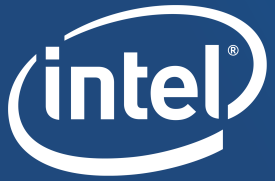


BAREFOOT ACADEMY

Программирование Arista 7170 на языке P4

Владимир Гуревич

Директор Академии Barefoot

December 7, 2019

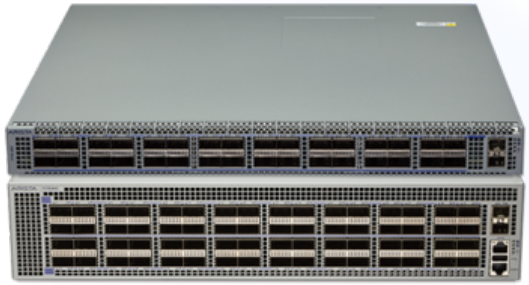
Краткое содержание

- Arista 7170 – коммутатор или пакетный процессор?
- Как его программировать?
- Дальнейшие шаги
- Вопросы и ответы

Системы Arista на основе Barefoot Tofino™

ARISTA 7170-32C, 7170-32D & 7170-64C SERIES OF **PROGRAMMABLE PLATFORMS**

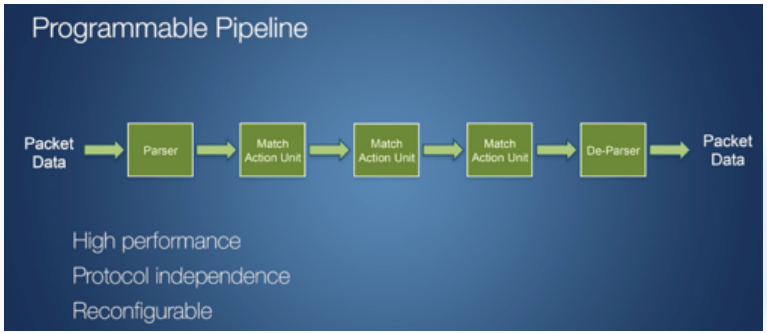
BAREFOOT



ARISTA

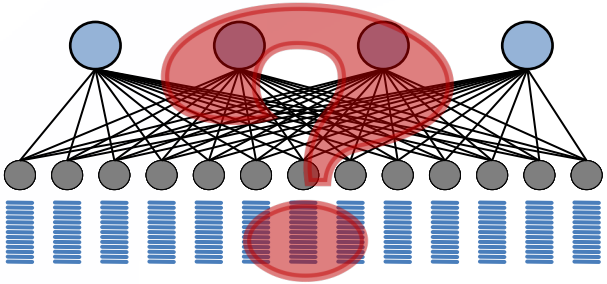
Мастер на все руки!

Barefoot Tofino Programmable Pipeline



А что тут удивительного?

Как разрабатывают сетевые устройства

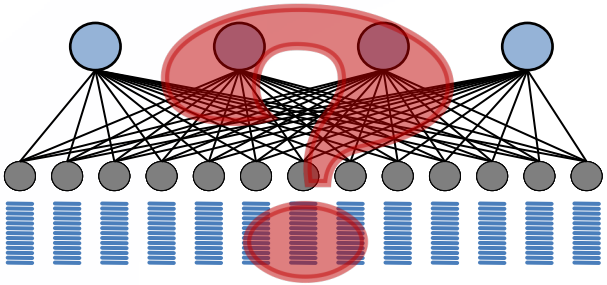


Я хочу строить сетевое оборудование вот так!

Требования

- **Алгоритм обработки пакетов <u>не</u> является частью большинства NOS ☹**
  - Встроен в микросхему коммутатора
  - Упор на стандартную обработку
  - Неформально документирован
  - Период разработки 1-2 года

А как разрабатывалась Arista 7170



Я хочу строить сетевое оборудование вот так!

Требования

- **Алгоритмы уровня обработки пакетов разработаны в Arista**
  - Написаны на языке P4
  - Работают на программируемой микросхеме Tofino

