**构建基于tendermind区块链应用（概念部分）**

**前言：**

本文参考了众多的技术文档和博客，包括不限于下面。

《http://blog.csdn.net/lizhe\_dashuju/article/details/79313071》

《http://chainx.org/news/index/detail/id/24.html》

《https://zhuanlan.zhihu.com/p/33154604》

相关代码请参考如下链接：（其中tendermin core太大无法上传，后面补上）

https://github.com/jenyouhu/tendermint\_hujiyou

https://github.com/jenyouhu/mininet\_hujiyou

1、比特币称为第一代区块链，特点是把通讯、共识、存储、业务逻辑耦合一起

2、以太坊称为第二代区块链，特点是屏蔽了底层的共识、通讯，可以用高级语

言直接开发区块链应用。

3、COSMOS称为第三代区块链，特点是可以在链与链直接交互，而这款神器

正是基于Tendermint 构建的，Tendermin core 是一个共识引擎，可以为

其他任何区块链平台提供 共识插件。

**一、本文面向的读者：**

1、具有区块链的初步知识（了解点对点通讯、共识机制、分布式存储、加密概念）。

2、具有一点点 golong语言的基本知识。

3、懂得CentOS操作系统的基本命令。

4、了解git 版本控制工具的常用命令。

**二、 Tendermint 是什么**

[**Tendermint**](http://link.zhihu.com/?target=https%3A//tendermint.com/)**（以下简称 TM）**，先来了解一下 TM 的原理，TM 主要包含两部分：（稍微啰嗦一下 Tendermin是一家公司，同时也是该公司区块链底层技术产品的名称 ）

1、Tendermint Core：区块链共识引擎。它负责两件事情：节点之间的数据同步有序传输

拜占庭共识机制的实现。（分布式事务管理引擎）

2、ABCI：区块链应用接口，它被设计成一组有接口规范的协议，目的是可以使用多种语言

实现区块链应用逻辑（支持C++,JAVA,GOLANG......包括C语言 ）。

3、Tendermin 的这个架构设计，实际上把区块链的底层技术进行了解耦，[Tendermint Core](https://github.com/tendermint/tendermint" \t "_blank) （“共识引擎”）通过一个满足 [ABCI](https://github.com/tendermint/abci) 标准的 socket 协议与应用进行交流，

4、举个大家比较熟悉的例子，比特币，比特币是一个加密货币区块链，其中的每个节点维护了一个完全经过审计的 UTXO 数据库。如果有人想要在 ABCI 之上创建一个类似比特币的系统.

