Table of Contents

BNG/BRAS reservation

The following replication scheme is used in the Stingray Service Gateway ver. 8.3+ to align subscriber data on all fastDPI servers: fastPCRF sends authorization responses and CoA requests to all the servers listed in the fdpi_server configuration parameters.

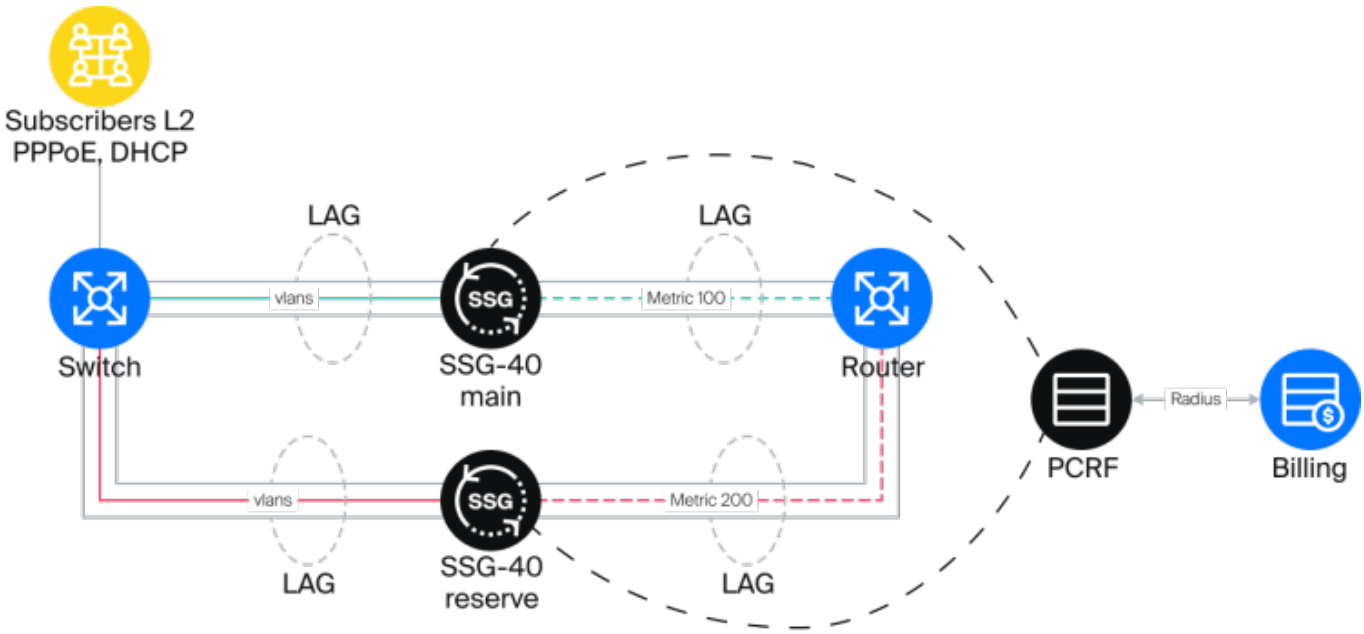
Authorization parameters are sent by using a persistent queue so even if some of the fastDPI servers are inactive at the time of data transmission, they will receive all the data missed during their idle time once they are activated.

Ways the data is applied on the DPI

When receiving authorization data, the fastDPI server identifies whether it was response to its own request or it was a response to another request (there is a special label in the packet for this purpose). If this is a response to its own request, the data will be applied completely: a DHCP or PPPoE session will be created in case of DHCP or PPPoE authorization request and the data will be stored in UDR. If this is an answer to another request, the fastDPI will simply store corresponding "extraneous" data in the UDR. Thus, when the main fastDPI server becomes unavailable all the load would be imposed on the backup fastDPI server and the latter will already have contain all the subscriber properties in its UDR: subscriver services, its policing, L2 properties - MAC address, VLAN, etc. That is, the UDR of the main and backup servers will essentially contain the consistent data.

BRAS L2 backup

Backing up of BRAS in L2 mode involves the connecting up of two Stingray Service Gateways in one L2 broadcast domain. One of them in Master mode and the other in Slave one. Master SSG carries out traffic processing along with users authorization through the PCRF server. Slave does not pass traffic through itself, dpdk interfaces are in the traffic standby mode (down). Subscribers information is synchronized through the PCRF server. Slave monitors the availability and performance of the Master and when the last fails, Slave will activate (up) dpdk interfaces and start to process traffic automatically or manually. An example of DPI connection and routes the traffic passes through it are presented in the diagram below.



