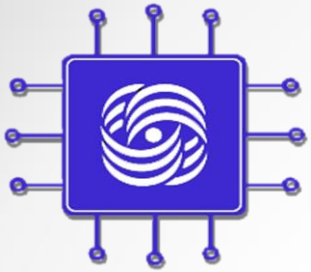


APPLIED
RESEARCH
CENTER FOR
COMPUTER
NETWORKS

Проблематика программируемости OpenFlow приложений

Математический спецкурс
“Программно Конфигурируемые Сети”

к.ф.-м.н., м.н.с., Шалимов А.В.

ashalimov@lvk.cs.msu.su

@alex_shali

@arccnnews

Часть 0: SDN/OpenFlow

Что такое SDN/OpenFlow?

SDN = **S**oftware **D**efined **N**etworking

Внедрения

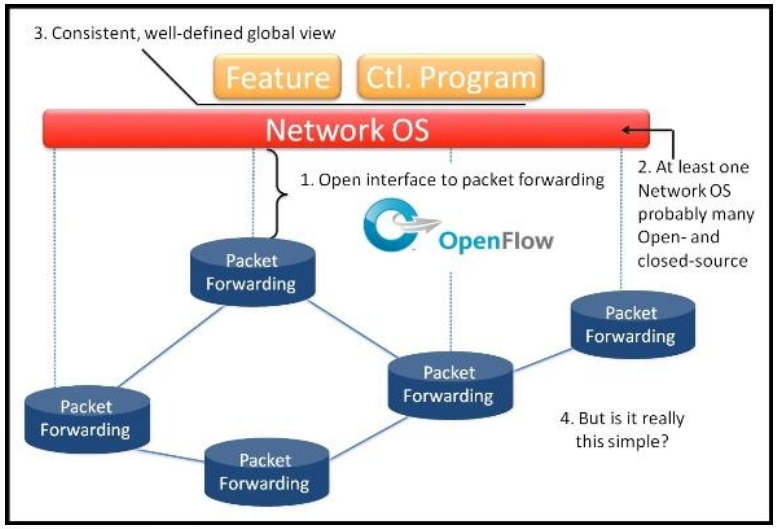
Google

NTT Communications

-
-
-

Основные принципы

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

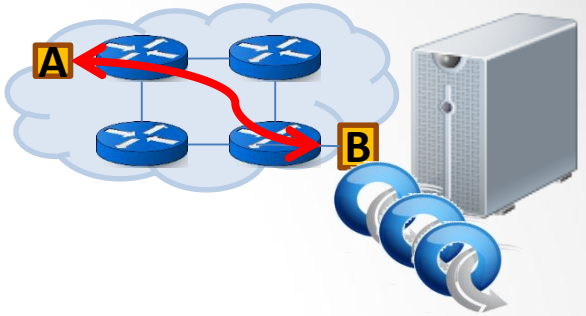


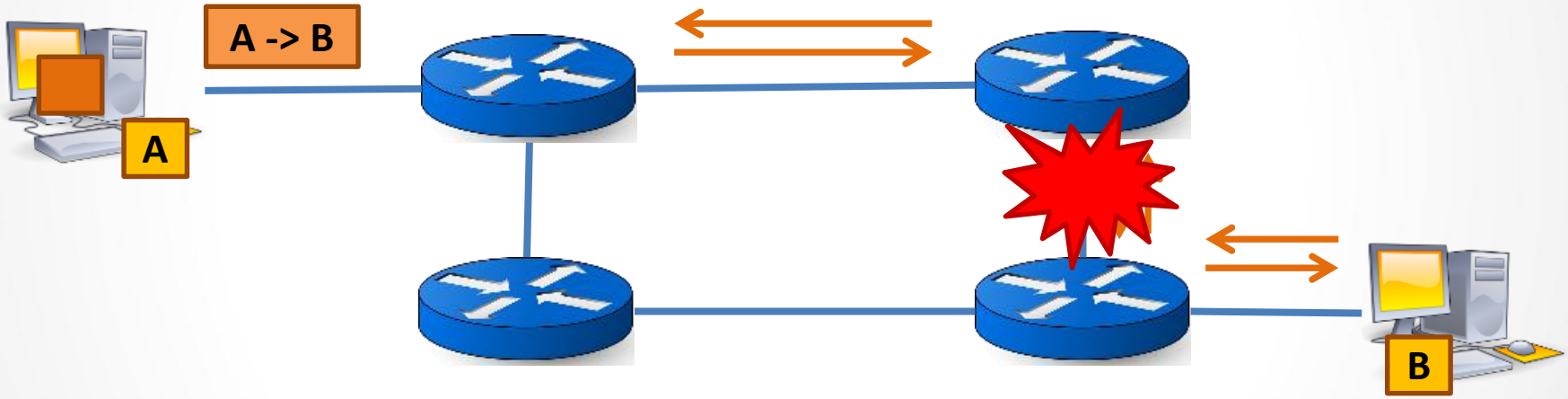
Преимущества

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

"SDN means thinking differently about networking"

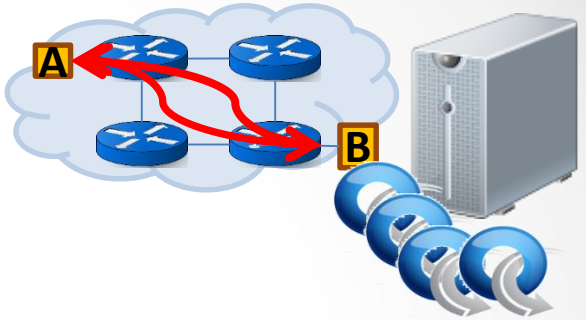
Маршрутизация с SDN/OpenFlow

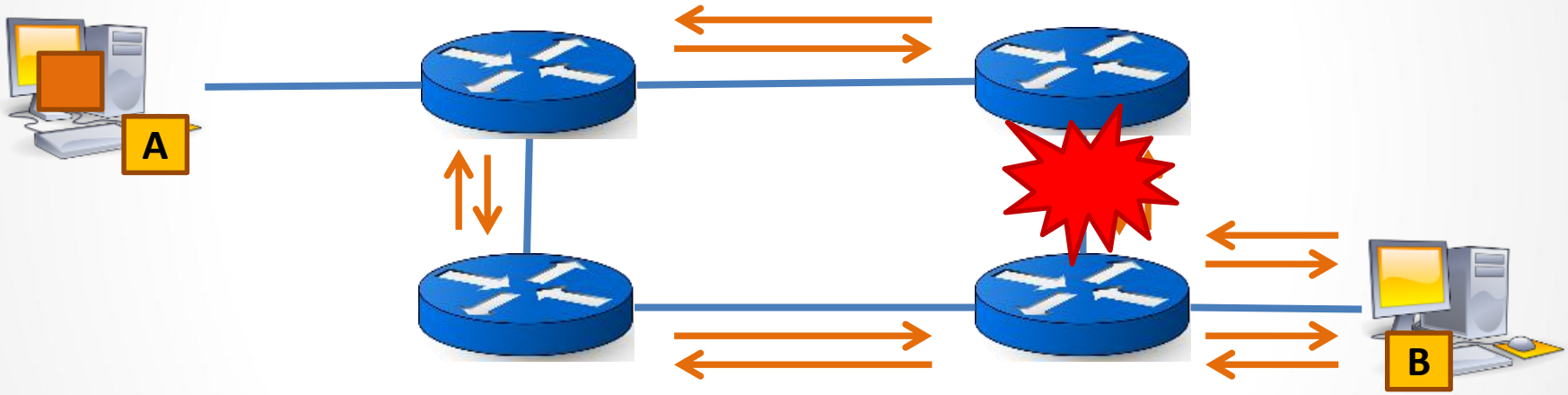




- Неизвестный пакет отправляется на контроллер (OF_PACKET_IN).
- Контроллер вычисляет лучший маршрут через всю сеть (с наименьшей стоимостью и удовлетворяющий политикам маршрутизации).
- Соответствующие правила OpenFlow устанавливаются на коммутаторы + сразу и обратный маршрут (OF_PACKET_OUT/FLOW_MOD).