**Tendermint Core 将会负责：**

1、在节点间共享区块和交易

2、建立交易（区块链）的标准/不可变顺序

**而构建在ABCI上的应用将会负责：**

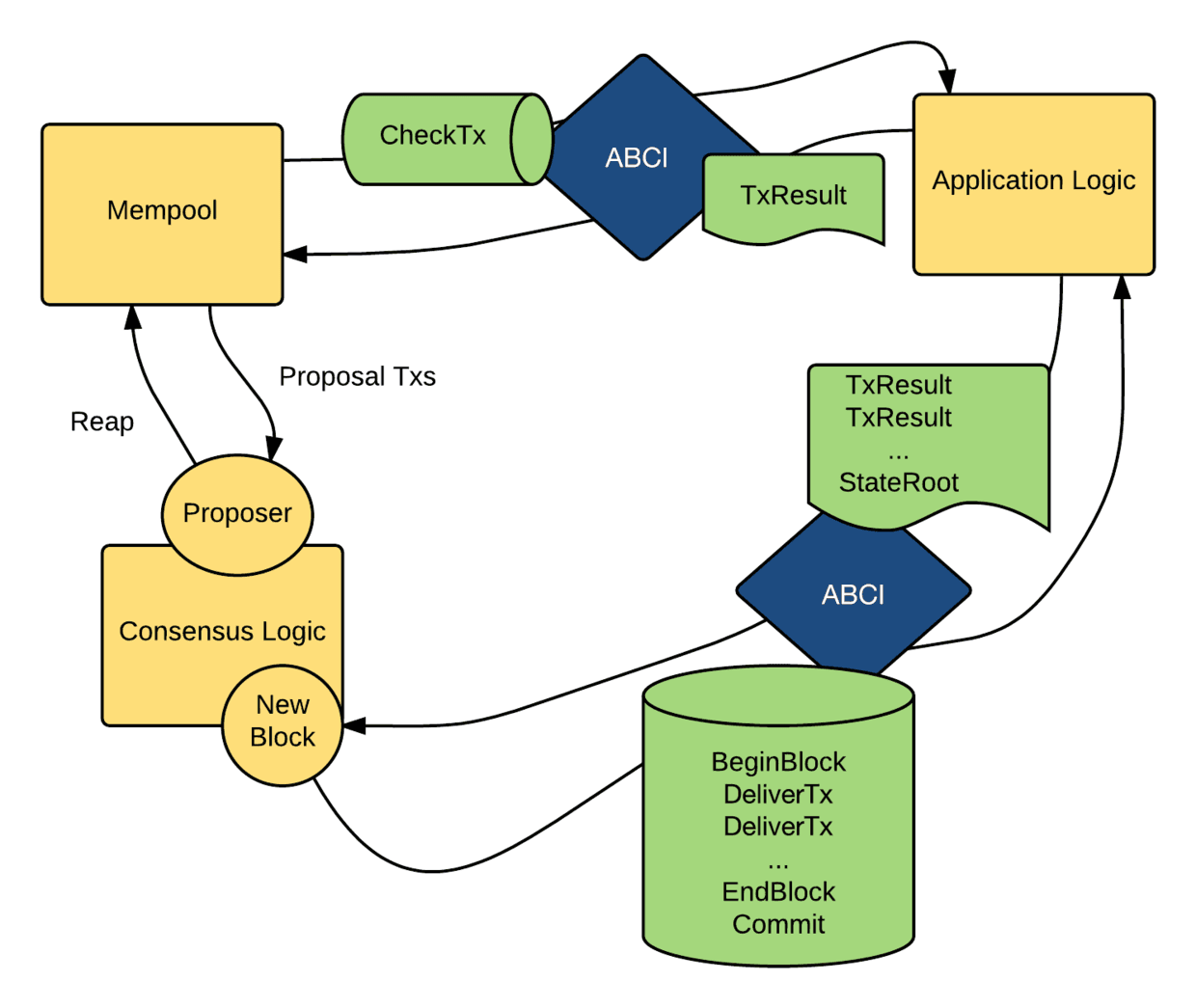
1 、维护 UTXO 数据库

2、验证交易的加密签名

3、阻止花费尚未存在的交易

4、允许客户端查询 UTXO 数据库

Tendermint 能够通过在应用过程和共识过程之间，提供一个非常简单的 API （也就是 ABCI）来分解区块链设计，下面这个图是官方文档里面，用来帮助理解 Tendermint的消息交互过程。



**三、Tendermin官方文档专有名词理解**

**看到下面这堆似是而非的东西，是不是有点蒙圈，别着急，本人开始接触这些概念也是一样, 不过确实有点烧脑啊。**

1、ABCI Server：

2、ABCI Client

3、Tendermint Core

4、ABCI Application

5、ABCI App

我们来这样理解吧，一个基于Tendermint的区块链应用，至少包括 Tendermint 这个底层工具，和我们自己构建的区块链应用，那么。。。。

**Client ：**区块链外部，真正的客户端，通过 Tendermint 提供的服务对区块链

进行读写访问（对于上面就是 **ABCI App**）

**Tendermit :** 区块链内部，（**ABCI Client/Tendermint Core**），启动后与 ABCI 提供的 Socket 服务创建 3 条连接（Connect），同时会提供 HTTP 服务给区块链外部客户端，服务地址：http://localhost:46657

**ABCI**：区块链内部，**ABCI Server**，启动后加载 **ABCI Application/App**，为 Tendermint 提供 Socket 服务，服务地址：tcp://localhost:46658

(**http://localhost:46657**) (**tcp://localhost:46658**)

区块链外部应用**(app)**------------------->**Tendermint Core** ----------------->**ABCI**

(**ABCI Client**) (**ABCI Server**)

