DHT-crawler

BitTorrent Distributed Hash Table Monitor

Developer documentation

Martin Vaško, Libor Polčák

TARZAN project VI20172020062 document
Faculty of Information Technology, Brno University of Technology

Last change: February 28, 2020

DHT-crawler

Generated by Doxygen 1.8.16

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 dht_crawler Namespace Reference

Namespaces

- arg_parse
- exec
- handshake
- monitor
- process_output
- torrent_dht

4.2 dht_crawler.arg_parse Namespace Reference

Functions

- def argument_parser ()
- def parse_input_args ()

4.2.1 Detailed Description

```
Created by Martin Vaško
Argument parser
```

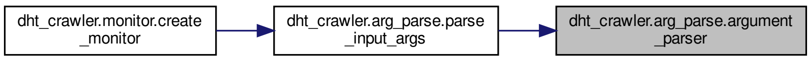
4.2.2 Function Documentation

4.2.2.1 argument_parser()

```
def dht_crawler.arg_parse.argument_parser ( )
```

```
This part is about parsing input arguments. Using argparse for
standardized use
```

Here is the caller graph for this function:

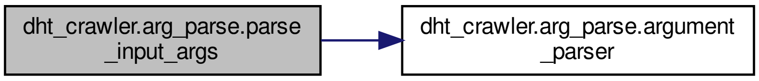

dht_crawler.monitor.create _monitor
dht_crawler.arg_parse.parse _input_args
dht_crawler.arg_parse.argument _parser

4.2.2.2 parse_input_args()

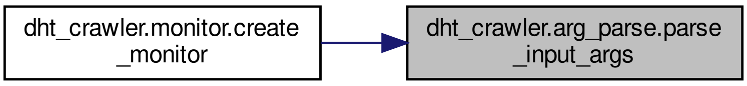
```
def dht_crawler.arg_parse.parse_input_args ( )
```

```
Parse arguments from argParse class
```

Here is the call graph for this function:


dht_crawler.arg_parse.parse _input_args
dht_crawler.arg_parse.argument _parser

Here is the caller graph for this function:


dht_crawler.monitor.create _monitor
dht_crawler.arg_parse.parse _input_args

4.3 dht_crawler.exec Namespace Reference

Variables

- CRAWL = create_monitor(False)

4.3.1 Detailed Description

```
Execution script
```

4.3.2 Variable Documentation

4.3.2.1 CRAWL

```
dht_crawler.exec.CRAWL = create_monitor(False)
```

4.4 dht_crawler.handshake Namespace Reference

Classes

- class TorrentHandshake

4.4.1 Detailed Description

```
Created by Martin Vaško
part of BT library which should implement handshake methods.
```

4.5 dht_crawler.monitor Namespace Reference

Classes

- class Monitor

Functions

- def kill_sender_reciever (thread1, thread2=None)
- def init_socket (port)
- def create_monitor (verbosity=False)

4.5.1 Detailed Description

```
Created by Martin Vasko
3BIT, Brno, Faculty of Information Technology.

Brief information:
This is implementation of monitoring BiTtorrent with Kademlia DHT.
Whole monitor class which will be presented next is going to be supported by
torrentDHT implementation, which was implemented by Martin Vasko.
```

4.5.2 Function Documentation

4.5.2.1 create_monitor()

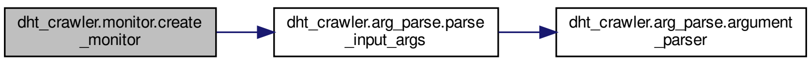
```
def dht_crawler.monitor.create_monitor (
              verbosity = False )
```

```
creates monitor class object. TorrentDHT creates udp socket which is
binded on `bind_port`. Monitor needs this `dht_socket` and command line
arguments to be created successfully. Then change of hash and parsing
can change resolution of crawl. When they are not specified then
global bootstrap nodes are used instead.

Parameters
----------
verbosity : bool
    Indicate verbose output

Returns
-------
object
    Monitor object with initialized DHT socket and parsed arguments.
```

Here is the call graph for this function:

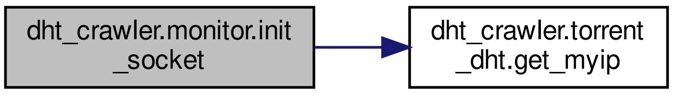

dht_crawler.monitor.create_monitor
dht_crawler.arg_parse.parse_input_args
dht_crawler.arg_parse.argument_parser

4.5.2.2 init_socket()

```
def dht_crawler.monitor.init_socket (
              port )
```

```
Initialize empty socket to send announce peer messages
```

Here is the call graph for this function:

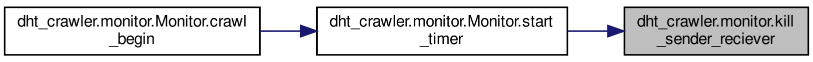

dht_crawler.monitor.init_socket
dht_crawler.torrent_dht.get_myip

4.5.2.3 kill_sender_reciever()

```
def dht_crawler.monitor.kill_sender_reciever (
              thread1,
              thread2 = None )
```

```
kill sender reciever and TorrentDHT socket when there is
continuous bootstrap.
```

Here is the caller graph for this function:


dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_timer
dht_crawler.monitor.kill_sender_reciever

4.6 dht_crawler.process_output Namespace Reference

Classes

- class ProcessOutput

4.6.1 Detailed Description

```
Create by Martin Vasko
3BIT, Brno, Faculty of Information Technology.
```

4.7 dht_crawler.torrent_dht Namespace Reference

Classes

- class TorrentArguments
- class TorrentDHT

Functions

- def entropy (length)
- def random_infohash ()
- def get_neighbor (target, infohash, end=10)
- def has_node (id_node, host, port, info_pool)
- def decode_krpc (message)
- def send_krpc (message, node, sock)
- def decode_nodes (value, info_pool)
- def decode_peers (infohash, peers, info_pool, token, unique=None)
- def get_myip ()

4.7.1 Detailed Description

```
Create by Martin Vasko
3BIT, Brno, Faculty of Information Technology.

This should be used as part of library, where you can create,
bind socket and send all torrent DHT messages over UDP.
BOOTSTRAP_NODES are well known nodes which should begin torrent
peer detection.
```

