BHBox

The BHBox is a software developed in C++ that aims to emulate the BH (Beam-Hopping) on a satellite telecommunication system. To do so, the emulator adds a temporal intermittency on an interface. It uses *netfilter* to intercept incoming packets in order to send them only when the recipient's beam is illuminated by the satellite.

The BHBox therefore represents a solution to study the impact of the implementation of Beam-Hopping on the end-to-end quality of service and on telecommunication services.

Manuals:

Please follow the instructions in the installation manual section and the user manual section to familiarize yourself with the software. The OpenSAND manual section provides a solution to use the BHBox in an OpenSAND platform.

Design of BHBox:

The BHBox is a multilayer filter. It operates at the network level and then at the application level.

At the network level, the BHBox uses *netfilter* to handle incoming packets which have to be concerned by the beam-hopping. Then *netfilter* allows these packets to be redirected to a local socket instead of being routed to the end user.

At the application level, the binary *run_bh* retrieves packets arriving on the socket. Then, it adds them to a FIFO pipe. It will then unpack the packets and re-send them according to the lighting profile of the recipient defined in the parameter.

For more information, see the *Architecture of BHBox* section.

Specific vocabularies:

In order to understand all the documentation of the BHBox, the following vocabulary is important:

- *BHS (Beam-Hopping Slot)*: the smallest unit of time during which a beam can be considered as switched on.
- *TimeLine:* the lighting profile of the recipient. It is composed of 1 and 0. State 1 corresponds to a beam illuminated by the satellite. During state 0, the receiver is not illuminated. The TimeLine will be repeated indefinitely until the end of the program execution.

User Manual

Iptables configuration:

Filtering rules based on *iptables* are necessary to redirect incoming packets to the BHBox.

```
iptables –I FORWARD –j NFQUEUE
```

Thanks to this iptables rule and netfilter, the traffic will be redirected to the BHBox interface.

It may be interesting to add some parameters to the filtering rule :

- *-p* : Filter only such types of protocols (e.g.: UDP)
- -o : Filter only packets intended for an interface
- *-i* : Filter only packets arriving from an interface
- *-d* : Filter only packets destined to an IP address
- -s : Filter only packets sent by an IP address

To delete the filter rule, simply execute the following command:

```
iptables -F
```

Execution of the beam-hopping

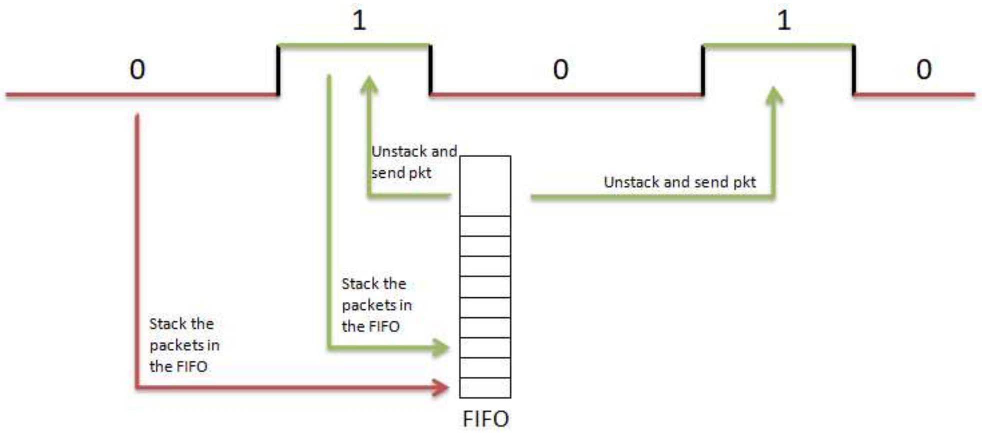
In order to launch the BHBox, and only after setting up the above filtering, the following command can be executed

```
./run_bh -bs $bhs -f $freq (-d $duration --debug)
```

The launching of the BHBox therefore requires at least 2 parameters:

- *bhs* : duration in us (microsecond) of a timeslot.
- *freq*: Frequency of beam illumination. The beam will be switched on once every freq timeslot (Example: for freq=6, we will have a timeline [100000]).
- If you want to operate the BHBox for a defined time, you can enter a *duration*. This duration must be entered in second. Otherwise, the executable will run until the user stop the program manually (ctrl+c, ctrl+z, ctrl+\)
- You can also activate the *debug mode* to follow the evolution of the timeline and fifo filling over time

The program exploits these parameters to establish the timeline. Then all the incoming packets are stored on the FIFO. They are only retransmitted to the recipient when the beam is considered switched ON. (See *Architecture of BHBox* for more information about the parameters and the design of the program).



