Skoltech

Skolkovo Institute of Science and Technology

Lomonosov Moscow State University

OpenFlow контроллеры

Лекция 5

Василий Пашков

pashkov@lvk.cs.msu.su

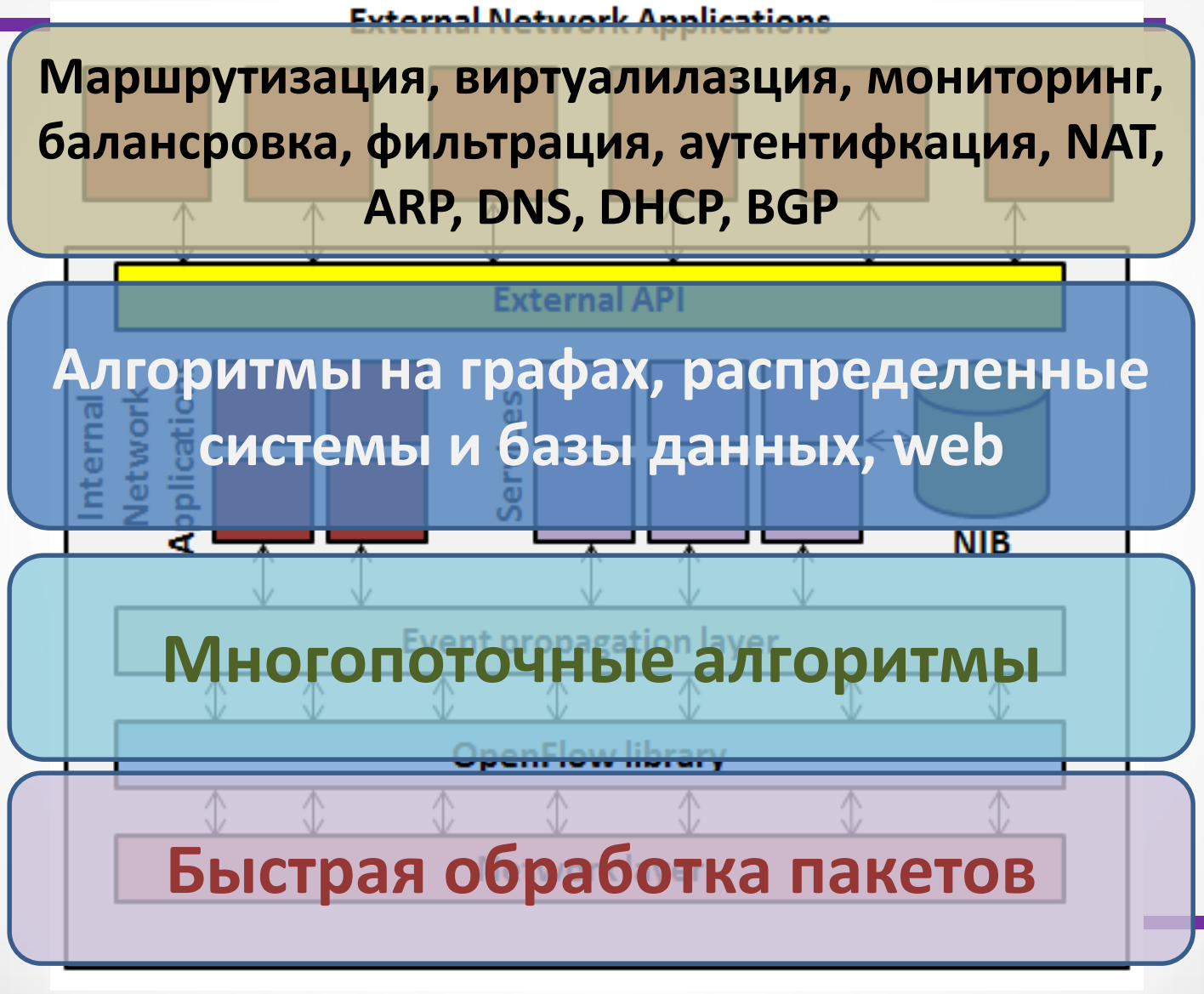
Skoltech
Skolkovo Institute of Science and Technology

План лекции

1. Классификация контроллеров
2. Архитектура
3. Проекты OpenFlow контроллеров

Skoltech
Skolkovo Institute of Science and Technology

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 >10M событий в секунду
- Реактивные контроллеры более “чувствительные”