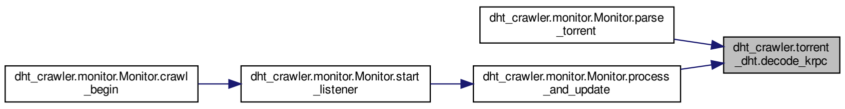
4.7.2 Function Documentation

4.7.2.1 decode_krpc()

```
def dht_crawler.torrent_dht.decode_krpc (
              message )
```

```
decode with bencoding. When exception is thrown, return None.
```

Here is the caller graph for this function:

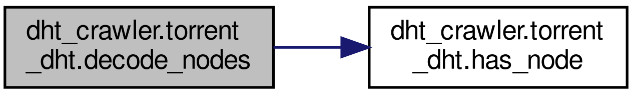

dht_crawler.monitor.Monitor.parse_torrent
dht_crawler.torrent_dht.decode_krpc
dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_listener
dht_crawler.monitor.Monitor.process_and_update

4.7.2.2 decode_nodes()

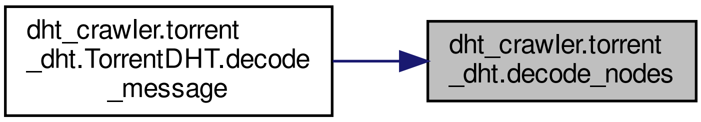
```
def dht_crawler.torrent_dht.decode_nodes (
              value,
              info_pool )
```

```
decode nodes from response message
```

Here is the call graph for this function:


dht_crawler.torrent _dht.decode_nodes
dht_crawler.torrent _dht.has_node

Here is the caller graph for this function:

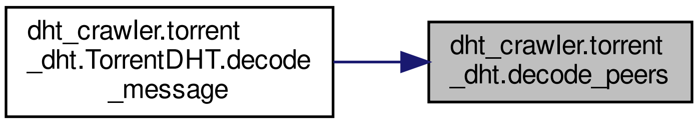

dht_crawler.torrent _dht.TorrentDHT.decode _message
dht_crawler.torrent _dht.decode_nodes

4.7.2.3 decode_peers()

```
def dht_crawler.torrent_dht.decode_peers (
              infohash,
              peers,
              info_pool,
              token,
              unique = None )
```

```
decodes peers from get_peers response. They have only ip address and port
within message. When unique specified get only ip address as key
```

Here is the caller graph for this function:

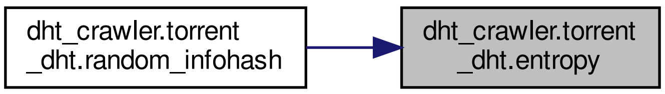

dht_crawler.torrent _dht.TorrentDHT.decode _message
dht_crawler.torrent _dht.decode_peers

4.7.2.4 entropy()

```
def dht_crawler.torrent_dht.entropy (
              length )
```

```
entropy to generate infohash
```

Here is the caller graph for this function:

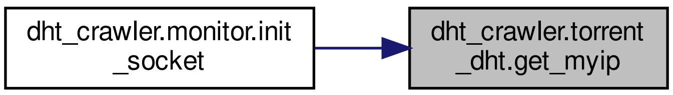

dht_crawler.torrent _dht.random_infohash
dht_crawler.torrent _dht.entropy

4.7.2.5 get_myip()

```
def dht_crawler.torrent_dht.get_myip ( )
```

```
get my global ip_address, by connecting to google and get sockname
```

Here is the caller graph for this function:

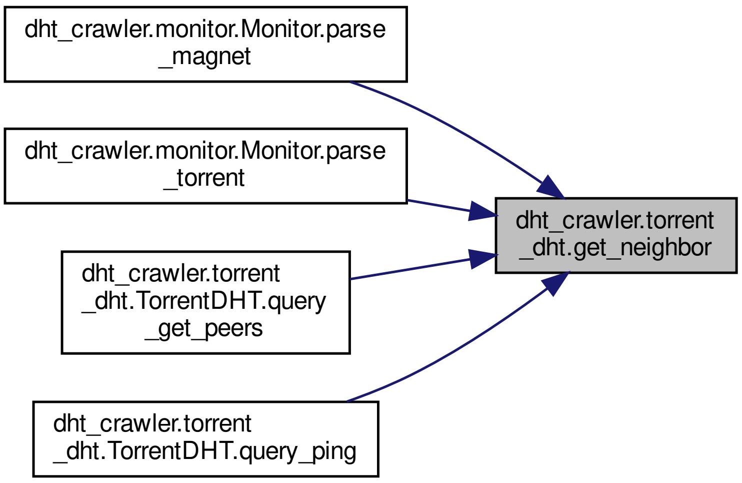

dht_crawler.monitor.init _socket
dht_crawler.torrent _dht.get_myip

4.7.2.6 get_neighbor()

```
def dht_crawler.torrent_dht.get_neighbor (
              target,
              infohash,
              end = 10 )
```

```
mixture of infohashes
```

Here is the caller graph for this function:

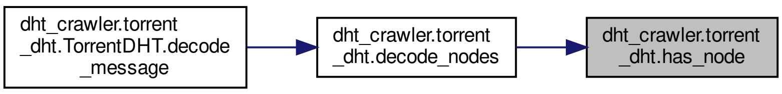

dht_crawler.monitor.Monitor.parse_magnet
dht_crawler.monitor.Monitor.parse_torrent
dht_crawler.torrent_dht.TorrentDHT.query_get_peers
dht_crawler.torrent_dht.TorrentDHT.query_ping
dht_crawler.torrent_dht.get_neighbor

4.7.2.7 has_node()

```
def dht_crawler.torrent_dht.has_node (
              id_node,
              host,
              port,
              info_pool )
```

```
when node is in infopool clear duplicities.
```

Here is the caller graph for this function:

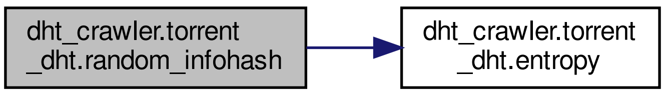

dht_crawler.torrent_dht.TorrentDHT.decode_message
dht_crawler.torrent_dht.decode_nodes
dht_crawler.torrent_dht.has_node

4.7.2.8 random_infohash()

```
def dht_crawler.torrent_dht.random_infohash ( )
```

```
generates random 20 bytes infohash
```

Here is the call graph for this function:

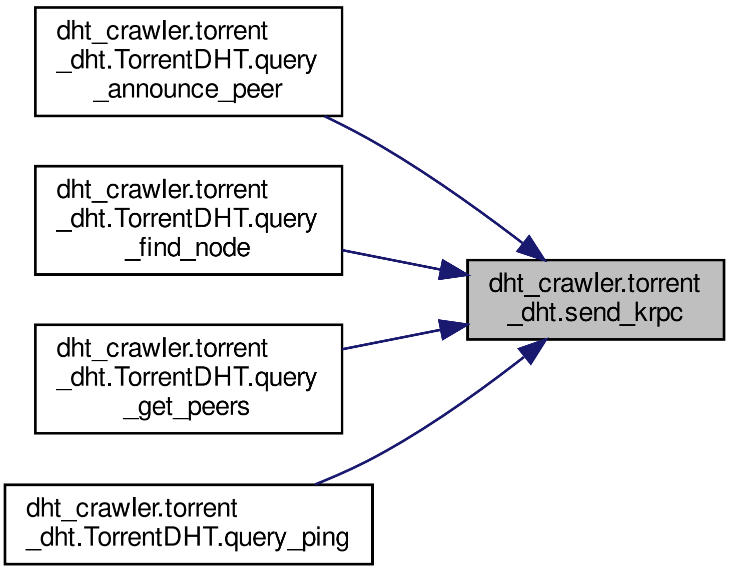

dht_crawler.torrent_dht.random_infohash
dht_crawler.torrent_dht.entropy

4.7.2.9 send_krpc()

```
def dht_crawler.torrent_dht.send_krpc (
              message,
              node,
              sock )
```

```
sends bencoded krpc to node ip address and node port
```

Here is the caller graph for this function:

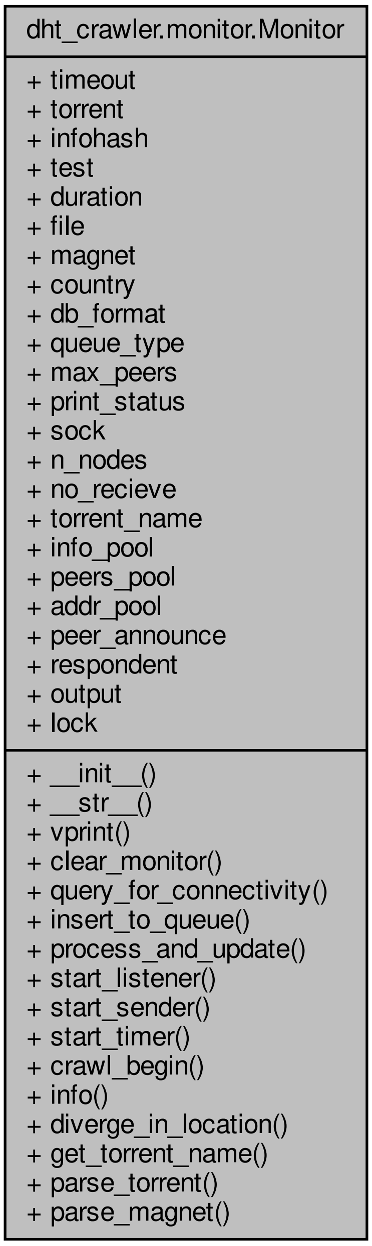

dht_crawler.torrent_dht.TorrentDHT.query_announce_peer
dht_crawler.torrent_dht.TorrentDHT.query_find_node
dht_crawler.torrent_dht.TorrentDHT.query_get_peers
dht_crawler.torrent_dht.TorrentDHT.query_ping
dht_crawler.torrent_dht.send_krpc

Chapter 5

Class Documentation

5.1 dht_crawler.monitor.Monitor Class Reference

Collaboration diagram for dht_crawler.monitor.Monitor:



Public Member Functions

- def __init__ (self, arguments, torrent)
- def __str__ (self)
- def vprint (self, msg)
- def clear_monitor (self)
- def query_for_connectivity (self)

  START OF CRAWLING #.
- def insert_to_queue (self, nodes)
- def process_and_update (self, ready, last_time)
- def start_listener (self)
- def start_sender (self, test=False)
- def start_timer (self, thread1, thread2)
- def crawl_begin (self, torrent=None, test=False)
- def info (self)
- def diverge_in_location (self, nodes)
- def get_torrent_name (self, value)
- def parse_torrent (self)
- def parse_magnet (self)

Public Attributes

- timeout
- torrent
- infohash
- test
- duration
- file
- magnet
- country
- db_format
- queue_type
- max_peers
- print_status
- sock
- n_nodes
- no_recieve
- torrent_name
- info_pool
- peers_pool
- addr_pool
- peer_announce
- respondent
- output
- lock

5.1.1 Detailed Description

```
Parse it from class methods to monitor class where we want to exchange
this information.
Start monitoring and initialize all necessary things at first
```

5.1.2 Constructor & Destructor Documentation

5.1.2.1 __init__()

```
def dht_crawler.monitor.Monitor.__init__ (
              self,
              arguments,
              torrent )
```

```
Construct a new 'Foo' object.

:param name: The name of foo
:param age: The ageof foo
:return: returns nothing
```

5.1.3 Member Function Documentation

5.1.3.1 __str__()

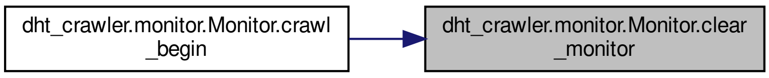
```
def dht_crawler.monitor.Monitor.__str__ (
              self )
```

5.1.3.2 clear_monitor()

```
def dht_crawler.monitor.Monitor.clear_monitor (
              self )
```

```
clear monitor class before next crawl
```

Here is the caller graph for this function:


dht_crawler.monitor.Monitor.crawl
_begin
dht_crawler.monitor.Monitor.clear
_monitor

