Policy Component Reference Manual

Generated by Doxygen 1.4.7

Thu Sep 16 11:44:01 2010

Contents

1 Policy Component Hierarchical Index

1.1 Policy Component Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Policy Component Class Index

2.1 Policy Component Class List

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

3 Policy Component File Index

3.1 Policy Component File List

Here is a list of all documented files with brief descriptions:

4 Policy Component Class Documentation

4.1 _TransactionLog Struct Reference

`#include <gossip.h>`

Public Member Functions

- **_TransactionLog** (time_t time, boost::uuids::uuid id, const std::string &s)

Public Attributes

- time_t **t**
- boost::uuids::uuid **Tag**
- std::string **sFileName**

4.1.1 Detailed Description

This structure is used to save gossip message according to time so that if any neighbor doesn't receive a particular gossip message, that can be sent to him.

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

- gossip.h

4.2 CBootStrapClient Class Reference

`#include <BootStrapClient.h>`

Public Member Functions

- CBootStrapClient ()
- virtual ~CBootStrapClient ()
- int Download (uint32_t server_ip, uint16_t server_port, const std::string &sDownloadDir)
- EventMessage * **GetEventMessage** () const

Private Types

- typedef std::vector< CFileTransfer * > **FileTransfer**

Private Member Functions

- int **Connect** ()
- int **InitSocketPair** ()
- int HandleNotification (int)

Private Attributes

- uint32_t **m_uServerIP**
- uint16_t **m_uServerPort**
- std::string **m_sDownloadDir**
- int **m_iSockPairFd** [2]
- FileTransfer **m_filetrans**
- EventMessage * **m_pEventMsg**

4.2.1 Detailed Description

A class for communicating with Bootstrap server.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 CBootStrapClient::CBootStrapClient ()

Constructor of the class.

4.2.2.2 CBootStrapClient::~CBootStrapClient () `[virtual]`

Destructor of the class.

4.2.3 Member Function Documentation

4.2.3.1 int CBootStrapClient::Download (uint32_t *server_ip*, uint16_t *server_port*, const std::string & *sDownloadDir*)

Connects with the server for getting bootstraping information. Waits for notification from the CFileTrnafer.

4.2.3.2 int CBootStrapClient::HandleNotification (int) [private]

When CFileTransfer notifies, the notification is handled by this function.

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

- BootStrapClient.h
- BootStrapClient.cpp

4.3 CFileTransfer Class Reference

`#include <FileTransfer.h>`

Public Member Functions

- CFileTransfer (int iSockFd, const std::string &sSharedDir, const std::string &sDownloadDir, int iSearchID=-1)
- virtual ~CFileTransfer ()
- int **GetSockFd** ()
- std::string **GetFileName** ()
- int **GetFileSize** ()
- int **GetSearchID** ()
- std::string **GetSharedDir** ()
- std::string **GetDownloadDir** ()
- void **Cleanup** ()
- void **CloseSocket** ()
- void SetFileHash (const CMD4Hash &fHash)
- CMD4Hash **GetFileHash** ()
- void **SetSocketPair** (int iFd)
- int GetSocketPair ()
- void NotifyComplete (bool bSuccess)
- int HandShake (bool bPassive, bool bPush=false, const std::string &sFileName=std::string(), int iFileSize=0)
- int **HandShake** (bool bPassive, bool bPush, int iSockFd)
- uint8_t **GetFileType** ()
- void **SetFilePath** (const std::string &sPath)
- bool **IsCompleted** ()
- const std::string **GetExtraData** ()
- int StartFileTransfer ()

Static Public Member Functions

- static int ConnectPeer (uint32_t iRemoteIP, uint16_t iRemotePort)

Private Member Functions

- int **SendPacket** (int iSockFD, const char *szPayload, int iSize)
- int ParseHandShakePacket (const char *szBuffer, int iReceived, bool &bPull)
- int WaitForHandShakePacket (bool &bPull, bool bWaitForAck=false)
- int WaitForHandShakePacketAck ()

