P2P: databases

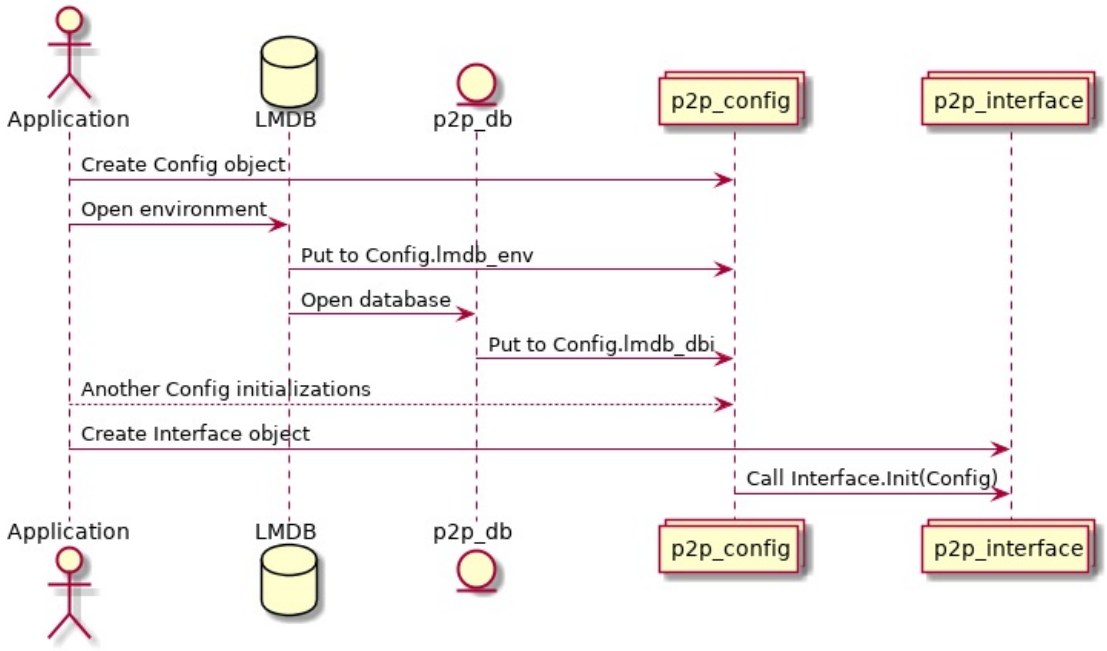
1. Overview

The p2p subsystem contains 2 databases that are available while the logos-core program is running. Also they are periodically saved to disk and read from disk the next time the logos-core program is started. These are databases of well-known p2p network hosts and banned ip addresses and subnets. The p2p_db database in the common LMDB storage of the logos-core program is used for permanent storage of these databases on disk.

2. Initialization

The p2p subsystem itself does not create an LMDB environment and does not open a database in it. This should be done by someone who uses the p2p subsystem. Work with databases is arranged differently in the programs logos_core, the p2p_standalone test application and the unit_test. Each of these applications uses the p2p subsystem, but first itself initializes the database.

When creating an object of class p2p_interface, the structure p2p_config is fed to it. The structure and the class are described in the file logos/p2p/p2p.h. In the p2p_config structure there are 2 fields related to databases. These are lmdb_env and lmdb_dbi. One need to place a pointer to the lmdb environment in the first field, and the pointer to the database in this environment in the second one. For example, a pointer to the database p2p_db.



3. External interfaces

There are 2 external interfaces to the p2p subsystem that deal with databases. This is the high-level interface located in the class ContainerP2p in the file logos/consensus/ p2p/ consensus_p2p.hpp, and the low-level interface in the class p2p_interface in the logos/p2p/p2p.h. The difference between the two is that the high-level interface uses the logos::endpoint type based on the corresponding boost type to represent the network addresses while the low-level interface uses a string representation of the addresses of the type char *.

The high-level interface includes the following functions:

```
int get_peers(int session_id, vector<logos::endpoint> &nodes, uint8_t count);
void close_session(int session_id);
void add_to_blacklist(const logos::endpoint &e);
bool is_blacklisted(const logos::endpoint &e);
```

The low-level interface contains functions:

```
int add_peers(char **nodes, uint8_t count);
int get_peers(int *next, char **nodes, uint8_t count);
void add_to_blacklist(const char *addr);
bool is_blacklisted(const char *addr);
bool load_databases();
bool save_databases();
```

High level functions invoke the corresponding low level functions. The latter, in turn, call functions from the internal class CConnman of the p2p subsystem and the underlying class CAddrMan. The relationship between the functions is shown in the table. Some functions of the low level have no analogues at a high level because they are used mainly in tests.

ContainerP2p	p2p_interface	CConnman	CAddrMan
	add_peers	AddNewAddresses	
get_peers	get_peers		get_peers
close_session			
add_to_blacklist	add_to_blacklist	Ban	
is_blacklisted	is_blacklisted	IsBanned	
	load_databases	LoadData	
	save_databases	DumpData	

The high level interface get_peers() function allows you to get a list of all known hosts for several function calls. These calls are combined into a session. The first time one call get_peers() one must submit -1 as session_id, and the function will return real session_id. For subsequent calls one need to use this session_id. At the end one should call the function close_session(). For each call the maximum number of hosts that the function can return is indicated. Calls related to different sessions may be interspersed with each other.