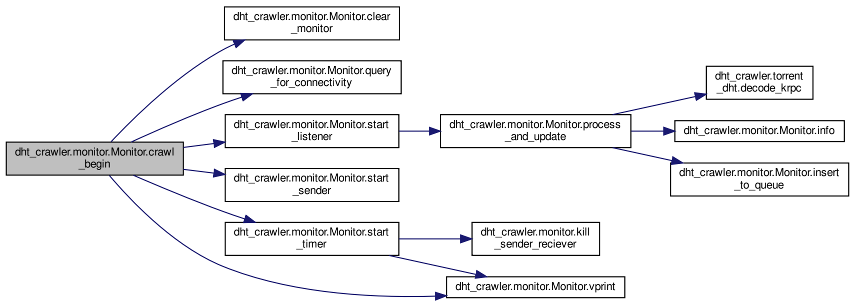
5.1.3.3 crawl_begin()

```
def dht_crawler.monitor.Monitor.crawl_begin (
              self,
              torrent = None,
              test = False )
```

```
Create all threads, duration to count how long program is executed.
When Ctrl+C is pressed kill all threads

Parameters
----------
torrent : infohash
    20 bytes long infohash which should be used as part of monitoring.
test : bool
    This paramter is for testing connection.
```

Here is the call graph for this function:


dht_crawler.monitor.Monitor.clear _monitor
dht_crawler.monitor.Monitor.query _for_connectivity
dht_crawler.monitor.Monitor.start _listener
dht_crawler.monitor.Monitor.crawl _begin
dht_crawler.monitor.Monitor.start _sender
dht_crawler.monitor.Monitor.start _timer
dht_crawler.monitor.Monitor.process _and_update
dht_crawler.torrent _dht.decode_krpc
dht_crawler.monitor.Monitor.info
dht_crawler.monitor.Monitor.insert _to_queue
dht_crawler.monitor.kill _sender_reciever
dht_crawler.monitor.Monitor.vprint

5.1.3.4 diverge_in_location()

```
def dht_crawler.monitor.Monitor.diverge_in_location (
              self,
              nodes )
```

```
After climbing to another teritory, do not access it,
return adjusted list of nodes.
```

5.1.3.5 get_torrent_name()

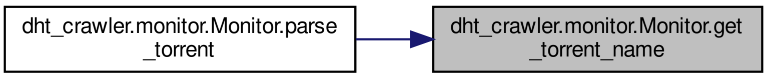
```
def dht_crawler.monitor.Monitor.get_torrent_name (
              self,
              value )
```

```
get name of torrent from torrent file

Parameters
----------
value : dict
    This should contain encoded name of torrent.

Returns
-------
self.torrent_name
    Parsed torrent name from value dictionary.
```

Here is the caller graph for this function:

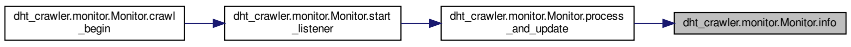

dht_crawler.monitor.Monitor.parse_torrent
dht_crawler.monitor.Monitor.get_torrent_name

5.1.3.6 info()

```
def dht_crawler.monitor.Monitor.info (
              self )
```

```
Print info for current state of crawling.
```

Here is the caller graph for this function:

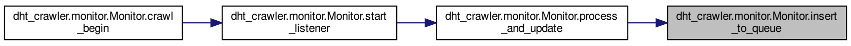

dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_listener
dht_crawler.monitor.Monitor.process_and_update
dht_crawler.monitor.Monitor.info

5.1.3.7 insert_to_queue()

```
def dht_crawler.monitor.Monitor.insert_to_queue (
              self,
              nodes )
```

```
Inserts nodes to queue by given queue type.
```

Here is the caller graph for this function:

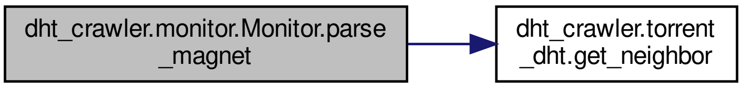

dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_listener
dht_crawler.monitor.Monitor.process_and_update
dht_crawler.monitor.Monitor.insert_to_queue

5.1.3.8 parse_magnet()

```
def dht_crawler.monitor.Monitor.parse_magnet (
              self )
```

```
parse magnet link
```

Here is the call graph for this function:

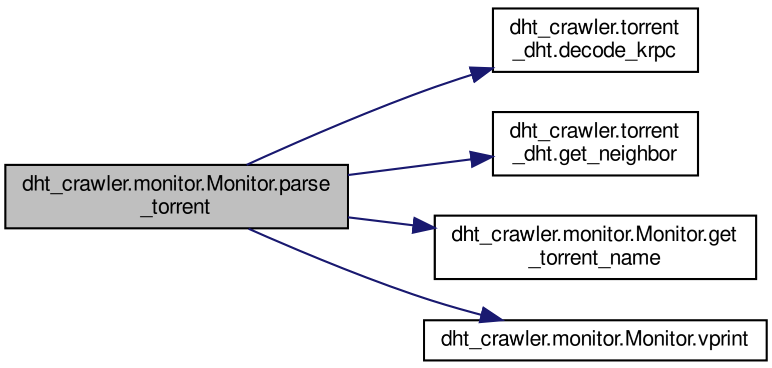

dht_crawler.monitor.Monitor.parse_magnet
dht_crawler.torrent_dht.get_neighbor

5.1.3.9 parse_torrent()

```
def dht_crawler.monitor.Monitor.parse_torrent (
              self )
```

```
parse torrent file to get infohash and announce list of nodes for
better bootstrap.
```

Here is the call graph for this function:

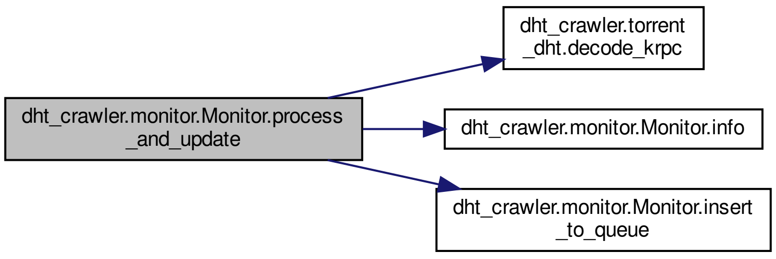

dht_crawler.monitor.Monitor.parse_torrent
dht_crawler.torrent_dht.decode_krpc
dht_crawler.torrent_dht.get_neighbor
dht_crawler.monitor.Monitor.get_torrent_name
dht_crawler.monitor.Monitor.vprint

5.1.3.10 process_and_update()