- int SendHandShakePacket (bool bPush)
- int **StartFileTransferThread** ()
- void ReceiveFile ()
- void SendFile ()

Static Private Member Functions

- static void * **FileTransferThread** (void *arg)

Private Attributes

- int **m_iSockFd**
- std::string **m_sFileName**
- std::string **m_sFilePath**
- int **m_iFileSize**
- pthread_t **m_pTID**
- int **m_iSearchID**
- std::string **m_sSharedDir**
- std::string **m_sDownloadDir**
- CMD4Hash **m_fileHash**
- int **m_iSocketPairFd**
- uint8_t **m_8FileType**
- std::string **m_sExtraData**
- bool **m_bCompleted**
- bool **m_bPassive**
- bool **m_bPush**

4.3.1 Detailed Description

CFileTransfer class. It is used to make any file transfer between two nodes over TCP. This class is used for different file tranfer such as: Kad file tranfer, gossip file transfer, missing gossip file transfer, pollution attack robustness file trnafer etc.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CFileTransfer::CFileTransfer (int *iSockFd*, const std::string & *sSharedDir*, const std::string & *sDownloadDir*, int *iSearchID* = -1)

Constructor. It takes the socket as its argument. Because, before creating an object, initiator node needs to call static function ConnectPeer function. If that is successful, it returns the socket fd, which is used in this argument. SharedDir is used to download certificate files which have been searched from p2p KAD network. Download directory is used to download Gossip Messages.

4.3.2.2 CFileTransfer::~CFileTransfer () [virtual]

Destructor of the class.

4.3.3 Member Function Documentation

4.3.3.1 int CFileTransfer::ConnectPeer (uint32_t *iRemoteIP*, uint16_t *iRemotePort*) [static]

A static function to connect peer using iRemoteIP and iRemotePort. If the connectionis successful, then the socket fd is returned, else -1 is returned.

4.3.3.2 int CFileTransfer::GetSocketPair () [inline]

A socket pair is used to communicate between sender/receiver threads and the main thread.

4.3.3.3 int CFileTransfer::HandShake (bool *bPassive*, bool *bPush* = false, const std::string & *sFileName* = std::string(), int *iFileSize* = 0)

File tranfer is used both gossiping (push based) and certificate download using kademlia (pull based). So, to make the sender and receiver dynamic, this function is used instead of static sender or receiver. HandShake funtion sends EventMessage to the peer. After handshaking, sender and receiver becomes fixed.

4.3.3.4 void CFileTransfer::NotifyComplete (bool *bSuccess*)

Sends an EventMessage to the network thread after file tranfer is done.

4.3.3.5 int CFileTransfer::ParseHandShakePacket (const char * *szBuffer*, int *iReceived*, bool & *bPull*) [private]

Parses the Handshake EventMessage to figure out who is sender, who is receiver of a particular file.

4.3.3.6 void CFileTransfer::ReceiveFile () [private]

Receives a file.

4.3.3.7 void CFileTransfer::SendFile () [private]

Sends a file.

4.3.3.8 int CFileTransfer::SendHandShakePacket (bool *bPush*) [private]

Sends the handshake packet.

4.3.3.9 void CFileTransfer::SetFileHash (const CMD4Hash & *fHash*) [inline]

File Hash is stored so that later the corresponding object can be retrived.

4.3.3.10 int CFileTransfer::StartFileTransfer ()

Starts the file tranfer thread.

4.3.3.11 int CFileTransfer::WaitForHandShakePacket (bool & *bPull*, bool *bWaitForAck* = false) [private]

Waits for handshake packet in the poll.

4.3.3.12 int CFileTransfer::WaitForHandShakePacketAck () [private]

Waits for handshake packet Acknowledgement in the poll.