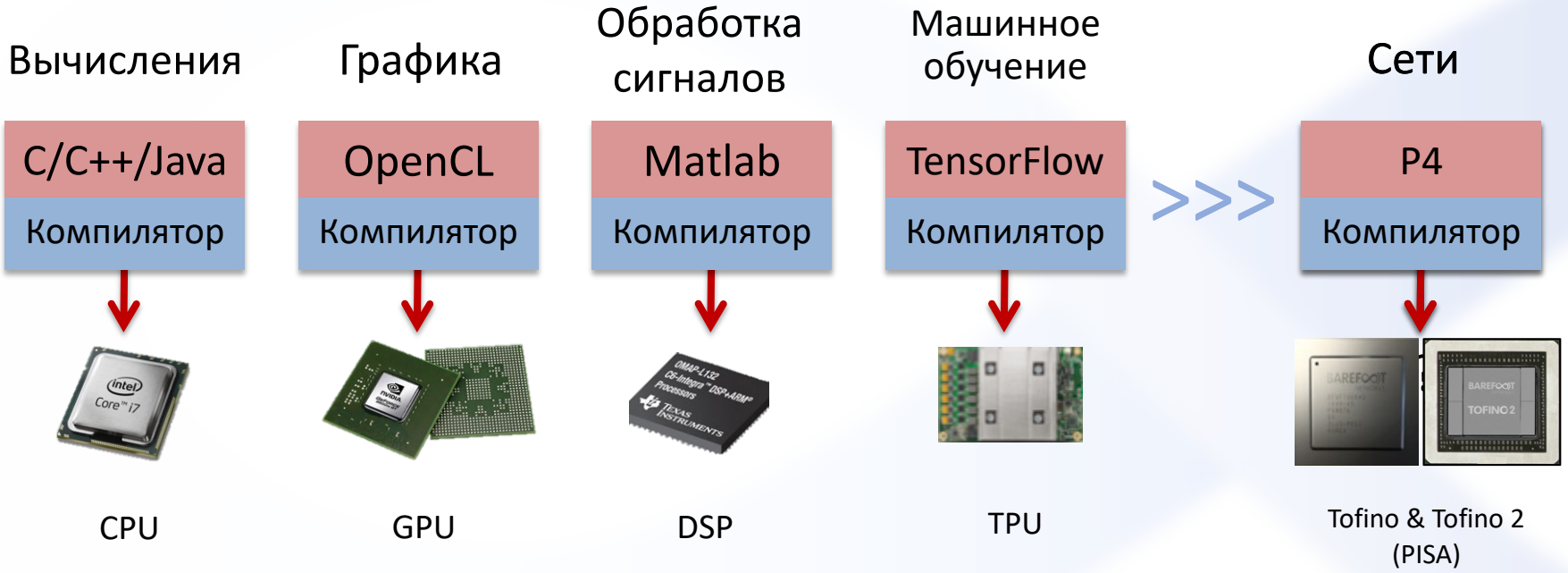
Результат налицо

	7170-64C	7260CX3	
L2/L3 Throughput	6.4Tb/s	6.4Tb/s	Same
Number of 100G Ports	64	64	Same
Availability	Yes	Yes	Same
Max Forwarding Rate	5B packets per sec	4.2B packets per sec	**20% better**
Max 25G/10G Ports	256/258	128/130	**2X**
Programmability	Yes (P4) – optimize for power	No	
Typical System Power draw	4.2W per port (Typ: 270W)	5.3W per port (Typ: 340W)	**25% better**
Large Scale NAT	Yes (100k)	No	
Large scale stateful ACL	Yes (100k)	No	
Large Scale Tunnels	Yes (192k)	No	
Packet Buffer	Unified	Segmented	**2X burst**
Segment Rtg/Bare Metal	Yes/Yes	No/No	
LAG/ECMP Hash Algorithm	Full entropy, programmable	Hash seed, reduced entropy	
ECMP	256 way	128 way	**2X**
Telemetry and Analytics	Line-rate per flow stats	Sflow (Sampled)	**~1000X**

Внешне системы идентичны:

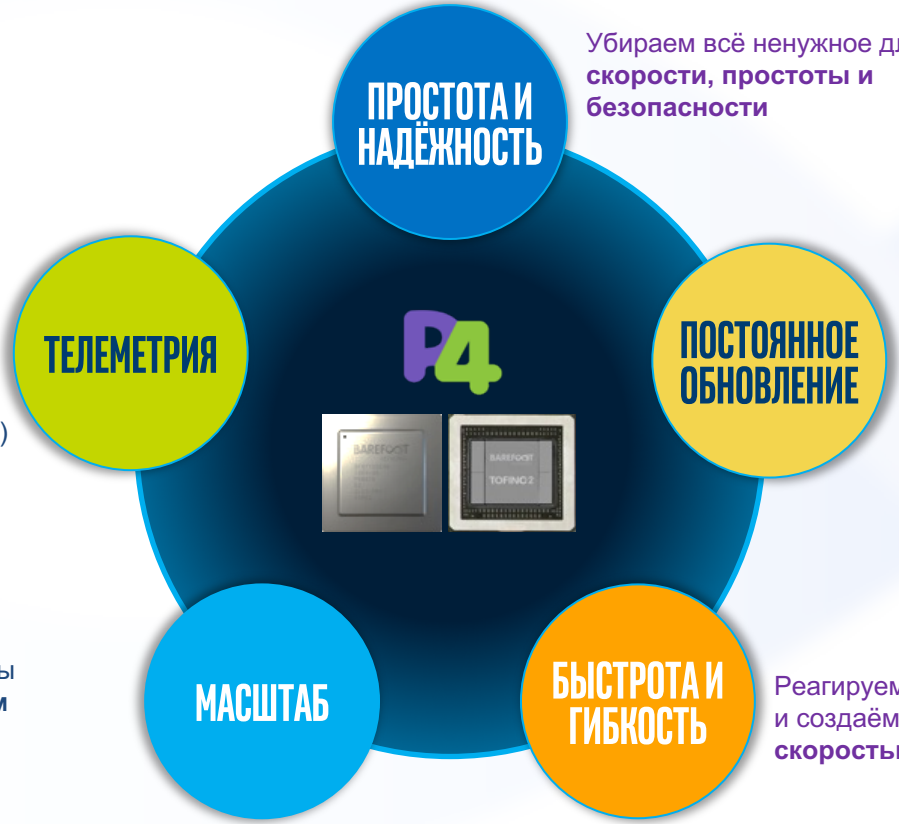
- Число портов
- Процессор
- Блоки питания

Эволюция в сторону программируемости



- **От фиксированного алгоритма – к программированию**
  - Специализированное устройство → специализированная архитектура
  - Фиксированный алгоритм → специализированный язык программирования

Преимущества программируемости



Убираем всё ненужное для **скорости, простоты и безопасности**

Barefoot **SPRINT**™
- **S**calable
- **P**rogrammable,
- **R**eal-Time,
- **I**nband **N**etwork **T**elemetry (**INT**)

Разрабатываем новый код на **существующем** оборудовании

Масштабируем ресурсы согласно **требованиям заказчиков**

Реагируем на новые требования и создаём новые решения **со скоростью программистов**

Что может делать P4

Описывать заголовки пакетов

```
header ethernet_h {
    bit<48>     dst_addr;
    bit<48>     src_addr;
    bit<16>     ether_type;
}

header vlan_tag_h {
    bit<3>       pcp;
    bit<1>       dei;
    bit<12>      vid;
    bit<16>      ether_type;
}
```

Разбирать заголовки пакетов

```
state parse_ethernet {
    pkt.extract(hdr.ethernet);
    transition select(hdr.ethernet.ether_type) {
            0x8100:  parse_vlan_tag;
            0x0800:  parse_ipv4;
            default: accept;
    }
}

state parse_vlan_tag {
    pkt.extract(hdr.vlan_tag);
    transition select(hdr.vlan_tag.ether_type) {
        0x0800:  parse_ipv4;
        default: accept;
    }
}
```

Что может делать P4

Описывать действия

```
action send(PortId_t port) {
    ig_tm_md.ucast_egress_port = port;
}

action drop() {
    ig_dprsr_md.drop_ctl = 1;
}

action l3_switch(bit<48>  new_da,
                 bit<48>  new_sa,
                 bit<12>  new_vid,
                 PortId_t port)
{
    hdr.ethernet.dst_addr   = new_da;
    hdr.ethernet.src_addr   = new_sa;
    hdr.ethrenet.ether_type = 0x8100;
    hdr.vlan_tag = { 0, 0, new_vid, 0x0800};
    hdr.ipv4.ttl = hdr.ipv4_ttl - 1;
    send(port);
}
```

Системная переменная

Модификация полей заголовка

Добавление нового заголовка

Описывать таблицы match-action

```
table ipv4_host {
    key = {
        meta.vrf_id       : exact;
        hdr.ipv4.dst_addr : exact;
    }
    actions = { send; drop; l3_switch; NoAction; }
    const default_action = NoAction();
    size = 131072;
}

table ipv4_lpm {
    key = {
        meta.vrf_id       : ternary;
        hdr.ipv4.dst_addr : lpm;
    }
    actions = { send; drop; l3_switch; }
    size = 32768;
    default_action = send(CPU_PORT);
}
```

