从 0 到 1 建立自己的区块链

[**帅初**](http://www.8btc.com/author/12240)**2015-11-09 10:23 发布在**[**区块链**](http://www.8btc.com/blockchain)**,**[**技术指南**](http://www.8btc.com/how)[**7**](http://www.8btc.com/build-your-own-blockchain#comment)**22880**



写作说明

自从 2009 年 Satoshi Nakamoto 发布了 Bitcoin 的开源项目,区块链技术作为其背后的支撑技术也在慢慢发展,2014 年以来,越来越多的科技巨头和产业资本逐渐进入了这个领 域,对区块链技术的发展也起到了极大的促进作用。

我们可以把 bitcoin 看做基于区块链技术的第一个金融应用,它是一个去中心化、无需 信任单个节点、隐私又透明、在共识机制下由人人维护的货币系统和支付清算网络。这样 的一个去中心化的清算网络正在成为另外一种互联网的基础设施,就像计算能力、带宽是 互联网的基础设施一样,比特币的支付网络正成为另外一种互联网基础设施—“信任”, 这样的一个基础设施每天由全世界的矿工共同维护,并越来越散发出迷人的魔力,并成为 Fintech 创新的温床和基础设施,也吸引了越来越多的资源进入该领域。

除了在支付和金融领域的应用,区块链作为一种去中心化的、共享的、无需信任单个节 点的公开透明的数字账本,也在逐渐应用到物联网、身份认证、供应链管理、版权管理、 去中心化的应用等众多领域。区块链技术解决了计算机领域著名的“拜占庭将军”问题, 使得在无需信任单个节点的情况下,创建一个去中心化的共识网络。

目前大多数和区块链技术相关的项目都在国外,国内区块链技术的开发和应用相对滞后, 因此计划在闲暇时间写一些简单的开发手册,希望可以帮助到从事这个行业的个人或者小 公司。

暂定的写作计划分为两部分

第一部分:《从 0 到 1,建立自己的区块链》 :重温比特币网络的发布过程,并发行自 己的区块链网络(私链)

第二部分:《从 0 到 1,开发区块链的应用》 :在公有链和私有链上,开发自己的区块 链应用。

第一部分:从 0 到 1 建立自己的区块链 目录:

1.1 从模仿开始,初识区块链  
1.2 区块链的基础:共识机制剖析  
1.3 共识机制的设计原理和设计方法  
1.4 如何快速克隆一条区块链  
1.5 如何把比特币变成自己的私链–分叉比特币  
1.6 如何把以太坊变成自己的私链–分叉以太坊  
1.7 如何把 Ripple 变成自己的私链–分叉 ripple  
1.8 如何把 stellar 变成自己的私链–分叉 stellar 1.9 如何搭建一个矿池,并挖出自己的创始区块  
1.10 如何开发自己的区块链钱包(Windows 和 MAC) 1.11 如何开发自己的区块链钱包(Android 和 IOS) 1.12 如何开发一个类似于 blockchain.info 的在线钱包 1.13 如何增加自己的区块链网络的安全性和鲁棒性 1.14 如何利用 coind 来处理充值提现业务  
1.15 如何利用资金池搭建一个混币服务  
1.16 如何设计一种新的挖矿算法

1.1 从模仿开始,初识区块链

对理工科的人来说,学习的最好方式就是直接上手,区块链的概念我们了解了很多,今 天就让我们解开区块链的神秘面纱,让我们直接动工吧,让我们自己从零开始发布自己的区块链,并且在发布过程中学习!

目前主流的共识机制有 POW,POS,Dpos,还有 ripple 和 stellar 的共识协议,以及[ethereum](http://www.8btc.com/ethereum) 的共识协议等等,其实还有一些 altcoin 也有一些独特的共识协议,目前关注的人 较少(这些共识协议,我们会在后面具体剖析)。

模仿需要一个源码,考虑到 Bitcoin 源码经过 6 年来的迭代,已经和最初版本有较大的差 异,我们选取 sunny king 大神的 pos 协议为基础,来发布自己的区块链,这里我们在 github 上面选取一个较好的代码库:cravecoin 的原始代码服务器配置:阿里云租用 Ubuntu14.04 版本服务器 2Core 4GB 登陆服务器 root@your password  
从 github 上面下载示例源码:

wget https://github.com/industrialcoinmagic/crave/archive/bbf8914fef9bff080ce0826e4ee6b5b76ac4690a.zip