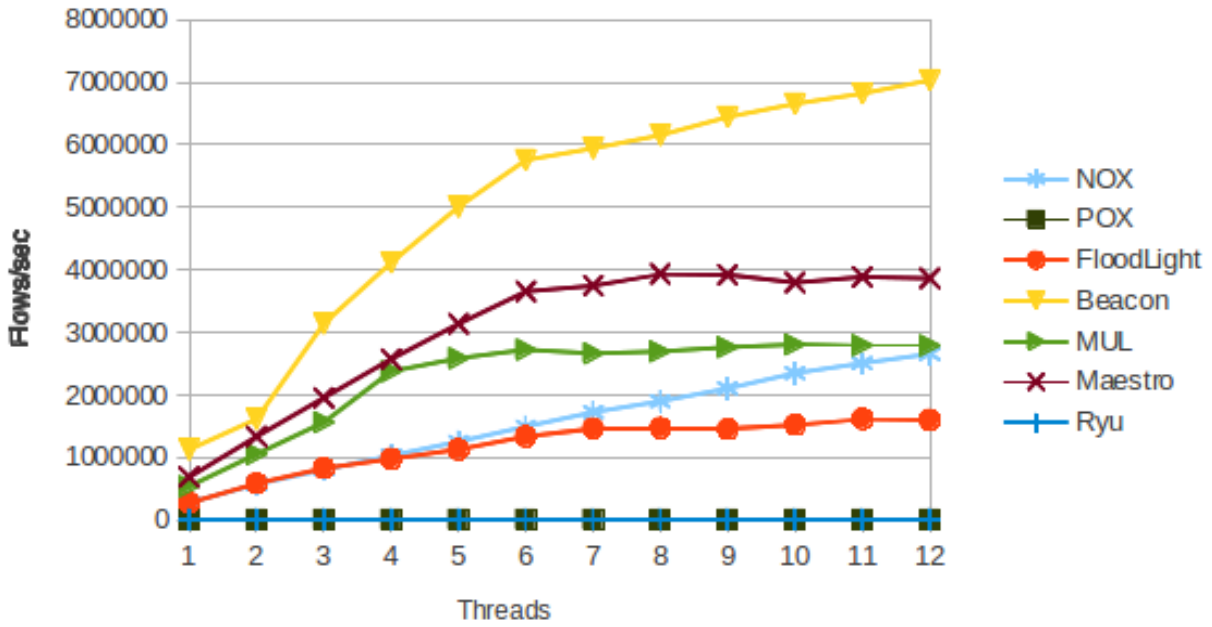
Полный список контроллеров (2013)

- 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)



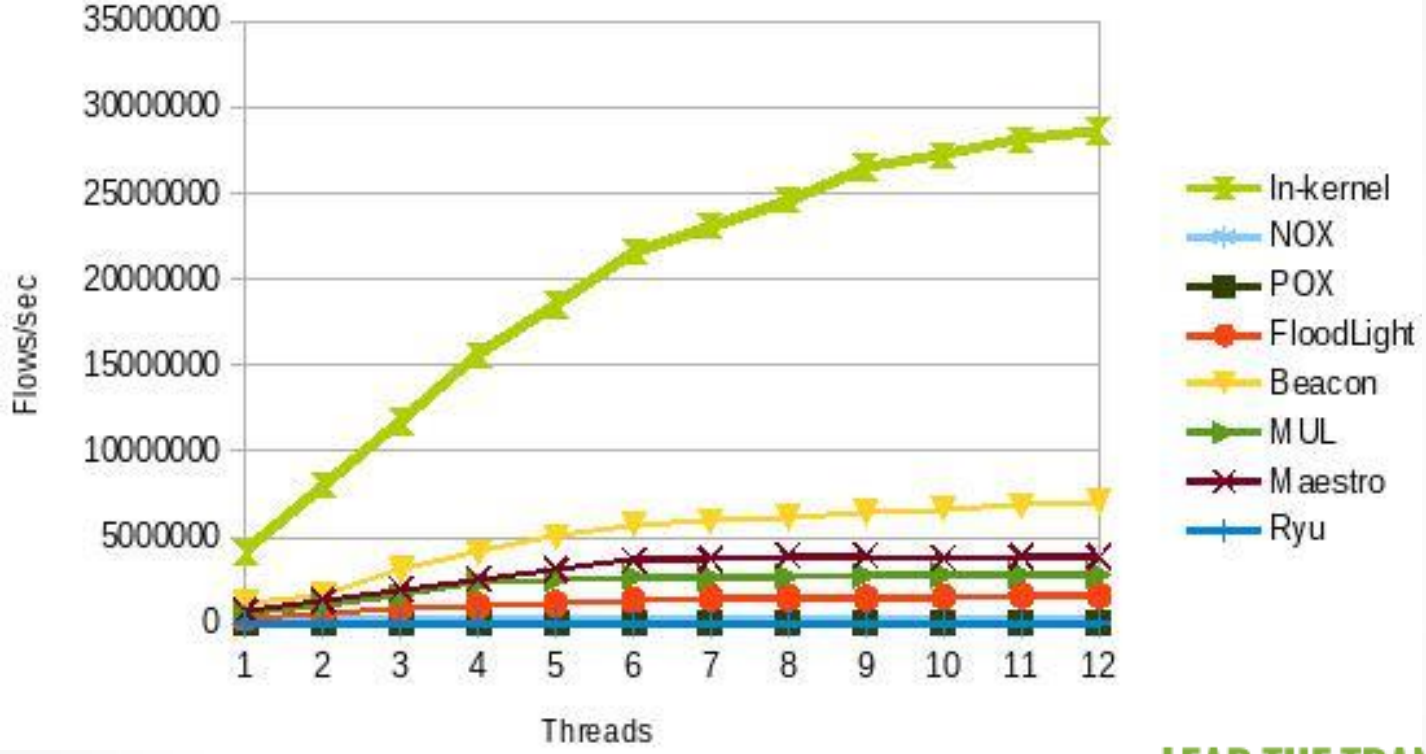
- Максимальная производительность **7 000 000 потоков в секунду**.
- Минимальное время задержки **от 50 до 75 мкс**.
- Недостатки:
  - Надежность контроллеров вызывала вопросы
  - Производительность была не достаточна (DC >10M fps)