**四、Tendermint 工作原理及流程**

还是举例说明，便于理解：

假设吉友哥在使用我写的的区块链记账 APP（用iOS开发的）：

1. Client 就是区块链记账 APP，强东哥记录了一条新数据 ‘Jack Pony $10’，然后保存
2. Client 会发起请求 http://localhost:46657/broadcast\_tx\_commit?tx="Jack Pony $10"
3. Tendermint 收到 tx="Jack Pony $10"，通过 Socket 向 ABCI 发出指令 CheckTx:"Jack Pony $10"
4. ABCI 运行了我写的 ABCI App（用GO开发的）里的 CheckTx 方法，作用是验证 "Jack Pony $10" 是否符合数据规范，如果不符合就要通知 Temdermint 拒绝这条 tx
5. 假设 ABCI CheckTx 验证通过后通知 Tendermint
6. Tendermint 把 "Jack Pony $10" 暂存在内存池（mempool）里，并把这条 tx 通过 P2P 网络复制给其它 Tendermint 节点（杰克马、坡尼马和雷布斯各自运行的 Tendermint 节点）
7. Tendermint 发起了对 "Jack Pony $10" 这条 tx 的拜占庭共识投票，所有4个 Tendermint 节点都参与了。投票过程分三轮，第一轮预投票（PreVote），超过 2/3 认可后进入第二轮预提交（PreCommit），超过 2/3 认可后进入最后一轮正式提交（Commit）
8. Tendermint 提交（Commit）时依次向 ABCI 发送指令：BeginBlock -> DeliverTx \* n次 -> EndBlock -> Commit，大致意思就是：开始接受新区快 -> 正在接受区块内容 \* n条 -> 区块内容接受完了 -> 提交到区块链去吧
9. ABCI 提交（Commit）成功后会通知 Tendermint
10. 至此，区块链多了一个区块记录了 1 条 transaction "Jack Pony $10" ，当然我们也可以记录 n 条后再保存，那么一个区块就会记录 n 条 transaction

**最后再总结几点：**

**abci-cli dummy**

**abci-cli console**

1. 说回刚才那个坑人的 abci-cli，当他运行 'abci-cli dummy' 时其实是运行了 ABCI，当他运行了 'abci-cli console' 时其实是运行了 Tendermint，所以一起运行后会进行 Socket 连接（Connect）
2. Tendermint 和 ABCI 之间的 Socket 连接有 3 条：1 条用于验证数据（CheckTx 指令），1 条用于查询数据（Query 指令），1 条用于共识数据（BeginBlock, DeliverTx, EndBlock, Commit 指令）
3. Tendermit 提供给 Client 的 HTTP 接口有很多，浏览器访问 http://localhost:46657 就都知道了

玩家也可以亲自拜读一下[官方文档](http://link.zhihu.com/?target=https%3A//tendermint.readthedocs.io/en/master/index.html)

**构建基于tendermind区块链应用（操作部分）**

**一、安装git ，可以直接在lunix上用git命令，从github网站下载github上面的项目（CVS版本控制工具）**

yum -y install git

yum -y instatll mercurial

**二、安装golang语言（编程语言）**

1、 由于tendermint 是用golang 语言编写，所以需要安装go语言以及相关编译工具。

2、 下载 go1.9.3.linux-amd64.tar.gz 按照包，直接解压到你在/etc/profile设置的GOROOT目录 (千万别用yum install golang安装，因为默认的版本是 1.8.3 后续一堆的坑)

**三、设置go语言的相关环境变量**

编辑 /etc/profile 文件，加入如下变量

export GOROOT=/home/go/go

export GOBIN=$GOROOT/bin

export GOPKG=$GOROOT/pkg/tool/linux\_amd64

export GOARCH=amd64

export GOOS=linux

export GOPATH=/Golang

export PATH=$PATH:$GOBIN:$GOPKG:$GOPATH/bin

**四、安装glide工具 （go语言用glide 管理依赖库）。**

go get github.com/Masterminds/glide （安装glide，go语言使用glide管理依赖库）

A、编译 **cd** github.**com**/Masterminds/glide

make build

**go** build -**o** glide -ldflags "-X main.version=v0.11.0" glide.**go**

**五、安装tendermint 。**

go get github.com/tendermint/tendermint/cmd/tendermint（安装tendermint源码）

(这个实际比较长，请耐心等待，当然也可以直接从 www.tendermint.com官方网站下载）

在安装 tendermint 源码时，由于无法访问 google , golang.org等网站，导致无法下载

相关库，会报错：例如package golang.org/x/sys/unix: unrecognized import path "golang.org/x/sys/unix" (https fetch: Get https://golang.org/

x/sys/unix?go-get=1: dial tcp 216.239.37.1:443: i/o timeout) 暂时不管。

六**、要编译源码，必须按照下面三个工具**

1、 go get github.com/mitchellh/gox   (安装gox 跨平台编译器)

