Satoshi Bitcoin Client Details

Andrew Stone

Developer, Bitcoin Unlimited

Nov 29, 2017

Major Functional Sections

- Blockchain
- UTXO
- Network
- Wallet
- Mempool

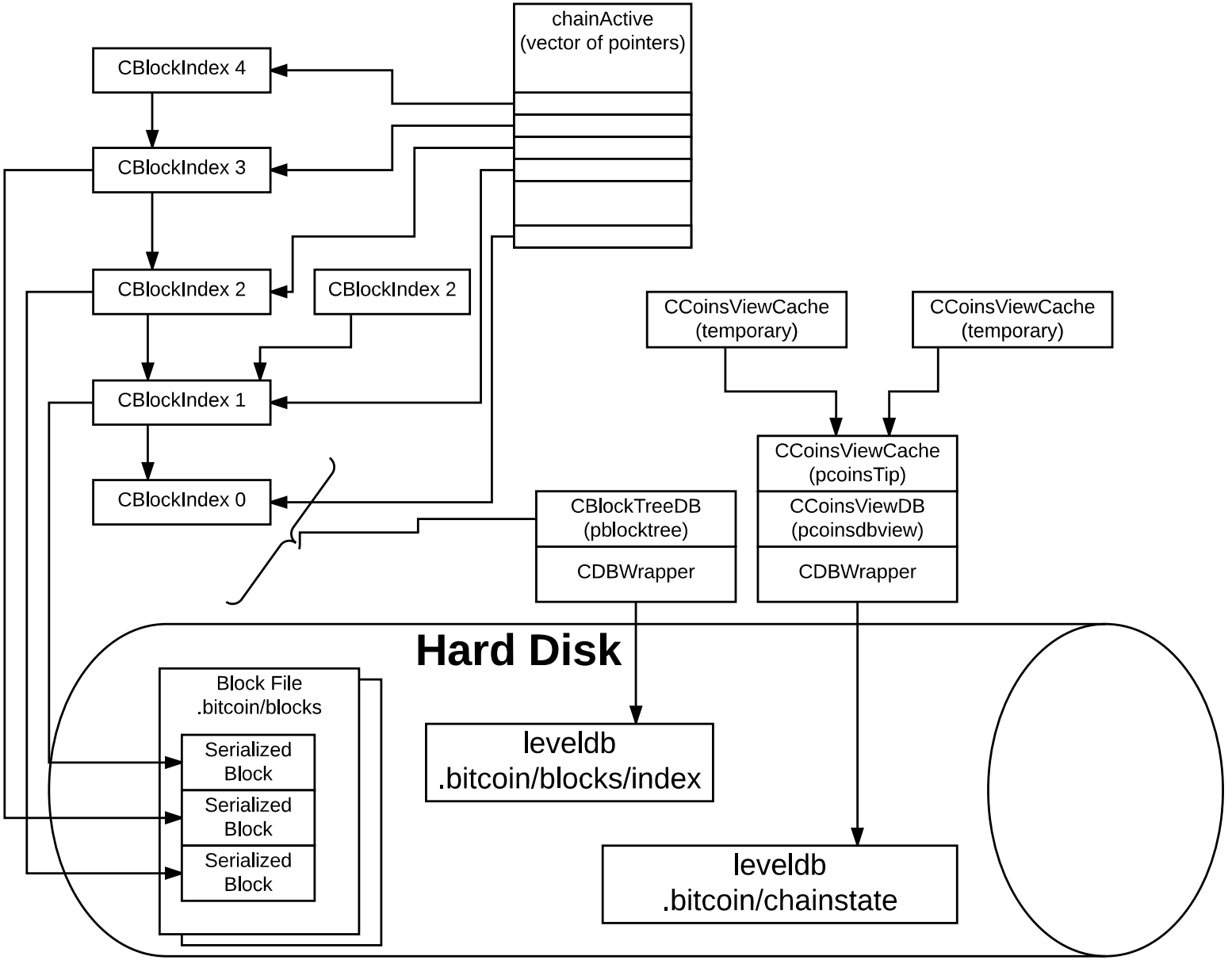
Code Subsections

- Source (src)
  - GUI (src/qt)
  - Wallet (src/wallet)
  - CLI commands (src/rpc)
  - Blockchain data structures (src/primitives)
  - Consensus params (src/consensus)
- Unit test (white-box) (src/test)
  - Based on boost unit test framework
- System Test (black-box) (qa)
  - python
- Reproducible Build
  - https://github.com/devrandom/gitian-builder