解压下载的文件 unzip bbf8914fef9bff080ce0826e4ee6b5b76ac4690a.zip

其中 crave.pro 是 QT 客户端编译需要的文件Doc

是一些基本的 docs,包含在源码里面

Share has some needed libs build’s used by the build system so we always need it

Src 包含客户端编译所需要的源码,以及图像和 logo

Pro 文件为客户端编译所需要的文件,在阿里云 Ubuntu14.04 上面,我们可以编译 Linux 钱包客户端和 Windows 钱包客户端

为了 alive 自己的区块链,我们需要把 CRAVE 的区块链,替换成自己的的区块链,因为 NEO 是中是《TheMatrix》正义的化身和代言人,我们为 NEO 来发布一条区块链区块链吧。



我们把 crave.pro 重命名成为 neocoin.pro (这里你可以重命名为任何你喜欢的名字,比如 Xcoin or bankcoin. Or your company coin)

从 POS2.0 协议开始,区块链网络的参数存储在 chainparams.cpp 中,在最原始的 pos 源码中,网络 参数存储在 main.cpp and main.h,从 2014 年以后,主流的 pos 协议都在采用这种方式。

nano chainparams.cpp 打开 chainparams.cpp 文件

修改一个区块链最重要的是要修改 pchMessage,后面的 16 进制标识符,是区块链的一个 ID,用来区别两条不一样的区块链。否则,如果不修改,会发生自己的区块链同步到其他区块链上面的情况。 源码中为:

chMessageStart[2] = 0x05;

chMessageStart[2] = 0xa5;

chMessageStart[2] = 0xb6;

chMessageStart[2] = 0x1b;

其中 0x 保持不变,数字范围 0-6,字母范围 a-f 我们修改为:

pchMessageStart[0] = 0xaf;

pchMessageStart[1] = 0x25;

pchMessageStart[2] = 0x04;

pchMessageStart[3] = 0x32;

现在我们区块链有一个唯一的 ID 了  
const char\* pszTimestamp = 8 Mar 2015 The rebirth starts.这里一般用来记录区块链 alive 的那一天发生的大事情,大家知道中本聪在这里写的什么吗?

你猜?

￼下一步我们需要修改 txNew.nTime = 1426700641; 这个时间如何得到呢?  
txNew.nTime 代表区块链发布的 unix 时间,可以在这里把北京时间转换成为 unix 时间

http://tool.chinaz.com/Tools/unixtime.aspx

同时也需要修改 genesis.nTime = 1446708703

把 genesis.nNonce = 17113 修改成为 genesis.nNonce = 0 区块从 0 开始,nNonce 用来计算,计算 找到第一个 block 需要的区块数,一旦找到创始区块,将会重新计数 block

下一步我们需要移除旧的创始区块和 merkle 的值 (remove the old genesis and merkle) Old merkle 是:

assert(hashGenesisBlock == uint256("0x00000a336bf3e2be21c2ce9a3f9bc9849$

assert(genesis.hashMerkleRoot == uint256("0x281554338b643722b01ea689ab6$

我们删除 unit256(“0x 之后的内容,当我们第一次编译的时候,会报错。在 debug.log 里面会产生一个新 的 value,到时候把新的 value 放入这个地方即可。

然后我们修改 nLastPOWBlock = 10000; 代表 POW 的最后一个 block,这里的数值可以自己选择, 取决于挖矿时间的长短。但是如果数值太小,会照成 coin 没有成熟,从而无法 stake,照成网络停滞。这 里我修改成了 1000

同时在 chainparams.cpp 文件中,可以发现一个 testnet,同时修改 testnet 中的区块链标识符

pchMessageStart[0] = 0x1f;

pchMessageStart[1] = 0x22;

pchMessageStart[2] = 0x05;

pchMessageStart[3] = 0x30;

在 bitcoin 的 testnet 网络中,我们可以测试一些新的 BIP 协议,比如我们可以把区块大小设置成 20MB, 来观察网络的运行状态。在自己的 private chain 上面,我们可以测试任何我们想要的功能。而不会对mainnet 形成干扰 或者破坏 mainnet 的完整性。

同时修改 testnet 的 Genesis block assert(hashGenesisBlock == uint256(“0x00000a336bf3e2be21c2ce9a3f9bc9849$ 删除 0x 后面的 value

