Table of Contents

20 Router 3

General Description 3

The Internal Router Architecture 4

System Requirements 4

Setting veth-nterface Names 4

20 Router

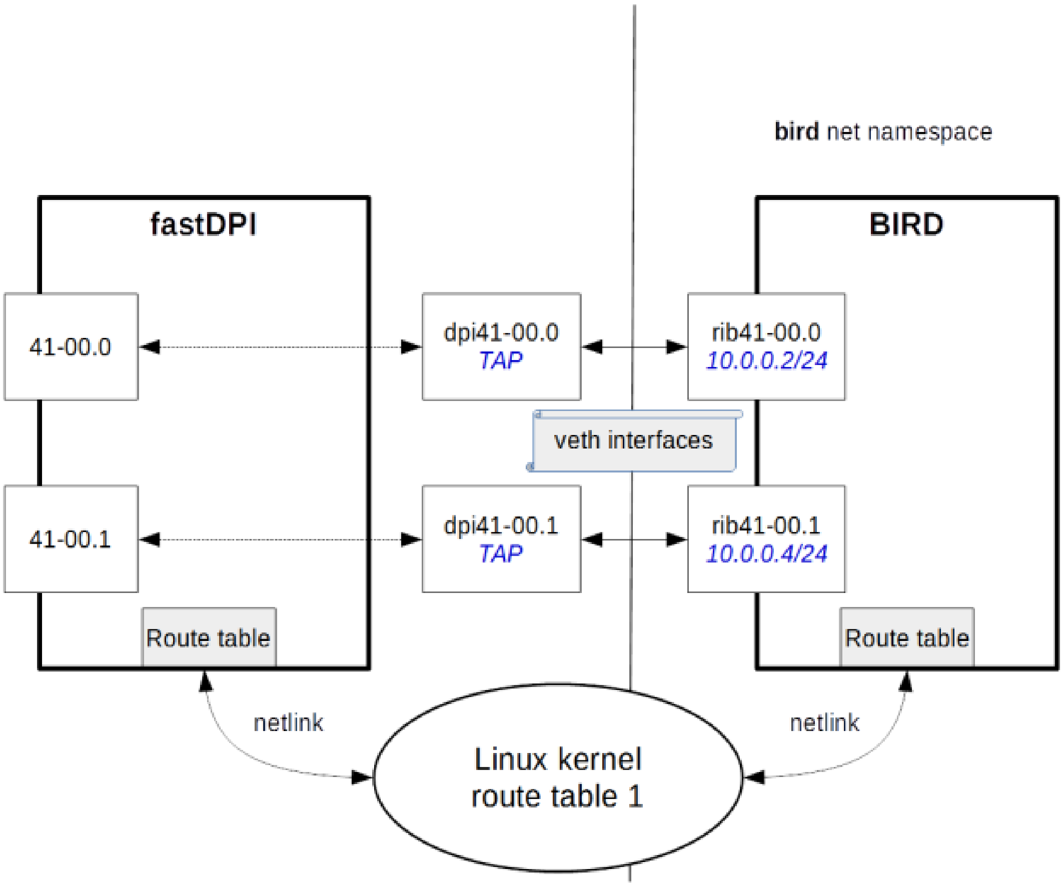
This is a provisional description, and may be substantially modified in the future based on test results

The router only works with the CentOS-8 version of Stingray Service Gateway, and only in L2 BRAS mode

<html><div class="menu"></html>

General Description

</note> SSG itself does not build the routing table. It delegates this work to proven specialised tools. The example uses the BIRD root daemon. The router daemon processes the required routing protocols (BGP, OSPF, etc.) and uses them to build a common routing table which it loads into the kernel. SSG performs routing of packets using this table.



Instead of BIRD, any other daemon that builds a routing table in the Linux kernel can be used, such as FRRouting, QUAGGA, Juniper CRPD, etc. since Stingray Service Gateway only uses the standard Linux interface to read the routing table and is therefore compatible with any daemon. In future versions, in order to save memory, it is possible to introduce optional specialized APIs for communication with a particular daemon to bypass kernel route table construction and communicate with the daemon directly.

Since BIRD builds the routing table in the OS kernel, to avoid application of these rules by the Linux server itself, the BIRD root daemon runs in a separate net namespace (in the diagram it is `bird` netns). Routing protocol packets are received by SSG in/out devices in general traffic. For each in/out device, a veth pair of "shadow" interfaces with predefined names is created: the DPI interface of the veth pair works as a TAP interface, the rib interface works as a normal system interface in BIRD's netns.

The Internal Router Architecture

Data from the kernel route table is read (rtnetlink) in the router's RIB. RIB is a prefix tree, it is convenient to modify it by kernel route table change events (delete/add entries). But it is impossible to use RIB in routing because it does not support multithreaded access from work threads (it requires locking, which is unacceptable).Therefore, in SSG, RIB is in router thread and unavailable for worker threads.

The worker threads use FIB. This structure is designed for multi-threaded search (LPM - longest prefix match), but is not designed for modifications (deletion/addition of new records). FIB can only be built from scratch by RIB and then used for LPM. Therefore, there are two FIBs in SSG - the current one (which is currently used for routing by worker threads) and the 'future one'. SSG checks every `router_fib_refresh` seconds to see if there have been any changes to the RIB since the current FIB was built. If there were changes, SSG builds (in router thread) new FIB in place of "future" one, and then switches current FIB to new one. By doing this, the worker threads will meet any changes that have occurred in the routing table.

System Requirements

Router mode in SSG requires quite a lot of memory, especially with BGP full view. Plus, memory is required for the BIRD daemon that builds the routing table via BGP, OSPF, etc. Router mode (especially BGP full view) requires at least 4-8G additional memory.

Setting veth-nterface Names

The fastdpi.conf describes all TAP-interfaces associated with the devices:

```
# Description of one router interface
# WARNING! '{' must be on the same line as the router_device section name!
router_device {
        # Device name from in_dev/out_dev
```

```
    device=
        # TAP interface name for the device (default='dpi' + device)
    #tap=
        # Name of the paired TAP interface in netns for the device
(default='rib' + device)
    #peer=
# WARNING! '}' must be on a separate line!
}
```

For example, for this configuration

```
in_dev=41-00.0
out_dev=41-00.1
```

where only `out_dev` is connected to the router, the description would be:

```
in_dev=41-00.0
out_dev=41-00.1

router_device {
        # Device name from in_dev/out_dev
    device=41-00.1
        # TAP interface name for the device (default='dpi' + device)
    tap=tap41
        # Name of the paired TAP interface in netns for the device
(default='rib' + device)
    peer=bgp41
}
```

It is possible not to specify the names of the `tap` and `peer` interfaces (default names are implied in this case), but the `router_device` should be described:

```
in_dev=41-00.0
out_dev=41-00.1
   # TAP for out_dev:
router_device {
    device=41-00.1
}

   # TAP for in_dev
router_device {
    device=41-00.0
}
```

In this case the TAP interface names are assumed to be as follows:

- for `in_dev=41-00.0`: `dpi41-00.0` on the SSG side, `rib41-00.0` on the BIRD side
- for `out_dev=41-00.1`: `dpi41-00.1` on the SSG side, `rib41-00.1` on the BIRD side