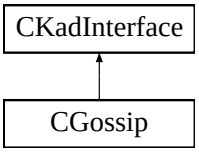
The documentation for this class was generated from the following files:

- FileTransfer.h
- FileTransfer.cpp

4.4 CGossip Class Reference

`#include <gossip.h>`

Inheritance diagram for CGossip::



Public Member Functions

- virtual ~CGossip ()
- void NotifyApp (const std::string &sFileName, uint8_t filetype, uint8_t relatedModule)
- int GetTransactionLog (std::string &transList) const
- int FindMissingGossip (const std::string &sTransList, std::string &sMissingList) const
- int SendMissingGossip (const std::string &sMissingList, uint32_t ip, uint16_t port)

Static Public Member Functions

- static CGossip * getGossipInstance (std::string)

Private Types

- typedef std::deque< CFileTransfer * > **FileTransfer**

Private Member Functions

- CGossip (std::string sFileName)
- void ProcessGossipPacket (const GossipPacket &gPack, uint32 ip, uint16 port, size_t length)
- int SendGossipPacket (const GossipPacket &packet, int pktLen, uint32_t destinationHost, uint16_t destinationPort)
- void OnPacketReceived (uint32 ip, uint16 port, byte *buffer, size_t length)
- int OnCommandReceived (int iFd)
- int SendCommand (const GossipPacket &gPack, int len, uint32_t srcIP, uint16_t srcPort)
- int SendTcpCommand (const std::string &sFileToSend, const boost::uuids::uuid &Tag, uint32_t srcIP, uint16_t srcPort)
- int SendCommand (const std::string &sCommand, bool bTcp=false, uint8_t module=MODULE_GOSSIP)

- int NotifyCommand (const std::string &)
- int GetTransactionID (const std::string &sFile, boost::uuids::uuid &id)
- int SendTcpGossipPacket (const std::string &sFileName, uint32_t destinationHost, uint16_t destinationPort)
- void ProcessTcpGossipPacket (const std::string &sFileName, uint32 ip, uint16 port, uint8_t module)
- std::string **CopyFileToTmpDir** (const std::string &sFile)
- int GetKadBootstrapInfo (int iServerIP, int iPort)
- int DoTransactionLog (const boost::uuids::uuid &Tag, const std::string &sFileName)
- std::string **BackupFileToDownloadDir** (const std::string &sFile, const boost::uuids::uuid &Tag)
- int SendMissingGossip (const boost::uuids::uuid &Tag, uint32_t ip, uint16_t port)

Private Attributes

- MapFwHist **mFwHist**
- MapFwHist **mRecvHist**
- FileTransfer **m_gfiletrans**
- CBootStrapClient * **m_pBootStrapClient**
- std::deque< TransactionLog > **m_qTransactionLog**

Static Private Attributes

- static bool **bInstance** = false
- static CGossip * **instance** = NULL

4.4.1 Detailed Description

CGossip class. It extends functionality of CKadInterface class since the gossiping works on top of kademlia protocol.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CGossip::~CGossip () `[virtual]`

CGossip Destructor.

4.4.2.2 CGossip::CGossip (std::string *sFileName*) `[private]`

CGossip Constructor; Takes configuration file name as its argument. It is private due to make the class as a singleton pattern.

4.4.3 Member Function Documentation

4.4.3.1 int CGossip::DoTransactionLog (const boost::uuids::uuid & *Tag*, const std::string & *sFileName*) `[private]`

After receiving or sending any TCP Gossip message, the message id, time and the file name is saved so that id list can be sent to the neighbor in PING message.

4.4.3.2 int CGossip::FindMissingGossip (const std::string & *sTransList*, std::string & *sMissingList*) const

From the KAD PING message, this method is used to find if the node missed any gossip message.

4.4.3.3 CGossip * CGossip::getGossipInstance (std::string) [static]

This method is used to get an instance of CGossip class. It is used to make this architecture as a singleton pattern.

4.4.3.4 int CGossip::GetKadBootstrapInfo (int *iServerIP*, int *iPort*) [private]

Invokes Bootstrap Client for getting bootstrap IP and port.

4.4.3.5 int CGossip::GetTransactionID (const std::string & *sFile*, boost::uuids::uuid & *id*) [private]

Get the transaction id as UUID from sFile. sFile is supposed to be in SMIME format and id should be found in "FROM:" field.