Вид сопоставления

Макс. число записей в таблице

Вид сопоставления

Что делать если запись не найдена

Что может делать P4

Описывать алгоритмы обработки

```
apply {
    if (ig_parser_md.parser_error != 0 ||
        meta.ipv4_checksum_err) {
        drop();
    }

    if (hdr.ipv4.isValid()) {
        if (hdr.ipv4.ttl > 1) {
            if (ipv4_host.apply().miss) {
                ipv4_lpm.apply();
            }
        }
    }
}

apply {
    pkt.emit(hdr.ethernet);
    pkt.emit(hdr.vlan_tag);
    pkt.emit(hdr.ipv4);
}
```

Результат сопоставления

Выполнение операции сопоставления-действия

Вывод заголовков пакета в нужном порядке

Предоставлять доступ с специальным функциям

Определение циклического кода на основе заданного многочлена

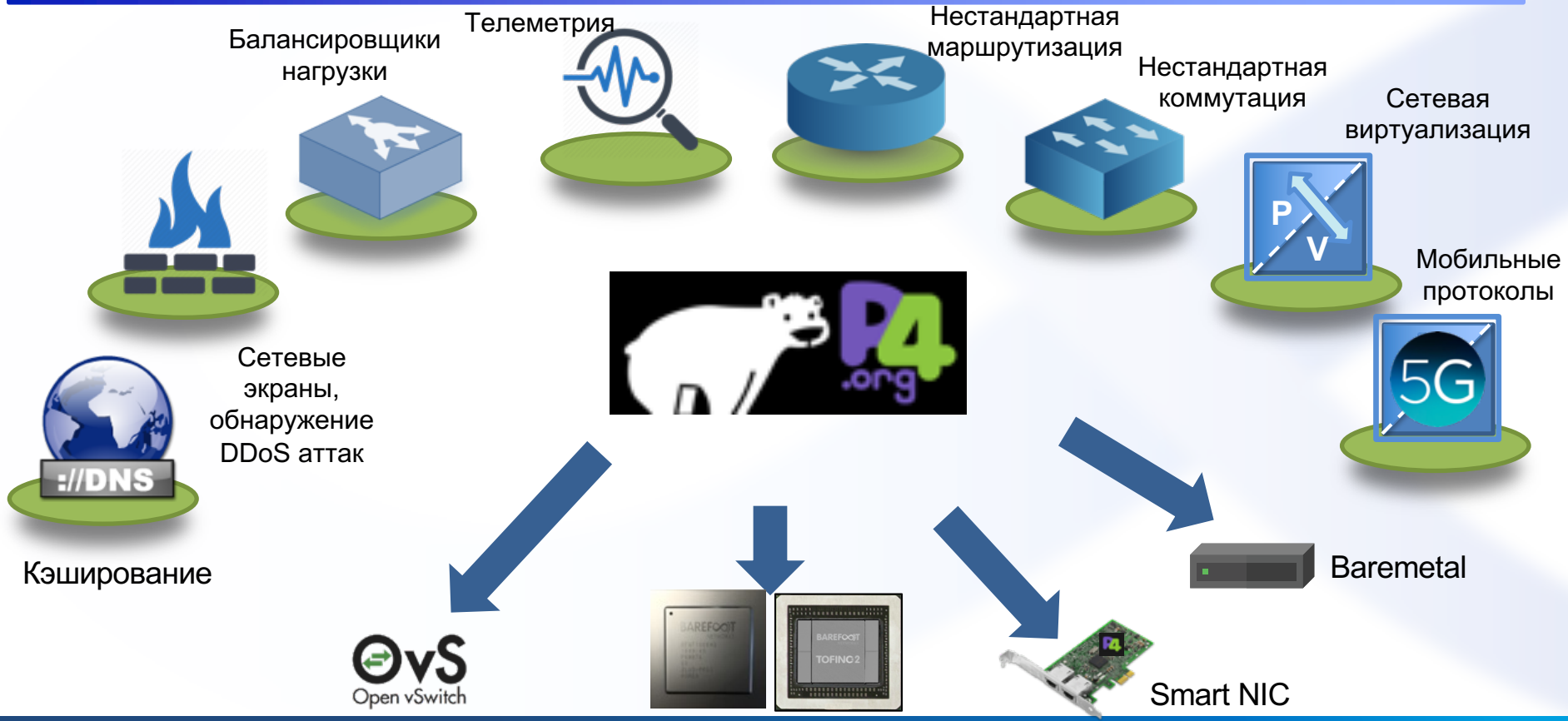
```
CRCPolynomial<bit<32>>(coeff=0xA833992b)      crc32_e;
Hash<bit<32>>(HashAlgorith_t.CUSTOM, crc32_e) my_hash_func;

apply {
    hdr.udp.src_port = my_hash_func.get({
                            hdr.ethernet.src_addr,
                            hdr.ethernet.dst_addr[23:0],
                            hdr.ipv4.src_addr[15:0],
                            hdr.ipv4.protocol,
                            hdr.ipv4.dst_addr})[23:8];
}
```

