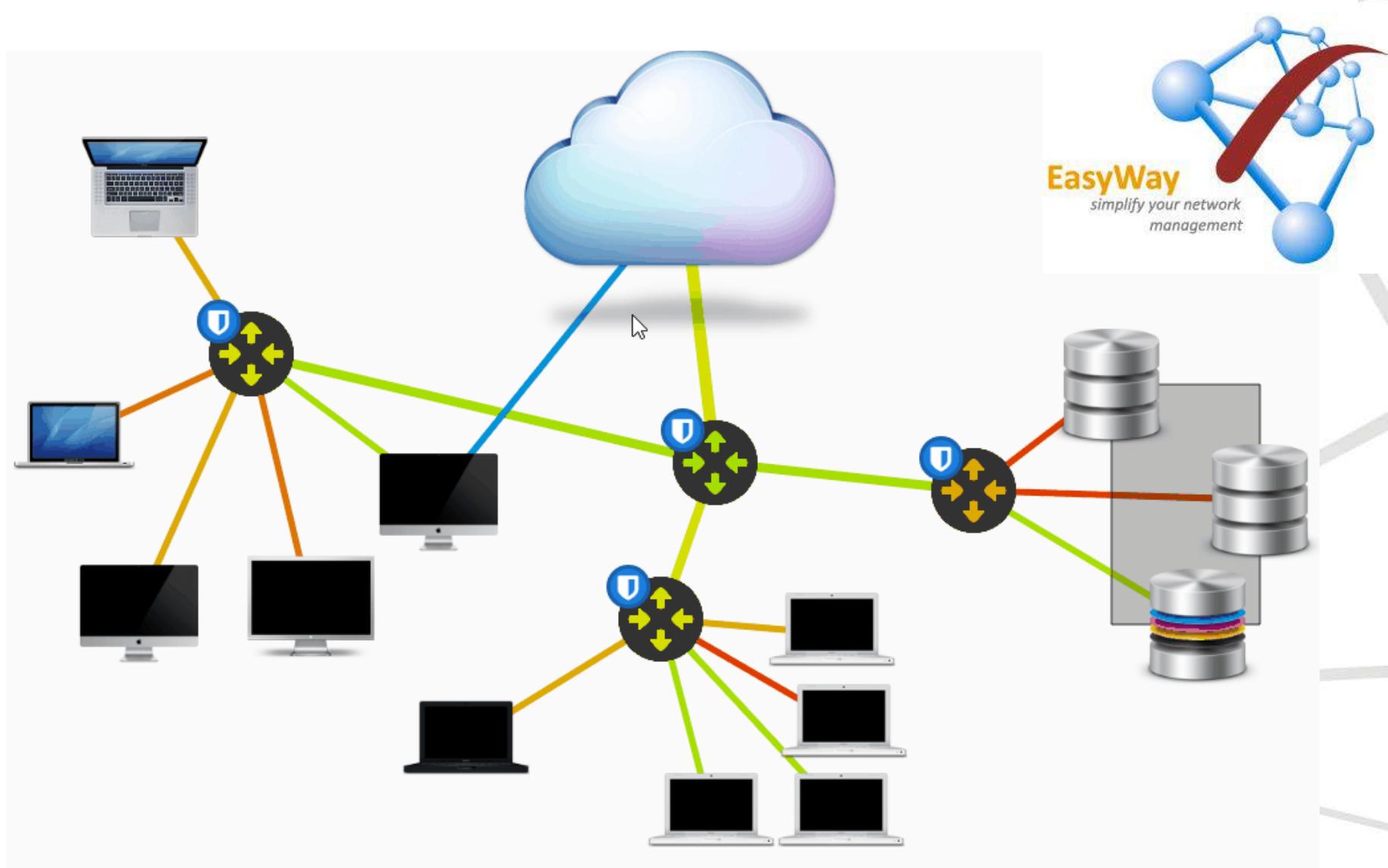
Реализация

Ключевые слова: C++11, 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

Пример графического интерфейса EasyWay



Описания релизов

Сейчас версия 0.5

- ядро контроллера
- построение топологии
- построение маршрута через всю сеть
- первая версия системы генерации правил
- 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** - следующий большой релиз (конец сентября)

- *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.
- *Обновление системы генерации правил* — повышена скорость работы и улучшена генерация правил (по количеству правил и числу приоритетов).
- Возможность статического связывания модулей.
- Система тестов.