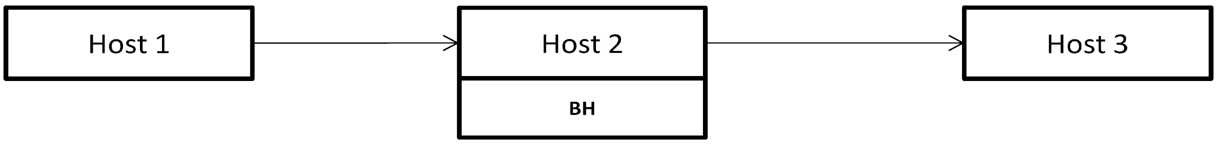
At the end of the execution, if the debug mode is activated, the program gives access to two output files:

- *profil_bh.txt* : Evolution of the beam state over time
- *profil_fifo.txt* : Evolution of the FIFO filling over time. The unit is the bytes.

Network profile after passing through the BHBox:

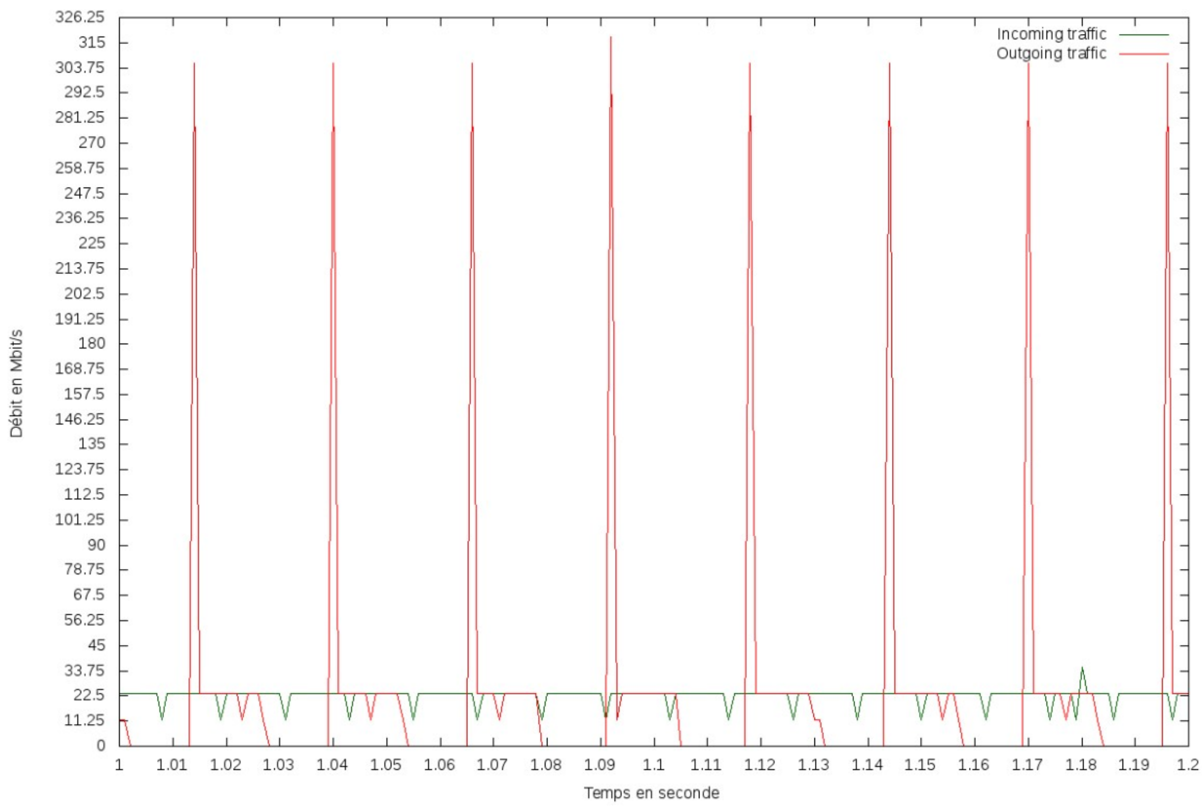
This section aims to show how the BHBox modifies the network profile and to check that it corresponds to expectations, i.e. a network profile that follows the timeline given in the BHBox parameter (No network traffic leaving the box when the beam state is 0, then restarting traffic when the beam state is 1).