Определение хэш-функции на основе циклического кода

Вычисление хэш-функции для заданных полей пакета

Области применения P4

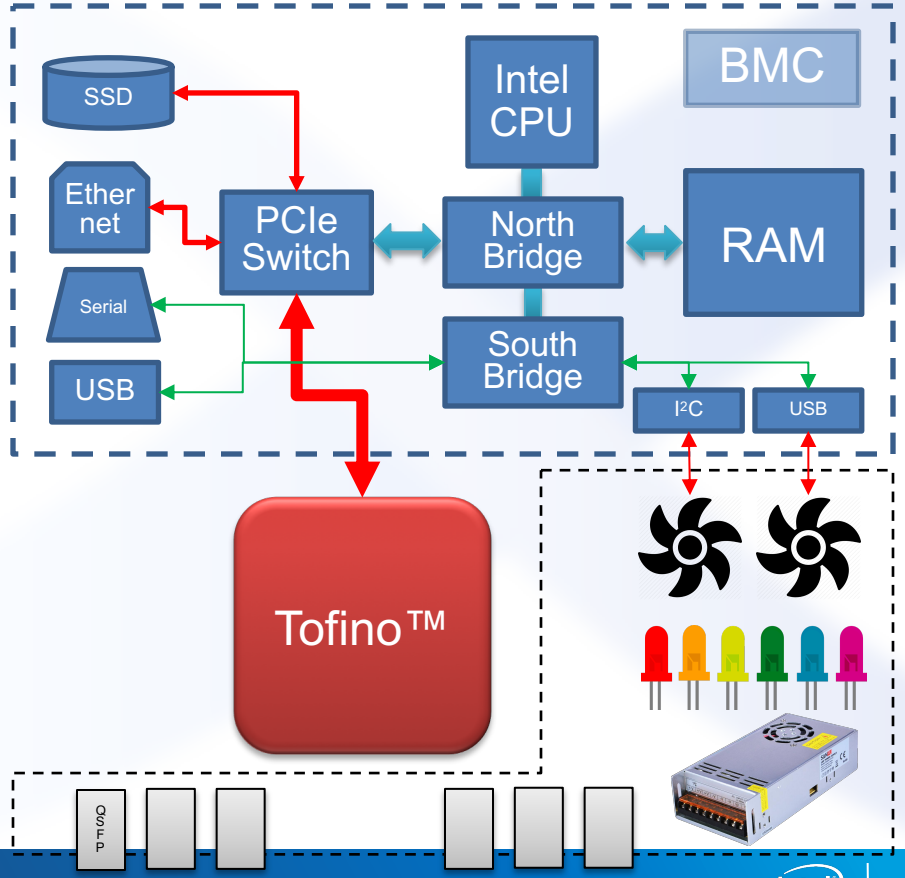


Но это – только начало!

