libcoin

Michael Gronager, PhD
Director, Ceptacle

gronager@ceptacle.com

Bitcoin Convention 2012 San Antonio, TX
March 17 2012

Bitcoin Convention 2012 San Antonio, TX

libcoin - intro

- It is not another crypto currency
- It is not a rewrite of bitcoin
- It is not yet another client

ceptacle

libcoin - intro

- It is not another crypto currency
- It is not a rewrite of bitcoin
- It is not another client

- It is the refactorization of bitcoin into a modular library
- It is chain agnostic
- You can build bitcoind, and bitcoin-Qt on libcoin

ceptacle

libcoin - motivation

- Building thin clients in C++
- Faster / More predictable http interface
- Support ideas from other chains / crypto currencies (e.g. freicoin) and consolidate efforts.
- Separate client(s) from core functionality
- Facilitate education, research and innovation

libcoin - paradigm

- No globals
- High level of Concurrency
- Clean classes with interfaces
- Keep as much as possible of the original code

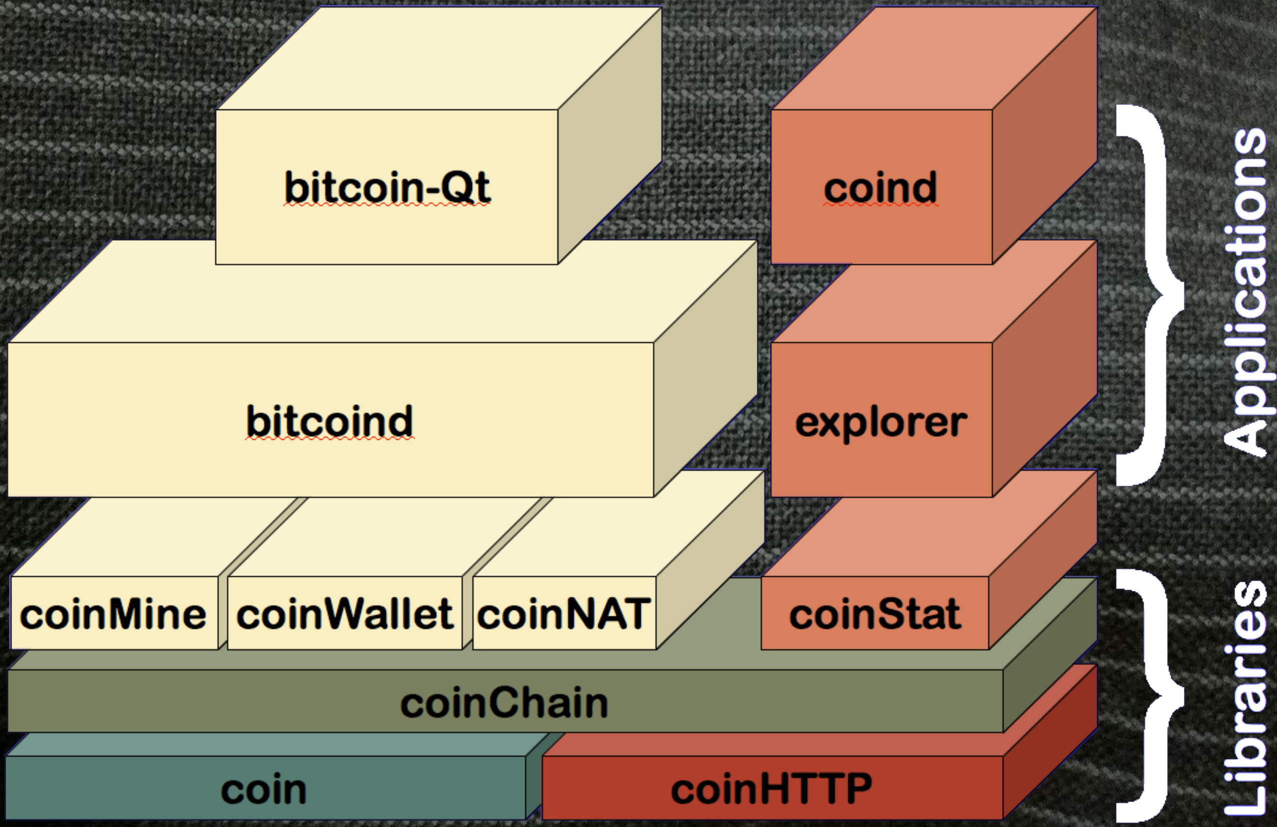
- Keep bitcoind functional at all times!
- Faster chain download