Проект с открытым исходным кодом

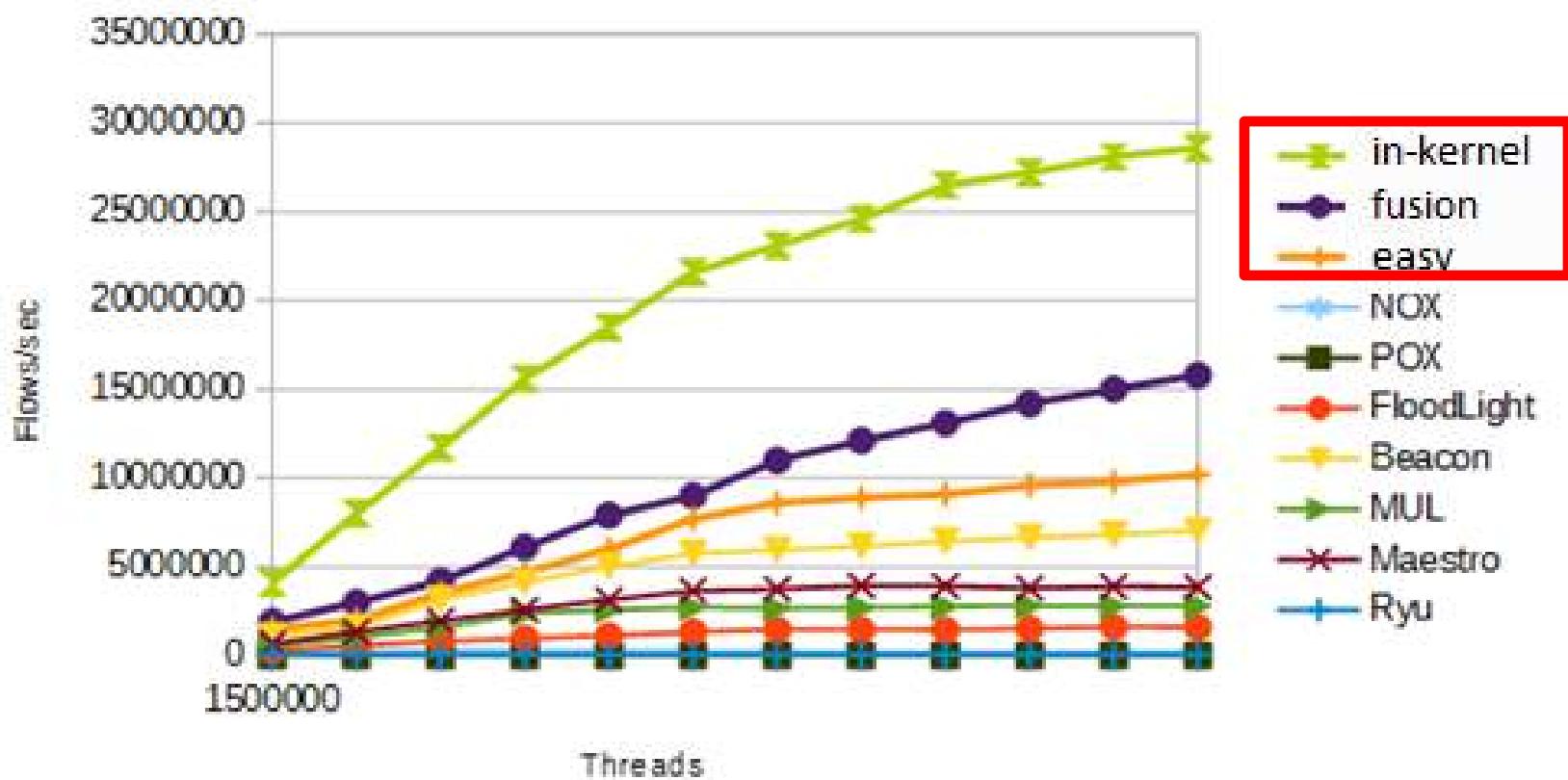
- Исходный код <http://arccn.github.io/runos/>
 - Apache, version 2.0
- Tutorial (Readme.md)
 - Как установить, запустить, написать свое первое приложение
- Виртуальная машина
 - Уже собранный контроллер
 - Средства для работы с OpenFlow
- Список рассылки
 - Google group **runos-ofc**



Программируемость

- Algorithmic policies (rule generation)
 - Arrangement of priorities of rules, combining of rules
 - LOAD, MATCH, READ abstractions
 - MAPLE based
- Client-friendly API using EDSL grammar (low level details are hidden inside the runtime – overloading, templates)
 - “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()”
- Application composition (parallel and sequential composition)
 - dpi + (lb >> forwarding)