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)



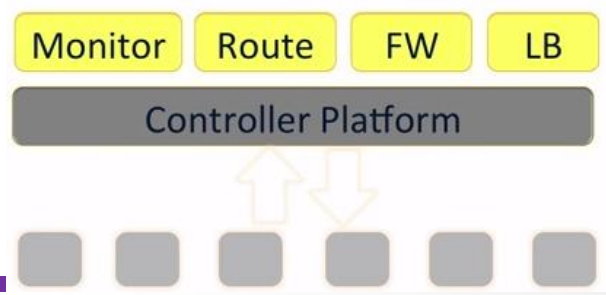
- Производительность равна **30M fps**
- Задержка **45us**

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



Skolte

Skolkovo Institute of Science and

Низкоровневые детали OpenFlow

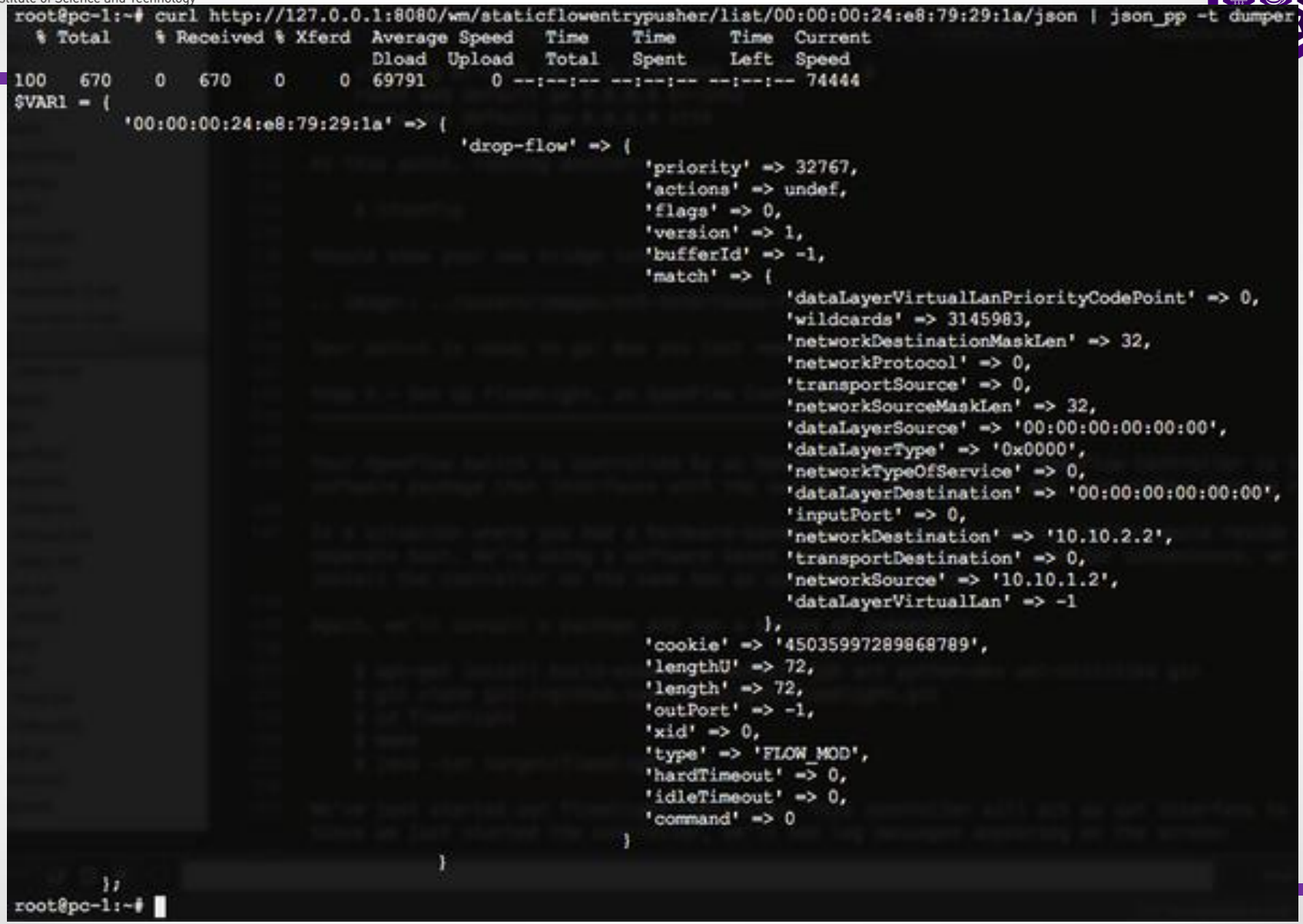
A.3.4.1 Modify Flow Entry Message

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);
```

Skoltech
Skolkovo Institute of Science and Technology