4.4.3.6 int CGossip::GetTransactionLog (std::string & *transList*) const

It is used to get the log of already received Gossip Message Transaction ID to attached in the KAD PING message.

4.4.3.7 void CGossip::NotifyApp (const std::string & *sFileName*, uint8_t *filetype*, uint8_t *relatedModule*) [virtual]

Overrides the functionality of NotifyApp of CKadInterface to send EventMessage to the Policy Component.

Reimplemented from CKadInterface.

4.4.3.8 int CGossip::NotifyCommand (const std::string &) [private]

This actually sends data to the Policy Component.

4.4.3.9 int CGossip::OnCommandReceived (int *iFd*) [private, virtual]

This method is invoked after receiving EventMessage from Policy Component.

Reimplemented from CKadInterface.

4.4.3.10 void CGossip::OnPacketReceived (uint32 *ip*, uint16 *port*, byte * *buffer*, size_t *length*) [private]

This method is invoked after receiving any Kademlia message.

4.4.3.11 void CGossip::ProcessGossipPacket (const GossipPacket & *gPack*, uint32 *ip*, uint16 *port*, size_t *length*) [private]

Process any received UDP Gossip Message.

4.4.3.12 void CGossip::ProcessTcpGossipPacket (const std::string & *sFileName*, uint32 *ip*, uint16 *port*, uint8_t *module*) [private, virtual]

Process any received TCP Gossip Message.

Reimplemented from CKadInterface.

4.4.3.13 int CGossip::SendCommand (const std::string & *sCommand*, bool *bTcp* = false, uint8_t *module* = MODULE_GOSSIP) [private]

Sends a gossip message in the network.

4.4.3.14 int CGossip::SendCommand (const GossipPacket & *gPack*, int *len*, uint32_t *srcIP*, uint16_t *srcPort*) [private]

It sends a UDP Gossip Message.

4.4.3.15 int CGossip::SendGossipPacket (const GossipPacket & *packet*, int *pktLen*, uint32_t *destinationHost*, uint16_t *destinationPort*) [private]

Send a gossip message using UDP protocol.

4.4.3.16 int CGossip::SendMissingGossip (const boost::uuids::uuid & *Tag*, uint32_t *ip*, uint16_t *port*) [private]

If a neighbor sends a gossip message ID in PONG message, the message is sent to the neighbor.

4.4.3.17 int CGossip::SendMissingGossip (const std::string & *sMissingList*, uint32_t *ip*, uint16_t *port*)

If any missing Gossip Message is found, the transaction is written in the KAD PONG message to retrieve it.

4.4.3.18 int CGossip::SendTcpCommand (const std::string & *sFileToSend*, const boost::uuids::uuid & *Tag*, uint32_t *srcIP*, uint16_t *srcPort*) [private]

It sends a TCP Gossip Message. Tag is used as a unique ID for this gossip message.

4.4.3.19 int CGossip::SendTcpGossipPacket (const std::string & *sFileName*, uint32_t *destinationHost*, uint16_t *destinationPort*) [private]

Sends a gossip message over TCP transport protocol.

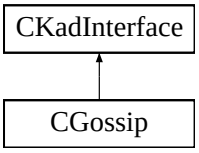
The documentation for this class was generated from the following files:

- gossip.h
- gossip.cpp

4.5 CKadInterface Class Reference

`#include <kademlia_interface.h>`

Inheritance diagram for CKadInterface::