The low level interface get_peers() function does the same thing, except that the service information about the state of the current session is saved to a variable by pointer next and it is read from there on the next call. The initial value of this variable must be zero.

The add_peers() function allows one to add multiple hosts to the database. In the functions get_peers() and add_peers(), the host address consists of both the IP address and the TCP port number. In the string representation for the low-level interface the IP address and the port are separated by a colon. For example, "80.12.7.55:1432".

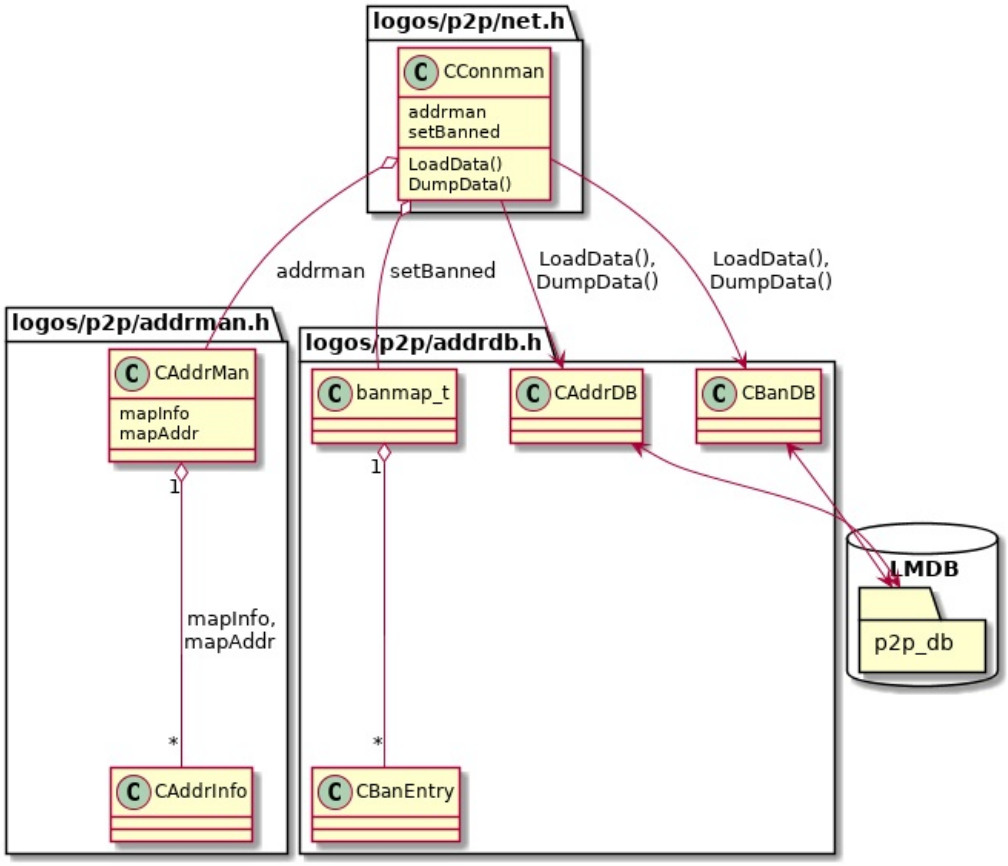
The function add_to_blacklist() of both interfaces allow one to add a host to the black list. The function is_blacklisted() returns true if this host is in the black list. For all functions that work with the black list, the argument is only the IP address of the host without specifying the port. For example, "80.12.7.55".

Finally, the load_databases() function allows one to reload both databases from a file on disk to memory, and the save_databases() function allows one to save both databases from memory to disk. Note that these functions should be used only in tests. During the real work of the program logos_core everything happens automatically. Namely, the databases are once loaded into memory when the p2p subsystem is initialized, and then periodically saved to disk, and also saved when the p2p subsystem completes.

4. Engine components

The p2p databases submodule involves classes located in the files logos/p2p/addrman.(h, cpp) and logos/p2p/addrb.(h, cpp). The first group of files contains the classes CAddrMan and CAddrInfo. The class CAddrMan implements a database of known hosts stored in memory. The class CAddrInfo implements one record in this database. The second group of files contains the class CBanEntry and the type banmap_t based on it which implement the database of banned ip addresses stored in memory. The second group of files also contains the classes CAddrDB and CBanDB which are used to write the corresponding databases to disk and read them from the disk.

The main p2p engine class CConnman implemented in the files logos/p2p/net.(h, cpp) contains the member addrman of the class CAddrMan and the member setBanned of the type banmap_t. The class CConnman carries out ongoing work with databases with the help of these members. The methods DumpData() and LoadData() of the class CConnman are called for writing databases to disk and reading it from disk. These methods in turn create temporary instances of the classes CAddrDB and CBanDB classes and then perform read/write operations with serialized representations of the objects addrman and setBanned.



5. Unit tests

There are two unit tests that check the saving of databases to disk and the operation of interfaces. The first test named VerifyPeersInterface checks the low-level interface and is located in the file logos/p2p/test/storage_test.cpp. The second test is also called VerifyPeersInterface. It checks the high-level interface and is located in the file logos/unit_test/p2p_test.cpp.

Each of these tests makes changes to the database, namely, adds new hosts and new banned hosts. After this, the test verifies that the changes are correctly made to the databases stored in memory. Then these databases are saved to disk, after which another instance of the p2p subsystem is started, the databases are read from the disk and checked again.