在 source 中 valertpubkey 也非常关键,可以增加网络的安全性,可以用来通知新的 block,在我们第 一次运行 daemon 的时间,会产生一个网络的 publickey,publickey 可以由 checkpoint 生成。

下面我们修改网络的端口号

nDefaultPort = 30104; 为 p2p port,网络用来与 daemons 通信

nRPCPort = 30105; 为 RPC 端口,所有 exchange,pool, explorer 通过 rpc port 与 daemon 通信 可以修改成为 10000 to 60000 的任何端口号,我们修改成 26666 和 26667

nDefaultPort = 26666;

nRPCPort = 26667;

修改完成后,ctrl+x 保持文件,并退出。

然后进入 main.cpp,找到

unsigned int nStakeMinAge = 8 60 60; // 8 hours

unsigned int nModifierInterval = 8 \* 60; // time to elaps

这里是最少的 staketime,在 pos2.0 的协议当中,我们需要 mini stake age,在 pos3.0 协议中,已经不再需 要。通用的是 8 个小时,但是你可以修改成任何时间,通常情况下,为了网络安全,建议时间大于 1 个小 时。但是在 pos3.0 协议中,coin 从 block1 开始 stake!

int nCoinbaseMaturity = 188; 为 coin 成熟的区块数,可以修改成任何数字。这里我改为了 100

int64\_t nSubsidy = 100 \* COIN; 为区块奖励的数额,这里可以修改你想要的 block reward,当然也可以 在这里设定自己想要的 block reward structure,控制 pow 和 pos 出币的多少。

static const int64\_t nTargetTimespan = 24 \* 60; // 24 mins 为区块难度调整的时间,通常如果是一个 public chain,在 挖矿的刚开始,我们希望难度系数尽快调整,可以防止 instamine。

int nTargetSpacing = 60; 为区块时间,比特币这里是 600s,我们这里设置为 30s。时间太短,容易分叉网 络。

下面我们修改 main.h 文件

static const unsigned int MAX\_BLOCK\_SIZE = 40000000;为区块大小,在 bitcoin 的 source code 中这里为 1MB,我们把 40000000 修改成 20MB,但是我们的网络不会超过 1MB 了。。。

