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 16-bits 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!
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={utctime:%Y%m%d%H%M%S}_{platform:4s}_{number:2s}.temp
max_connections=2
station=norrköping
coordinates=16.148649 58.581844 0.02
tle_files=/var/opt/2met/data/polar/orbitalelements/*.tle
schedule_file=/var/opt/2met/data/polar/schedule/schedule.txt
schedule_format=scisys
mirror=my_receiver_server
output_file=/tmp/{utctime:%Y%m%d%H%M%S}_{platform:4s}_{number:2s}.trollcast.hmf
publisher=trollcast_receiver

[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 used 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** gives the type of data to be exchanged. 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 pattern (trollsift syntax) 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 useful 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.

• **schedule_file**: schedule file to give trollcast server the knowledge of passes to come.

• **schedule_format**: the format of the schedule file. Supported at the moment: **scisys** and **kongsberg_metno**.

• **mirror**: the hostname or ip address of the trollcast server the current server has to mirror.

• **output_file**: the file to write data to, in trollsift syntax.

• **publisher**: the name under which to publish new incoming files. If you don’t want to publish anything (which is usually the case when you don’t have a posttroll based trigger to handle messages), just omit this option.

### 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. Some options are also available:

```

positional arguments:
  config_file

optional arguments:
  -h, --help            show this help message and exit
  -l LOG, --log LOG     File to log to.
  -m MAIL, --mail MAIL  Mail to log to.
  -v, --verbose         Print out debug messages also.
```

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

Note: Don’t forget to prepend “nohup” to the command if you want to make sure the process doesn’t shut down when you logout from the server.

3.2 Client mode, retrieving data

The client mode retrieves data.

Here is the usage of the client:

```
usage: trollcast_client [-h] [-t TIMES TIMES] [-o OUTPUT] -f CONFIG_FILE [-v]
                      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 conjunction with -t)
- `f CONFIG_FILE, --config_file CONFIG_FILE` eg. sattorrent_local.cfg
- `l LOG, --log LOG` File to log to.
- `-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_file.cfg
  ```

- 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.
4.1 Client

Trollcast client. Leeches all it can :) todo: - connection between institutes is shutdown after a while (2 hours ?) - filename are wrong (1 year to old) - Option for new log file every day? Now log files are quite big after few days. - resets connection to mirror in case of timeout.

```python
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.

class trollcast.client.SimpleRequester (host, port)
Base requester class.
connect ()
Connect to the server
request_retries = 3
reset_connection ()
Reset the socket
send_and_recv (msg, timeout=1000000)
stop ()
Close the connection to the server

class trollcast.client.Subscriber (addresses, translate=False)

addr_sub
recv (timeout=None)
Receive a message, timeout in seconds.
reset (addr)
stop ()
Stop the subscriber
sub_addr

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_publisher (cfgfile)
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

New version of the trollcast server

TODO:
    • add lines when local client gets data (if missing)
    • check that mirror server is alive

class trollcast.server.CADU
    The cadu reader class
    static is_it(data)

class trollcast.server.Cleaner(holder, delay)
    Dummy watcher for test purposes
    clean()
        Clean the db
    run()
    stop()
        Stop adding stuff

class trollcast.server.DummyWatcher(holder, uri)
    Dummy watcher for test purposes
    run()
    stop()
        Stop adding stuff

class trollcast.server.FileWatcher(holder, uri, schedule_reader)
    run()
    start()
        Start the file watcher
    stop()
        Stop the file watcher

class trollcast.server.HRPT(sat, reftime)
    The hrpt reader class
    dtype
        hrpt_sync
        hrpt_sync_start
    static is_it(data)
    line_size = 22180
**Trollcast Documentation, Release v0.2.0**

```python
read(data, f_elev=None)
Read hrpt data.


static timecode(tc_array)
HRPT timecode reading

class trollcast.server.Heart (pub, address, interval, schedule_reader)
Send heartbeats once in a while.

run()
stop()
Cardiac arrest

class trollcast.server.Holder (pub, origin)
The mighty data holder

add(sat, key, elevation, qual, data)
Add some data.

delete(sat, key)
Delete item

get(sat, key)
get the value of sat and key

get_data(sat, key)
get the data of sat and key

get_sat(sat)
Get the data for a given satellite sat.

have(sat, key, elevation, qual)
Tell the world about our new data.

sats()
return the satellites in store.

class trollcast.server.MirrorWatcher (holder, host, pubport, reaport, sched)
Watches a other server.

run()
stop()
Stop the watcher

class trollcast.server.Publisher (port)
Publish stuff.

send(message)
Publish something

stop()
Stop publishing.

class trollcast.server.RequestManager (holder, port, station)
Manage requests.