2、 go get -u github.com/tcnksm/ghr （安装ghr ）

3、 go get -u gopkg.in/alecthomas/gometalinter.v2 （安装gometalinter.v2 ）

$ glide install

七**、执行 gilde install**

在执行 glide install 时，由于无法访问google相关网站，所以先执行下面的步骤，

目的是从 github网站下载与 google.golang.org，golang.org网站对应的关联代码。

1、git clone https://github.com/golang/sys.git （golang.org/x/sys）

2、git clone https://github.com/golang/text.git （golang.org/x/text）

3、git clone https://github.com/golang/net.git （golang.org/x/net）

4、git clone https://gihub.com/golang/crypto.git (golang.org/x/crypto)

5、git clone https://github.com/golang/tools.git (golang.org/x/tools)

6、git clone https://github.com/grpc/grpc-go.git （.google.golang.org/grpc）

7、git clone https://github.com/google/go-genproto.git（google.golang.org/genproto）

**cd $GOPATH/src/github.com/tendermint/tendermint**

在执行前，必须先修改 glide.lock 文件，将里面 google,golang.org网站名称及路径 替换成github网站上对应的文件路径，例如：

将 google.golang.org/grpc 替换成 github.com/grpc/grpc-go

然后，执行 glide install ,最后一行显示 Replacing existing vendor dependencies 表示执行成功。

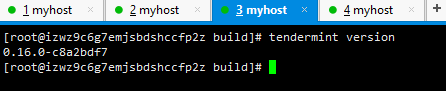
**八、编译 tendermint 代码**

运行 make ；这个地方坑太大了。。。。各种的错误，让你摸不到头脑，到网上去找，发现坑更多了，没办法，还是自力更生，花费了一番功夫才发现，其中最大的问题出在：/tendermint/tendermint/rpc/grpc/types.pb.go 文件里面，该文件是由第三方软件protobuf自动生成的，其中import了一个库文件 context。但关键这该死的 context 库有“双重身份”，在标准库里面有，在/golang.org/x/net/context"也有，更变态的是，在types.pb.go文件中竟然同时引用了 这两个“context”包里面的方法；为了区别他们的不同，我不得不 加入如下行 ：

import cotx "golang.org/x/net/context" ，其中别名取成 cotx 也是没办法，为了区分标准库里面的 context包。

折腾了2天，终于 make all 成功，可执行文件存放在 $GOPATH/src/github.com/tendermint/tendermint/build 下面：

执行tendermint version ，截止到我发稿是0.16.0版。



**九、编译基于tendermint的区块链接口 abci，提供了很多语言接口，包括**

**（** [**C++**](https://github.com/mdyring/cpp-tmsp)**，[JavaScript](https://github.com/tendermint/js-abci" \t "_blank)，[Java](https://github.com/jTendermint/jabci" \t "_blank) 和** [**Erlang**](https://github.com/KrzysiekJ/abci_server)**），本包里面我集成了 C语言接口在**

$GOPATH/src/github.com/tednermint/c-abci目录下。

编译该接口时，需要在：/etc/ld.so.conf.d/local.conf 文件中增加如下两行：

$GOPATH/src/github.com/tendermint/c-abci/lib

$GOPATH/src/github.com/tendermint/c-abci/type/protobuf-c

刷新编辑文件：ldconfig后，直接运行 make 编译，可执行文件存放在

~c-abci/demo/bin/c-dummy。下面主要重点说其他语言的abci接口编译

**1） 安装必要工具如下：**

确保你的Centos 机器上安装了 autoconf、automake、libtool、curle、make

g++,unzip ，如果没有 yum install xxxx就好。

**2） 下载protobuf源码包，解压，编译安装**

地址：https://github.com/google/protobuf/releases

选择Source code (tar.gz)下载

tar -zxvf protobuf-3.1.0.tar.gz -C /usr/local/

cd protobuf-3.1.0/

# 如果使用的不是源码，而是release版本 (已经包含gmock和configure脚本)，可以略过这一步

./autogen.sh

# 指定安装路径

./configure --prefix=/xxxxxxxx/protobuf

#编译

make (需要很长时间，耐心等待 )