ceptacle

libcoin - license

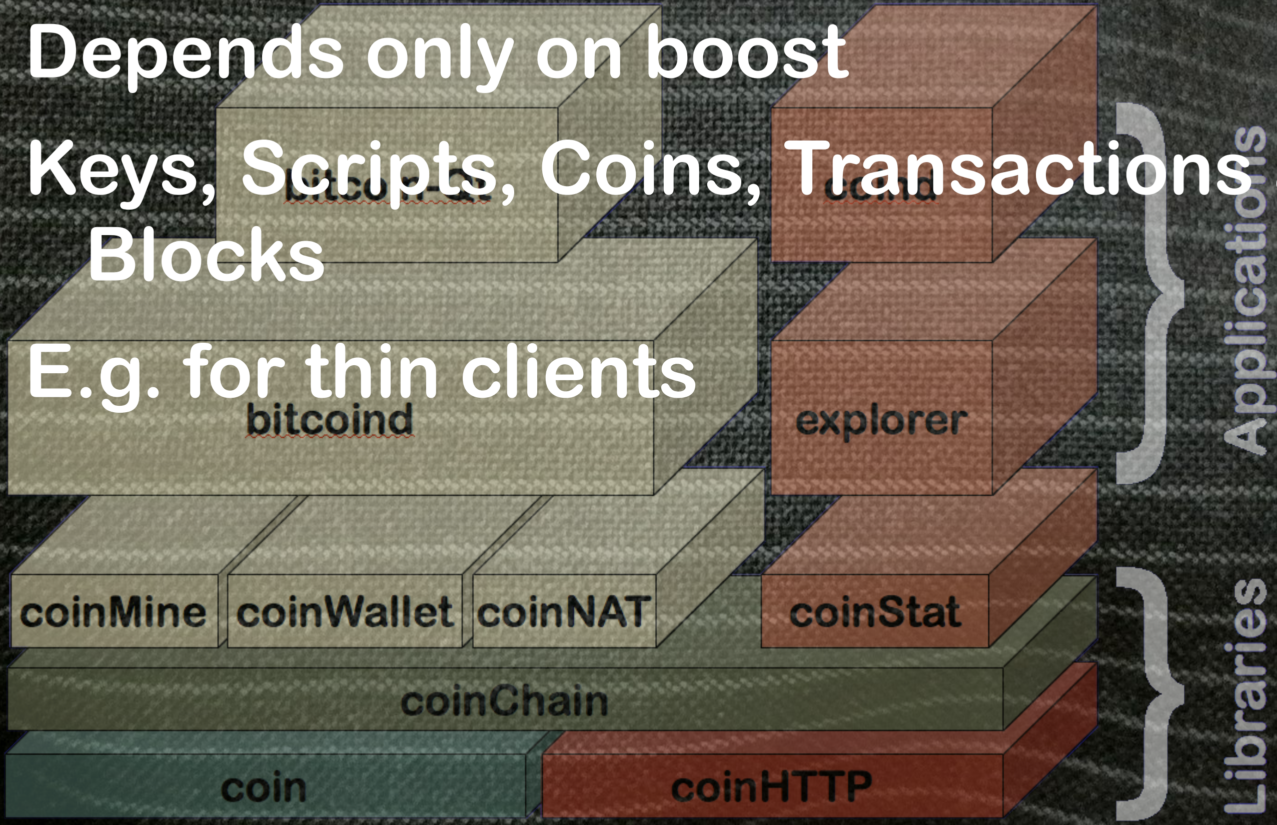
- It is a library – so LGPL3!
- Why move away from MIT?
  - libcoin is include and reuse as opposed to copy and compete
  - Aims at being a basic library for several crypto currencies
- Open for other arguments...