Маршрутизация с SDN/OpenFlow





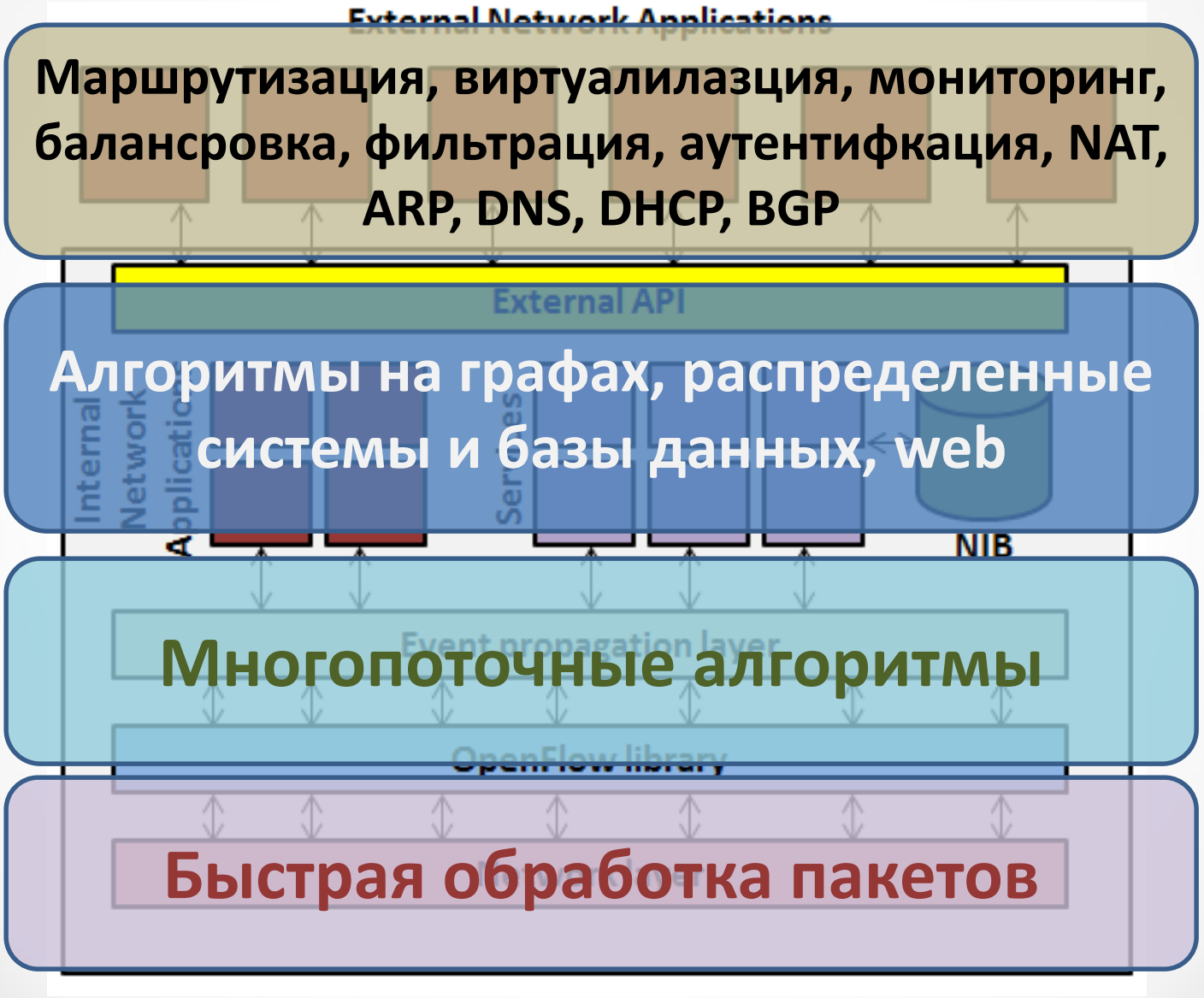
- Неизвестный пакет отправляется на контроллер (OF_PACKET_IN).
- Контроллер вычисляет лучший маршрут через всю сеть (с наименьшей стоимостью и удовлетворяющий политикам маршрутизации).
- Соответствующие правила OpenFlow устанавливаются на коммутаторы + сразу и обратный маршрут (OF_PACKET_OUT/FLOW_MOD).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