Public Types

- typedef std::deque< CFileTransfer * > **FileTransfer**

Public Member Functions

- CKadInterface (std::string sFilename)
- virtual ~CKadInterface ()
- int Run ()
- void StartKad ()
- void StopKad ()
- void BootstrapKad ()
- void GetAllContacts ()
- void **FindNodeByID** (const uint8_t *nodeID)
- uint32_t **GetPublicIP** (bool ignorelocal)
- int **GetSockFd** ()
- int **GetDaemonSockFd** ()
- int **GetClientTcpSockFd** ()
- int **GetPeerSockFd** ()
- void **SetPeerSockFd** (int iFd)
- void CleanUp ()
- void SearchSrcKad (CSearchFile *toadd)
- void KademliaSearchFile (uint32_t searchID, const Kademlia::CUInt128 *pcontactID, const Kademlia::CUInt128 *pbuddyID, uint8_t type, uint32_t ip, uint16_t tcp, uint16_t udp, uint32_t buddyip, uint16_t buddyport, uint8_t byCryptOptions)
- CPartFile * **GetFileByID** (const CMD4Hash &filehash) const
- std::string GetSharedDir ()
- std::string GetDownloadDir ()
- const int * **GetSocketPair** ()
- void PublishFile (const std::string &sFile)
- void SearchKeyKad (const std::string &sSearchStr)
- void AddSource (uint32_t searchID, uint32_t ip, uint16_t port)
- int CheckFileTransferStatus (int iFd)
- std::string **MakeAbsPath** (const std::string &sPath)
- void SetDownloadDir (const std::string &sPath)
- CMulticast * GetMulticastInstance ()
- bool IsGossipEnable ()
- bool IsMulticastEnable ()
- void BootstrapKad (uint32_t ip, uint16_t port)

Public Attributes

- FileTransfer **m_filetrans**
- CSharedFileList * **sharedfiles**
- CSearchList * **searchlist**

Private Types

- typedef std::deque< CPartFile * > **FileQueue**

Private Member Functions

- void StartNetworkThread ()
- void **SetLogger** ()
- int CreateSocket ()
- int CreateDaemonSocket ()
- int CreateClientTcpSocket ()
- void **CheckEvent** (int iFd)
- int **CheckDaemonEvent** (int iFd)
- void **DestroySocket** ()
- int ParseConfig ()
- std::string **GetString** (const std::vector< std::string > &vConf, const std::string &sLine)
- void **StrTrim** (std::string &) const
- void OnTimer ()
- void **SetSharedDir** (const std::string &sPath)
- virtual void OnPacketReceived (uint32_t ip, uint16_t port, byte *buffer, size_t length)
- virtual int OnCommandReceived (int iFd)
- virtual void NotifyApp (const std::string &sFileName, uint8_t filetype, uint8_t relatedModule)
- virtual void ProcessTcpGossipPacket (const std::string &sFileName, uint32 ip, uint16 port, uint8_t module)
- CPartFile * **GetFileByKadFileSearchID** (uint32 id) const
- CFileTransfer * **CheckFileTransByID** (const int iSearchID) const
- CFileTransfer * **CheckFileTransByHash** (const CMD4Hash &fHash) const
- int **InitSocketPair** ()
- void RetrySource (int iSearchID)
- void MakeDirs ()
- void **ShutdownNetworkThread** ()

Static Private Member Functions

- static void * **ThreadCallback** (void *arg)

Private Attributes

- int **m_iSockFd**
- int **m_iDaemonSockFd**
- int **m_iPeerSockFd**
- int **m_iClientTcpSockFd**
- std::string **m_sConfigFileName**
- std::string **m_sClientName**
- bool **m_bBootstrapEnable**
- bool **m_bKadLogEnable**
- uint32_t **m_uClientIP**
- uint16_t **m_uClientPort**
- uint32_t **m_uBootIP**
- uint16_t **m_uBootPort**

- pthread_t **m_Thread**
- uint16_t **m_uDaemonTcpPort**
- uint16_t **m_uClientTcpPort**
- CKnownFileList * **knownfiles**
- FileQueue **m_filelist**
- std::string **m_sSharedDir**
- std::string **m_sDownloadDir**
- int **m_iSockPairFd** [2]
- std::map< int, std::vector< AddrPortPair > > **m_SourceMap**
- bool **m_bMulticastEnable**
- bool **m_bGossipEnable**
- std::string **m_sMulticastInterface**
- uint32_t **m_uMulticastIP**
- uint16_t **m_uMulticastPort**
- CMulticast * **m_pMulticast**
- bool **m_bInit**