libcoin - structure



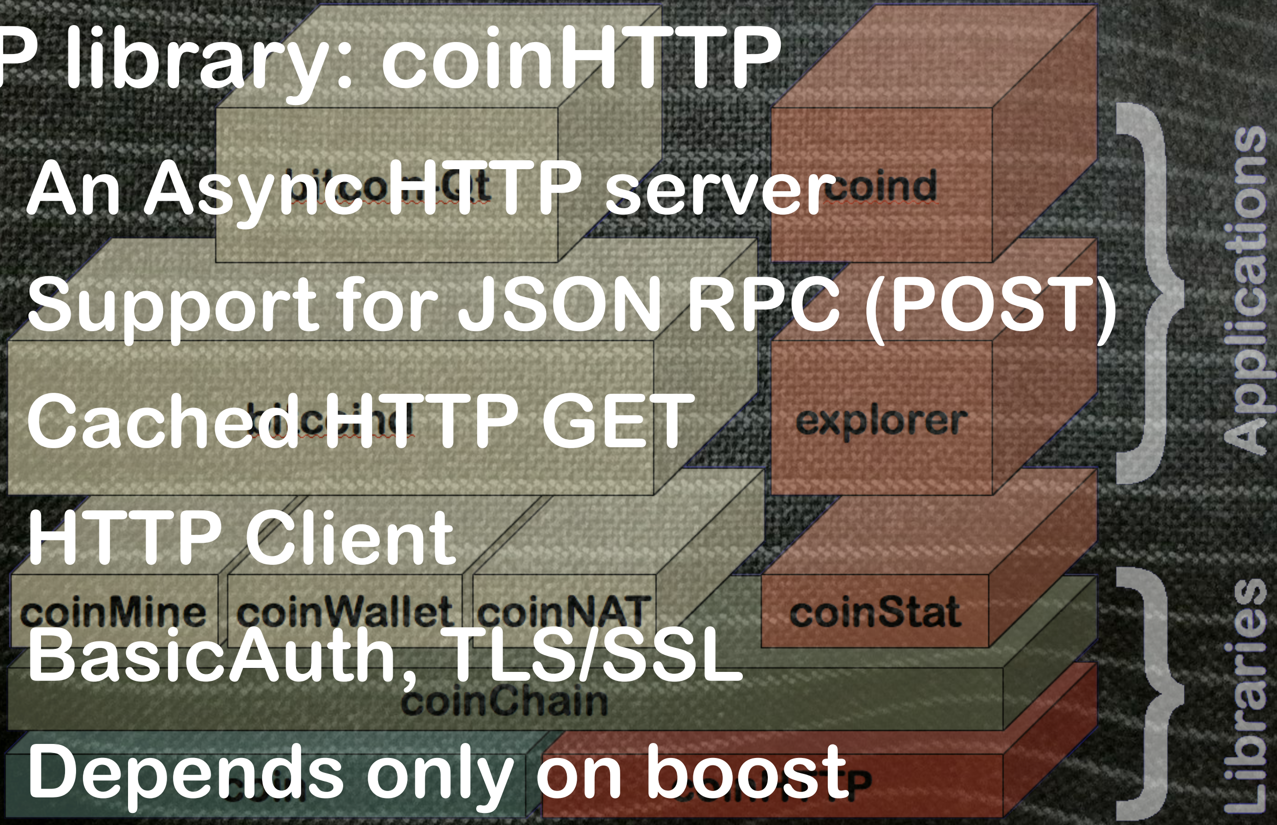
libcoin - structure

- Basic library: coin
  - Depends only on boost
  - Keys, Scripts, Coins, Transactions and Blocks
  - E.g. for thin clients



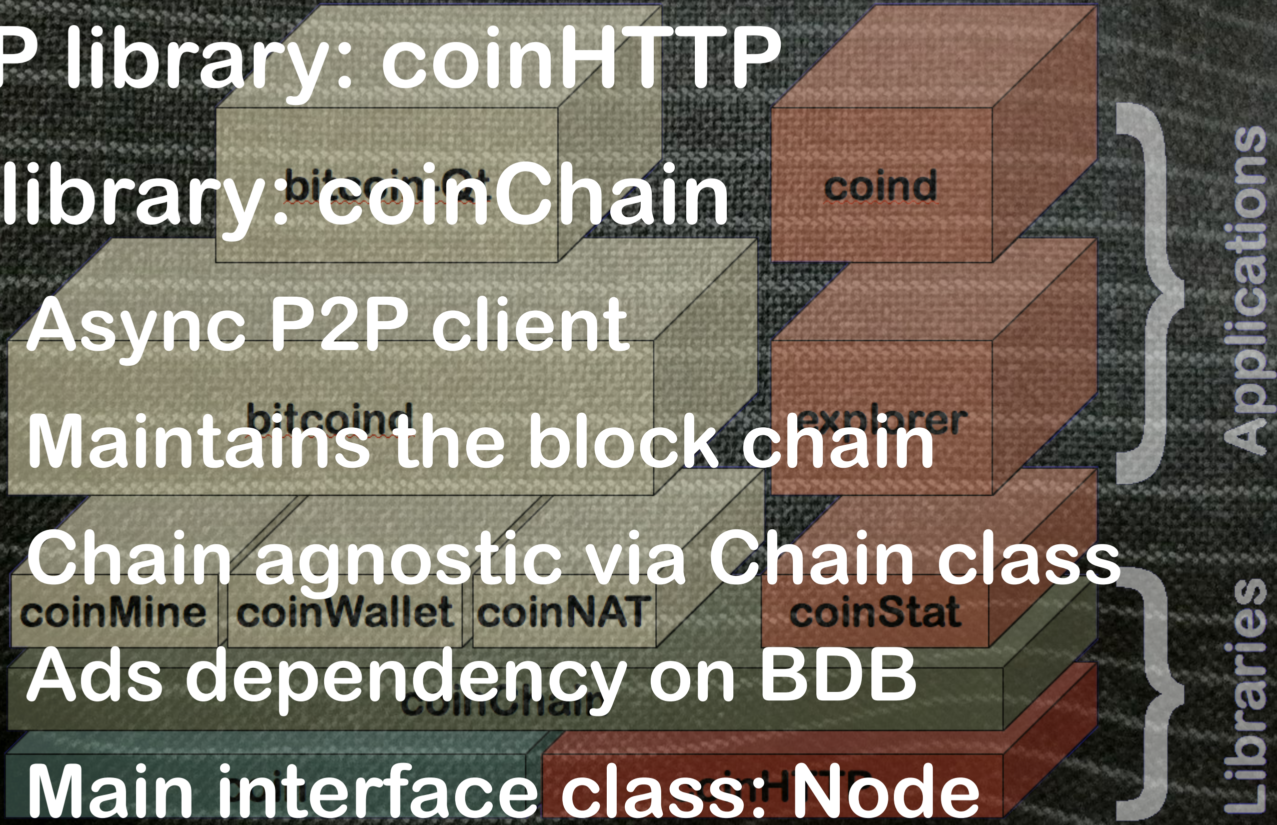
libcoin - structure

- Basic library: coin
- HTTP library: coinHTTP
  - An Async HTTP server
  - Support for JSON RPC (POST)
  - Cached HTTP GET
  - HTTP Client
  - BasicAuth, TLS/SSL
  - Depends only on boost
  - Main interface class: Server