To validate the implementation of the BHBox, we exploit the following architecture.



Impact on the goodput

For this test, we use as parameter *BHS*=13 ms and *freq*=2 (timeline [1, 0]). By sending a regular traffic of 22.5 Mbit/s with *iperf*, the profile of the incoming and outgoing traffic on the BHBox is as follows:



The rate profile follows the timeline profile. Flow peaks are observed each time the beam returns to the ON state. This corresponds to the moment when the program sends all the packets that have been stored in the FIFO while the beam was in the OFF state.

Impact on the delay and jitter

It can also be interesting to see how the BHBox modifies the latency and jitter of a communication.

This test was performed without the BHBox, with a BHBox and a timeline [1,1] and then with a timeline [1,0]. The BHS used is always 13ms. The test with the BHBox [1,1] is used to see if moving the packets up to application level before re-transmitting them affects the performances

	Without BHBox	With BHBox timeline [1,1]	With BHBox timeline [1,0]
Average jitter (ms)	0.0138	0.0121	0.232
Average latency (ms)	0.2895	0.3145	3.5655

Using the BHBox without really setting up beam-hopping (timeline [1,1]) only slightly modifies the latency and jitter of the communication

With the timeline [1, 0] and the BHS=13ms, we see an increase in latency of **3.251ms**. This value corresponds to the expected value. Indeed, with a timeline [1,0], half of the packets will be interrupted by the BHBox. With a BHS of 13ms, packets will wait on average 6.5 ms in the bow. The theoretical value (6.5*50%=3.25) therefore corresponds to the experimental value found. There is also an increase in the average jitter with the introduction of the beam-hopping in the communication.

Limitations of the BHBox:

The BHBox passes the packets through the application level for processing them. Thus, the processing time involved can causes some limitations directly linked to the throughout, the size of the packets and the BHS and timeline used. If the time during which the beam is considered as switched ON is no sufficient to send all the packets stored in the FIFO during the transmission interruption, packet losses will be observed.

On the tested computer, for a parameter *freq*=11, i.e. a timeline [10000000000], the BHBox can support at least 4250 packets/s (e.g. a rate of 50 Mbit/s for packets of 1470 bytes). With this reference value, for other freq values, limit values can be deducted.

These performances can be improved with the execution parameter –s (See Architecture of BHBox for more information).

If better performances are desired, the improvement of the program is necessary. The fact of using more powerful computers can also be tested.

Before any utilisation of the BHBox, it is highly recommended to test if the program has sufficient performance for the test needed in order to not make false conclusions about beam-hopping

In addition, it is not recommended to use a BHS below one millisecond. Indeed, the program will have difficulties managing the evolution of the timeline and the results could be truncated.

It is recommended to reproduce the tests presented in this section to assess the relevance of the exploited platform.

Installation Manual

Requirements:

The BHBox can be installed on any physical or virtual computer in order to set up a time interruption representative of the beam hopping. The implementation has been tested with Ubuntu version 16.04.

The *netfilter* Library:

It is compulsory to install the *netfilter* library on the computer where the BHBox will be deployed. To do this, simply execute the following command:

```
sudo apt-get install libnetfilter-queue-dev
```

Size of system pipe FIFO:

The maximum size of the FIFO system must be set to a high value. This size is located in the **/proc/sys/fs** folder and can be set using this command:

```
sudo sysctl fs.pipe-max-size=66781584
```

Installation of the BHBox:

The source code of the software can be downloaded from the address below:

```
git clone TBD
```

To compile the source code, you must do so with at least C++11 version and including the following libraries for the compilation:

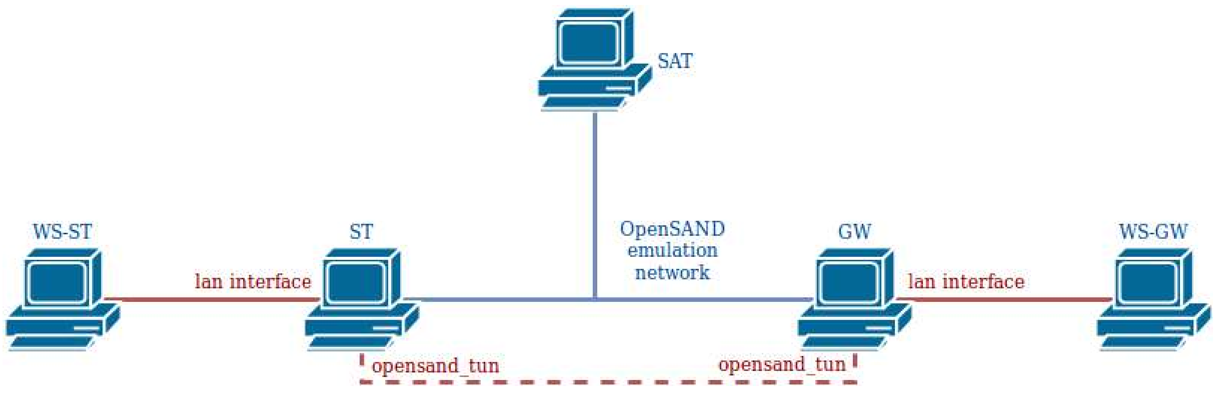
- *lpthread*
- *lnfnetlink*
- *lnetfilter_queue*

A makefile is available. Run the following command to obtain the binary *run_bh*:

```
make –f beam_hopping.mak
```

OpenSAND Manual

In order to study the impact of the beam-hopping implementation on a satellite communication, it will be necessary to set up the BHBox on an OpenSAND platform previously deployed. The image below describes the architecture used.



The BHBox will be deployed on the gateway. So, we need to filter all the packets arriving from WS_GW and destined for WS_ST. To do this, it is therefore necessary to set up the following filtering rule on GW:

```
iptables –I FORWARD –o opensand –j NFQUEUE
```

Then just run the run_bh binary on GW with the required parameters. Beam-Hopping is then implemented on the satellite communication.

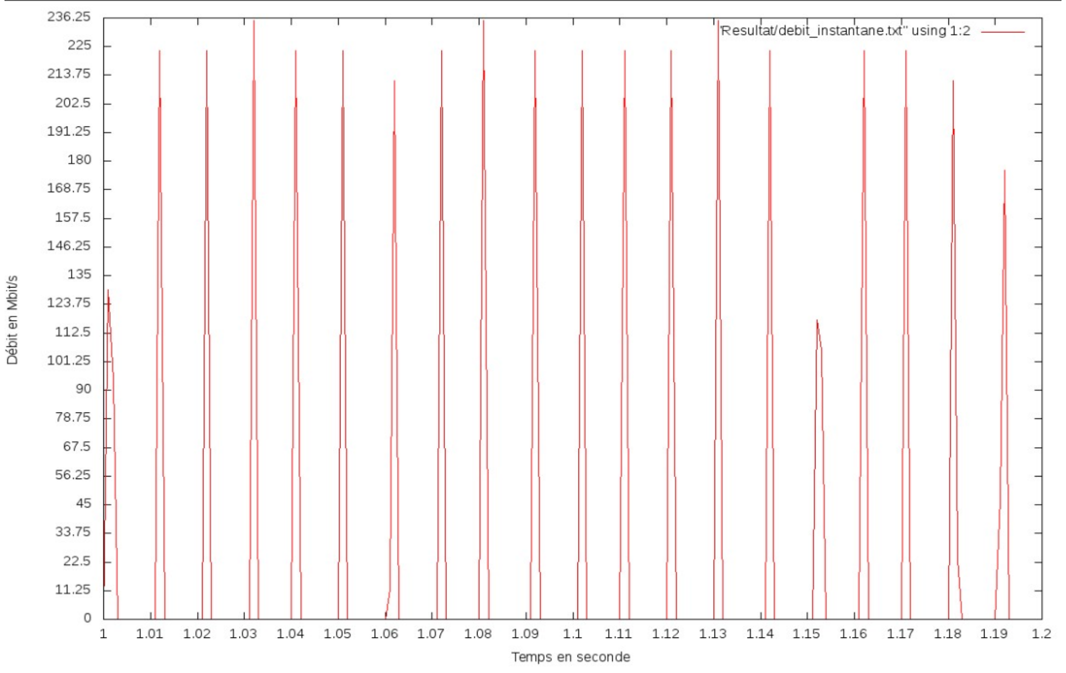
```
./run_bh -bs $bhs -f $freq (--debug –d $duration)
```

For more information about the parameters of the BHBox, the Architecture of BHBox section can be consulted.