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

Архитектура OpenFlow контроллера



Часть I: OpenFlow контроллеры

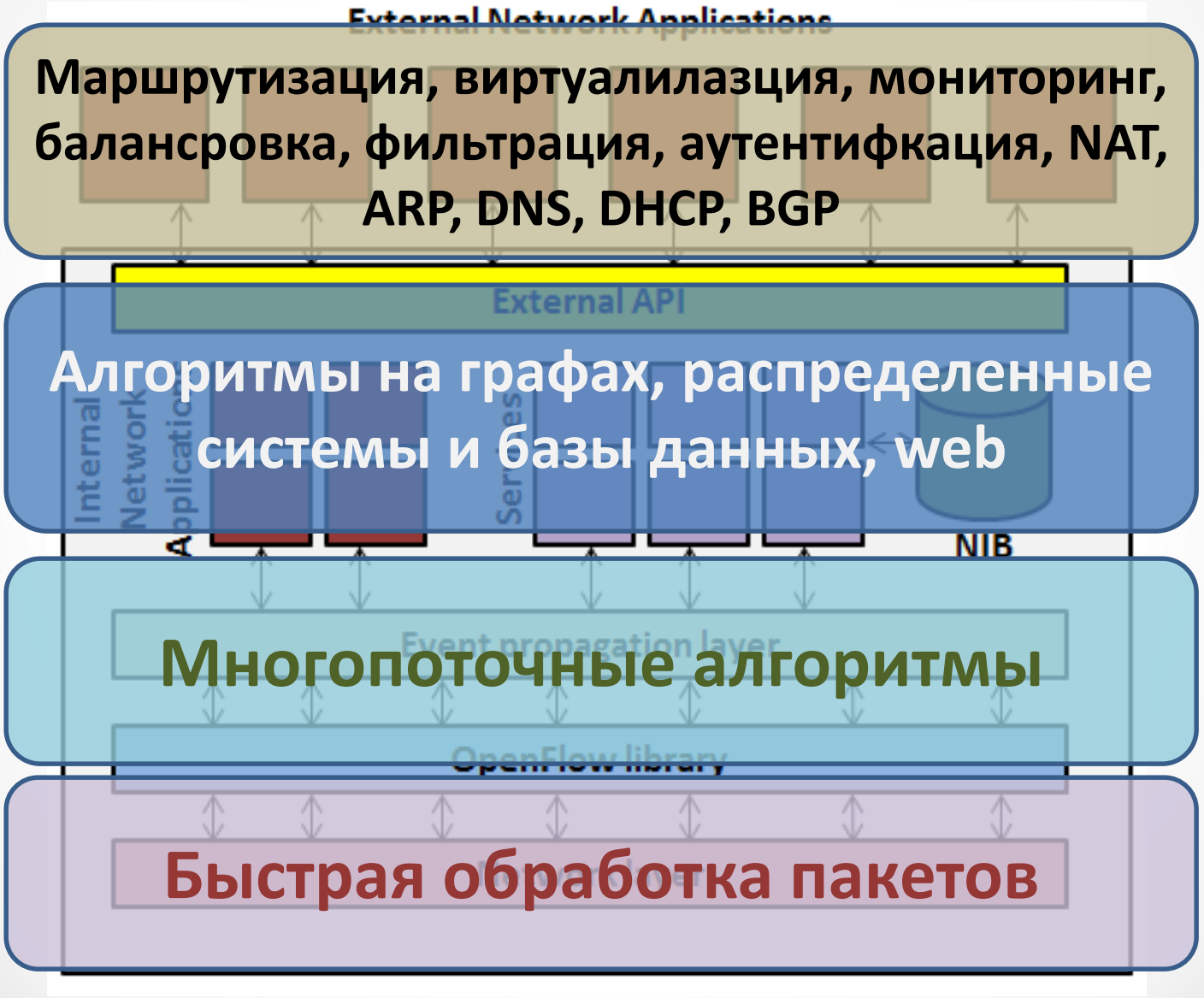
OpenFlow контроллер

- 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
  - 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

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

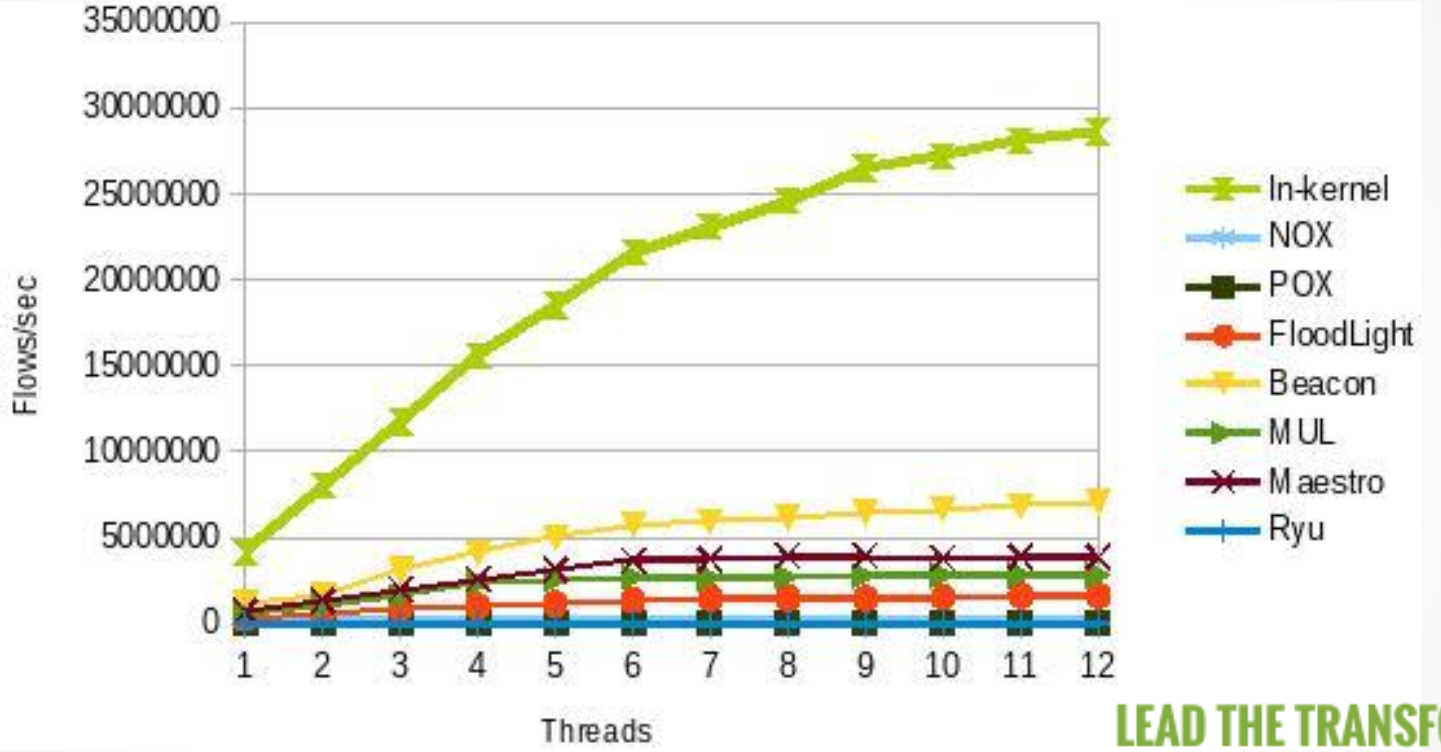
- ЦОД требует обработку >10М событий в секунду
- Реактивные контроллеры более “чувствительные”

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)

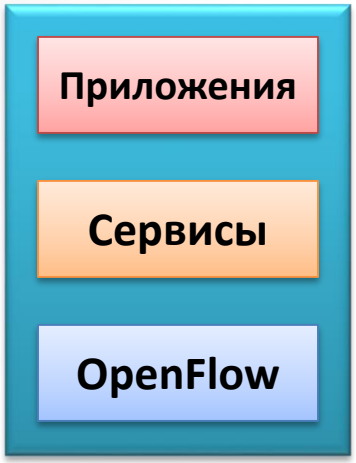


- Производительность равна **30M fps**
- Задержка **45us**

LEAD THE TRANSFORMATION
OPEN NETWORKING SUMMIT 2014
MARCH 3 - 5, 2014
SANTA CLARA CONVENTION CENTER & HYATT

RUNOS – это линейка OpenFlow контроллеров

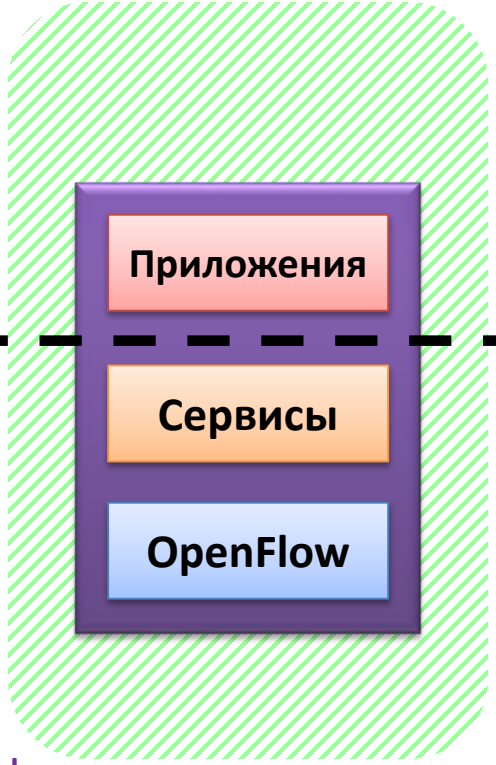
+: большая фунци-ть
-: низкая произв-ть



NOX, Pox, Floodlight, OpenDaylight, MUL, etc

User space
Kernel space

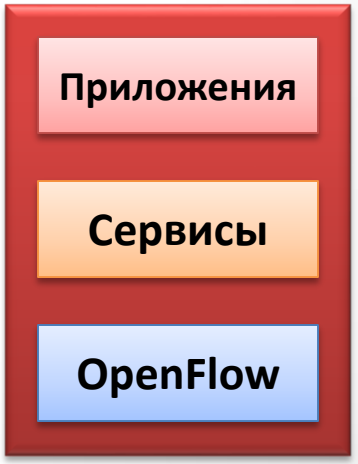
Что может быть по середине?