libcoin - structure

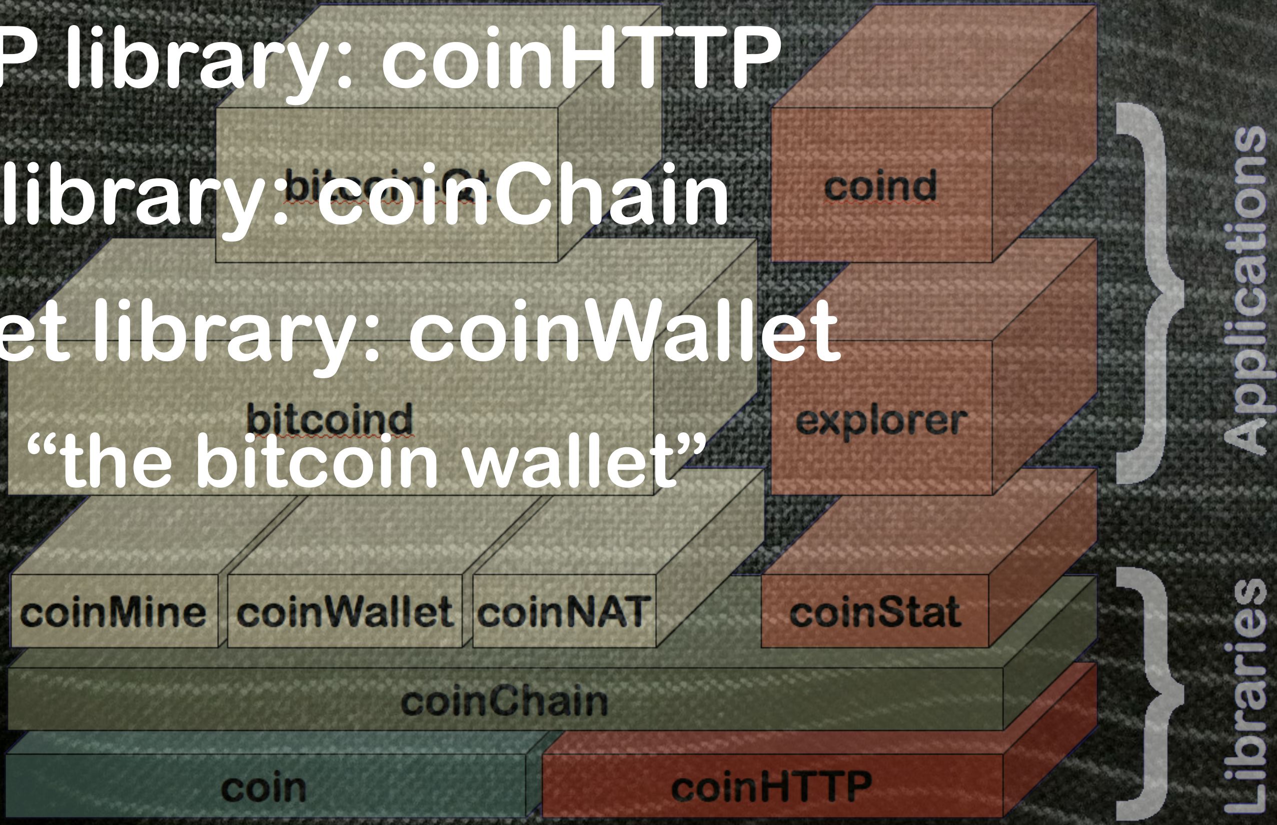
- Basic library: coin
- HTTP library: coinHTTP
- P2P library: coinChain
  - Async P2P client
  - Maintains the block chain
  - Chain agnostic via Chain class
  - Ads dependency on BDB
  - Main interface class: Node