# 测试，这一步很耗时间

make check （需要很长时间，耐心等待 ）

make install

# refresh shared library cache.

ldconfig

注意make check这一步会花费比较多的时间

**3） 添加环境变量**

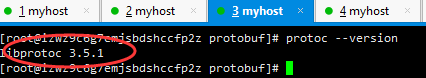
vi /etc/profile 加入

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$GOPATH/src/github.com/gogo/protobuf/lib

export LIBRARY\_PATH=$LIBRARY\_PATH:$GOPATH/src/github.com/gogo/protobuf/lib

export PATH=$PATH:$GOPATH/src/github.com/gogo/protobuf/bin

**4） 编译查看版本protoc --version**

****

**5） 由于abci接口用了 gogo 里面的protobuf**

mkdir gogo\_protobuf ，cd gogo\_protobuf

下载：git clone https://github.com/gogo/protobuf.git

将Makefile 文件中的 下面这行注释掉，因为golang.org网站无法访问，

如果碰到无法访问的类似网站，一律到 github.com网站上去找，替代golang.org网站对应的文件（例如 go get golang.org/x/tools/cmd/benchcmp 需要注释掉）

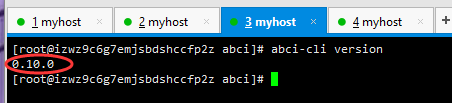
编译：make 成功。

**十 编译 tendermint 下的 abci**

**1) cd $GOPATH/** **github.com/tendermint/abci**

**2) 运行 make 成功后，在当前目录下生产 abci-cli**

**3） 运行 abci-cli version**

****

**至此，断断续续，我已经折腾了 差不多 2天，重要恭喜你成功安装 tendermint 了，**

**并且 是源码安装，源码安装的好处就是，当你成为大牛后，你就可以修改源代码。**

**十一 测试 tendermint , abci-cli是否成功**

**1） 运行官方例子，**

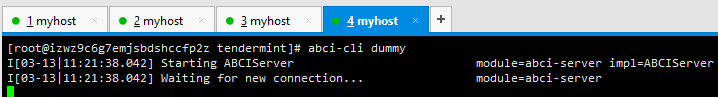
官方文档提供了两个栗子，已经集成在了刚才的安装里：

dummy：一个简单的键值存储区块链应用，使用起来有点像 Redis 或 ElasticSearch。

counter：一个简单的计数器区块链应用，写入区块的数字必须递增，否则将不被区块接受。

这里我们简单介绍一下 dummy 这个栗子，首先启动这个区块链应用，命令行输入：

abci-cli dummy （显示如下，表示成功）

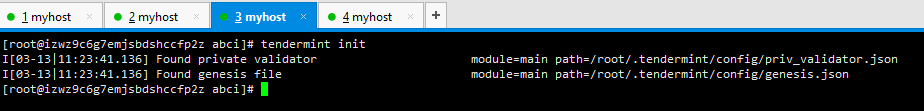


其实上面启动，是作为服务端启动的，其IP地址 和 端口的配置在如下文件中：

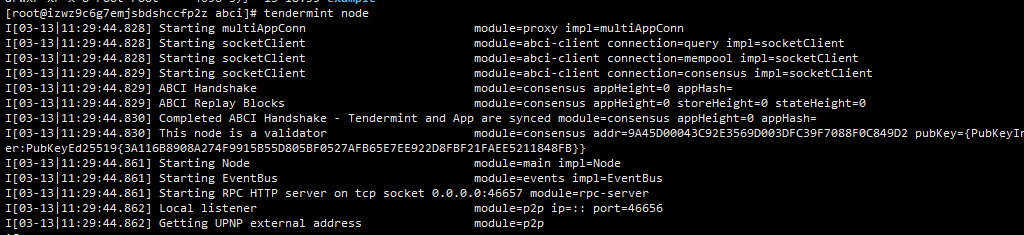
~/tendermint/abci/cmd/abci-cli/abci-cli.go (tcp://127.0.0.1:46658)

然后启动 tendermint ，命令行输入：（作为节点端启动---也就是客户端 ）

tendermint init （下面的显示表示以前启动过,可以用 tendermint unsafe\_reset\_all启动，清除数据，并且重新初始化私人秘钥 ）



tendermint node （显示如下表示成功）



顺利的话，可以看到 abci 和 tendermint 两个程序连通（Connect）了，并且 tendermint 会像心跳一样每一秒提交一个空区块

tendermint 初始化启动成功后，在 /root/.tendermint 目录下有两个目录 data和config ， 其中 data目录下存放数据， config 存放配置文件：

genesis.json ： 公钥文件

priv\_validator.json 私钥文件

addrbook.json 节点地址文件

config.toml tendermint的配置文件（例如IP地址、端口、协议、数据库类型等）

**十二 往区块链里面写实质内容**

**1、**命令行输入：

curl -s 'localhost:46657/broadcast\_tx\_commit?tx="jenyouhu"'

curl -s 'localhost:46657/broadcast\_tx\_commit?tx="name=hujiyou"'