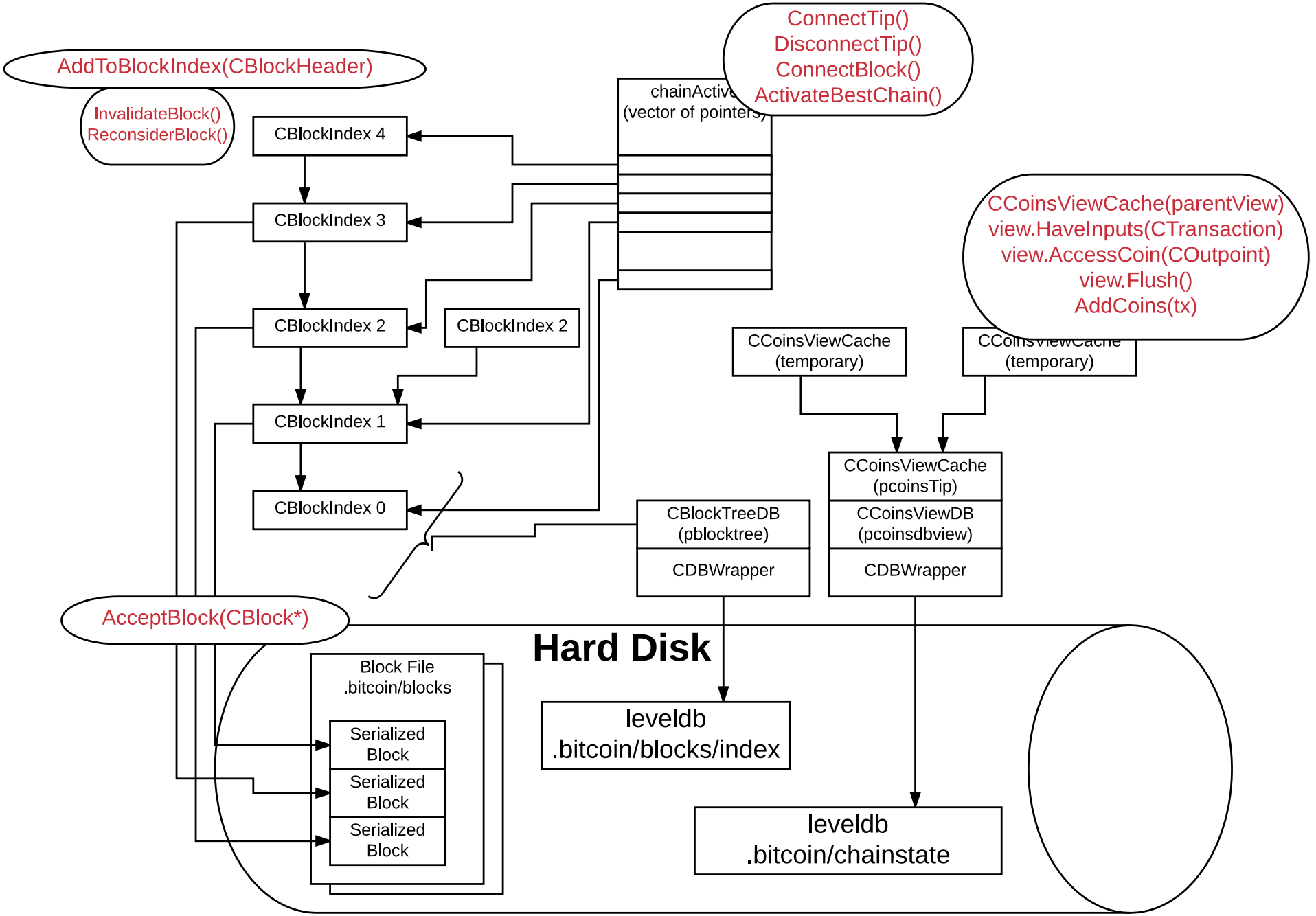
Important Classes

- blockchain:
  - CBlockIndex
    - block header + disk location of block + parent pointer
  - CChain
    - global singleton: chainActive
    - chainActive->Tip() is the latest block
  - CBlockHeader
  - CBlock (primitives/block.h)
  - CTransaction
  - COutPoint (uniquely identifies a coin by tx hash, vout index tuple)
- network:
  - CNode (net.h)
  - CRequestManager (requestManager.h)
- mempool (txmempool.h):
  - CTxMemPool
  - global singleton: mempool
- UTXO set (coins.h, txdb.h):
  - Coin
    - an unspent transaction output (UTXO)
  - CCoinsView (sort-of-abstract base class)
  - CCoinsViewCache
    - cache and local-modification space combined
  - CCoinsViewDb