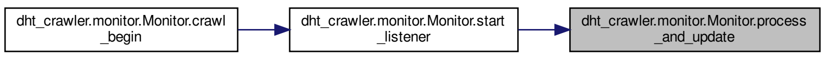
```
def dht_crawler.monitor.Monitor.process_and_update (
              self,
              ready,
              last_time )
```

```
process packet and update all necesseties like info_pool, peers_pool.
When decoding failed or not ready socket for recieving return back to
listener thread.
```

Here is the call graph for this function:


dht_crawler.monitor.Monitor.process_and_update
dht_crawler.torrent_dht.decode_krpc
dht_crawler.monitor.Monitor.info
dht_crawler.monitor.Monitor.insert_to_queue

Here is the caller graph for this function:


dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_listener
dht_crawler.monitor.Monitor.process_and_update

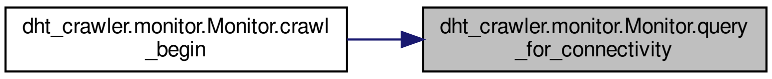
5.1.3.11 query_for_connectivity()

```
def dht_crawler.monitor.Monitor.query_for_connectivity (
              self )
```

START OF CRAWLING #.

```
Query all found peers for connectivity.
When respond, then connection is still there and peer is valid, else
peer is deleted from dictionary.
```

Here is the caller graph for this function:

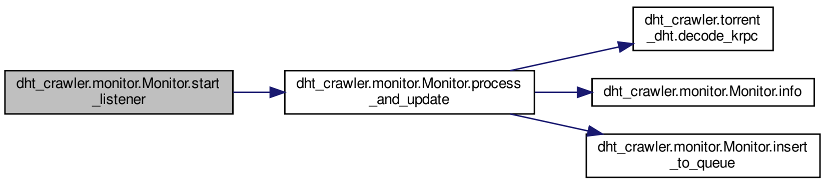

dht_crawler.monitor.Monitor.crawl _begin
dht_crawler.monitor.Monitor.query _for_connectivity

5.1.3.12 start_listener()

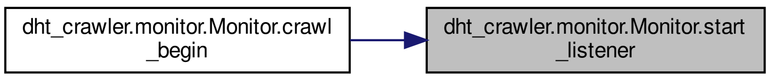
```
def dht_crawler.monitor.Monitor.start_listener (
              self )
```

```
start listener thread. Recieve query packet and decode its body.
There is shared queue between listener and sender thread.
```

Here is the call graph for this function:


dht_crawler.monitor.Monitor.start _listener
dht_crawler.monitor.Monitor.process _and_update
dht_crawler.torrent _dht.decode_krpc
dht_crawler.monitor.Monitor.info
dht_crawler.monitor.Monitor.insert _to_queue

Here is the caller graph for this function:

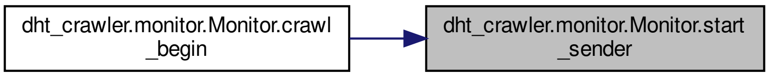

dht_crawler.monitor.Monitor.crawl _begin
dht_crawler.monitor.Monitor.start _listener

5.1.3.13 start_sender()

```
def dht_crawler.monitor.Monitor.start_sender (
              self,
              test = False )
```

```
start sender thread. There is test parameter to test connection for
unit testing. Otherwise continuous connection is performed till
we dont get all nodes from k-zone or duration is exhausted.
```

Here is the caller graph for this function:

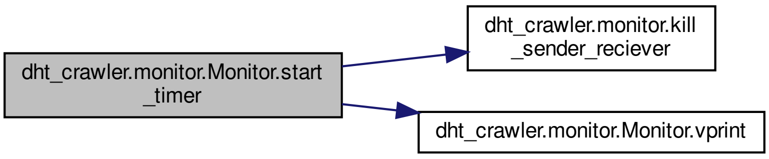

dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_sender

5.1.3.14 start_timer()

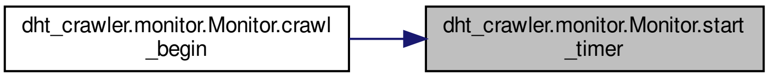
```
def dht_crawler.monitor.Monitor.start_timer (
              self,
              thread1,
              thread2 )
```

```
start thread timer for duration, when exhausted kill threads
and exit program.
```

Here is the call graph for this function:


dht_crawler.monitor.Monitor.start_timer
dht_crawler.monitor.kill_sender_reciever
dht_crawler.monitor.Monitor.vprint

Here is the caller graph for this function:

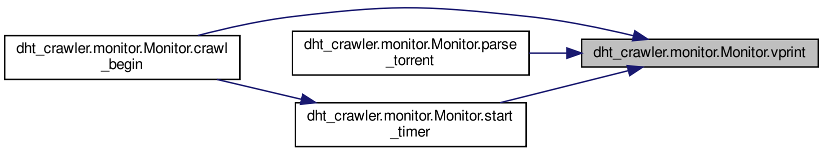

dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.start_timer

5.1.3.15 vprint()

```
def dht_crawler.monitor.Monitor.vprint (
              self,
              msg )
```

```
Print only when -v parameter is present
```

Here is the caller graph for this function:


dht_crawler.monitor.Monitor.crawl_begin
dht_crawler.monitor.Monitor.parse_torrent
dht_crawler.monitor.Monitor.vprint
dht_crawler.monitor.Monitor.start_timer