ceptacle

libcoin - structure

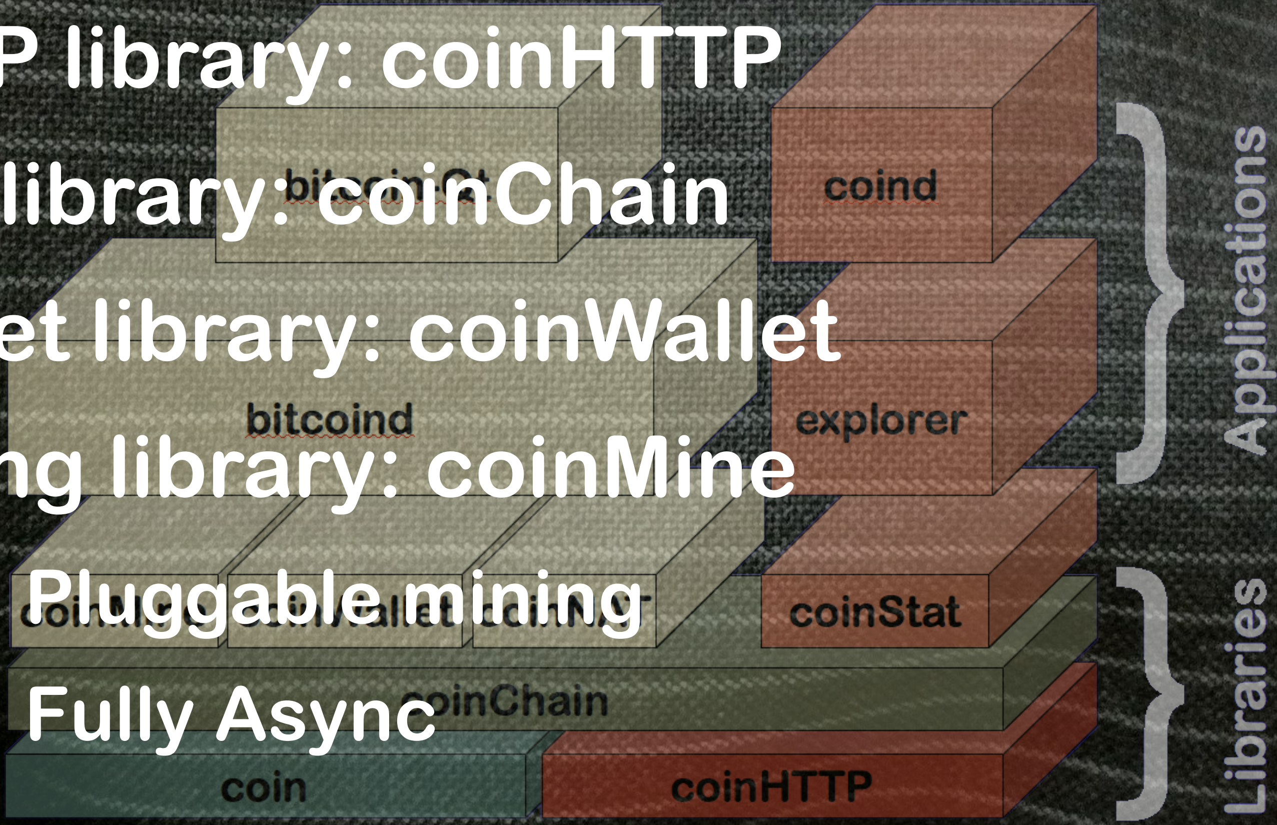
- Basic library: coin
- HTTP library: coinHTTP
- P2P library: coinChain
- Wallet library: coinWallet
  - "the bitcoin wallet"



ceptacle

libcoin - structure

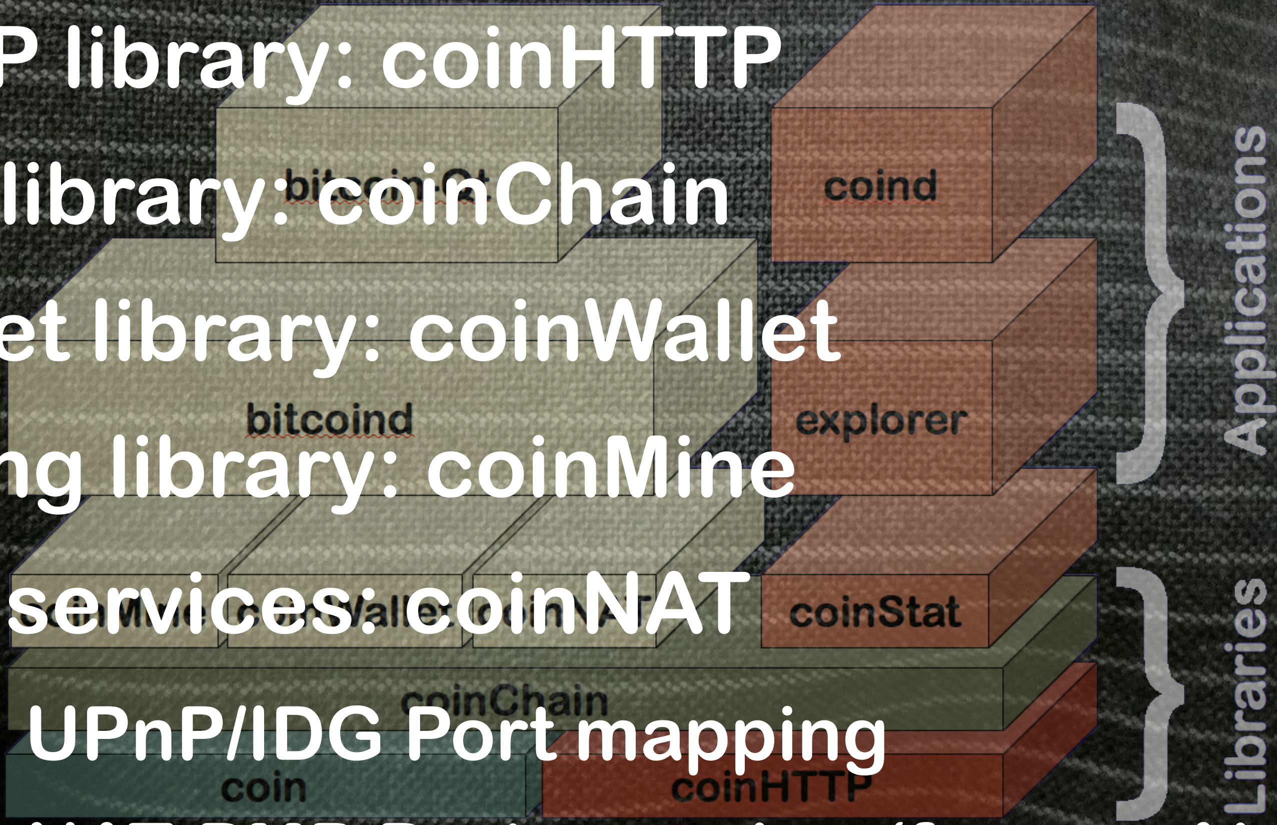
- Basic library: coin
- HTTP library: coinHTTP
- P2P library: coinChain
- Wallet library: coinWallet
- Mining library: coinMine
  - Pluggable mining
  - Fully Async