Архитектура:

- Прилож-я в userspace
- Часто используемые сервисы в ядре (топология)
- Интерфейс взаимодействия

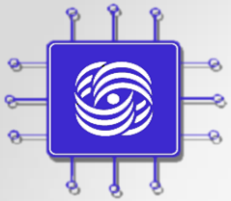
+: большая фунци-ть
+: высокая произв-ть



In-kernel

+: высокая произв-ть
-: мало функций

Часть II: Программируемость

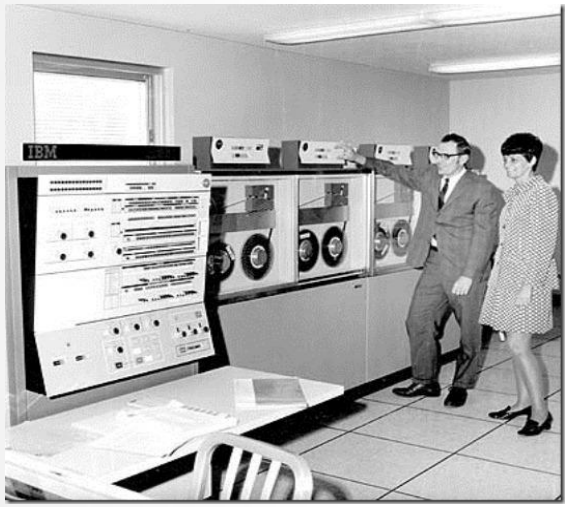


Абстракция

- 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

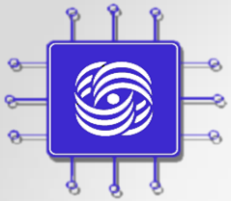
Операционные системы

Windows iOS

Открытый интерфейс

Микропроцессоры

intel AMD QUALCOMM



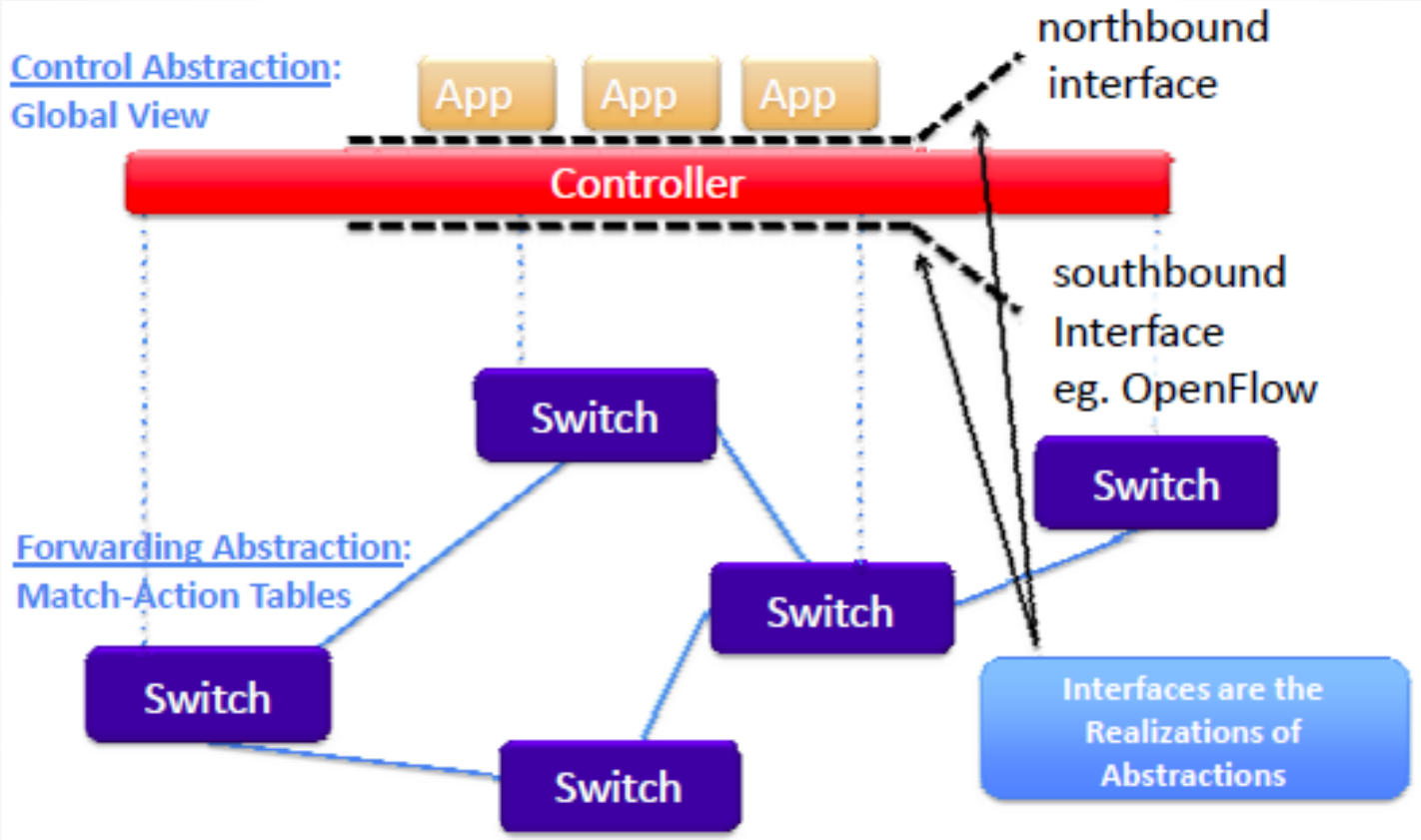
Абстракция в ЯП

- **Машинные языки программирования - нет никакой абстракции**
- **Высоко-уровневые языки программирования, ОС + другие абстракции**
  - Структуры данных, функции, переменные, файлы, виртуальная память, ...
- **Современные языки программирования – еще больше абстракций**
  - Объекты, нити, семафоры, сборщик мусора

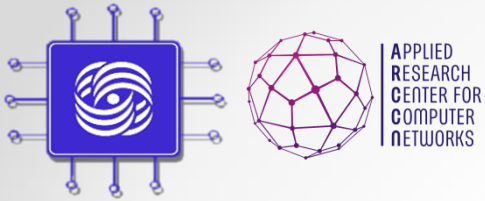
Абстракции упрощают программирование: проще писать, проще поддерживать, думать об алгоритме

Можно ли такое же получить для компьютерных сетей?

Абстракция в SDN?