static const int64\_t MIN\_TX\_FEE = 1000; 为网络的传输收付费,单位为聪,如果太少,网络可能会停滞。

static const int64\_t MAX\_MONEY = 2000000000 \* COIN;为单次发送的最大量,并不是 total amount

Ok main.cpp done

我们找到 checkpoints.cpp

static MapCheckpoints mapCheckpoints =

boost::assign::map\_list\_of

( 0, uint256("0x00000a336bf3e2be21c2ce9a3f9bc9849c697475d0de85e201bdc$

为创始区块的 check points,防止网络分叉,现在我们还没有这个 value,等待 daemon 第一次运行,会产 生这个 value

const std::string CSyncCheckpoint::strMasterPubKey =

MasterPubKey 告诉网络如何同步节点,为一个 main node,在网络早期,大家需要尽量同步到 main node

Checkpoints 告诉网络某一个特定 hash 值属于哪个 block

修改完成后,我们 ctrl+x 退出

然后找到 版本管理文件 clientversion.h

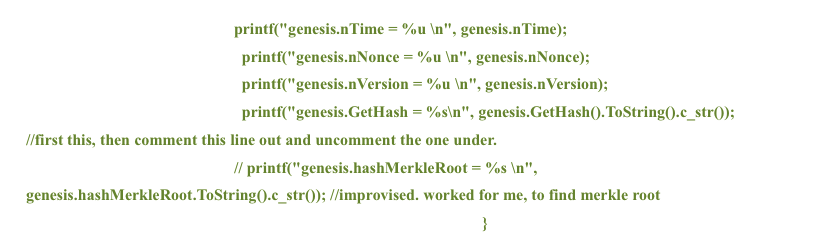
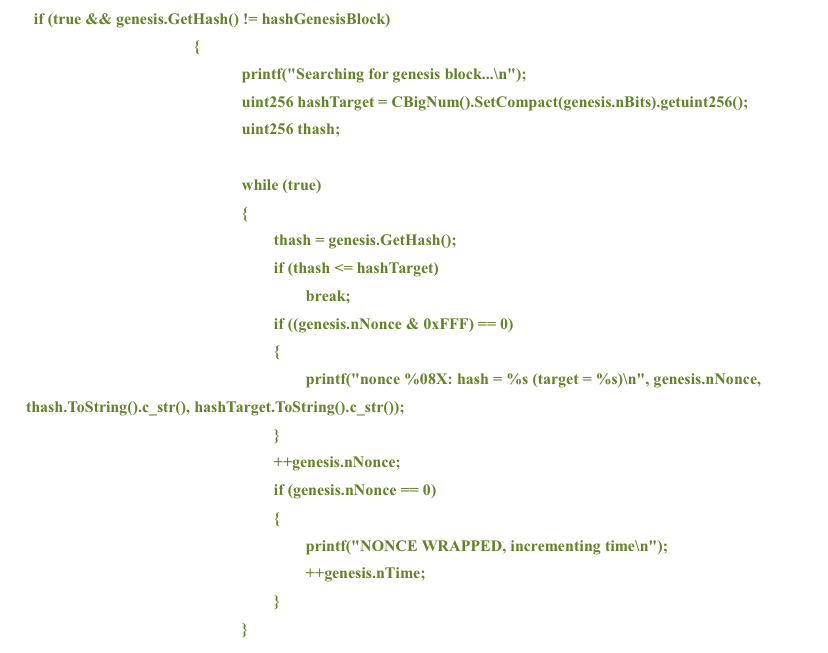
define CLIENT\_VERSION\_MAJOR 1

#define CLIENT\_VERSION\_MINOR 2

#define CLIENT\_VERSION\_REVISION 0

#define CLIENT\_VERSION\_BUILD 0

代表版本号 1.2.0 可以修改成 1.0

下面我们来产生,Genesis block,用下面代码来产生,把下述产生创始区块的代码,粘贴在 nNonce 下面  
刚好在 hashGenesisBlock  


这段代码会检测是否有创始区块产生,如果没有,就会产生一个创始区块。并把创始区块信息打印出来, 我们可以后面使用。  
一些最简单的参数修改已经完成,下面我们开始配置编译环境。

首先需要更新 Ubuntu 系统到最新版本 type : apt-get update

安装编译环境: apt-get install git build-essential libssl-dev libboost-all-dev git libdb5.1++-dev libminiupnpc-dev make zip  
其中

\* build essential package (为一个编译环境最基本的东西)  
\* libssl-dev :openssl 需要  
\*libboost :c++ boost libraries for the coin  
\* libdb5.1 :编译 wallet.dat files 需要  
Miniupnpc:钱包连接网络需要 wallets to look for open ports and connect  
make :用来编译钱包

配置完成后,我们使用 make -f makefile.unix  
来进行第一次编译… Hmm 写到这里,忘了一件事情有没有。。。



我们还没有修改这个 private chain 的名字有没有。。。查找替换你会不会。。? 把所有 Cravecoin or CRAVE or CRAVECOIN 统统替换成你想要的名字。。。。

我们用一个超炫的命令: grep -rl ‘crave’ ./ | xargs sed -i ’s/crave/neocoin/g’ 替换名字 还需要你自己的来替换 CRAVE or Cravecoin。。同样命令用三次

话说我们 logo 我也忘记了。。后面再替换

一切替换完成后,我们再一次编译

make -f makefile.unix

忘了我们是双核的 cpu,我们可以加快编译速度

make -j2 -f makefile.unix

编译成功,./neocoind

genesis.nNonce = 95200 把原来的 genesis.nNonce = 0 替换成 95200

得到创始区块:

genesis.GetHash = 00002acaf860e2ae594becc9ce1f20c4714b9aa7fc153f80585d8f562bba5e 1b

添加到 chainparams.cpp 和 checkpoints.cpp 当中

添加完成后,再次运行 neocoind,这次找到 merkle root,在编译之前,删除 用来产生 Genesisblock 的 那一段代码

运行得到 merkle root: 8465c95064313d48688e08e19d80915c0e03310a9c85c57d97cb355 c04b31b0e

下面我们为 coind 配置一个 config file,然后运行 daemon,配置简单的端口即可。 如果要修改 public key 的前缀地址,可以通过 base58 编码表修改:

https://en.bitcoin.it/wiki/List\_of\_address\_prefixes

./neocoind stop 停止  
./neocoind getaccountaddress “”