Производительность



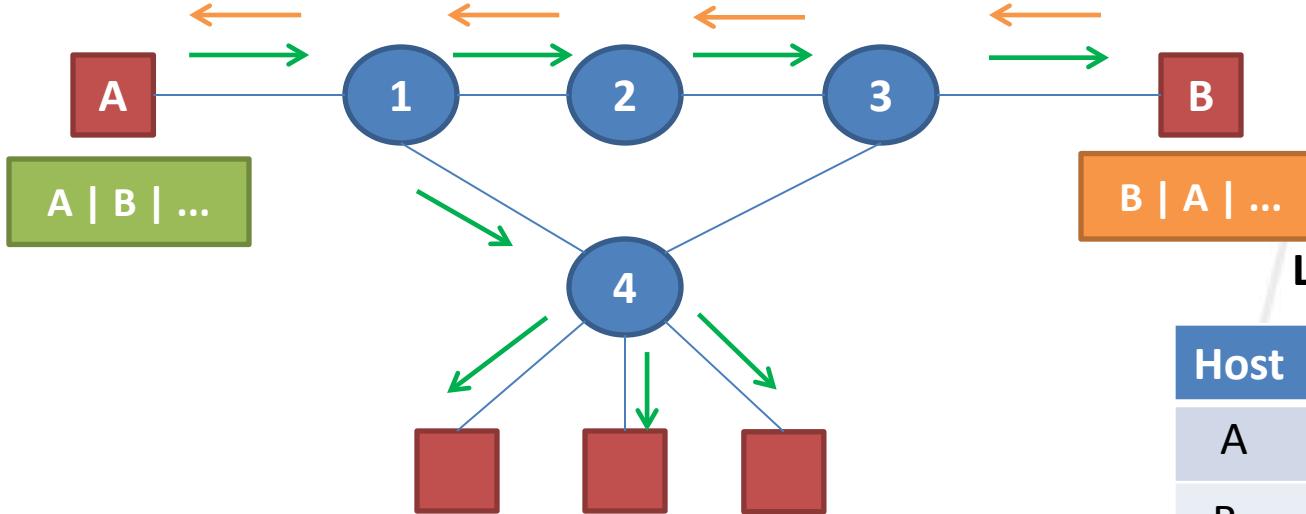
- Производительность : **10 млн. потоков в секунду**
- Задержка: **55 мкс**

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>

First application – L2 learning



L2 learning table

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

- What is L2 learning?
 - L2 table – where particular 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)

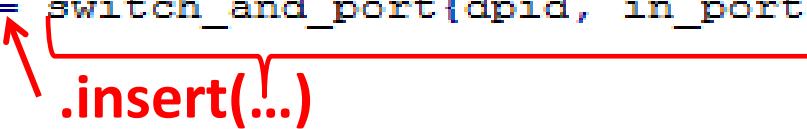
Host Databases

```
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;
    }
};
```



L2 forwarding application

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

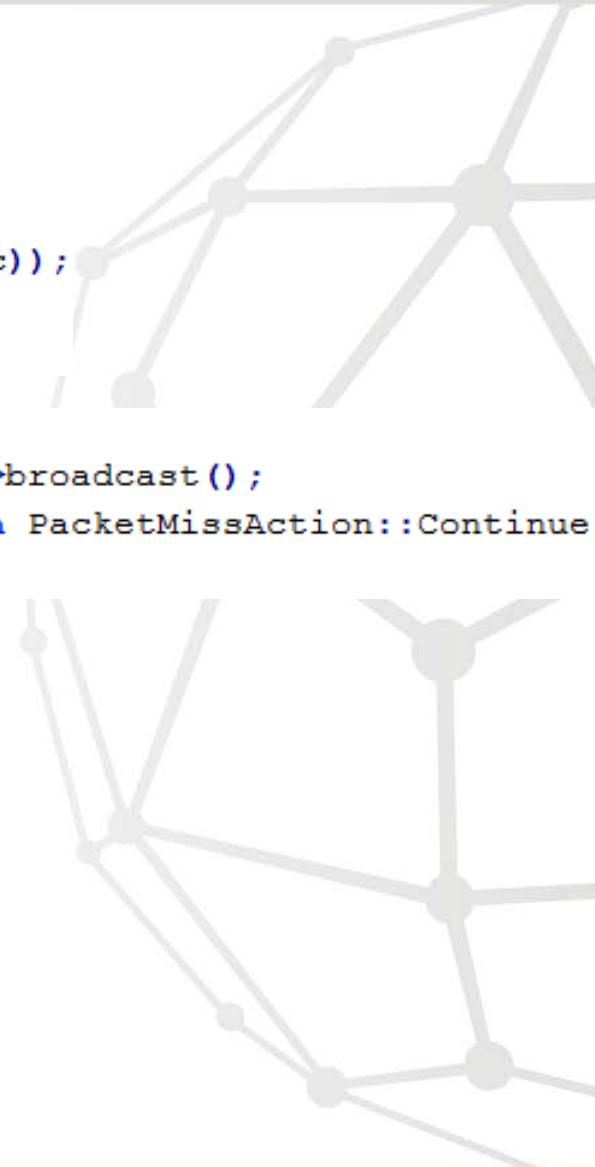
db->learn(connection->dpid(),
           pkt.load(ofb_in_port),
           packet_cast<TraceablePacket>(pkt).watch(ofb_eth_src));

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;
}
```



Заключение

- SDN/OpenFlow позволяет значительно упростить управление сетью.
- Проект RUNOS находится в открытом доступе
 - OpenFlow контроллер arccn.github.io/runos
 - Система управления корпоративной сетью EasyWay.
- RUNOS уже используется в университетах и в промышленности (сервис провайдеры, телеком операторы, ЦОД).



ЦЕНТР
ПРИКЛАДНЫХ
ИССЛЕДОВАНИЙ
КОМПЬЮТЕРНЫХ
СЕТЕЙ