Influence of the BHBox on network performance:

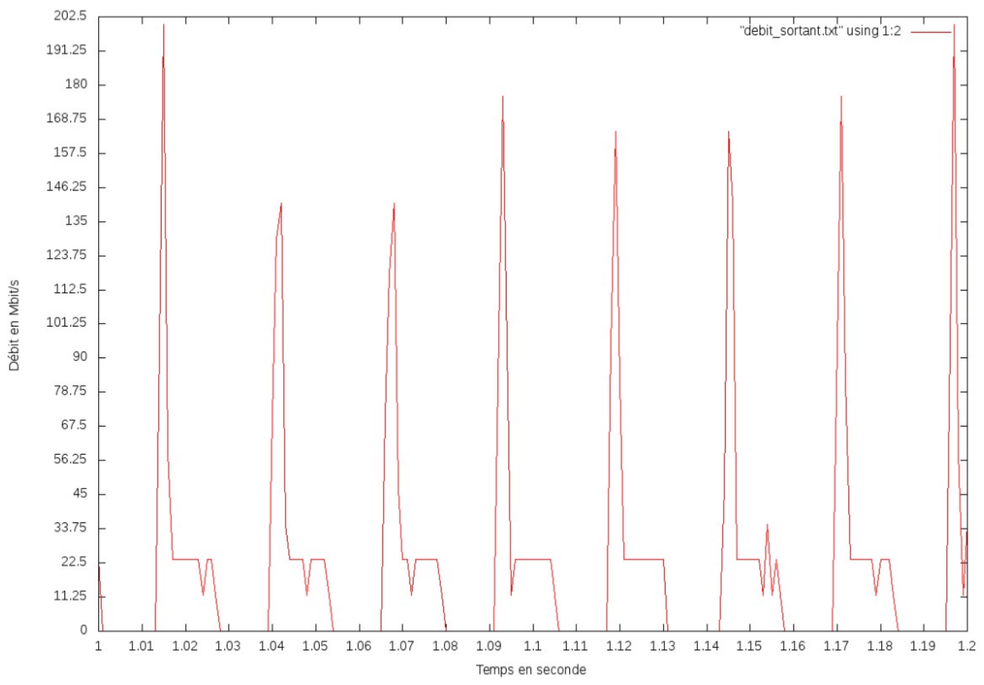
The purpose of this section is to see how the BHBox changes the flow profile, latency and jitter on an OpenSAND platform.

In the case of non-beam-hopped communication (No BHBox or BHBox with timeline[1.1]), if a rate of 22.5 Mbit/s is sent by iperf from WS-GW to WS-ST, the profile rate received by the end user is as follows:

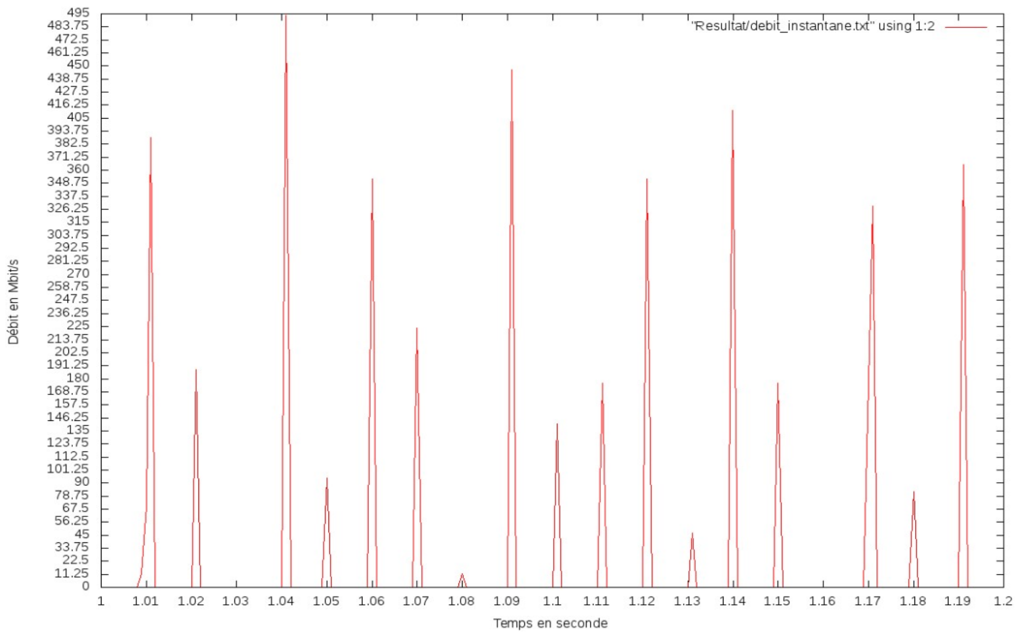


Flow peaks are observed every 10ms. Indeed, OpenSAND buffers the packets arriving on the GW to retransmit them every 10ms (default value) to the terminal by adding the desired delay.