notice(message)
Reply to notice message

pong()
Reply to ping
```
run()

**scanline**(message)
Reply to scanline request

**send**(message)
Send a message

stop()
Stop the request manager.

**unknown**(message)
Reply to any unknown request.

class trollcast.server.ScheduleReader(filename, fileformat)
Reads and handles a schedule

**get_next_pass**()
Get the next pass from the schedule

trollcast.server.get_f_elev(satellite)
Get the elevation function for a given satellite

trollcast.server.serve(configfile)
Serve forever.

trollcast.server.set_subject(station)

### 4.3 Schedule readers

Read schedule files.

trollcast.schedules.kongsberg_metno(filename)
Read a kongsberg schedule

trollcast.schedules.scisys(filename)
Read a scisys schedule
CHAPTER 5

Indices and tables

• genindex
• modindex
• search
t

trollcast.client, 9

trollcast.schedules, 13

trollcast.server, 11
Index

A
add (trollcast.server.Holder method), 12
add_queue (trollcast.client.HaveBuffer method), 9
addr_sub (trollcast.client.Subscriber attribute), 10
alert (trollcast.client.RTimer method), 9

C
CADU (class in trollcast.server), 11
clean (trollcast.server.Cleaner method), 11
Cleaner (class in trollcast.server), 11
Client (class in trollcast.client), 9
compute_line_times (in module trollcast.client), 10
connect (trollcast.client.SimpleRequester method), 10
create_publisher (in module trollcast.client), 10
create_requesters (in module trollcast.client), 10
create_subscribers (in module trollcast.client), 10
create_timers (in module trollcast.client), 11

D
del_queue (trollcast.client.HaveBuffer method), 9
delete (trollcast.server.Holder method), 12
dtype (trollcast.server.HRPT attribute), 11
DummyWatcher (class in trollcast.server), 11

F
FileWatcher (class in trollcast.server), 11

G
get (trollcast.server.Holder method), 12
get_all (trollcast.client.Client method), 9
get_data (trollcast.client.HRPT method), 12
get_f_elev (in module trollcast.server), 13
get_line (trollcast.client.Requester method), 10
get_lines (trollcast.client.Requester method), 10
get_next_pass (trollcast.server.ScheduleReader method), 13
get_sat (trollcast.server.Holder method), 12
get_slice (trollcast.client.Requester method), 10

H
have (trollcast.server.Holder method), 12
HaveBuffer (class in trollcast.client), 9
Heart (class in trollcast.server), 12
Holder (class in trollcast.server), 12
HRPT (class in trollcast.server), 11
hrpt_sync (trollcast.server.HRPT attribute), 11
hrpt_sync_start (trollcast.server.HRPT attribute), 11

I
is_it (trollcast.server.CADU static method), 11
is_it (trollcast.server.HRPT static method), 11

K
kongsberg_metno (in module trollcast.schedules), 13

L
line_size (trollcast.server.HRPT attribute), 11

M
MirrorWatcher (class in trollcast.server), 12

N
notice (trollcast.server.RequestManager method), 12

O
order (trollcast.client.Client method), 9

P
ping (trollcast.client.Requester method), 10
pong (trollcast.server.RequestManager method), 12
Publisher (class in trollcast.server), 12

R
read (trollcast.server.HRPT method), 11
recv (trollcast.client.Requester method), 10
recv (trollcast.client.Subscriber method), 10
request_retries (trollcast.client.SimpleRequester attribute), 10
Requester (class in trollcast.client), 10
RequestManager (class in trollcast.server), 12
reset() (trollcast.client.RTimer method), 9
reset() (trollcast.client.Subscriber method), 10
reset_connection() (trollcast.client.SimpleRequester method), 10
reset_subscriber() (in module trollcast.client), 11
RTimer (class in trollcast.client), 9
run() (trollcast.client.HaveBuffer method), 9
run() (trollcast.client.RTimer method), 10
run() (trollcast.server.Cleaner method), 11
run() (trollcast.server.DummyWatcher method), 11
run() (trollcast.server.FileWatcher method), 11
run() (trollcast.server.Heart method), 12
run() (trollcast.server.MirrorWatcher method), 12
run() (trollcast.server.RequestManager method), 13
s
satellites (trollcast.server.HRPT attribute), 12
sats() (trollcast.server.Holder method), 12
scanline() (trollcast.server.RequestManager method), 13
ScheduleReader (class in trollcast.server), 13
scisys() (in module trollcast.schedules), 13
send() (trollcast.client.Requester method), 10
send() (trollcast.server.Publisher method), 12
send() (trollcast.server.RequestManager method), 13
send_and_recv() (trollcast.client.SimpleRequester method), 10
send_lineinfo() (trollcast.client.Requester method), 10
send_lineinfo_to_server() (trollcast.client.Client method), 9
send_to_queues() (trollcast.client.HaveBuffer method), 9
serve() (in module trollcast.server), 13
set_subject() (in module trollcast.server), 13
SimpleRequester (class in trollcast.client), 10
start() (trollcast.server.FileWatcher method), 11
stop() (trollcast.client.Client method), 9
stop() (trollcast.client.HaveBuffer method), 9
stop() (trollcast.client.RTimer method), 10
stop() (trollcast.client.SimpleRequester method), 10
stop() (trollcast.client.Subscriber method), 10
stop() (trollcast.server.Cleaner method), 11
stop() (trollcast.server.DummyWatcher method), 11
stop() (trollcast.server.FileWatcher method), 11
stop() (trollcast.server.Heart method), 12
stop() (trollcast.server.MirrorWatcher method), 12
stop() (trollcast.server.Publisher method), 12
stop() (trollcast.server.RequestManager method), 13
sub_addr (trollcast.client.Subscriber attribute), 10
Subscriber (class in trollcast.client), 10
T
timecode() (trollcast.server.HRPT static method), 12
trollcast.client (module), 9
trollcast.schedules (module), 13
trollcast.server (module), 11
U
unknown() (trollcast.server.RequestManager method), 13

Index