# Trollcast Documentation

***Release v0.2.0***

**Martin Raspaud**

August 18, 2014

To the source code page.

Trollcast is a tool to exchange polar weather satellite data. It aims at providing near real time data transfer between peers, and should be adaptable to any type of data that is scan-based. At the moments it works on hrpt minor frame data (both big and little endian).

The protocol it uses is loosely based on bittorrent.

**Warning:** This is experimental software, use it at your own risk!

CHAPTER 1

# Installing trollcast

Download trollcast from the source code page and run:

```
python setup.py install
```

# Setting up trollcast

A trollcast config file describes the different parameters one needs for running both the client and the server.

```
[local_reception]
localhost=nimbus
remotehosts=safe
data=hrpt
data_dir=/data/hrpt
file_pattern=*.temp
max_connections=2
station=norrköping
coordinates=16.148649 58.581844 0.02
tle_files=/var/opt/2met/data/polar/orbitalelements/*.tle

[safe]
hostname=172.29.0.236
pubport=9333
reqport=9332

[nimbus]
hostname=172.22.8.16
pubport=9333
reqport=9332
```

## 2.1 The *local_reception* section

- *localhost* defines the name of the host the process is going to run on locally. This name will be user further down in the configuration file as a section which will hold information about the host. More on this later.
- *remotehosts* is the list of remote hosts to communicate with.
- *data* give the type of data to be exchange. Only *hrpt* is available at the moment.
- *data_dir* is the place where streaming data from the reception station is written.
- *file_pattern* is the fnmatch pattern to use to detect the file that the reception station writes to. Trollcast will watch this file to stream the data to the network in real time.
- *max_connections* tells how many times the data can be sent. This is usefull for avoiding too many clients retrieving the data from the same server, putting unnecessary load on it. Instead, clients will spread the data among each other, creating a more distributed load.
- *station*: name of the station

- *coordinates*: coordinates of the station. Used for the computation of satellite elevation. Lon/lats in degrees, altitude in kilometers.
- *tle_dir*: directory holding the latest TLE data. Used for the computation of satellite elevation.

## 2.2 The host sections

- *hostname* is the hostname or the ip address of the host.
- *pubport* on which publishing of messages will occur.
- *reqport* on which request and transfer of data will occur.

CHAPTER 3

# Modes of operation

## 3.1 Server mode, giving out data to the world

The server mode is used to serve data to remote hosts.

**It is started with::** trollcast_server my_config_file.cfg

This will start a server that watches a given file, as specified in the configuration file. Add a `-v` if you want debugging info.

**Note:** In the eventuality that you want to start a sever in gateway mode, that is acting as a gateway to another server, add `mirror=name_of_the_primary_server` in your configuration file.

## 3.2 Client mode, retrieving data

The client mode retrieves data.

Here is the usage of the client:

```
usage: client.py [-h] [-t TIMES TIMES] [-o OUTPUT] -f CONFIG_FILE
                 satellite [satellite ...]

positional arguments:
  satellite             eg. noaa_18

optional arguments:
  -h, --help            show this help message and exit
  -t TIMES TIMES, --times TIMES TIMES
                        Start and end times, <YYYYMMDDHHMMSS>
  -o OUTPUT, --output OUTPUT
                        Output file (used only in conjuction with -t)
  -f CONFIG_FILE, --config_file CONFIG_FILE
                        eg. sattorrent_local.cfg
  -v, --verbose
```

**There are two ways of running the client:**

- The first way is to retrieve a given time interval of data. For example, to retrieve data from NOAA 18 for the 14th of November 2012, between 14:02:23 and 14:15:00, the client has to be called with:

```
trollcast_client -t 20121114140223 20121114141500 -o noaa18_20121114140223.hmf -f config_fil
```

- The second way is to retrieve all the data possible data and dump it to files:

  ```
  trollcast_client -f config_file.cfg noaa_15 noaa_16 noaa_18 noaa_19
  ```

  In this case, only new data will be retrieved though, contrarily to the time interval retrieval where old data will be retrieved too if necessary.

Contents:

CHAPTER 4

# API

## 4.1 Client

Trollcast client. Leeches all it can :)

**class** `trollcast.client.`**`Client`**(*cfgfile='sattorrent.cfg'*)

The client class.

**`get_all`**(*satellites*)

Retrieve all the available scanlines from the stream, and save them.