4.5.1 Detailed Description

Placement of Communication component in TCP/IP stack

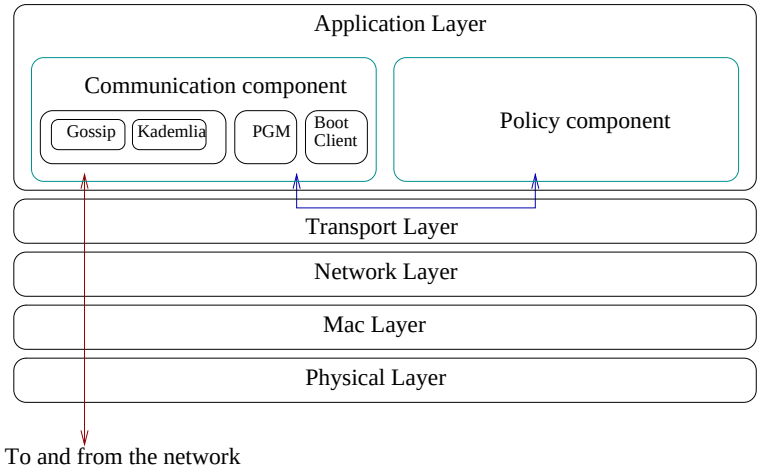


Figure 1: C2 System Architecture

4.5.2 Constructor & Destructor Documentation

4.5.2.1 CKadInterface::CKadInterface (std::string *sFilename*)

Constructor of the class. Takes configuration file name of the communication component as its argument. An example of such configuration file is:

```
#Name is not important; should be removed later
client_name = client_2

#IP address of the node
client_ip = 130.55.115.201
```

```
#Kademlia UDP port
client_port =  4672

#Kademlia TCP port
client_tcp_port = 5672

#Kademlia logging should enabled?
kad_log_enable = yes

#Port on which Communication component listens for Policy component.
daemon_tcp_port = 7772

#Should enable bootstrap?
#This are only for testing since bootstrapping is done
#by policy component now.
bootstrap_enable = yes
bootstrap_ip = 130.55.115.201
bootstrap_port =  4671


#Name of the directory where communication component stores the certificates
#collected from the p2p network and republishes.
shared_dir = shared_dir

#Name of the directory where communication component stores all gossip
#messages.
download_dir = download_dir

#Is multicast module enabled?
multicast_enable = yes

#Multicast interface name
multicast_interface = eth0

#Multicast IP address
multicast_ip = 239.192.0.1

#Multicast Port number
multicast_port = 7500

#Is Kademlia module enabled?
gossip_enable = yes
```

4.5.2.2 CKadInterface::~CKadInterface () [virtual]

Destructor of the class.

4.5.3 Member Function Documentation

4.5.3.1 void CKadInterface::AddSource (uint32_t *searchID*, uint32_t *ip*, uint16_t *port*)

Stores the source ip and port for a specific kad search ID.

4.5.3.2 void CKadInterface::BootstrapKad (uint32_t *ip*, uint16_t *port*)

Asks kademlia to bootstrap the specified ip and port.

4.5.3.3 void CKadInterface::BootstrapKad ()

Asks kademlia to bootstrap on some specific ip and port in kad network.

4.5.3.4 int CKadInterface::CheckFileTransferStatus (int *iFd*)

After file transfer notifies the network thread, this function checks the file transfer status result and invokes corresponding action.

4.5.3.5 void CKadInterface::CleanUp ()

Cleans up any allocated resource such as sockets, threads, etc.

4.5.3.6 int CKadInterface::CreateClientTcpSocket () [private]

Creates a TCP socket for kademlia.

4.5.3.7 int CKadInterface::CreateDaemonSocket () [private]

Creates a TCP socket for communicating with policy component.

4.5.3.8 int CKadInterface::CreateSocket () [private]

Creates a UDP socket for kademlia.