可以得到一个地址 NcrzS8YiRbfmDzza8sQB7CkyEaSttDZy8m 在另一台电脑上同样编译,得到一个新的节点,这里我的节点是:。。。。。 打开 chainparams.cpp,找到  
vSeeds.push\_back(CDNSSeedData(“112.74.98.71″, “112.74.98.71″));

添加节点。

然后重新编译 source,这次我们找到网络的 public key 和 private key。。 Never leak your network private key..否则别人可以控制整个网络,就悲剧了。

编译完成后,./neocoind makekeypair

得到网络的 publickey:

0403dc07ec3f1215b636179659ed7033015cd775baf5fa76c4f70e5d603002d9a261a3adb3bdcbbb8ddb50f19047 5b3a3cd779d2ba0830d770febcd5d7c310d437

Privatekey 我会告诉你。。



Hmm 我们的项目 还没有一个 logo。。意味着到时候客户端没有 logo。找到一个 logo 在这里  


下面我们来编译让人头痛的 windows 开发环境,而且是在 Ubuntu 系统下面编译

直接用下面的逆天的命令:

apt-get install \

autoconf automake autopoint bash bison bzip2 cmake flex \

gettext git g++ gperf intltool libffi-dev libgdk-pixbuf2.0-dev \

libtool libltdl-dev libssl-dev libxml-parser-perl make openssl \

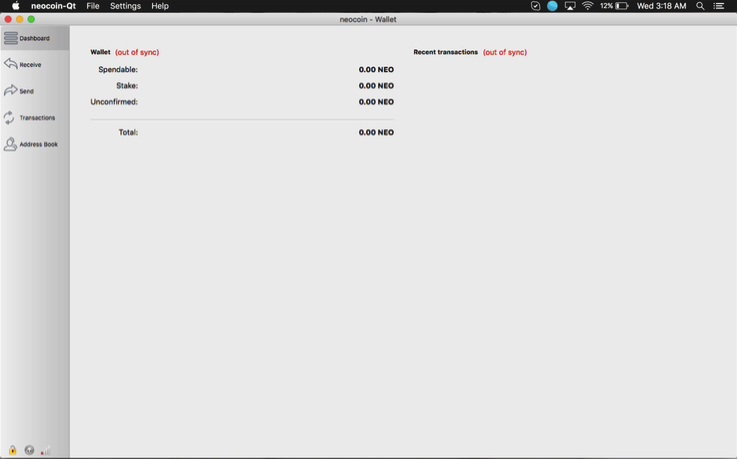
p7zip-full patch perl pkg-config python ruby scons sed \

unzip wget xz-utils

会下载一大堆东西,是什么。。你自己慢慢去 Google 一下,只想说 windows 还好。。到时候编译 MAC clients 的时候,你会发现有多头大。

git clone https://github.com/mxe/mxe.git 下载 cross compiler

make -j2 qt5 boost 编译一个定制化的 QTversion under Linux 需要编译等待 3-5 个小时

这个时候我已经编译好了 mac wallet。为什么这么快? 因为我一直编译过啊  
￼￼￼  


Cool?

到/src/qt/res/images/ 里面修改 logo 吧

可以到这里生成 logo 文件 https://iconverticons.com/online/

Windows 下面 wget http://download.oracle.com/berkeley-db/db-4.8.30.tar.gz

berkley db4.8 用来生成 wallet.dat file

然后修改路径:export PATH=/root/mxe/usr/bin/:${PATH};CC=i686-w64-mingw32-gcc CXX=i686-w64-mingw32-g++

告诉编译终端使用 MXE gcc and g++

然后 ../dist/configure –disable-replication –enable-mingw –enable-cxx –host x86\_64 –prefix=/root/mxe/usr/i686-w64-mingw32.static/  
配置 db

然后 type make 开始编译。

如果你第一次就编译成功了,你可以去买彩票。如果你没有编译成功,恭喜你。你需要花好几天弄明白到 底是什么原因。

但是如果你以为你编译成功,就可以发布自己的虚拟货币或者区块链服务了,那你太天真了。要不然 中本聪如何叫做大神 大神(=@\_\_@=)呐。前后各种版本 100 万行的 bitcoin 代码不是盖的。。如果真想学 习 satoshi nakamoto,一步一步来吧。。这个是科普读物。。被骗了吧?