ceptacle

libcoin - structure

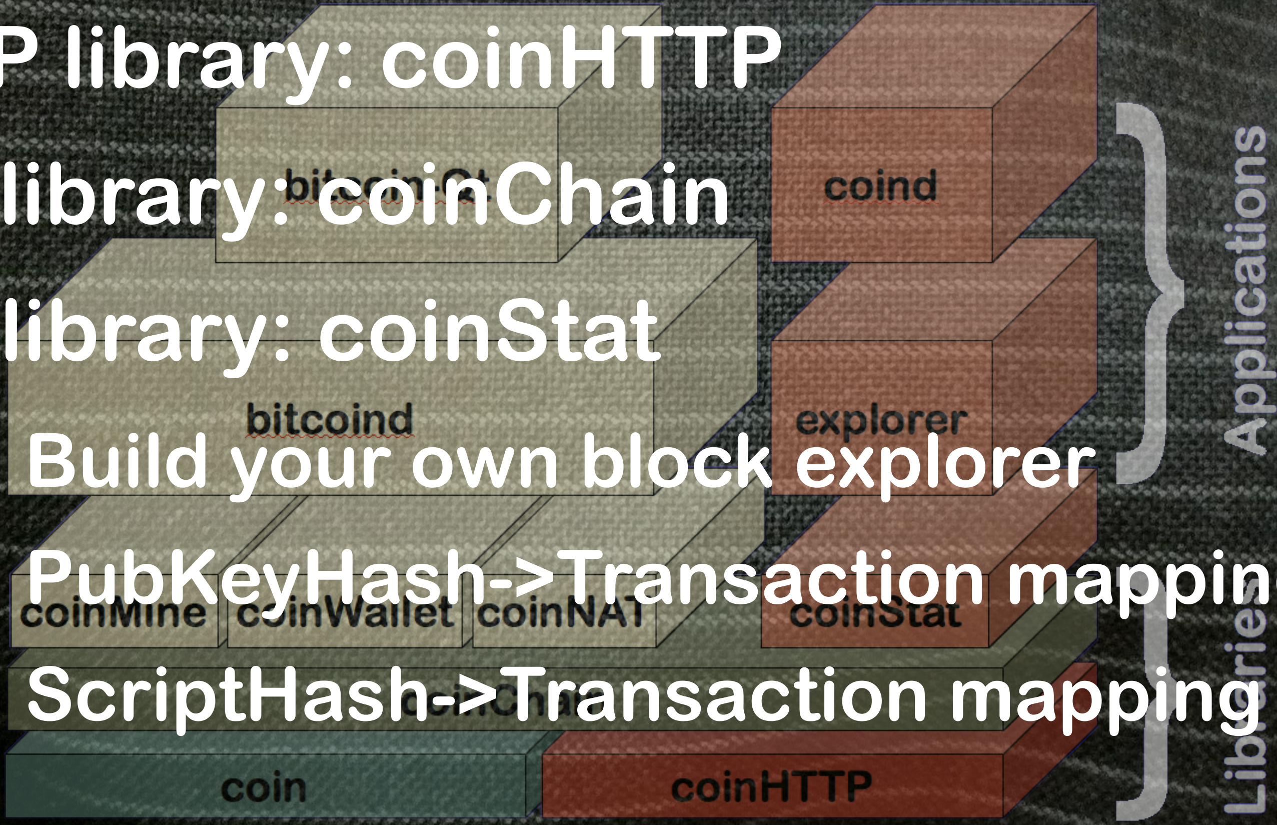
- Basic library: coin
- HTTP library: coinHTTP
- P2P library: coinChain
- Wallet library: coinWallet
- Mining library: coinMine
- NAT services: coinNAT
  - UPnP/IDG Port mapping
  - NAT-PMP Port mapping (for e.g. AirPort)