5.1.4 Member Data Documentation

5.1.4.1 addr_pool

```
dht_crawler.monitor.Monitor.addr_pool
```

5.1.4.2 country

```
dht_crawler.monitor.Monitor.country
```

5.1.4.3 db_format

```
dht_crawler.monitor.Monitor.db_format
```

5.1.4.4 duration

```
dht_crawler.monitor.Monitor.duration
```

5.1.4.5 file

```
dht_crawler.monitor.Monitor.file
```

5.1.4.6 info_pool

```
dht_crawler.monitor.Monitor.info_pool
```

5.1.4.7 infohash

```
dht_crawler.monitor.Monitor.infohash
```

5.1.4.8 lock

```
dht_crawler.monitor.Monitor.lock
```

5.1.4.9 magnet

```
dht_crawler.monitor.Monitor.magnet
```

5.1.4.10 max_peers

```
dht_crawler.monitor.Monitor.max_peers
```

5.1.4.11 n_nodes

```
dht_crawler.monitor.Monitor.n_nodes
```

5.1.4.12 no_recieve

```
dht_crawler.monitor.Monitor.no_recieve
```

5.1.4.13 output

```
dht_crawler.monitor.Monitor.output
```

5.1.4.14 peer_announce

```
dht_crawler.monitor.Monitor.peer_announce
```

5.1.4.15 peers_pool

```
dht_crawler.monitor.Monitor.peers_pool
```

5.1.4.16 print_status

```
dht_crawler.monitor.Monitor.print_status
```

5.1.4.17 queue_type

```
dht_crawler.monitor.Monitor.queue_type
```

5.1.4.18 respondent

```
dht_crawler.monitor.Monitor.respondent
```

5.1.4.19 sock

```
dht_crawler.monitor.Monitor.sock
```

5.1.4.20 test

```
dht_crawler.monitor.Monitor.test
```

5.1.4.21 timeout

```
dht_crawler.monitor.Monitor.timeout
```

5.1.4.22 torrent

```
dht_crawler.monitor.Monitor.torrent
```

5.1.4.23 torrent_name

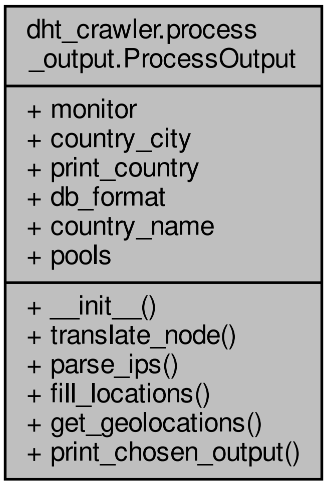
```
dht_crawler.monitor.Monitor.torrent_name
```

The documentation for this class was generated from the following file:

- dht_crawler/monitor.py

5.2 dht_crawler.process_output.ProcessOutput Class Reference

Collaboration diagram for dht_crawler.process_output.ProcessOutput:



Public Member Functions

- def __init__ (self, monitor, country_print, country)
- def translate_node (self, nodes)
- def parse_ips (self)
- def fill_locations (self, location_info, iplist=None)
- def get_geolocations (self)
- def print_chosen_output (self)

Public Attributes

- monitor
- country_city
- print_country
- db_format
- country_name
- pools

5.2.1 Detailed Description

```
Process output to process regex or to remove duplicities
```

5.2.2 Constructor & Destructor Documentation

5.2.2.1 __init__()

```
def dht_crawler.process_output.ProcessOutput.__init__ (
              self,
              monitor,
              country_print,
              country )
```

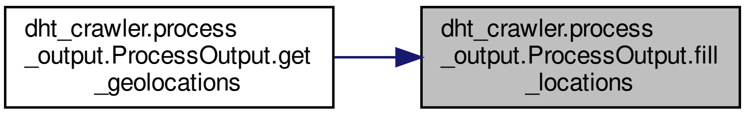
5.2.3 Member Function Documentation

5.2.3.1 fill_locations()

```
def dht_crawler.process_output.ProcessOutput.fill_locations (
              self,
              location_info,
              iplist = None )
```

```
fill various information to dictionary
```

Here is the caller graph for this function:

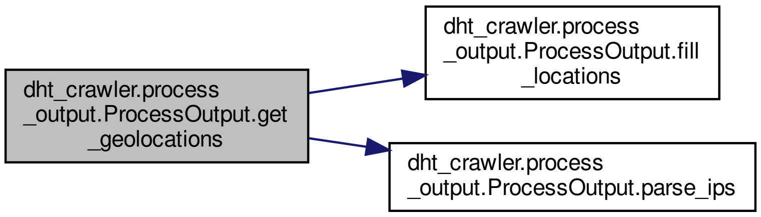

dht_crawler.process _output.ProcessOutput.get _geolocations
dht_crawler.process _output.ProcessOutput.fill _locations

5.2.3.2 get_geolocations()

```
def dht_crawler.process_output.ProcessOutput.get_geolocations (
              self )
```

```
get real locations of ip addresses
```

Here is the call graph for this function:

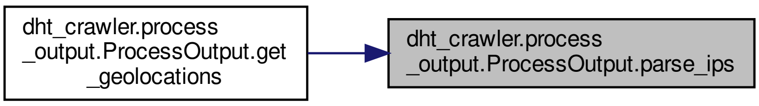

dht_crawler.process _output.ProcessOutput.get _geolocations
dht_crawler.process _output.ProcessOutput.fill _locations
dht_crawler.process _output.ProcessOutput.parse_ips

5.2.3.3 parse_ips()

```
def dht_crawler.process_output.ProcessOutput.parse_ips (
              self )
```

```
initiate pool of ip addresses port and infohashes for geolocation.
```

Here is the caller graph for this function:


dht_crawler.process _output.ProcessOutput.get _geolocations
dht_crawler.process _output.ProcessOutput.parse_ips

5.2.3.4 print_chosen_output()

```
def dht_crawler.process_output.ProcessOutput.print_chosen_output (
              self )
```

```
print geolocation when argument --print_as_country is specified, else
print as json object with no resolution.
```

5.2.3.5 translate_node()

```
def dht_crawler.process_output.ProcessOutput.translate_node (
              self,
              nodes )
```

```
translates nodes ip address to equivalent country name, check wether
they corelate with name given as parameter, returns adjusted ip
addresses which should be deleted from `nodes` list.
```

5.2.4 Member Data Documentation

5.2.4.1 country_city

```
dht_crawler.process_output.ProcessOutput.country_city
```

5.2.4.2 country_name

```
dht_crawler.process_output.ProcessOutput.country_name
```

5.2.4.3 db_format

```
dht_crawler.process_output.ProcessOutput.db_format
```

5.2.4.4 monitor

```
dht_crawler.process_output.ProcessOutput.monitor
```

5.2.4.5 pools

```
dht_crawler.process_output.ProcessOutput.pools
```

5.2.4.6 print_country

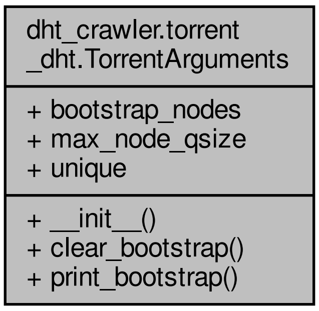
```
dht_crawler.process_output.ProcessOutput.print_country
```

The documentation for this class was generated from the following file:

- dht_crawler/process_output.py

5.3 dht_crawler.torrent_dht.TorrentArguments Class Reference

Collaboration diagram for dht_crawler.torrent_dht.TorrentArguments:



Public Member Functions

- def __init__ (self, bootstrap_nodes=None, max_node_qsize=200)
- def clear_bootstrap (self)
- def print_bootstrap (self)

Public Attributes

- bootstrap_nodes
- max_node_qsize
- unique

5.3.1 Detailed Description

```
bootstrap arguments for DHT class
```

5.3.2 Constructor & Destructor Documentation

5.3.2.1 __init__()

```
def dht_crawler.torrent_dht.TorrentArguments.__init__ (
              self,
              bootstrap_nodes = None,
              max_node_qsize = 200 )
```

5.3.3 Member Function Documentation

5.3.3.1 clear_bootstrap()

```
def dht_crawler.torrent_dht.TorrentArguments.clear_bootstrap (
              self )
```

clear

5.3.3.2 print_bootstrap()

```
def dht_crawler.torrent_dht.TorrentArguments.print_bootstrap (
              self )
```

print

5.3.4 Member Data Documentation

5.3.4.1 bootstrap_nodes

`dht_crawler.torrent_dht.TorrentArguments.bootstrap_nodes`

5.3.4.2 max_node_qsize

`dht_crawler.torrent_dht.TorrentArguments.max_node_qsize`

5.3.4.3 unique

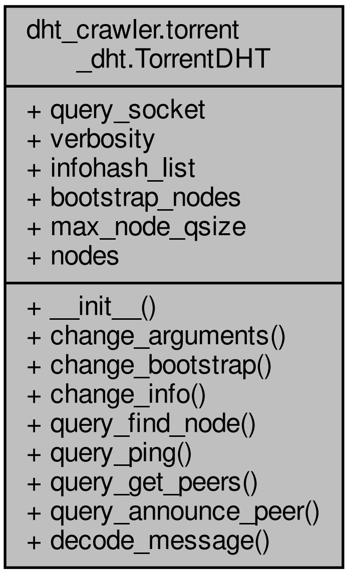
`dht_crawler.torrent_dht.TorrentArguments.unique`

The documentation for this class was generated from the following file:

- dht_crawler/torrent_dht.py

5.4 dht_crawler.torrent_dht.TorrentDHT Class Reference

Collaboration diagram for dht_crawler.torrent_dht.TorrentDHT:



Public Member Functions

- def __init__ (self, arguments, bind_port=6882, verbosity=False)
- def change_arguments (self, length, queue_type)
- def change_bootstrap (self, infohash, nodes, queue_type)
- def change_info (self, infohash)
- def query_find_node (self, node, sock)
- def query_ping (self, node, sock, infohash=None)
- def query_get_peers (self, node, infohash, sock)
- def query_announce_peer (self, node, infohash, port, sock)
- def decode_message (self, msg, info_pool, peer_announce=None, addr=None)

Public Attributes

- query_socket
- verbosity
- infohash_list
- bootstrap_nodes
- max_node_qsize
- nodes

5.4.1 Detailed Description

```
Class which perform query and response of dht messages.
```

5.4.2 Constructor & Destructor Documentation

5.4.2.1 __init__()

```
def dht_crawler.torrent_dht.TorrentDHT.__init__ (
              self,
              arguments,
              bind_port = 6882,
              verbosity = False )
```

5.4.3 Member Function Documentation

5.4.3.1 change_arguments()

```
def dht_crawler.torrent_dht.TorrentDHT.change_arguments (
              self,
              length,
              queue_type )
```

```
change class arguments
```

5.4.3.2 change_bootstrap()

```
def dht_crawler.torrent_dht.TorrentDHT.change_bootstrap (
              self,
              infohash,
              nodes,
              queue_type )
```

```
change bootstrap nodes when parsed magnet-link or .torrent file
```

5.4.3.3 change_info()

```
def dht_crawler.torrent_dht.TorrentDHT.change_info (
              self,
              infohash )
```

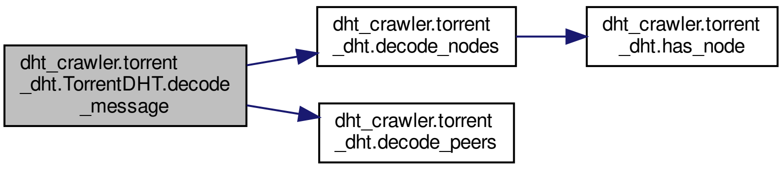
```
change infohash in nodes queue
```

5.4.3.4 decode_message()

```
def dht_crawler.torrent_dht.TorrentDHT.decode_message (
              self,
              msg,
              info_pool,
              peer_announce = None,
              addr = None )
```

```
decodes response message. When nodes decode nodes when peers
decode peers and return them as result.
```

Here is the call graph for this function:

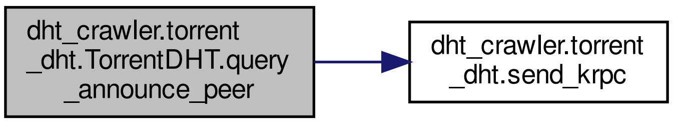

dht_crawler.torrent _dht.TorrentDHT.decode _message
dht_crawler.torrent _dht.decode_nodes
dht_crawler.torrent _dht.has_node
dht_crawler.torrent _dht.decode_peers

5.4.3.5 query_announce_peer()

```
def dht_crawler.torrent_dht.TorrentDHT.query_announce_peer (
              self,
              node,
              infohash,
              port,
              sock )
```

```
send announce_peer query
```

Here is the call graph for this function:

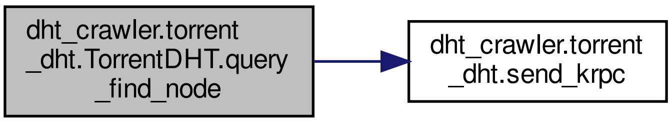

dht_crawler.torrent _dht.TorrentDHT.query _announce_peer
dht_crawler.torrent _dht.send_krpc

5.4.3.6 query_find_node()

```
def dht_crawler.torrent_dht.TorrentDHT.query_find_node (
              self,
              node,
              sock )
```

```
send query find_node to node with our infohash
```

Here is the call graph for this function:

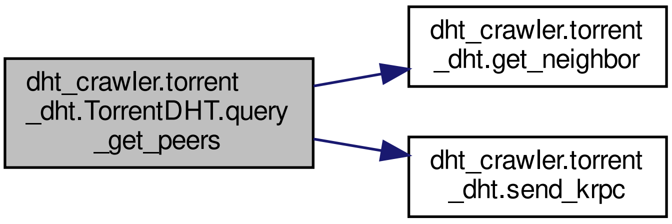

dht_crawler.torrent_dht.TorrentDHT.query_find_node
dht_crawler.torrent_dht.send_krpc

5.4.3.7 query_get_peers()

```
def dht_crawler.torrent_dht.TorrentDHT.query_get_peers (
              self,
              node,
              infohash,
              sock )
```

```
send simple get_peers with our infohash to node
```

Here is the call graph for this function:

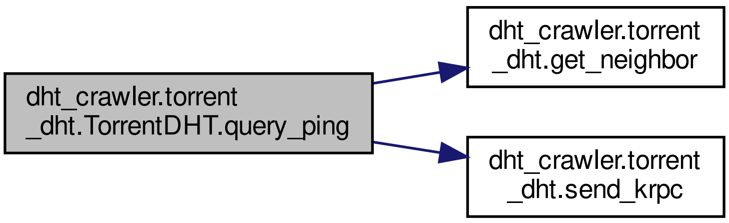

dht_crawler.torrent_dht.TorrentDHT.query_get_peers
dht_crawler.torrent_dht.get_neighbor
dht_crawler.torrent_dht.send_krpc

5.4.3.8 query_ping()

```
def dht_crawler.torrent_dht.TorrentDHT.query_ping (
              self,
              node,
              sock,
              infohash = None )
```

```
send query ping to node
```

Here is the call graph for this function:


dht_crawler.torrent_dht.TorrentDHT.query_ping
dht_crawler.torrent_dht.get_neighbor
dht_crawler.torrent_dht.send_krpc

5.4.4 Member Data Documentation

5.4.4.1 bootstrap_nodes

```
dht_crawler.torrent_dht.TorrentDHT.bootstrap_nodes
```

5.4.4.2 infohash_list

```
dht_crawler.torrent_dht.TorrentDHT.infohash_list
```

5.4.4.3 max_node_qsize

```
dht_crawler.torrent_dht.TorrentDHT.max_node_qsize
```

5.4.4.4 nodes

```
dht_crawler.torrent_dht.TorrentDHT.nodes
```

5.4.4.5 query_socket

```
dht_crawler.torrent_dht.TorrentDHT.query_socket
```

5.4.4.6 verbosity

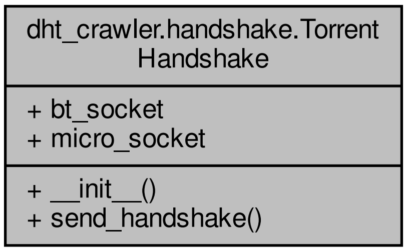
```
dht_crawler.torrent_dht.TorrentDHT.verbosity
```

The documentation for this class was generated from the following file:

- dht_crawler/torrent_dht.py

5.5 dht_crawler.handshake.TorrentHandshake Class Reference

Collaboration diagram for dht_crawler.handshake.TorrentHandshake:



Public Member Functions

- def __init__ (self, port)
- def send_handshake (self, peer)

Public Attributes

- bt_socket
- micro_socket

5.5.1 Detailed Description

```
Torrent handshake class to process initial handshake with peer to prove
its connectivity and filter it from peer pool if non respondend.
```

5.5.2 Constructor & Destructor Documentation

5.5.2.1 __init__()

```
def dht_crawler.handshake.TorrentHandshake.__init__ (
              self,
              port )
```

5.5.3 Member Function Documentation

5.5.3.1 send_handshake()

```
def dht_crawler.handshake.TorrentHandshake.send_handshake (
              self,
              peer )
```

```
send handshake message for bitTorrent connection
```

5.5.4 Member Data Documentation

5.5.4.1 bt_socket

```
dht_crawler.handshake.TorrentHandshake.bt_socket
```

5.5.4.2 micro_socket

```
dht_crawler.handshake.TorrentHandshake.micro_socket
```

The documentation for this class was generated from the following file:

- dht_crawler/handshake.py

Chapter 6

File Documentation

6.1 dht_crawler/__init__.py File Reference

Namespaces

- dht_crawler

6.2 dht_crawler/arg_parse.py File Reference

Namespaces

- dht_crawler.arg_parse

Functions

- def dht_crawler.arg_parse.argument_parser ()
- def dht_crawler.arg_parse.parse_input_args ()

6.3 dht_crawler/exec.py File Reference

Namespaces

- dht_crawler.exec

Variables

- dht_crawler.exec.CRAWL = create_monitor(False)

6.4 dht_crawler/handshake.py File Reference

Classes

- class dht_crawler.handshake.TorrentHandshake

Namespaces

- dht_crawler.handshake

6.5 dht_crawler/monitor.py File Reference

Classes

- class dht_crawler.monitor.Monitor

Namespaces

- dht_crawler.monitor

Functions

- def dht_crawler.monitor.kill_sender_reciever (thread1, thread2=None)
- def dht_crawler.monitor.init_socket (port)
- def dht_crawler.monitor.create_monitor (verbosity=False)

6.6 dht_crawler/process_output.py File Reference

Classes

- class dht_crawler.process_output.ProcessOutput

Namespaces

- dht_crawler.process_output

6.7 dht_crawler/torrent_dht.py File Reference

Classes

- class dht_crawler.torrent_dht.TorrentArguments
- class dht_crawler.torrent_dht.TorrentDHT

Namespaces

- dht_crawler.torrent_dht

Functions

- def dht_crawler.torrent_dht.entropy (length)
- def dht_crawler.torrent_dht.random_infohash ()
- def dht_crawler.torrent_dht.get_neighbor (target, infohash, end=10)
- def dht_crawler.torrent_dht.has_node (id_node, host, port, info_pool)
- def dht_crawler.torrent_dht.decode_krpc (message)
- def dht_crawler.torrent_dht.send_krpc (message, node, sock)
- def dht_crawler.torrent_dht.decode_nodes (value, info_pool)
- def dht_crawler.torrent_dht.decode_peers (infohash, peers, info_pool, token, unique=None)
- def dht_crawler.torrent_dht.get_myip ()

Index