**`get_lines`**(*satellite*, *scanline_dict*)

Retrieve the best (highest elevation) lines of *scanline_dict*.

**`order`**(*time_slice*, *satellite*, *filename*)

Get all the scanlines for a *satellite* within a *time_slice* and save them in *filename*. The scanlines will be saved in a contiguous manner.

**`send_lineinfo_to_server`**(**args*, ***kwargs*)

Send information to our own server.

**`stop`**()

**class** `trollcast.client.`**`HaveBuffer`**(*cfgfile='sattorrent.cfg'*)

Listen to incomming have messages.

**`add_queue`**(*queue*)

Adds a queue to dispatch have messages to

**`del_queue`**(*queue*)

Deletes a dispatch queue.

**`run`**()

**`send_to_queues`**(*sat*, *utctime*)

Send scanline at *utctime* to queues.

**`stop`**()

Stop buffering.

**class** `trollcast.client.`**`RTimer`**(*tries*, *warning_message*, *function*, **args*, ***kwargs*)

**`alert`**()

**`reset`**()

**run**()

**stop**()

**class** trollcast.client.**Requester**(*host*, *port*, *station*, *pubport=None*)
Make a request connection, waiting to get scanlines .

**get_line**(*satellite*, *utctime*)
Get the scanline of *satellite* at *utctime*.

**get_slice**(*satellite*, *start_time*, *end_time*)
Get a slice of scanlines.

**ping**()
Send a ping.

**recv**(*timeout=None*)
Receive a message. *timeout* in ms.

**send**(*msg*)
Send a message.

**send_lineinfo**(*sat*, *utctime*, *elevation*, *filename*, *pos*)
Send information to our own server.

**stop**()
Close the socket.

trollcast.client.**compute_line_times**(*utctime*, *start_time*, *end_time*)
Compute the times of lines if a swath order depending on a reference *utctime*.

trollcast.client.**create_requesters**(*cfgfile*)
Create requesters to all the configure remote hosts.

trollcast.client.**create_subscriber**(*cfgfile*)
Create a new subscriber for all the remote hosts in cfgfile.

trollcast.client.**create_timers**(*cfgfile*, *subscriber*)

trollcast.client.**reset_subscriber**(*subscriber*, *addr*)

## 4.2 Server

Trollcast, server side.

Trollcasting is loosely based on the bittorrent concepts, and adapted to satellite data.

**Limitations:**

- HRPT specific at the moment

**TODO:**

- Include files from a library, not only the currently written file to the list of scanlines
- Implement choking
- de-hardcode filename

**class** trollcast.server.**FileStreamer**(*holder*, *configfile*, **args*, ***kwargs*)
Get the updates from files.

TODO: separate holder from file handling.

**on_created**(*event*)
Callback when file is created.

**on_modified**(*event*)

**on_opened**(*event*)
Callback when file is opened

**update_satellite**(*satellite*)
Update satellite and renew the orbital instance.

**class** trollcast.server.**Heart**(*holder*, **args*, ***kwargs*)

**run**()

**stop**()

**class** trollcast.server.**Holder**(*configfile*)

**add_scanline**(*satellite*, *utctime*, *elevation*, *line_start*, *filename*, *line=None*)
Adds the scanline to the server. Typically used by the client to signal newly received lines.

**get**(**args*, ***kwargs*)

**get_scanline**(*satellite*, *utctime*)

**send_have**(*satellite*, *utctime*, *elevation*)
Sends 'have' message for *satellite*, *utctime*, *elevation*.

**send_heartbeat**(*next_pass_time='unknown'*)

**class** trollcast.server.**Looper**

**stop**()

**class** trollcast.server.**MirrorStreamer**(*holder*, *configfile*)
Act as a relay...

**run**()

**stop**()
Stop streaming.

**class** trollcast.server.**Responder**(*holder*, *configfile*, **args*, ***kwargs*)

**forward_request**(*address*, *message*)
Forward a request to another server.

**run**()

**stop**()

**class** trollcast.server.**Socket**(*addr*, *stype*)

**class** trollcast.server.**SocketLooper**(**args*, ***kwargs*)

**class** trollcast.server.**SocketLooperThread**(**args*, ***kwargs*)

trollcast.server.**serve**(*configfile*)
Serve forever.

trollcast.server.**timecode**(*tc_array*)

CHAPTER 5

# Indices and tables

- *genindex*
- *modindex*
- *search*

# Python Module Index