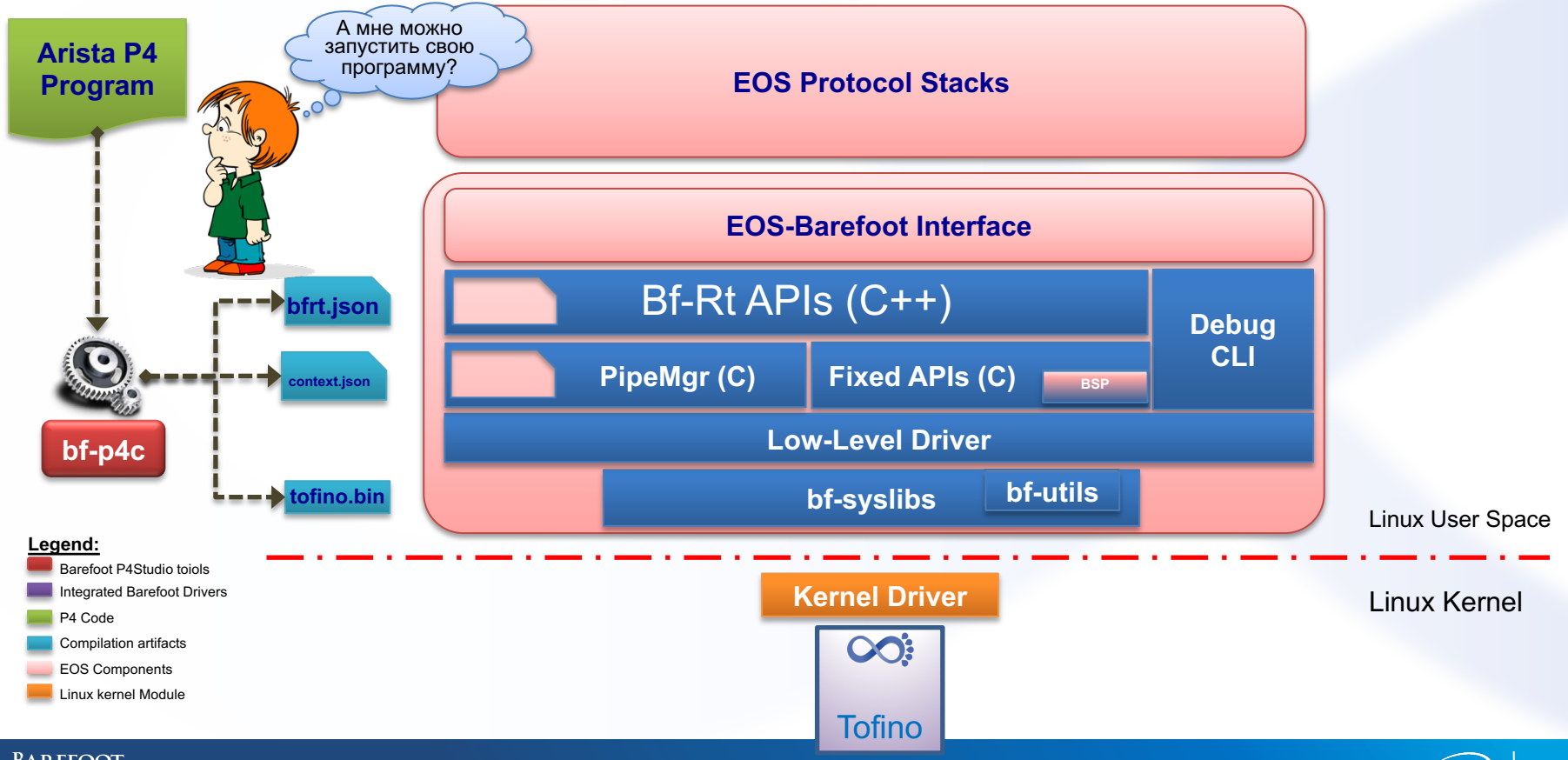
- Теперь вы можете строить такие устройства сами!