4.5.3.9 void CKadInterface::GetAllContacts ()

Retrieves contact list of neighbor from kademlia protocol. This contact list is useful for gossiping.

4.5.3.10 std::string CKadInterface::GetDownloadDir () [inline]

Retrieves the download directory. Any gossip file is stored in this directory.

4.5.3.11 CMulticast* CKadInterface::GetMulticastInstance () [inline]

Returns the multicasting instance.

4.5.3.12 std::string CKadInterface::GetSharedDir () [inline]

Retrieves the name of the shared directory. Kademlia publishes files who are located in this shared directory. After searching a file in the kad network and downloading it, the file is also stored in this shared directory for further publishing.

4.5.3.13 bool CKadInterface::IsGossipEnable () [inline]

Checks whether the gossip module is enabled or not. In the configurationfile, it is possible to enable or disable gossip module and multicast module.

4.5.3.14 bool CKadInterface::IsMulticastEnable () [inline]

Checks whether the multicast module is enabled or not. In the configurationfile, it is possible to enable or disable gossip module and multicast module.

4.5.3.15 void CKadInterface::KademliaSearchFile (uint32_t *searchID*, const Kademlia::CUInt128 * *pcontactID*, const Kademlia::CUInt128 * *pbuddyID*, uint8_t *type*, uint32_t *ip*, uint16_t *tcp*, uint16_t *udp*, uint32_t *buddyip*, uint16_t *buddyport*, uint8_t *byCryptOptions*)

This function is called after a source of a file is found.

4.5.3.16 void CKadInterface::MakeDirs () [private]

Creates Shared Dir, Download Dir and .tmp Dir.

4.5.3.17 void CKadInterface::NotifyApp (const std::string & *sFileName*, uint8_t *filetype*, uint8_t *relatedModule*) [private, virtual]

It sends EventMessage to the Policy component. sFileName, filetype and relatedModule are encoded in the EventMessage.

Reimplemented in CGossip.

4.5.3.18 virtual int CKadInterface::OnCommandReceived (int *iFd*) [inline, private, virtual]

Invoked after a command from policy component is received. Override by the CGossip class. Communication is done using EventMessage class.

Reimplemented in CGossip.

4.5.3.19 virtual void CKadInterface::OnPacketReceived (uint32_t *ip*, uint16_t *port*, byte * *buffer*, size_t *length*) [private, virtual]

Invoked after a Kademlia packet is arrived.

4.5.3.20 void CKadInterface::OnTimer () [private]

Executes on timeout in the network thread. The interval is 1 sec. Calls Kademlia::Process()

4.5.3.21 int CKadInterface::ParseConfig () [private]

Parse the configuration file. Returns -1 on error in the configuration file.

4.5.3.22 void CKadInterface::ProcessTcpGossipPacket (const std::string & *sFileName*, uint32 *ip*, uint16 *port*, uint8_t *module*) [private, virtual]

After receiving a Gossip Message on TCP, this method is invoked.

Reimplemented in CGossip.

4.5.3.23 void CKadInterface::PublishFile (const std::string & *sFile*)

Asks kademlia to publish the sFile in the KAD network.

4.5.3.24 void CKadInterface::RetrySource (int *iSearchID*) [private]

If a file tranfer is failed for downloading a file from the KAD network, retry is done by invoking this function.

4.5.3.25 int CKadInterface::Run ()

Main function to run the communication component. Returns -1 on failure in running the component. It runs Kademlia, multicast and network modules.

4.5.3.26 void CKadInterface::SearchKeyKad (const std::string & *sSearchStr*)

Asks kademlia to search any keyword in the kad network.

The filename extension. May be empty.

The smallest filesize in bytes to accept, zero for any.

The largest filesize in bytes to accept, zero for any.

4.5.3.27 void CKadInterface::SearchSrcKad (CSearchFile * *toadd*)

Search source of a file in the KAD network.

4.5.3.28 void CKadInterface::SetDownloadDir (const std::string & *sPath*)

Sets the path of the download directory as specified in the configuration file.