- Абстракция уровня управления
- Абстракция уровня передачи данных



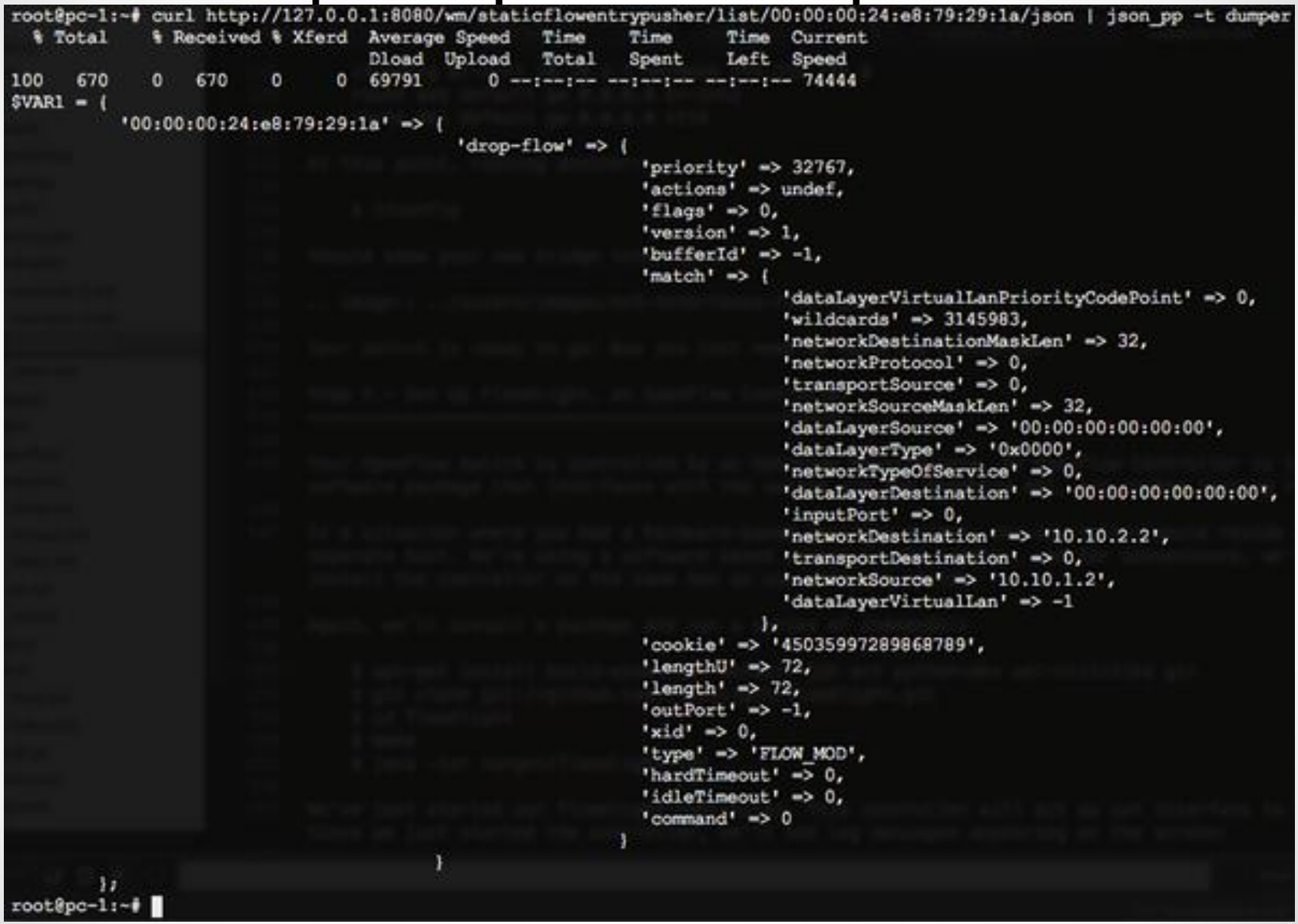
Абстракции OpenFlow

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

Программируемость

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

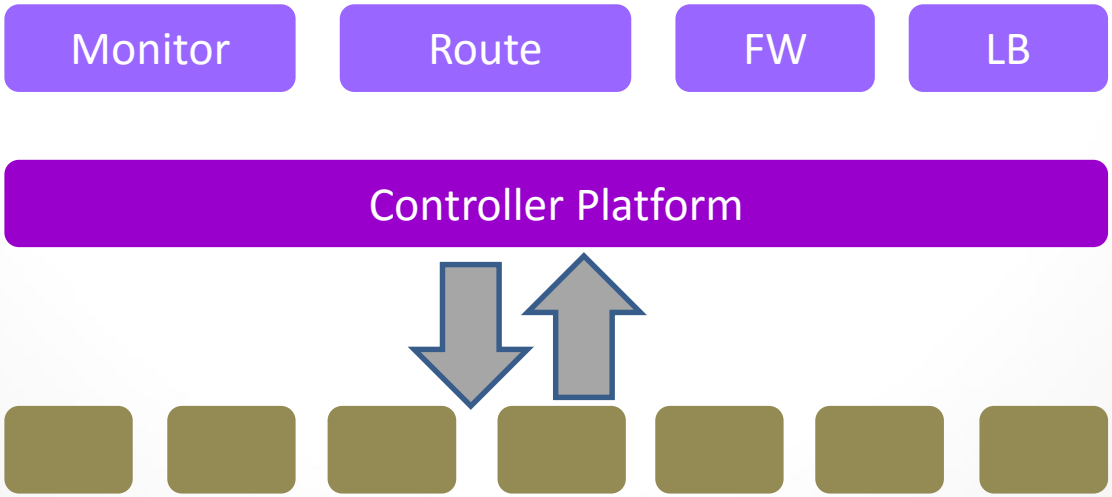
Пример REST запроса



```
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     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:~#
```

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



Низкоуровневые 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);
```

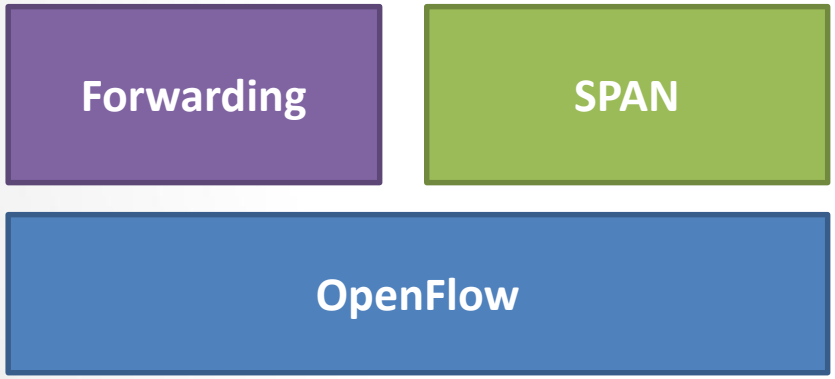
Низкоуровневые детали OpenFlow