我们的区块链里就有了一个记录为“jenyouhu”的区块和另一个记录为“name=hujiyou”的区块;

**2) 接下来，我们查询一下是否OK**

如果需要对区块链内容进行查询，命令行输入：

curl -s 'localhost:46657/abci\_query?data="name"'

你会发现在 tendermint 节点上显示：

I[03-13|13:06:00.212] ABCIQuery module=rpc path= data=6E616D65 result="log:\"exists\" index:-1 key:\"name\" value:\"hujiyou\" " 表示查询成功了。

curl 是 Linux 常用工具，我们也可以通过网页浏览器直接输入 localhost:46657/abci\_query?data="name"

**十三 构建我们自己的区块链**

总算到了构建自己的区块链的环节，我们先把项目架构整理一下：

1、ClientApp：区块链记账 APP，可以用任意语言任意平台实现

（iOS/Android/H5/其它），这个版本里我们用 GO 做了最简单的控制台命

令行实现，甚至没有 UI 界面。

2、TendermintApp：区块链服务程序，这个里面包括 Tendermint 和 ABCI 两

部分。

3、ABCI：因为是接口协议，设计上可以用任何语言去实现，我们使用了 GO 语

言。开发上主要是实现 CheckTx, DeliverTx, Commit 等 ABCI 指令的具体

逻辑

4、Tendermint：几乎没有需要开发的地方，前面提到了它已经为我们做了两

件事情，P2P网络同步有序传输数据和拜占庭共识引擎。我们在这里就将它

运行即可。

**我们主要讲解一部分核心逻辑：**

**1. ClientApp**

模拟了5个区块，每个区块记录10条记录，将这些记录转成 json 提交给 Tendermint

**1. ClientApp**

模拟了5个区块，每个区块记录10条记录，将这些记录转成 json 提交给 Tendermint

blocksNumber := 5 // how many blocks

transactionsPerBlock := 10 // how many transactions in each block

players := []string{"Lei", "Jack", "Pony", "Richard"} // 4 players

random := rand.New(rand.NewSource(time.Now().UnixNano()))

json := jsoniter.ConfigCompatibleWithStandardLibrary

for i := 0; i < blocksNumber; i++ {

time.Sleep(time.Second \* 1)

transactions := []controllers.Transaction{}

for j := 0; j < transactionsPerBlock; j++ {

from := players[random.Intn(len(players))]

to := players[random.Intn(len(players))]

for from == to {

to = players[random.Intn(len(players))]

}

btc := float32(random.Intn(10) + 1)

tran := controllers.Transaction{

From: from,

To: to,

Bitcoin: btc,

}

\_, \_ = tran.Create()

transactions = append(transactions, tran)

}

bytes, \_ := json.Marshal(&transactions)

data := strings.Replace(string(bytes), "\"", "'", -1)

tx := data

// tmAsync(tx)

tmCommit(tx)

}

func tmCommit(tx string) {

url := "http://localhost:46657/broadcast\_tx\_async?tx=\"" + tx + "\""

txHandle(url)

}

**2. Tendermint**

通过命令行运行一个定义好的 Shell 脚本，并且把运行结果打印到控制台和日志文件中去

f, err := os.Create("logs/tendermint.log")

if err != nil {

fmt.Println("Tendermint log init error:", err)

}

multiWriter := io.MultiWriter(f, os.Stdout)

go func() {

cmd := exec.Command("bash", "-c", "sh run-tm.sh")

cmd.Stdout = multiWriter

cmd.Start()

}()

run-tm.sh 脚本内容：

echo tendermint start

tendermint init

tendermint unsafe\_reset\_all

tendermint node --consensus.create\_empty\_blocks=false

echo tendermint end

*unsafe\_reset\_all 作用是每次重置本机的区块链数据，仅供开发使用* *consensus.create\_empty\_blocks 作用是关闭 Tendermint 自带的每秒生成新的空区块功能*

