Table of Contents

Настройка on-stick и LAG/LACP 3
Настройка on-stick 3
Агрегация портов LAG/LACP 5
Настройка LAG 5
Применение балансировки к исходящему трафику LAG 6

Настройка on-stick и LAG/LACP

Настройка on-stick

[FastDPI 12+]

On-stick позволяют экономить на физическом оборудовании. FastDPI обычно работает с мостами, замыкая два физических порта (девайса). Для on-stick девайса физический порт один, на котором fastDPI сам создает виртуальные порты — со стороны абонентов (`subs`) и интернета (`inet`).

Каждый on-stick порт описывается особым образом: сначала описывается базовый физический порт с помощью `dpdk_device`, затем описываются виртуальные порты, основанные на базовом.
Описание базового девайса:

```
dpdk_device=port1:pci:04:00.0
```

Описание on-stick на основе девайса `port1`:

```
onstick_device {
    base=port1
    filter=<выражение фильтра со стороны  subs>
    subs=subs1
    inet=inet1
}
```

где:
`base` — базовый девайс
`filter` — логическое выражение для определения направления пакета (фильтр). Если это выражение возвращает `true`, значит, пакет со стороны `subs`, иначе — со стороны `inet`
`subs` — имя девайса со стороны `subs`
`inet` — имя девайса со стороны `inet`

Задаем мосты.

Базовый девайс `port1` НЕ МОЖЕТ входить ни в какие мосты

```
in_dev=subs1
out_dev=inet1
```

Везде, где требуется указание девайса, следует использовать виртуальные девайсы (в данном примере - `subs1` и `inet1`). Базовый on-stick девайс `port1` указывается только при описании on-stick девайса и нигде более.

В описании on-stick порта наиболее важная часть — это выражение `filter` для определения

направления пакета (subs → inet или inet → subs). Направление пакета — важный атрибут пакета в fastDPI, от которого зависит обработка. `filter` задает логическое выражение над L2-свойствами пакета. Если это выражение возвращает `true` — пакет со стороны `subs` (абонентов), иначе — со стороны `inet` (uplink, интернета).

Основой выражения `filter` являются термы, которые с помощью логических операторов & (И) и | (ИЛИ), скобок `(` и `)`, а также отрицания `!` объединяются в логическое выражение. Оператор & более приоритетен, чем |; по аналогии с арифметическими выражениями, можно считать, что & — это умножение, а | — это сложение, — именно исходя из этого нужно расставлять скобки.

Термы задают элементарные выражения над L2-свойствами пакета. Существуют следующие термы (регистр важен):

- `vlan(`список`)` — пакет single VLAN с указанными номерами VLAN, например: `vlan(56,78,890)`
- `vlan` — пакет с любым single VLAN
- `qinq` — Q-in-Q-пакет
- `pppoe` — PPPoE-пакет
- `smac(MAC-`адрес`)` — source MAC-адрес пакета, пример: `smac(01:02:03:04:05:06)`
- `dmac(MAC-`адрес`)` — destination MAC-адрес пакета, пример: `dmac(01:02:03:04:05:07)`

Примеры (напомним, что `filter` задает выражение для стороны subs):

- сеть Q-in-Q со стороны абонентов терминируется в single VLAN: `filter=qinq`
- гетерогенная сеть: со стороны абонентов Q-in-Q или PPPoE в VLAN: `filter=qinq | pppoe`. Здесь то, что PPPoE заключено в VLAN, неважно: PPPoE терминируется BRAS'ом, так что PPPoE со стороны inet невозможен.
- single VLAN-сеть, со стороны inet VLAN=609, все остальные VLAN - subs: `filter=vlan && !vlan(609)`. Здесь надо пояснить более подробно. Для стороны inet выражение фильтра выглядело бы так: `filter=vlan(609)`, но фильтр у нас задает выражение для стороны subs, так что казалось бы, достаточно отрицания: `filter=!vlan(609)`. Но это выражение будет истинно для любого пакета, кроме пакета с VLAN=609, даже без VLAN. Поэтому следует указать, что пакет **должен** содержать тег single VLAN, но за исключением VLAN=609: `filter=vlan && !vlan(609)`
- со стороны inet MAC-адрес бордера 3c:fd:fe:ed:b8:ad: `filter=!smac(3c:fd:fe:ed:b8:ad)` - все пакеты с source MAC, не равным MAC-адресу бордера, являются пакетами со стороны subs.