```
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;
```

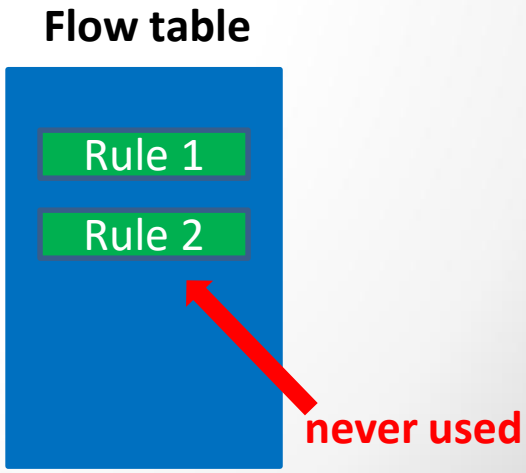
Possible problems in OpenFlow controllers

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!**



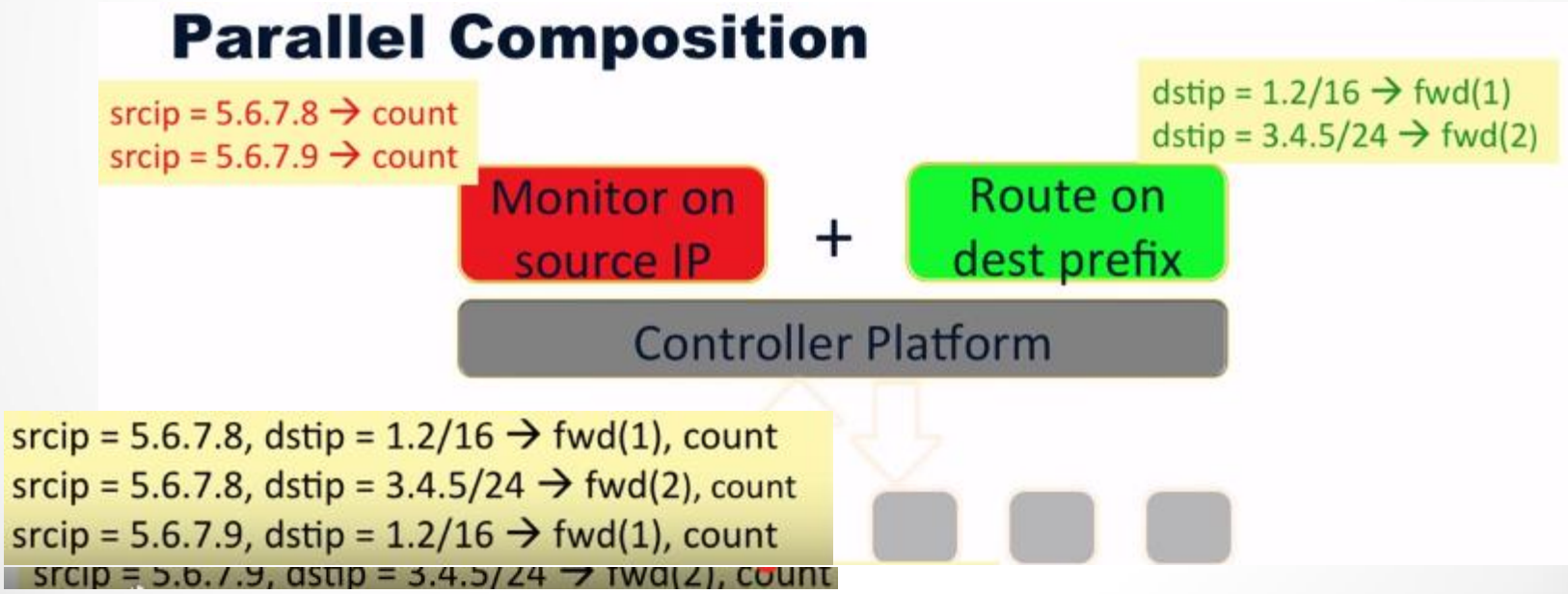
Rule 1: ip_src:10.0.0.1 -> output:1
Rule 2: ip_src:10.0.0.1 -> output:5

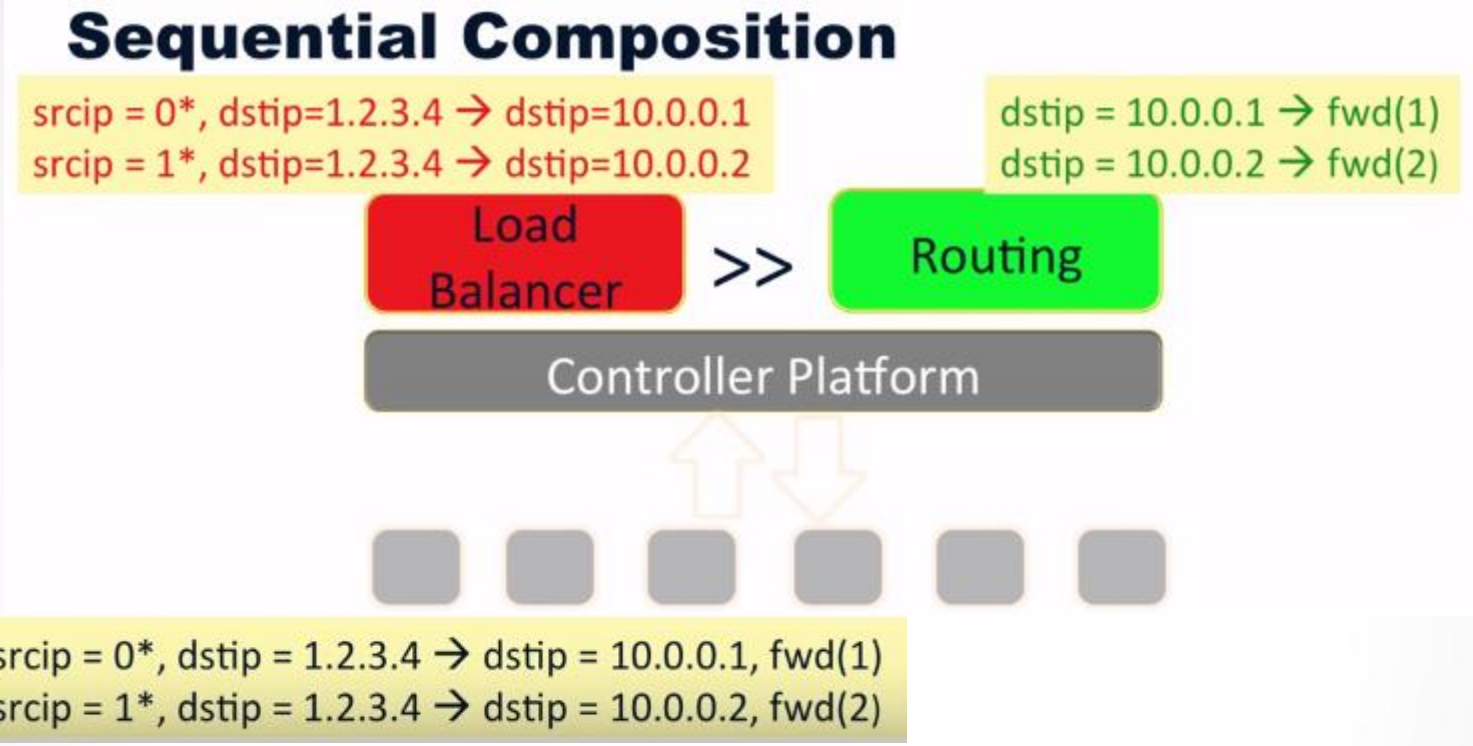


New packet

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)





- **Pyretic**
  - (match(dstip=2.2.2.8) >> fwd(1)) +
    (match(dstip=2.2.2.9) >> fwd(2))

Pyretic пример LB

As another example, consider a server load-balancing policy that splits request traffic directed to destination 1.2.3.4 over two backend servers (2.2.2.8 and 2.2.2.9), depending on the first bit of the source IP address (packets with sources starting with 0 fall under IP prefix 0.0.0.0/1 and are routed to 2.2.2.8). This results in the policy:

```
L = match(dstip='1.2.3.4') >>
    ((match(srcip='0.0.0.0/1') >> modify(dstip='2.2.2.8')) +
     (~match(srcip='0.0.0.0/1') >> modify(dstip='2.2.2.9')))
```

- http://frenetic-lang.org/publications/pyretic-login13.pdf

Maple

Пример проблемы

- Безопасность: TCP потоки на 22 порт идут безопасным маршрутом
- Остальные потоки идут кратчайшим маршрутом
- При этом надо запоминать, где какие конечные узлы находятся

```
def f(pkt):
  srcSw = pkt.switch(); srcInp = pkt.inport()
  if locTable[pkt.eth_src()] != (srcSw,srcInp):
    invalidateHost(pkt.eth_src())
    locTable[pkt.eth_src()] = (srcSw,srcInp)
  dstSw = lookupSwitch(pkt.eth_dst())
  if pkt.tcp_dst_port() == 22:
    outcome.path = securePath(srcSw,dstSw)
  else:
    outcome.path=shortestPath(srcSw,dstSw)
  return outcome
```

В реальной жизни

```
def packet_in(pkt):
  srcSw = pkt.switch(); srcInp = pkt.inport()
  locTable[pkt.eth_src()] = (srcSw,srcInp)
  dstSw = lookupSwitch(pkt.eth_dst())
  if pkt.tcp_dst_port() == 22:
   (nextHop,path)=securePath(srcSw,dstSw)
  else:
   (nextHop,path)=shortestPath(srcSw,dstSw)
  fixEndPorts(path,srcSw,srcInp,pkt.eth_dst())
  for each sw in path:
   inport' = path.inportAt(sw)
   outport'= path.outportAt(sw)
   installRule(sw,inport',
               exactMatch(pkt),
               [output(outport')])
  forward(srcSw,srcInp,action,nextHop)
```

Что будет, если добавить это правило?

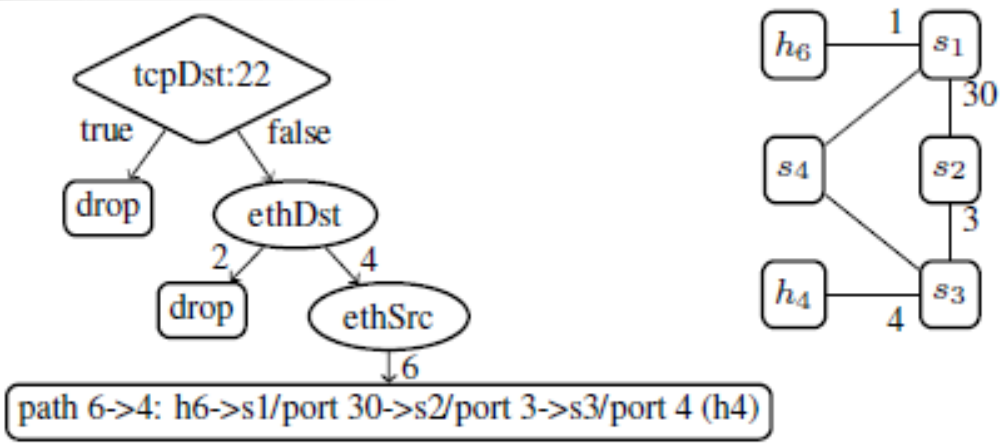
```
installRule(sw,inport',
            {from:pkt.eth_src(), to:pkt.eth_dst()},
            [output(outport')])
```

- Как решить проблему?
  - Приоритеты

Еще вариант

```
def packet_in(pkt):
 ...
 if pkt.tcp_dst_port() == 22:
   (nextHop,path)=securePath(srcSw,dstSw)
   fixEndPorts(path,srcSw,srcInp,pkt.eth_dst())
   for each sw in path:
     inport' = path.inportAt(sw)
     outport'= path.outportAt(sw)
     installRule(sw,inport',priority=HIGH,
                 {from:pkt.eth_src(),
                  to:pkt.eth_dst(),to_tcp_port:22},
                 [output(output')])
   forward(srcSw,srcInp,action,nextHop)
 else:
   (nexthop,path)=shortestPath(srcSw,dstSw)
   fixEndPorts(path,srcSw,srcInp,pkt.eth_dst())
   for each sw in path:
    inport' = path.inportAt(sw)
    outport'= path.outportAt(sw)
    installRule(sw,inport',priority=MEDIUM,
                                 {from:pkt.eth_src(),
                                  to:pkt.eth_dst(),to_tcp_port:22},
                                 [output(toController)])
                  installRule(sw,inport',priority=LOW,
                         {from:pkt.eth_src(),to:pkt.eth_dst()},
                         [output(outport')])
               forward(srcSw,srcInp,action,nextHop)
```

Trace Tree



- Три типа вершин
  - L – leaf
  - V – value
  - T – test
  - +empty

Flow table

```
1  Algorithm BUILDFT (t)
2      priority ← 0;
3      BUILD(t, any);
4      return;
5  Procedure BUILD (t, m)
6      if type_t = L then
7          emitRule(priority, m, value_t);
8          priority ← priority + 1;
9      else if type_t = V then
10         for v ∈ keys(subtrees_t) do
11             BUILD(subtrees_t[v], m ∧ (attr_t : v));
12     else if type_t = T then
13         BUILD(t_, m);
14         m_t = m ∧ (attr_t : value_t);
15         emitRule(priority, m_t, ToController);
16         priority ← priority + 1;
17         BUILD(t_+, m_t);
```