ceptacle

libcoin - structure

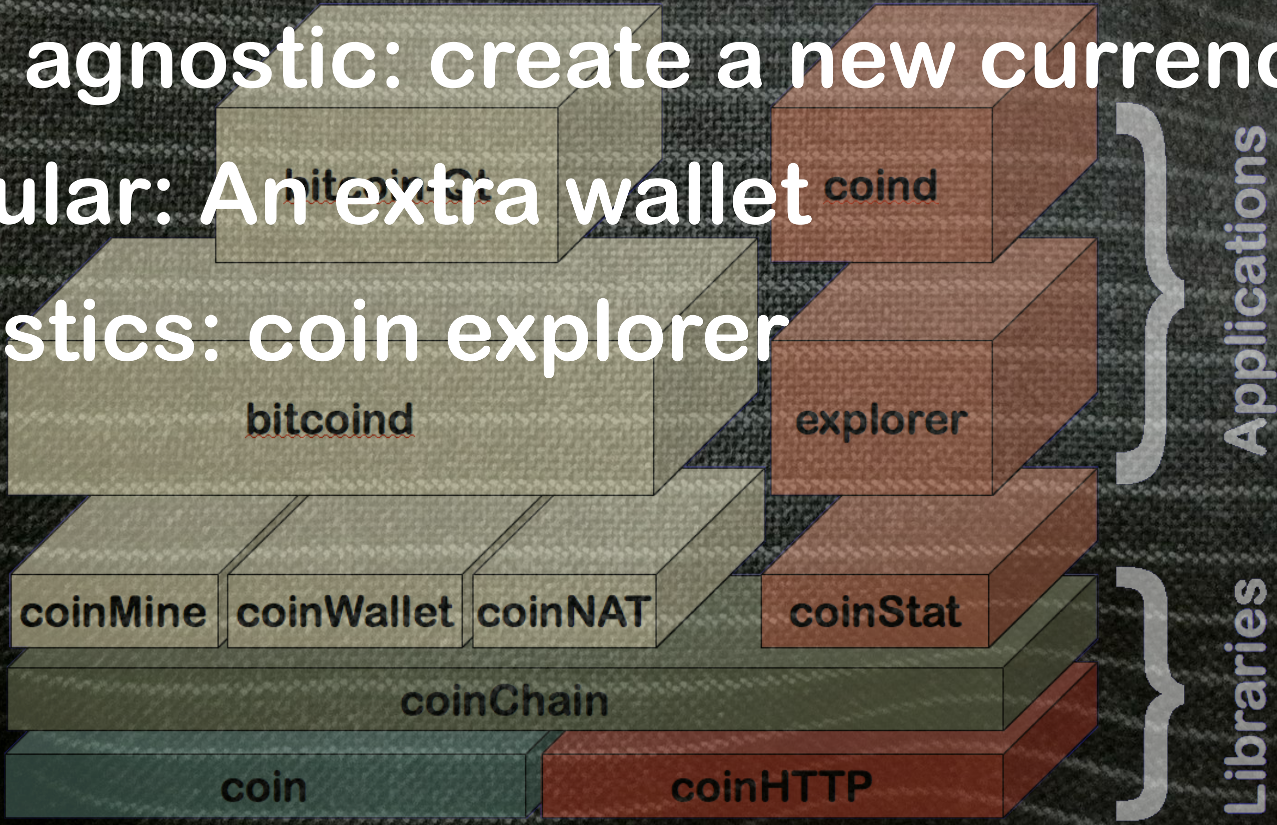
- Basic library: coin
- HTTP library: coinHTTP
- P2P library: coinChain
- Stat library: coinStat
    - Build your own block explorer
    - PubKeyHash->Transaction mapping
    - ScriptHash->Transaction mapping



ceptacle

libcoin - examples

- 20 lines client: simple client
- Coin agnostic: create a new currency
- Modular: An extra wallet
- Statistics: coin explorer



libcoin – simple client

- Write a bitcoin client in 20 lines
- Includes:

```
#include <coinChain/Node.h>
#include <coinChain/NodeRPC.h>

#include <coinHTTP/Server.h>

#include <coinWallet/Wallet.h>
#include <coinWallet/WalletRPC.h>

using namespace std;
using namespace boost;
```

libcoin – simple client

- Write a bitcoin client in 20 lines
- Setup node and wallet – Start the Node:

```
int main(int argc, char* argv[])
{
    logfile = CDB::dataDir(bitcoin.dataDirSuffix()) + "/debug.log";

    Node node; // deafult chain is bitcoin

    Wallet wallet(node); // add the wallet

    thread nodeThread(&Node::run, &node); // run this as a background thread
```

libcoin – simple client

- Write a bitcoin client in 20 lines
- Setup Server and register methods:

```
Server server;

server.registerMethod(method_ptr(new Stop(server)));
server.registerMethod(method_ptr(new GetBlockCount(node)));
server.registerMethod(method_ptr(new GetConnectionCount(node)));
server.registerMethod(method_ptr(new GetDifficulty(node)));
server.registerMethod(method_ptr(new GetInfo(node)));

// Register Wallet methods.
server.registerMethod(method_ptr(new GetBalance(wallet)));
server.registerMethod(method_ptr(new SendToAddress(wallet)),
                                          Auth("username","password"));
```

libcoin – simple client

- Write a bitcoin client in 20 lines
- Start Server and clean up:

```
    server.run();

    node.shutdown();
    nodeThread.join();

    return 0;
}
```