If the BHBox is set up on GW with parameters BHS=13 ms and freq=2 (timeline [1.0]), the output rate of the GW (GW opensand_tun interface) has the following profile:



This traffic will then pass through OpenSAND. The flow profile finally received by the ST and the WS-ST end user is as follows:



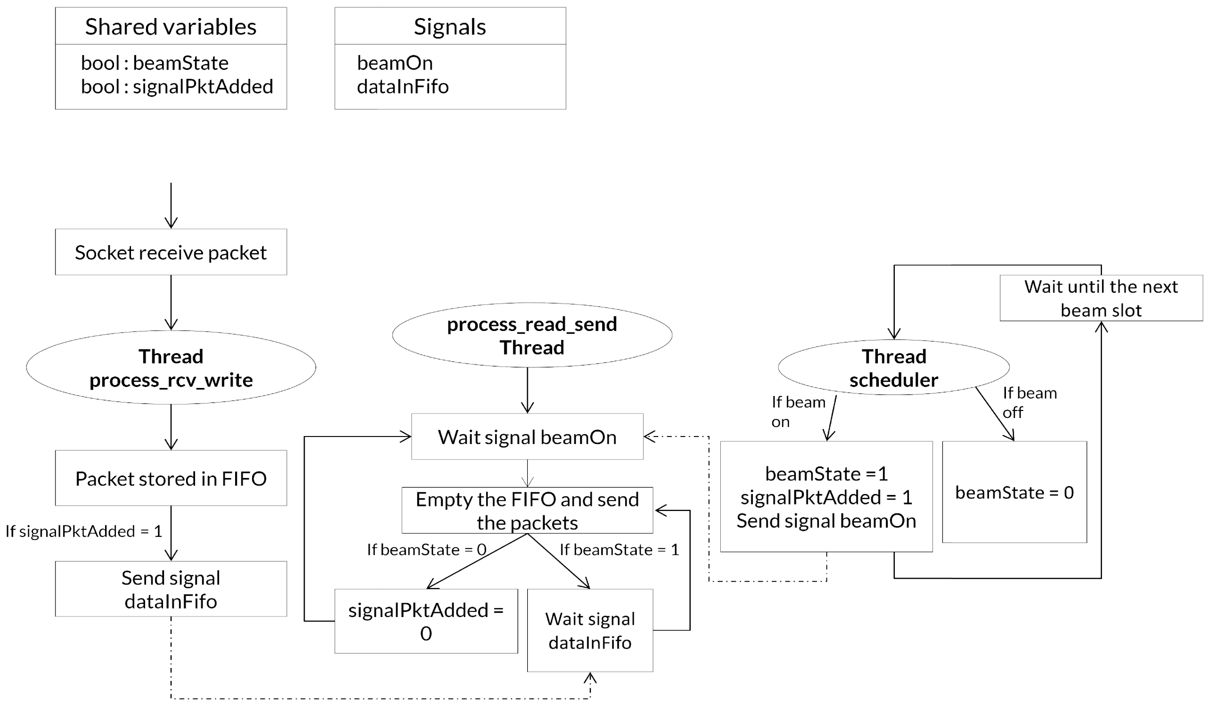
It could also be interesting to look how the implementation of the BHBox changes latency and jitter on satellite communication performed by OpenSAND. The OpenBACH jobs iperf, fping, owamp_client/owamp_server can be used to obtain these metrics in the 3 cases tested (without BHBox, with BHBox and timeline [1, 1] and finally with BHBox and timeline [1 ,0]):

	Without BHBox	With BHBox timeline [1,1]	With BHBox timeline [1,0]
Average jitter (ms)	0.753	0.758	0.770
Average latency (ms)	264.994	264.951	268.221

The integration of the BHBox on OpenSand almost does not change the jitter on the communication. By comparing the latency values for the two cases without beam-hopping (without BHBox and with BHBox and timeline [1, 1]), we identify that the packet passage at the application level does not have a major influence on latency. With the execution of the BHBox with a timeline [1.0], we observe an increase in the average latency of **3.27 ms**, which is very close to the theoretically expected value 3.25 ms.