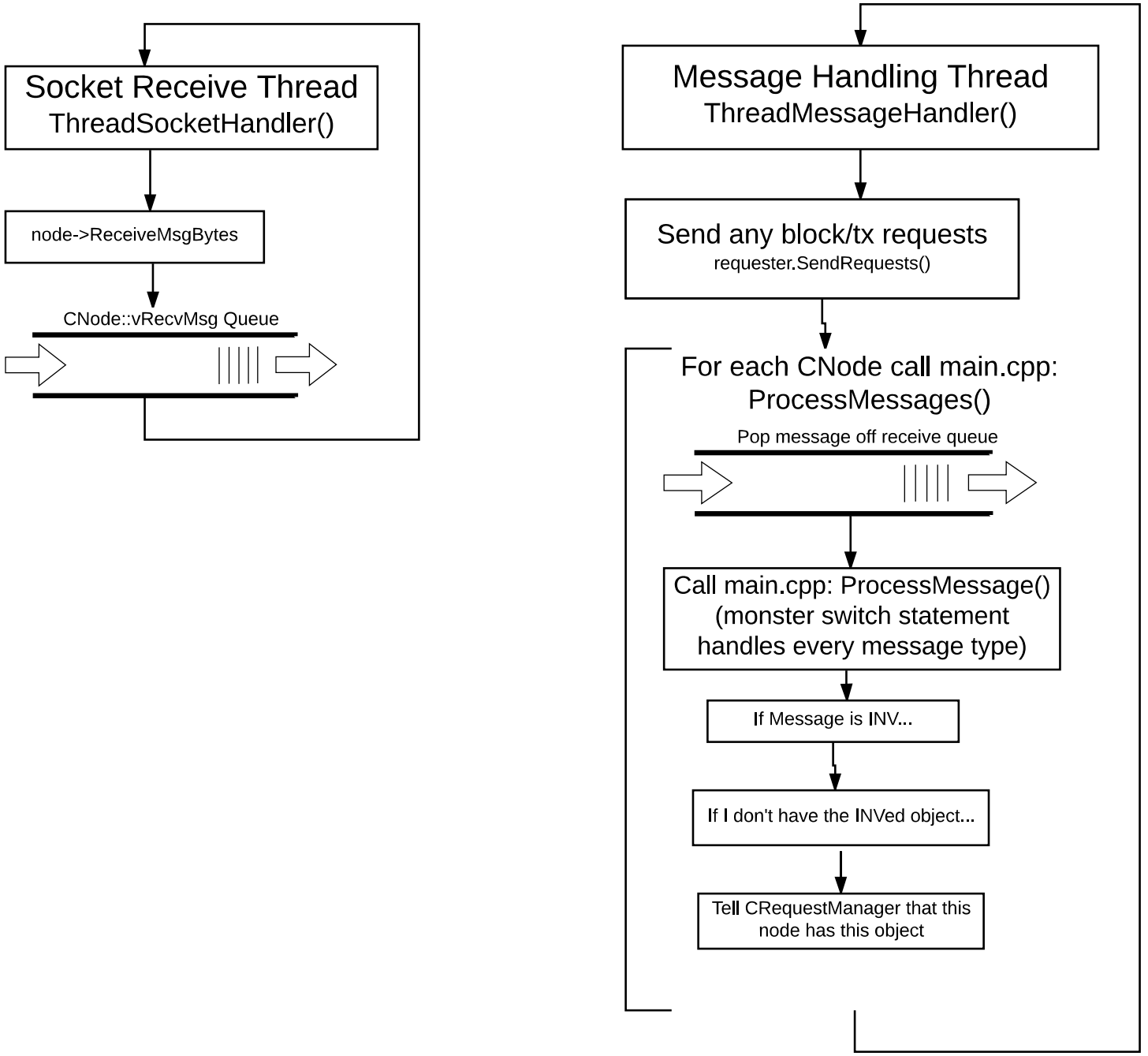
Blockchain Class Interactions



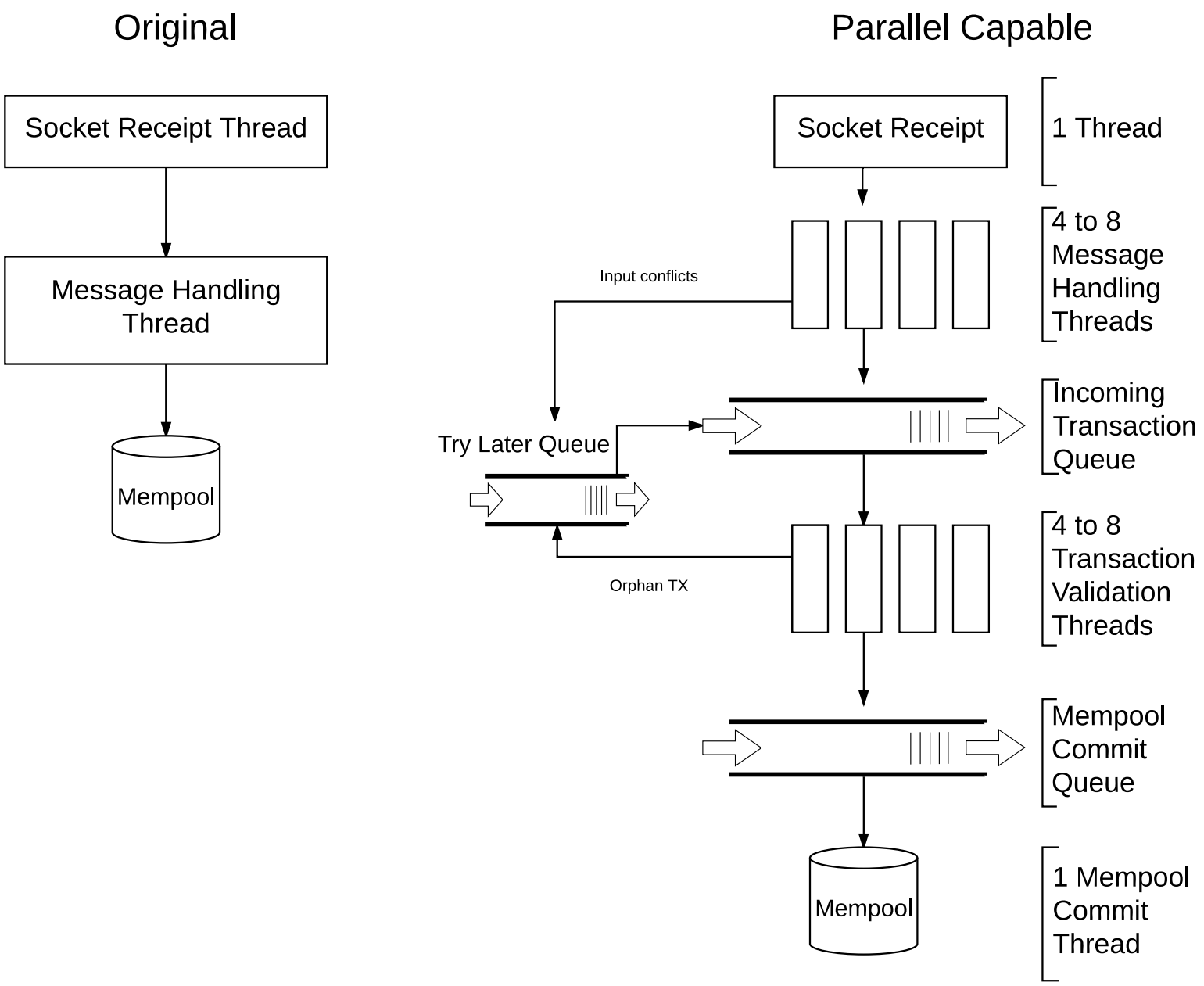
Useful Blockchain Functions



Network Protocol Processing



Transaction Processing Architecture



Useful Classes

- **CCriticalSection**
  - mutex
  - In declaration: Always comment what this mutex protects
  - In declaration: Always comment what mutex is protecting this object
- **CValidationState**
  - return error details
- **CTweak**
  - easily add a configuration parameter
- **CStat, CStatHistory**
  - create a statistic