Arista 7170 – программируемое оборудование

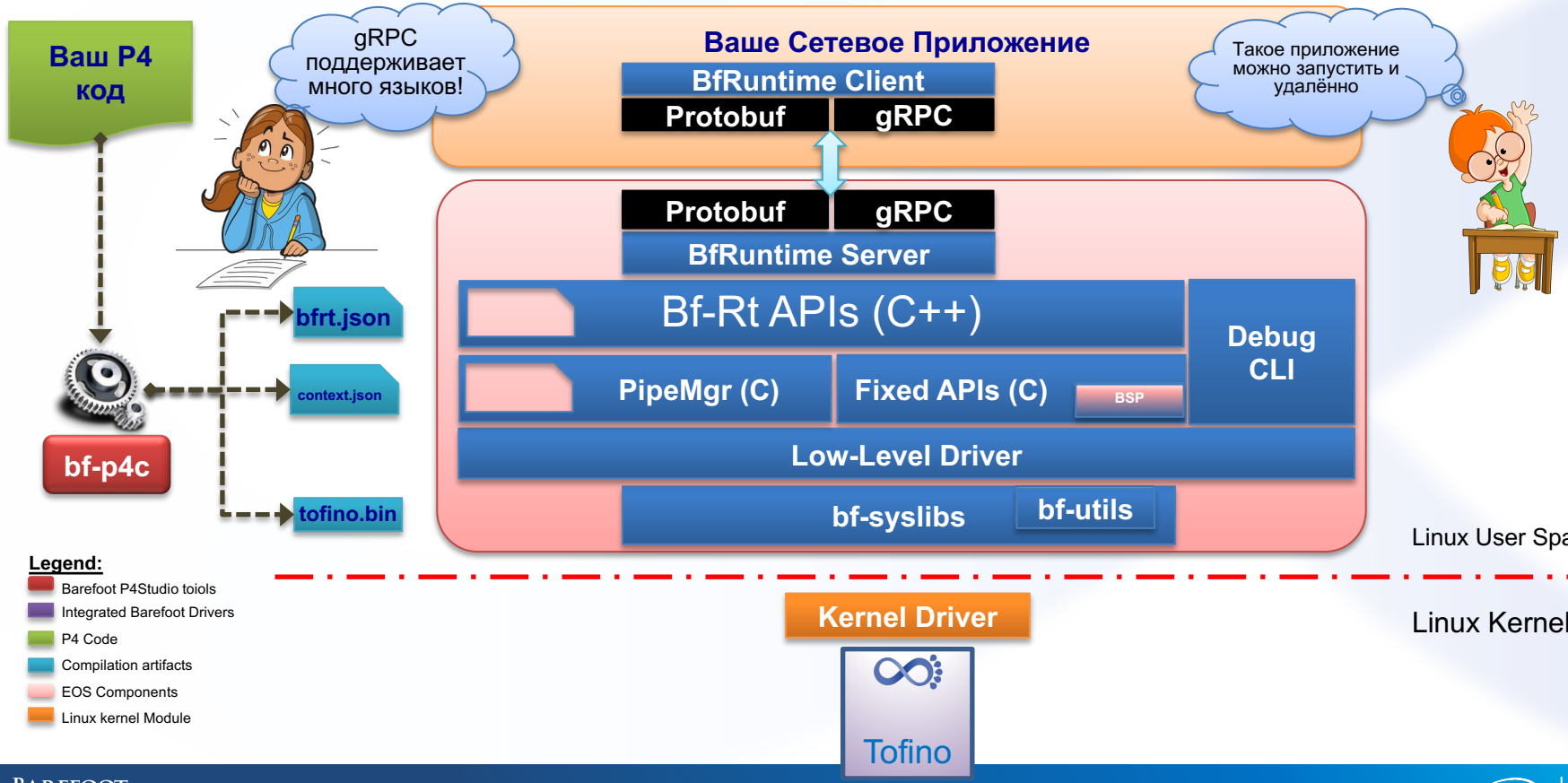
- **Стандартная архитектура на базе Intel x86_64**
  - Отлаженный Linux от Arista
- **Дополнительные устройства**
  - Отлаженные драйверы от Arista
  - Удобное управление через EOS
- **Микросхема программируемого коммутатора Tofino**
  - Драйверы от Barefoot интегрированы в EOS



Заглянем внутрь EOS…



Arista 7170 разрешает использовать свои программы!