```
[ (3, tcp_dst_port=22          , drop),
  (2, tcp_dst_port=22          , toController),
  (1, eth_dst=4 && eth_src=6, port 30),
  (0, eth_dst=2                , drop)]
```

Оптимизации

- По числу правил
- По приоритетам
- Генератор правил
- Добавление потоков

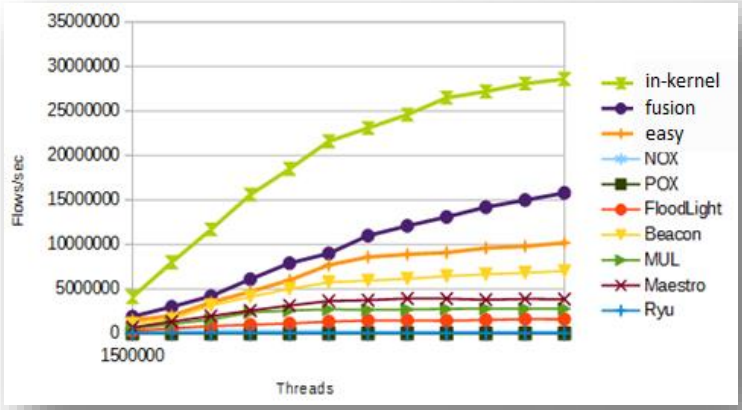
Часть III: Runos OpenFlow контроллер

Сетевая операционная система Runos

Система управления сетью первый российский SDN-контроллер RUNOS

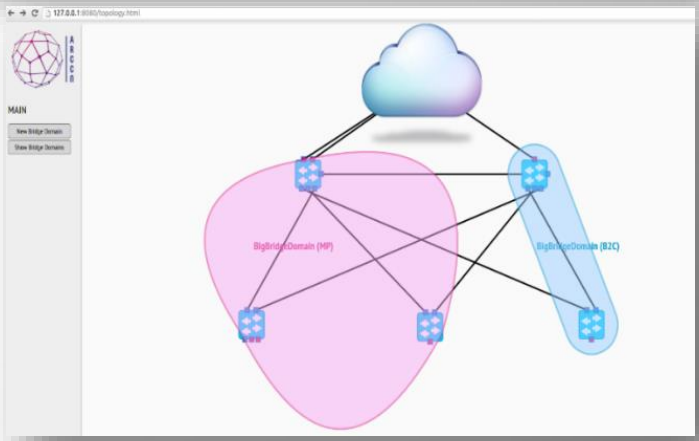
RUssian Network Operation System

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

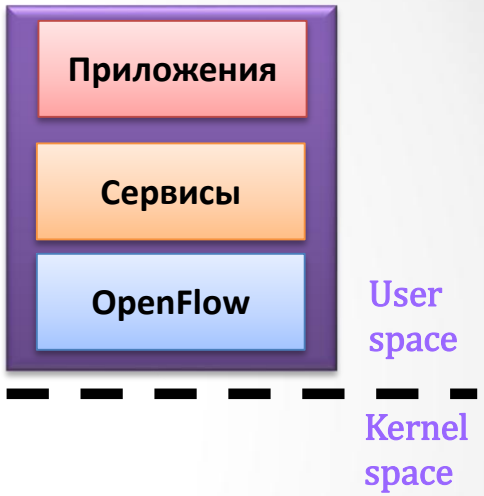
- **Внутренняя ядерная версия**
  - Супер-производительность 30 млн событий в секунду
  - Разработка приложений под заказчика



- **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

- **Алгоритмические политики (генерация правил)**
  - Распределение приоритетов, комбинация правил
  - LOAD, MATCH, READ абстракции
  - На основе MAPLE
- **Дружественный API на основе EDSL грамматики (низкоуровневые детали скрыты в системе выполнения – 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()”
- **Композиция приложений (параллельная и последовательная композация)**
  - dpi + (lb >> forwarding)



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)

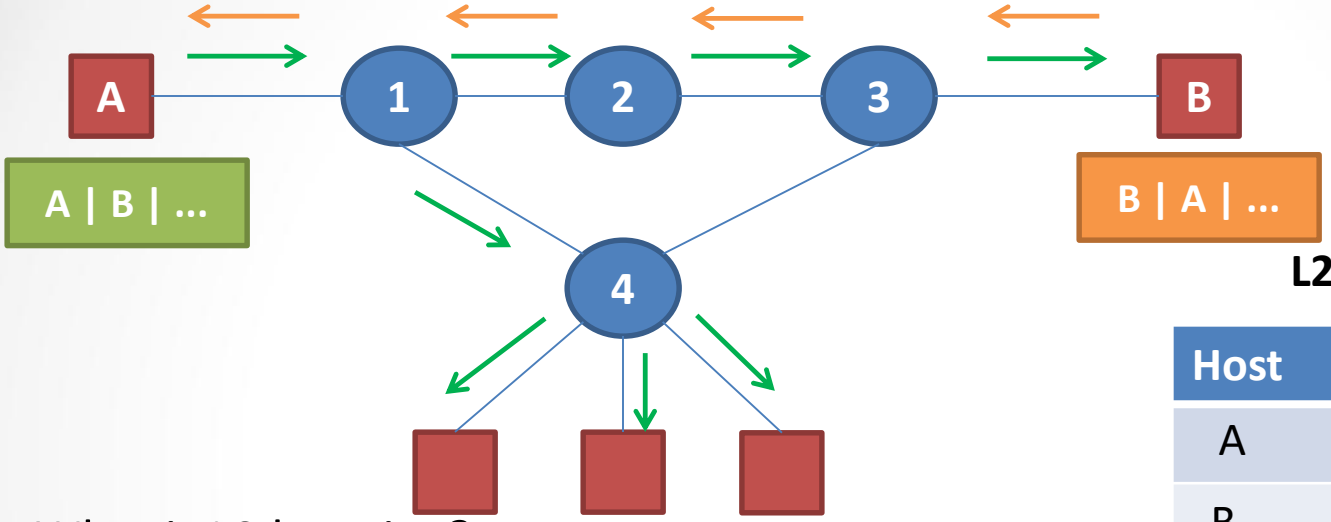
Описания релизов

Версия **0.6** - следующий большой релиз (апрель)

- **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 количетсву правил и числу приоритетов).
- Возможность статического связывания модулей.
- Система тестов.

Часть IV: Разработка приложений для Runos контроллер

First application – L2 learning



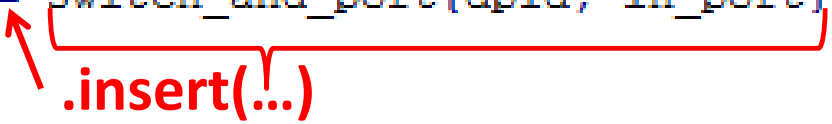
L2 learning table

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;
    }
};
```

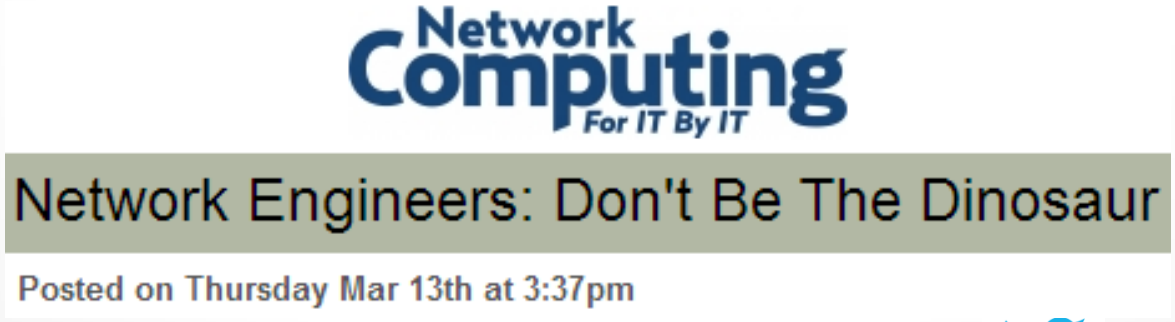


L2 forwarding application

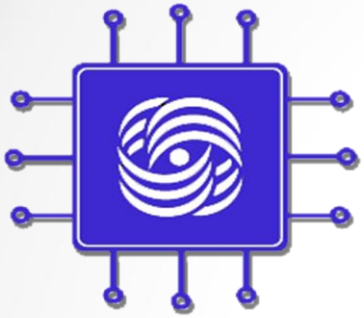
```
// Get required fields
ethaddr dst_mac = pkt.load(ofb_eth_dst);

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;
}
```

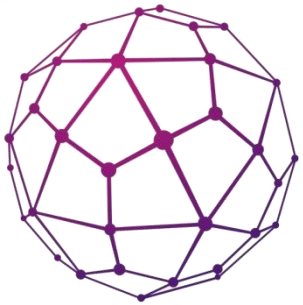
Заключение

- SDN это все об абстракциях программировать
- Runos проект предназначен для упрощения разработки новых приложений
  - В открытом доступе arccn.github.io/runos



http://arccn.ru/ ashalimov@lvk.cs.msu.su @alex_shali @arccnnews





ЦЕНТР
ПРИКЛАДНЫХ
ИССЛЕДОВАНИЙ
КОМПЬЮТЕРНЫХ
СЕТЕЙ

Спасибо за внимание!