Table of Contents

On-Stick devices support

[FastDPI 12+]

On-stick devices are a good way to save on physical hardware. Fastdpi usually bridges two physical ports (devices). For an on-stick device there is one physical port on which Fastdpi itself creates virtual ports – on the subscriber (subs) and internet (inet) sides.

Each on-stick port is described in a special way: first the base physical port is described with dpdk_device, then the virtual ports based on the physical port are described:

```
# base physical port description
dpdk_device=port1:pci:04:00.0

# on-stick (based on port1 device) description:
onstick_device {
    # base device
    base=port1

    # a logical expression to determine the direction of the packet (filter)
    # If this expression returns true, it means that the package is from the
subs side,
    # otherwise - from the inet side.
    filter=<subs-side filter expression>

    # Name of the device from the subs side
    subs=subs1

    # Name of the device from the inet side
    inet=inet1
}

# Set the bridges
# The base device port1 CANNOT be part of any bridges
in_dev=subs1
out_dev=inet1
```

Wherever you need to specify a device, you should use virtual devices (in this example `subs1` and `inet1`). The base on-stick device `port1` is specified only when describing the on-stick device and nowhere else.

The most important part of the on-stick port description is the `filter` expression to determine the packet direction (subs → inet or inet → subs). Packet direction is a significant packet attribute in fastdpi, on which the processing depends. `filter` specifies a logical expression over the L2 properties of the packet. If it returns ``true`` – the packet is from the subs (subscribers), otherwise – it is from inet side (uplink, internet). `Filter` is based on terms which are combined into a logical expression with the & (AND) and | (OR) operators, brackets (and), and the negation !. The & operator has higher priority than |; similarly to arithmetic expressions, we can think of & as multiplication and | as addition – this is the basis for bracketing. The terms specify elementary

expressions over the L2 properties of a package. There are the following terms (case matters):

- `vlan(list)` – single VLAN packet with the specified VLAN numbers, for example: `vlan(56,78,890)`
- `vlan` – packet with any single VLAN
- `qinq` – Q-in-Q-packet
- `pppoe` – PPPoE-packet
- `smac(MAC-адрес)` – source MAC address of the packet, e.g.: `smac(01:02:03:04:05:06)`
- `dmac(MAC-адрес)` – destination MAC address of the packet, e.g.: `dmac(01:02:03:04:05:07)`

Examples (recall that `filter` defines an expression for the subs side):

- the Q-in-Q network on the subscriber side is terminated in a single VLAN: `filter=qinq`
- heterogeneous network: Q-in-Q or PPPoE on subscriber side in VLAN: `filter=qinq | pppoe`. Here the fact that PPPoE is enclosed in a VLAN does not matter: PPPoE is terminated by BRAS, so that PPPoE on the inet side is not possible.
- single VLAN network, on the inet side VLAN=609, all other VLANs are subs: `filter=vlan && !vlan(609)`. Here we need to explain in more detail. For the inet side the filter expression would look like this: `filter=vlan(609)`, but the filter here sets the expression for the subs side, so it seems that a negation is enough: `filter=!vlan(609)`. But this expression will be true for any packet except the packet with VLAN=609, even without VLAN. Therefore, you should specify that the packet **must** contain a single VLAN tag, excluding VLAN=609: `filter=vlan && !vlan(609)`
- on the inet side, the MAC address of the border is 3c:fd:fe:ed:b8:ad: `filter=!smac(3c:fd:fe:ed:b8:ad)` – all packets with a source MAC not equal to the border MAC address are packets from the subs side.

A formal description of the `filter` expression:

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

vlan   ::= 'vlan' | 'vlan' '(' list_int ')'
qinq   ::= 'qinq'
pppoe  ::= 'pppoe'
smac   ::= 'smac' '(' mac_address ')'
dmac   ::= 'dmac' '(' mac_address ')'
mac_address ::= xx:xx:xx:xx:xx:xx
```