Architecture of BHBox

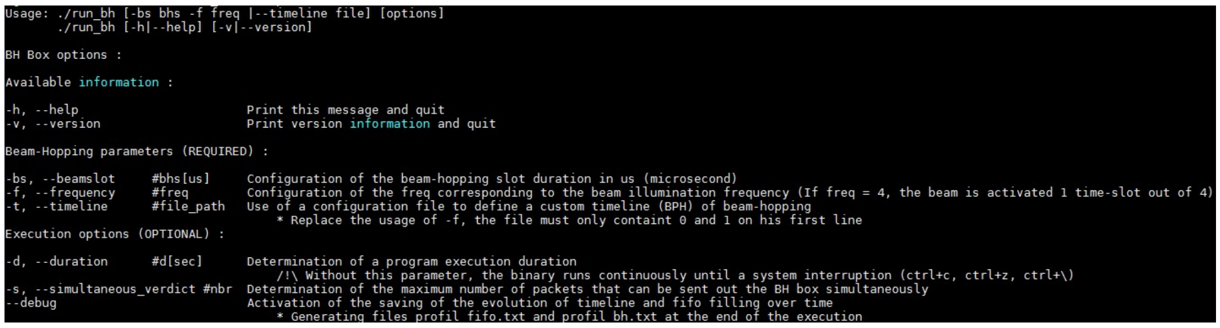
The program developed is a multithreaded application. Its design can be summarized by the diagram below:



Parameters of the BHBox

The help for setting the executable can be obtained by using the command:

```
./run_bh --help
```



```
Usage: ./run_bh [-bs bhs -f freq |--timeline file] [options]
       ./run_bh [-h|--help] [-v|--version]

BH Box options :

Available information :

-h, --help                         Print this message and quit
-v, --version                      Print version information and quit

Beam-Hopping parameters (REQUIRED) :

-bs, --beamslot     #bhs[us]     Configuration of the beam-hopping slot duration in us (microsecond)
-f, --frequency     #freq        Configuration of the freq corresponding to the beam illumination frequency (If freq = 4, the beam is activated 1 time-slot out of 4)
-t, --timeline      #file_path   Use of a configuration file to define a custom timeline (BPH) of beam-hopping
                                     * Replace the usage of -f, the file must only containt 0 and 1 on his first line
Execution options (OPTIONAL) :

-d, --duration      #d[sec]      Determination of a program execution duration
                                     /!\ Without this parameter, the binary runs continuously until a system interruption (ctrl+c, ctrl+z, ctrl+\)
-s, --simultaneous_verdict #nbr  Determination of the maximum number of packets that can be sent out the BH box simultaneously
--debug                          Activation of the saving of the evolution of timeline and fifo filling over time
                                     * Generating files profil fifo.txt and profil bh.txt at the end of the execution
```

Shared variables:

- *beamState:* Gives the illumination state. If the beam state is 1, the beam is considered to be ON, otherwise it is considered to be OFF. This variable can be viewed and modified by several threads, so it is protected by a mutex.

- *signalPktAdded:* this Boolean indicates at a given time t whether or not a signal mechanism should be used by the threads *processRecvWrite* and *processReadSend*. This variable can be viewed and modified by several threads, so it is protected by a mutex.

The threads:

- *scheduler*: this thread manages the timeline defined by the program's input parameter. If the timeline indicates an ON state, it sets the shared variable *beamState* to 1 and sends a signal to *processReadSend* to inform it that the beam is now ON. Otherwise, it sets the *beamState* variable to 0. After this step, it waits until the time of the next bhs.
- *processRecvWrite:* this thread receives the packets from the socket. It then stores them in a FIFO. If the shared variable *signalPktAdded* is 1, it sends a signal to *processReadSend* to inform it that a packet has just been added to the FIFO.
- *processReadSend*: This thread waits until receiving a signal informing it that the beam is ON. When the signal is received, it empties the FIFO and sends the packets. It then waits to receive a signal informing it that a new packet has been stored, while checking that the beam is still ON. If it does, it sends the packet. Otherwise, it waits for the next signal informing it that the beam is on.