Database synchronization

FastPCRF is responsible for synchronization, its configuration is described in the section Replication of authorization data.

Configuring the SSG Master mode

Configuring the SSG Slave mode

Algorithm description

Stingray Service Gateway backup concept - MASTER-SLAVE (L2-BRAS):

- 1. MASTER is running 99% of the time, it can be disabled or may fail
- 2. When being recovered MASTER always treacherously proceed to process the traffic
- 3. SLAVE just accepts replications from MASTER and saves them in UDR in 99% of the time
- 4. There is a third party that switches traffic to MASTER or to SLAVE, depending on the current situation:
- 4.1. MASTER is available, SLAVE is available, then the traffic will be switched to MASTER
- 4.2. MASTER is available, SLAVE isn't, then the traffic will be switched to MASTER
- 4.3. MASTER is not available, SLAVE is available, then the traffic will be switched to SLAVE
- 4.4. MASTER and SLAVE are not available, then the traffic will be switched to MASTER

MASTER→ SLAVE toggling:

- 1. The third party detects that MASTER becomes unavailable and switches all the traffic to SLAVE
- 2. Delays when switching are barely perceptible (physically and logically) due to 99% SLAVE's UDR contains replicated data

Bootstrap MASTER'a (SLAVE is active and process traffic):

- 1. MASTER has the fastdpi+fastpcrf services running and enabled (they were started on boot)
- 2. MASTER detects that SLAVE is active and stores relevant data
- 3. MASTER stops its fastdpi + fastpcrf services
- 4. MASTER backups UDR on SLAVE and takes it back
- 5. MASTER starts its fastdpi + fastpcrf
- 6. A third party detects that the MASTER becomes available and switches the traffic to it

Bootstrap MASTER'a (SLAVE is anavailable):

- 1. MASTER has the fastdpi+fastpcrf services running and enabled (they were started on boot)
- 2. MASTER determines that SLAVE is not available, considers that UDR it holds is more relevant than the one located on SLAVE, continues to work normally
- 3. A third party detects that the MASTER becomes available and switches the traffic to it

Bootstrap SLAVE'a (MASTER is active and process traffic):

- 1. SLAVE has the fastdpi+fastpcrf services running and enabled (they were started on boot)
- 2. SLAVE detects that MASTER is active and stores relevant data
- 3. SLAVE stops its fastdpi + fastpcrf services
- 4. SLAVE backups UDR on MASTER and takes it back
- 5. SLAVE starts its fastdpi + fastpcrf
- 6. SLAVE starts to replicate data

Bootstrap SLAVE'a (MASTER is unavailable):

- 1. SLAVE has the fastdpi+fastpcrf services running and enabled (they were started on boot)
- 2. SLAVE detects that MASTER is unavailable, considers that UDR it holds is more relevant than the one located on the currently unavailable MASTER, continues to work normally
- 3. A third party detects that SLAVE becomes available and switches the traffic to it

Script for active DPI reservation

The script should be installed on the reserve DPI where it runs in a continious loop monitoring the state of main dpi via SSH.
This script uss **4 checks** to confirm that main dpi is working:

1. The server is rachable via network (pingcheck)
2. The fastDPI process is present
3. FastDPI process PID did not change
4. The link state on the main DPI did not change (optional check). This check is disabled by default as it may not be necessary in some installations.

Installation process:

1. Download all files from the archive to your target reserve server.
2. Configure main server ip inside the `SRS.sh` script
3. Create an SSH key pair on the reserve server via

```
ssh-keygen -t ed25519
```

4. Create a new sudo user account on the main server
5. Copy private SSH keys from reserve server to the new account authorized keys file on the main server
6. Add permissions to installation scrip via

```
chomd +x install.sh
```

7. Run `install.sh`

Controlling the service:

1. Starting the service:

```
systemctl start fastsrs
```

2. Checking service status:

```
systemctl status fastsrs
```

3. Stoping the service:

```
systemctl stop fastsrs
```

4. Checking service logs:

```
journalctl -u fastsrs
```