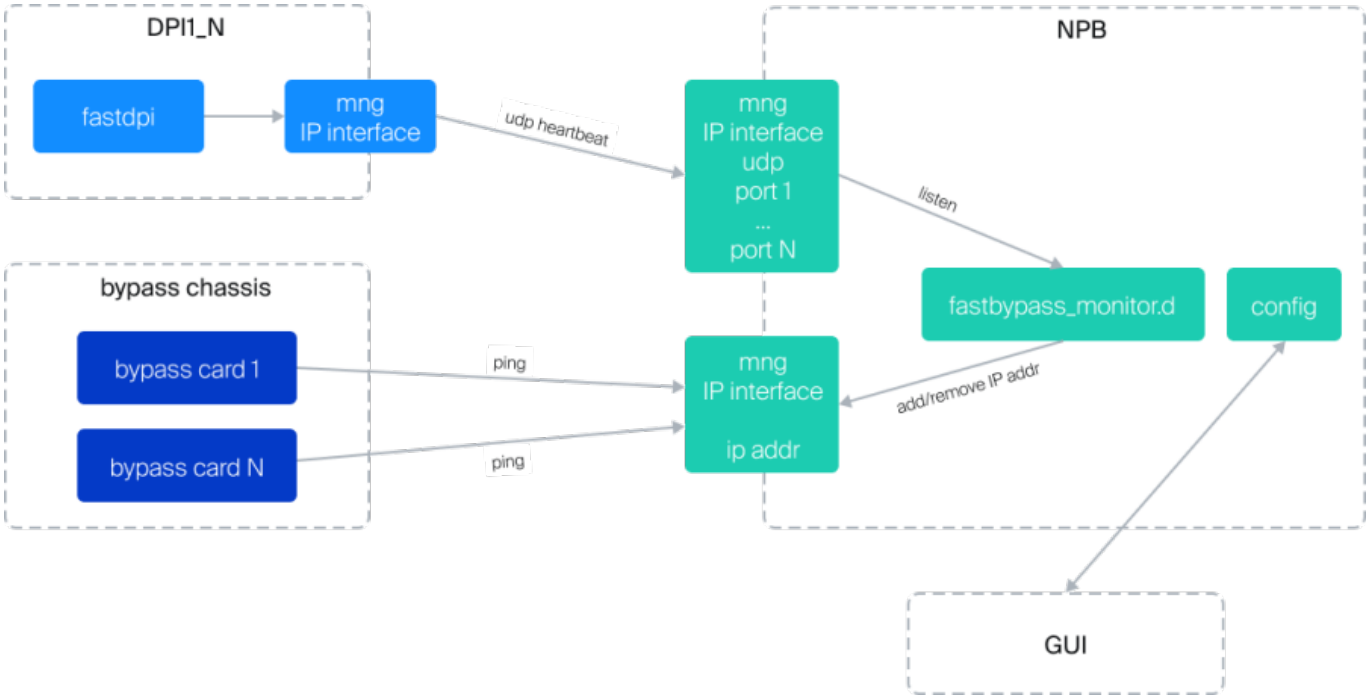
Содержание

FastBypass monitor

`fastbypass_monitor` (referred to as "daemon" further in the documentation and script) is a tool for monitoring and managing the state of network interfaces connected to Bypass network cards.

The daemon reacts to HEARTBEAT signals received from DPI on specific ports defined in the configuration file. If HEARTBEAT signals are not received according to the configuration rules, the daemon performs specific actions such as deleting or creating IP addresses connected to the Bypass cards and enabling or disabling certain network interfaces.



Hardware Requirements

OS: OpenSwitch 2+ / Debian 9+
Python: 2.7.9

Key Features

- Monitoring HEARTBEAT signals from DPI on specified ports.
- Dynamic management of IP addresses and network interfaces.

Installation

1. Copy the installation package `fastbypass_monitor-X.X.XX.deb` to the host machine.
2. Run the following command from the directory where the package is located:

```
sudo dpkg -i fastbypass_monitor-X.X.XX.deb
```

After installation, the daemon becomes manageable through the system manager (`systemctl`).

The configuration file is available at `/var/fastbypass_monitor/backend/.env`
A sample configuration file can be found at `/var/fastbypass_monitor/backend/sample.env`
Daemon logs are stored at `/var/fastbypass_monitor/backend/logs/`

Usage

After installation, the daemon runs automatically. Upon reboot, it starts after the `network` service has successfully launched.
Manage the daemon using system manager commands.

 Aliases (short command equivalents) can only be used with sudo. Use `sudo su -` and enter the password to enable this mode.

Start the daemon:

```
sudo systemctl start fastbypass_monitor
```

Alias:

```
fbypass_ctl start
```

The service starts in an unknown state, meaning it does not initially enable or disable bypass mode. After all receivers are initialized and their statuses are determined, the system switches to either normal or bypass mode depending on configuration and receiver status.