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:~#
```

Skoltech
Skolkovo Institute of Science and Technology

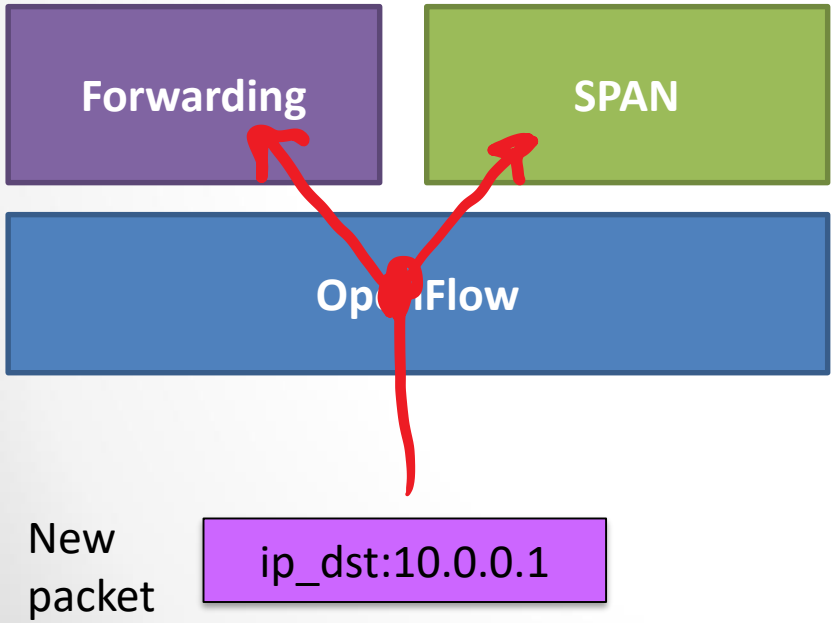
Низкоуровневые детали OpenFlow

```
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;
```

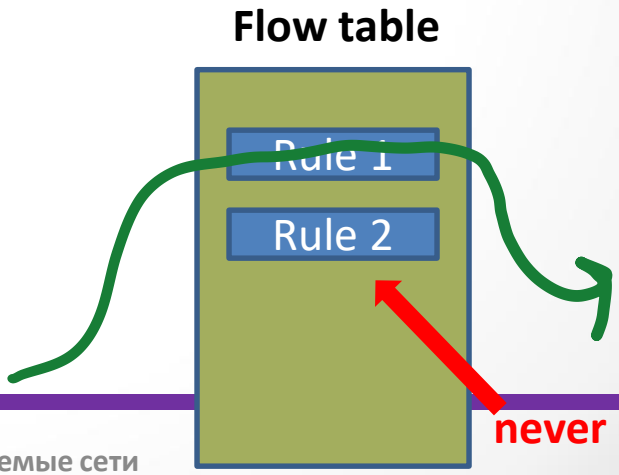
Possible problems in OpenFlow controllers

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



Skoltech
Skolkovo Institute of Science and Technology

Спасибо за внимание!

Василий Пашков
pashkov@lvk.cs.msu.su