Building Bitcoin

- **Setup**
  - ./autogen.sh
  - make distclean
- **Local debug build (3.5 min)**
  - ln -s /ramdisk/debug .; cd debug
  - <srcdir>/configure --enable-debug --enable-gperf --disable-hardening
  - make V=1 -j7
- **Cross-build**
  - cd depends; make -j11 HOST=i686-w64-mingw32
  - mkdir w64; cd w64
  - ./configure --prefix=`pwd`/../depends/i686-w64-mingw32/ --without-comparison-tool --disable-dependency-tracking --enable-reduce-exports LDFLAGS=-static-libstdc++ --host=i686-w64-mingw32
  - make -j11 V=1

Running Bitcoin

- conf file: ~/.bitcoin/bitcoin.conf
  - Start bitcoind or bitcoin-qt to create the .bitcoin directory (or create it yourself)
  - if bitcoin.conf doesn’t exist create it, otherwise modify to:

    regtest=1

    server=1

    discover=0

    rpcuser=rt

    rpcpassword=rt
- nohup <builddir>/src/bitcoind &

Using bitcoin-cli

- Basic Information
  - getinfo
  - getpeerinfo
  - getmempoolinfo
  - listbanned
  - help [command]
- Basic functions
  - addnode <ip:port> onetry
  - pushtx <ip>
  - clearbanned
  - setban <ip> “add|remove” <time>
  - generate N (regtest only)
- Configuration
  - get “*”
  - set <item>=<value>
    - NO spaces around equals: for example “set a=b” not “set a = b”
- Statistics
  - getstatlist
  - getstat memPool/size sec10 1

Troubleshooting bitcoin-cli

- Wrong conf file?
  - --conf=<directory>/bitcoin.conf
- Wrong blockchain?
  - use --testnet, --chain_nol, --regtest
- Wrong username/password?
  - bitcoin.conf:
    - rpcuser=foo
    - rpcpassword=yourpassword
  - or specify on command line
- Remote access allowed?
  - bitcoin.conf: rpcallowip=0/0
- bitcoin-qt server mode not enabled?
  - bitcoin.conf: server=1

Testing

- Test blockchains/networks
  - --testnet, --chain_nol, --regtest
- C++ Unit tests
  - uses boost unit test framework
  - cd <builddir>/src/test; ./test_bitcoin
- Python Regression tests
  - cd <builddir>/qa/pull_tester; ./rpc-tests.py --help
  - Run 1 regression test:
    - cd qa/rpc-tests
    - ./zmq_test.py --srcdir=<builddir>/src --tmpdir=/ramdisk/test --nocleanup --noshutdown
    - You can access the node:
      - cd /ramdisk/test
      - <builddir>/src/bitcoin-cli --conf=`pwd`/node0/bitcoin.conf getinfo
  - Use test_template.py to:
    - Make your own test
    - Quickly spawn a bunch of regtest connected nodes

Add the MYFEATURE Protocol Message

- Announce support
  - version.h:
    - Add MYFEATURE_VERSION definition
    - Bump PROTOCOL_VERSION
  - BUVERSION
    - Append your data to the BUVERSION message
- protocol.cpp, protocol.h:
  - Add a short string message identifier: “myfeat”
- Create an object that corresponds to your message
  - Add serialization member functions:

```
#include “serialize.h”

Class MyFeatureMsg
{
    <type> someData;
public:
    ADD_SERIALIZE_METHODS;
    template <typename Stream, typename Operation> inline void SerializationOp(Stream& s, Operation ser_action)
    {
        READWRITE(someData);
    }
};
```

- Add handler in main.cpp ProcessMessage()
  - unpack the message:
  - (Add special INV type to RequestManager)
- Send the new message:

```
MyFeatureMsg msg(...);
node->PushMessage(NetMsgType::MYFEATURE, msg);
```

Other topics

- Add a new CLI command
- Add a statistic
- Change the GUI

Before pull request

- squash commit
  - converts your many commits into one
  - don’t squash other people’s work if you cherry-picked or merged
- “git diff > diff.txt” and review for debugging you didn’t remove
- make check
- make check-formatting
  - if source formatting incorrect, follow instructions to fix
- run qa tests
- run compiled with --enable-debug to check mutex deadlocks

squash part of this branch:

git log (to figure out how many commits to squash)
git rebase -i HEAD~6
git push <origin> <branch-name> --force

squash all commits on a branch:

git rebase -i <parent branch>
git push <origin> <branch-name> --force