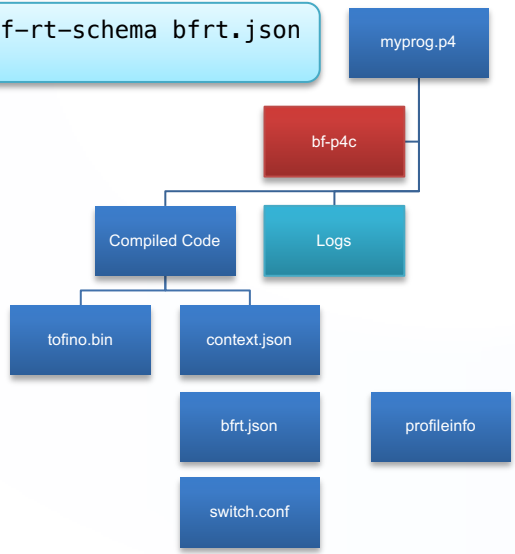


Что нужно для разработки

Barefoot P4Studio™ SDE

- **Компилятор P4**
- **Текстовый редактор**
- **Программа визуализации (P4i)**

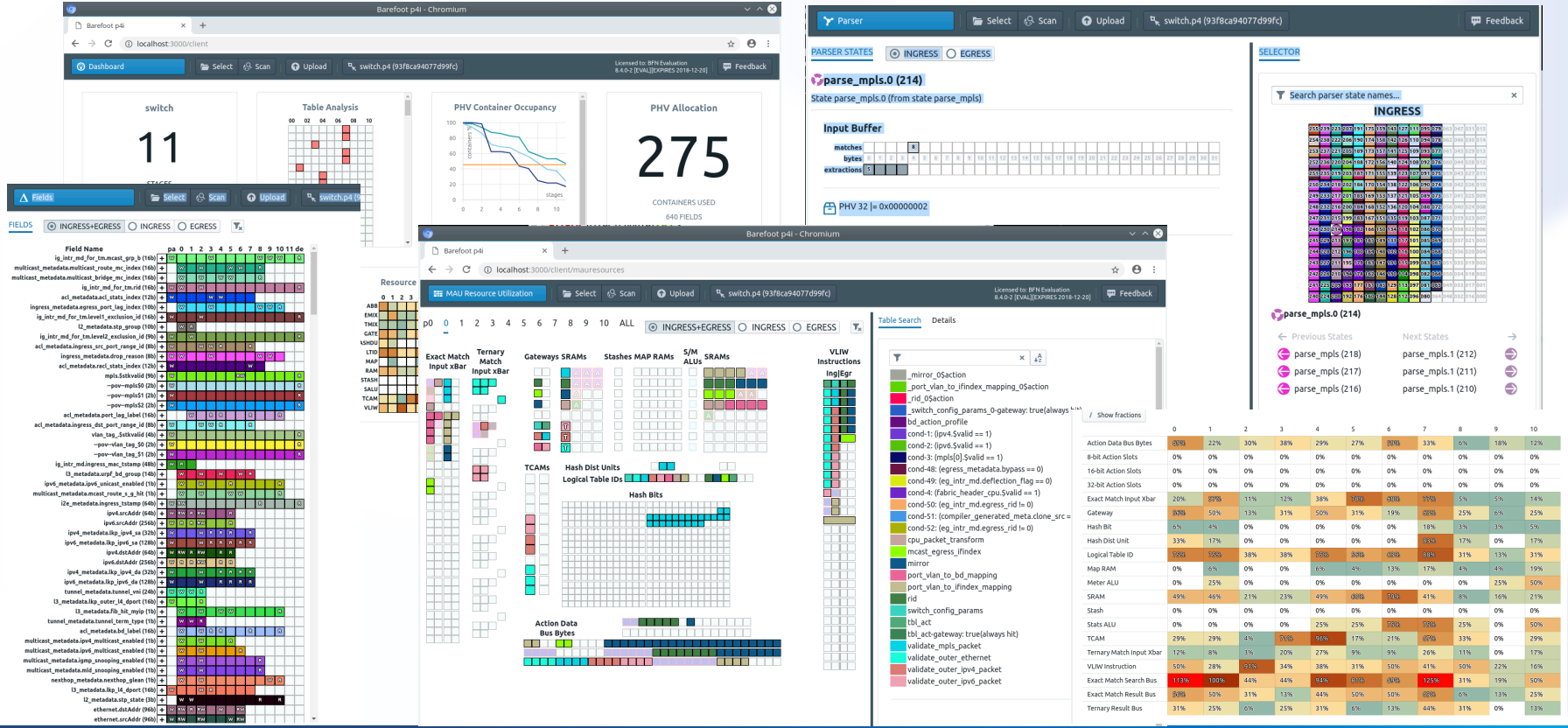
```
bf-p4c --bf-rt-schema bfrt.json
myprog.p4
```



Arista

- **Arista 7170**
  - Копируем файлы в `/mnt/flash/custom_profiles/myprog`
  - Проверяем `show platform barefoot profile`
  - Запускаем `platform barefoot profile myprog`

Визуализация программы на P4

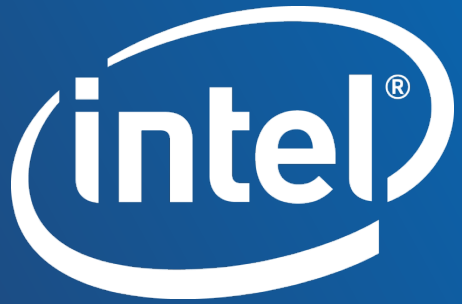


Преимущества использования Arista 7170

- **Высококачественное оборудование**
- **Надёжная базовая система (EOS)**
- **Простота в использовании**
  - Не нужно строить драйверы Barefoot
  - Порты (100G/50G/25G/10G) «просто работают»
  - Предоставляются дополнительные средства отладки (Snapshot)
- **Гибкость в применении:**
  - Локальный контроль
  - Удалённое управление (SDN)
- **Возможность использования стека EOS**
  - При написании программы на основе базового P4 кода от Arista
    - В недалёком будущем
- **Загружаем Barefoot P4Studio™ и начинаем творить!**

Дальнейшие шаги

- **Покупка Arista 7170**
  - emea-sales@arista.com
- **Получение доступа с Barefoot P4Studio™ -- бесплатно**
  - Коммерческие организации: sales@barefootnetworks.com
  - Академические организации: https://barefootnetworks.com/faster
- **Учиться, учиться и учиться (в Академии Barefoot)**
  - https://barefootnetworks.com/barefoot-academy


intel


INGENIUM RETICULUM PROGRAMMARE!
P4
EST. 2017
BAREFOOT ACADEMY