4.5.3.29 void CKadInterface::StartKad ()

Asks kademlia to startup.

4.5.3.30 void CKadInterface::StartNetworkThread () [private]

Starts the networking thread. It listens for any : Kademlia message, Gossip file transfer message, socket pair for communicating with the main thread, policy component message

4.5.3.31 void CKadInterface::StopKad ()

Asks kademlia to stop.

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

- kademlia_interface.h
- kademlia_interface.cpp

4.6 CMulticast Class Reference

`#include <multicast.h>`

Public Member Functions

- CMulticast (const std::string &sInterface, uint32_t ip, uint16_t port)
- virtual ~CMulticast ()
- int SendFile (const std::string &sFileName, const std::string &sFileLoc)
- int **SendPacket** (const std::string &s)

Static Public Member Functions

- static void FileReceivedCallback (fileinfo *f)
- static void **SetSocketPair** (int iFd)
- static int **GetSocketPairFd** ()

Private Attributes

- bool **m_bInit**

Static Private Attributes

- static int **m_iFd** = -1

4.6.1 Detailed Description

An interface class for communicating with OpenPGM library.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 CMulticast::CMulticast (const std::string & *sInterface*, uint32_t *ip*, uint16_t *port*)

Constructor of the class. Takes interface name, multicast IP address and port as its argument.

4.6.2.2 CMulticast::~CMulticast () [virtual]

Destructor of the class.

4.6.3 Member Function Documentation

4.6.3.1 void CMulticast::FileReceivedCallback (fileinfo * *f*) [static]

The network thread of the multicast library notifies about any downloaded file.

4.6.3.2 int CMulticast::SendFile (const std::string & *sFileName*, const std::string & *sFileLoc*)

Sends the multicast file in the network.

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

- multicast.h
- multicast.cpp

4.7 gossipPacket Struct Reference

`#include <gossip.h>`

Public Attributes

- uint8_t **m_iProtocol**
- boost::uuids::uuid **m_Tag**
- uint8_t **m_cPack** [MAXCHAR]

4.7.1 Detailed Description

Only used in UDP gossiping.

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

- gossip.h

5 Policy Component File Documentation

5.1 util.h File Reference

```
#include <vector>
```

Typedefs

- typedef std::pair< uint32_t, uint16_t > **AddressPortPair**

Functions

- int GetFileSize (const std::string &sFile)
- std::string GetDotFormatIP (uint32_t ip)
- void reuse (int fd)
- void nonblock (int fd)
- void nodelay (int fd)
- void **ByteOrderTesting** ()
- int RunAsDaemon (const char *stdoutname)
- int write_pid (const char *sFileName)
- AddressPortPair GetSockName (uint32_t sock)
- int StringTokenizer (const std::string sToParse, const char *cToken, std::vector< std::string > &v)

5.1.1 Detailed Description

Utility functions.

5.1.2 Function Documentation

5.1.2.1 std::string GetDotFormatIP (uint32_t *ip*)

Returns the dot formatted IP address from 4 byte ip.

5.1.2.2 int GetFileSize (const std::string & *sFile*)

Returns the size of the sFile in bytes.

5.1.2.3 AddressPortPair GetSockName (uint32_t *sock*)

Retrieves IP and port of s socket sock.

5.1.2.4 void nodelay (int *fd*)

Disables naggles algorithm.

5.1.2.5 void nonblock (int *fd*)

Makes socket fd non-blocking.

5.1.2.6 void reuse (int *fd*)

Makes socket fd reuseable.

5.1.2.7 int RunAsDaemon (const char * *stdoutname*)

Runs current process as daemon. write all output to stdoutname file.

5.1.2.8 int StringTokenizer (const std::string *sToParse*, const char * *cToken*, std::vector< std::string > & *v*)

A special string tokenizer. Used for finding gossip message ID from a list.

5.1.2.9 int write_pid (const char * *sFileName*)

Creates a pid file with the pid number of the daemon.

Index