Формальное описание грамматики выражения `filter`:

```
filter ::= and | and ‘|’ filter
and    ::= mult | mult ‘&’ and
mult   ::= ‘!’ mult | term | ‘(‘ filter ‘)’
term   ::= vlan | qinq | pppoe | smac | dmac

vlan   ::= ‘vlan’ | ‘vlan’ ‘(‘список_int ‘)’
qinq   ::= ‘qinq’
pppoe  ::= ‘pppoe’
smac   ::= ‘smac’ ‘(‘ mac_address ‘)’
dmac   ::= ‘dmac’ ‘(‘ mac_address ‘)’
```

```
mac_address ::= xx:xx:xx:xx:xx:xx
```

Агрегация портов LAG/LACP

Агрегация портов средствами СКАТ поддерживается для режимов in-line и on-stick. В LAG могут входить либо обычные порты, либо on-stick, смешение недопустимо. LAG на on-stick организуется на базовом (физическом) порту. LAG реализуется в fastDPI на логическом уровне: никакого единого bond-девайса нет, внутри fastDPI работа ведется с портами, как и раньше.

Возможны 3 разных конфигурации:

- LAG на стороне `subs`, на стороне `inet` нет LAG
- LAG на стороне `inet`, на стороне `subs` нет LAG
- LAG на стороне `subs` И на стороне `inet`

Настройка LAG

Требования к девайсам, входящим в LAG:

- девайс может входить только в один lag-девайс;
- все девайсы в LAG должны иметь одинаковую скорость;

В настоящее время конфигурирование LAG производится в fastdpi.conf без возможности применения на лету, то есть требуется рестарт fastdpi при изменении конфигурации LAG.

```
   # [cold] Описание LAG
   # Для каждого LAG - отдельная секция lag.
   # В LAG могут входить только девайсы либо in_dev, либо из out_dev,
   # смешение недопустимо.
lag {
    # Необязательное имя LAG, используется для вывода в лог
    #name=

    # перечисляются все девайсы, входящие в LAG (из in_dev/out_dev)
    # LAG должен содержать как минимум 2 девайса
    device=
    device=

    # поддержка LACP:
    # - 0 - LACP отключено, СКАТ не держит LAG, а свободно пропускает.
    # - 1 - LAG в пассивном режиме: не шлем периодических LACPDU,
    #       но отвечаем на пришедшие LACPDU
    # - 2 - LAG в активном режиме: шлем периодические LACPDU
    # Default value: 0 (LACP отключено)
    #lacp=0

    # MAC-адрес - system_id данного LAG
    # Если не задано - используется arp_mac
    #system_id=
```

```
# system priority для данного LAG
# число в диапазоне 1 - 65535, default 32768
#priority=32768

# short (off) или long (on) LACP timeout
#short_timeout=off
```

Применение балансировки к исходящему трафику LAG

Тип применяемого алгоритма балансировки задается параметром `lag.balance_algo`. Допустимые значения:

- `0` — балансировка по внутреннему `session_id` (балансировка по умолчанию). В качестве хеша берется `session_id`
- `1` — без балансировки — пакет будет отправлен в парный порт моста
- `2` — хеш от `flow key <srcIP, dstIP, srcPort, dstPort, proto>`. Если flow нет — балансируем по `session_id`

Дополнительные параметры конфигурации хеша в секции `lag`: `hash_seed`, `hash_offset`, `hash_bits`
Сколько значащих бит берем из 64-битного хеша при балансировке. Алгоритм балансировки в общем случае выглядит так:

- вычисляем 64-битный хеш от тех или иных полей пакета и `hash_seed`;
- из 64-битного хеша берем `hash_bits` бит, начиная с `hash_offset` бита;
- по получившемуся числу N определяем номер порта в LAG: `port := N mod LAG_active_port_count`, т.е.

  ```
  port := ((hash(packet, hash_seed) >> hash_offset) & (2^hash_bits - 1))
  mod LAG_active_port_count
  ```

  Пример:

  ```
  //       +-----------------------------------------------+
  // hash: |                                XXXXXXXXXX------|
  //       +-----------------------------------------------+
  //                                        ^         ^
  //                                        |         hash_offset = 6
  //                                        hash_bits = 10
  ```

Настроечные параметры: `hash_seed=0` `hash_offset=0` `hash_bits=64`

В описании LAG должны быть указаны только базовые девайсы для On-Stick. Смешение On-Stick и обычных девайсов в одном LAG не допускается.

При агрегации происходит трассировка балансировки трафика.

Возможна агрегация портов LACP.