libcoin – a new currency

- libcoin is chain agnostic – lets create a new currency: Ponzicoin!

```
class PonziChain : public Chain
{
public:
    PonziChain();
    virtual const Block& genesisBlock() const ;
    virtual const uint256& genesisHash() const { return _genesis; }
    virtual const int64 subsidy(unsigned int height) const ;
    virtual bool isStandard(const Transaction& tx) const ;
    virtual const CBigNum proofOfWorkLimit() const { return CBigNum(~uint256(0) >> 20); }
    virtual unsigned int nextWorkRequired(const CBlockIndex* pindexLast) const ;
    virtual bool checkPoints(const unsigned int height, const uint256& hash) const { return 
true; }
    virtual unsigned int totalBlocksEstimate() const { return 0; }
```

ceptacle

libcoin – a new currency

- libcoin is chain agnostic – lets create a new currency: Ponzicoin!

```
...
    virtual const std::string dataDirSuffix() const { return "ponzicoin"; }
    virtual ChainAddress getAddress(PubKeyHash hash) const { return ChainAddress(0xff,
hash); }
    virtual ChainAddress getAddress(ScriptHash hash) const { return ChainAddress(); }
    virtual ChainAddress getAddress(std::string str) const {
        ChainAddress addr(str);
        if(addr.version() == 0xff) addr.setType(ChainAddress::PUBKEYHASH);
        return addr;
    }
    virtual const MessageStart& messageStart() const { return _messageStart; };
    virtual short defaultPort() const { return 5247; }
    virtual std::string ircChannel() const { return "ponzicoin"; }
    virtual unsigned int ircChannels() const { return 1; } // number of groups to try
```

libcoin – a new currency

- Define subsidy and nextWorkRequired:

```
const int64 PonziChain::subsidy(unsigned int height) const {
    int64 s = 50 * COIN;

    // Subsidy is cut in half every week
    s >>= (height / 10080);

    return s;
}
...
```

ceptacle

libcoin – a new currency

- Define the node

```
Node node(ponzicoin);
```

- This is all we need!
- Note: Some currencies have further subtleties:
  - freicoin: demurrage (coins get old)
  - namecoin: extended protocol
  - …

libcoin - an extra wallet?

```
… define node stuff...

    Wallet wallet(node, "wallet.dat"); // add the wallet
    Wallet extra_wallet(node, "extra_wallet.dat"); // add the extra wallet

    // Register Wallet methods. - note that we don't have any auth, so anyone (on localhost) can
read your balance!
    server.registerMethod(method_ptr(new GetBalance(wallet)));
    server.registerMethod(method_ptr(new SendToAddress(wallet)),
Auth("username","password"));

    GetBalance* extragetbalance = new GetBalance(extra_wallet);
    extragetbalance->setName("extragetbalance");
    server.registerMethod(method_ptr(extragetbalance));

    SendToAddress* extrasendtoaddress = new SendToAddress(extra_wallet);
    extrasendtoaddress->setName("extrasendtoaddress");
    server.registerMethod(method_ptr(extrasendtoaddress), Auth("username","password"));
```

ceptacle

libcoin - coin explorer

```
Node node(chain);
Explorer explorer(node); // this will register notifications for new blocks and transactions
thread nodeThread(&Node::run, &node); // run this as a background thread

string search_page =
"<html><head><title>libcoin - Coin Explorer</title></head>"
"<body><form action=\"/search\" method=\"post\" enctype=\"text/plain\"><p>"
        "<input type=\"text\" name=\"params\" size=\"50\">"
        "<input type=\"submit\" value=\"Search\"></p>"
"</form></body></html>";
Server server(rpc_bind, search_page));

// Register Node methods.
server.registerMethod(method_ptr(new GetBlockCount(node)));
server.registerMethod(method_ptr(new GetConnectionCount(node)));
server.registerMethod(method_ptr(new GetDifficulty(node)));
server.registerMethod(method_ptr(new GetInfo(node)));
```

libcoin - coin explorer

- Register methods:
  - Getdebit, credit, coins, addrbalance, search

```
// Node methods relevant for coin explorer
server.registerMethod(method_ptr(new GetBlock(node)));
server.registerMethod(method_ptr(new GetBlockHash(node)));
server.registerMethod(method_ptr(new GetTransaction(node)));

// Register Explorer methods.
server.registerMethod(method_ptr(new GetDebit(explorer)));
server.registerMethod(method_ptr(new GetCredit(explorer)));
server.registerMethod(method_ptr(new GetCoins(explorer)));
server.registerMethod(method_ptr(new GetAddressBalance(explorer)));
server.registerMethod(method_ptr(new Search(explorer)));
```

libcoin – future

- coin: remove dependency of openssl/libcrypto
- coinHTTP: methods as plugins
- coinChain: header only, BDB->SQLite/memory
- coinMine: reuse puddinpop miners as plugins

- Update everything to 0.6 incl libcoin/Bitcoin-Qt

ceptacle

libcoin – try it – use it!

- Wiki:
  - github.com/ceptacle/libcoin/wiki
- Github:
  - github.com/ceptacle/libcoin
- Twitter:
  - follow libcoin
- Mailinglists:
  - Bitcoin-dev / forums

ceptacle