Restart the daemon:

```
sudo systemctl restart fastbypass_monitor
```

Alias:

```
fbypass_ctl restart
```

Reload the daemon without stopping:

```
sudo systemctl reload fastbypass_monitor
```

Alias:

```
fbypass_ctl reload
```

Stop the daemon:

```
sudo systemctl stop fastbypass_monitor
```

Alias:

```
fbypass_ctl stop
```

Check the daemon’s status:

```
sudo systemctl status fastbypass_monitor
```

Alias:

```
fbypass_ctl status
```

View the last few lines of the log file in real-time:

```
tail -f /var/fastbypass_monitor/backend/logs/fastbypass_monitor.log
```

Alias:

```
fbypass_ctl tailf
```

Output the last 100 lines of the log:

```
tail -n 100 /var/fastbypass_monitor/backend/logs/fastbypass_monitor.log
```

Alias:

```
fbypass_ctl tail 100
```

Stop the daemon and remove IPs from Bypass cards, forcing the system into bypass mode:

```
fbypass_ctl force_on
```

Stop the daemon and add IPs to Bypass cards, forcing the system into normal mode:

```
fbypass_ctl force_off
```

Add the daemon to startup:

```
fbypass_ctl enable
```

Remove the daemon from startup:

```
fbypass_ctl disable
```

To configure and launch the daemon with new settings, edit the configuration file and restart or stop and start the daemon.

The daemon configuration is located at `/var/fastbypass_monitor/backend/.env`

Upon startup and reload, the daemon reads the configuration file. If it can successfully configure the specified IPs and interfaces, it continues operation. Otherwise, it only launches successfully configured components.

In case of a critical error, the daemon will restart automatically.

Using `sudo systemctl reload fastbypass_monitor` will reload the configuration without stopping the daemon, shutting down removed components, and adding new ones.

During startup and reload, the daemon does not manage interfaces and IPs until all listeners report their statuses. After a restart, the daemon remains in its previous state until receiving updates from all listeners.

Local and Global States: Bypass Mode

The daemon manages interfaces based on either a **global** state (depending on all listeners) or a **local** state (specific to individual listeners).
For instance, if you list interfaces in the global settings, they will be enabled or disabled based on the daemon's overall state. If the daemon fails to receive enough signals, the interfaces are disabled.

Example:

```
LISTEN_CUBRO_IFS=<interface list>
LISTEN_SHUTDOWN_CUBRO_IFS_WHEN_BYPASS=1
```

Each listener can also have its own interface list that it manages based on its state.

Example:

```
LISTEN_CUBRO_IFS[0]=<interface list>
LISTEN_SHUTDOWN_CUBRO_IFS_WHEN_BYPASS[0]=1
```

If an interface appears in multiple listeners' lists, it switches to bypass mode if any listener stops receiving signals. The interface returns to normal mode only if all listeners are active.

If an interface appears in both local and global settings, it remains in bypass mode until the corresponding listener starts receiving signals and the daemon switches to normal mode.

Configuration

Minimal Configuration

A minimal configuration requires specifying at least one interface, IP address, and port for receiving HEARTBEAT signals, along with one interface and IP for Bypass cards.

Example:

```
LOG_LEVEL=INFO

LISTEN_HEARTBEAT_IFS=eth0
BYPASS_CARD_IFS=eth0

LISTEN_HEARTBEAT_FAILED=1
LISTEN_HEARTBEAT_ATTEMPTS=1
LISTEN_HEARTBEAT_TIMEOUT=3000

LISTEN_HB_HOST[0]=192.168.1.202
LISTEN_HB_PORT[0]=3000

LISTEN_HB_HOST[1]=192.168.1.202
LISTEN_HB_PORT[1]=3100

BYPASS_CARD_HOST[0]=192.168.1.211
BYPASS_CARD_HOST[1]=192.168.1.212
```

This example configures the daemon to receive HEARTBEAT signals on interface `eth0` at IP `192.168.1.202` and ports `3000` and `3100`. Bypass cards are connected via `eth0` at IPs `192.168.1.211` and `192.168.1.212`. Default listener values: `LISTEN_HEARTBEAT_ATTEMPTS`: 1 `LISTEN_HEARTBEAT_TIMEOUT`: 3000 ms

If a listener fails to receive a signal after one attempt within 3000 ms, it is marked as failed. If the number of failed listeners meets or exceeds the threshold (`LISTEN_HEARTBEAT_FAILED`), the daemon switches to bypass mode and removes IPs from Bypass cards. When signals are restored, the listener resumes normal operation. If the number of failed listeners falls below the threshold, the daemon switches back to NORMAL mode and restores the IPs for the Bypass cards.