**3. ABCI**

实现 ABCI 指令接口，这里我们直接使用了官方栗子 dummy 应用，来保存提交的 json 记录，并且在每个实现接口里做了日志打印（因篇幅有限，省略了长长的代码）

func (app \*DummyApplication) CheckTx(tx []byte) types.ResponseCheckTx {

lib.Log.Debug("CheckTx")

lib.Log.Notice(string(tx))

return types.ResponseCheckTx{Code: code.CodeTypeOK}

}

func (app \*DummyApplication) DeliverTx(tx []byte) types.ResponseDeliverTx {

lib.Log.Debug("DeliverTx")

lib.Log.Notice(string(tx))

// ...

return types.ResponseDeliverTx{Code: code.CodeTypeOK, Tags: tags}

}

func (app \*DummyApplication) Commit() types.ResponseCommit {

lib.Log.Debug("Commit")

// ...

lib.Log.Debug("Commit Hash", hash)

return types.ResponseCommit{Code: code.CodeTypeOK, Data: hash}

}

func (app \*DummyApplication) Query(reqQuery types.RequestQuery) (resQuery types.ResponseQuery) {

lib.Log.Debug("Query")

// ...

}

**4. 编译**

我们写好了编译脚本，命令行输入：

sh make.sh

编译脚本本质就是把三个运行程序编译出来：

go build ./TendermintApp/ABCIServer/

go build ./TendermintApp/ABCIClient/

go build -o Client ./ClientApp/

**5. 测试**

到了激动人心的一刻，我们已经离一条自造的区块链很接近了。

先运行 ABCI，命令行输入:

./ABCIServer

再运行 Tendermint，命令行输入：

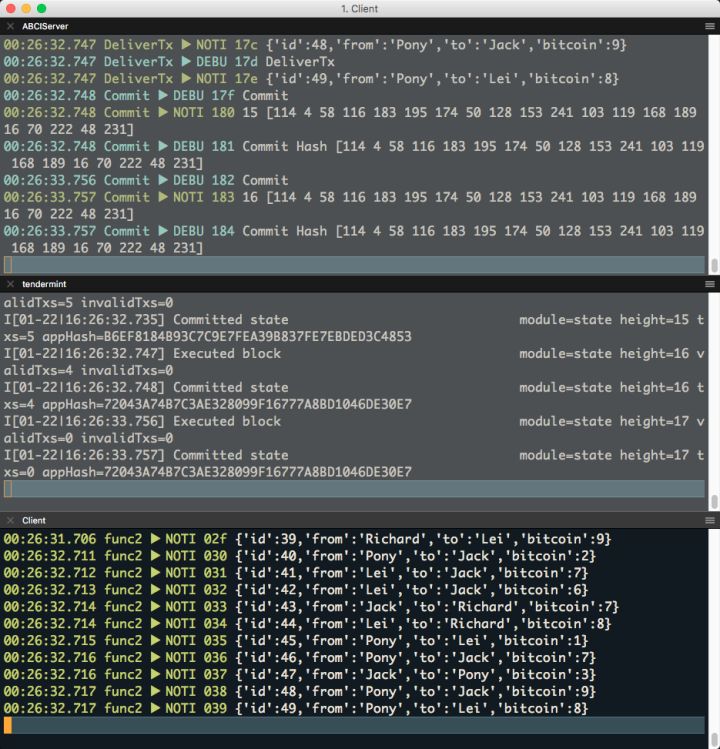
./ABCIClient

可以看到 Tendermint 和 ABCI 已经有 3 个连接（Connect） Socket 握手了，表示整个区块链服务已经准备就绪，最后我们只需要使用 Client 往区块链里写数据就行了，命令行输入：

./Client

把条命令理解成使用记账 APP 记录了 5 页，每页 10 条记录，然后点击保存后送去了区块链。

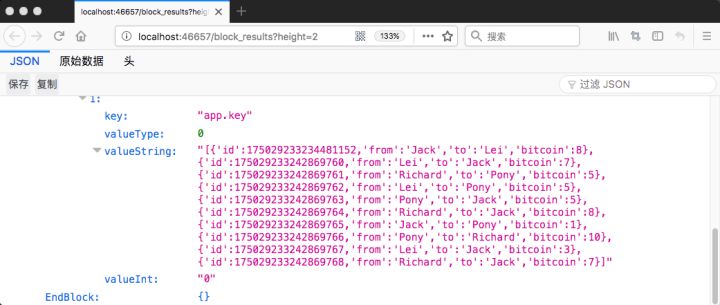
附上一张运行结果图，大致能观察到三个程序都已成功运行

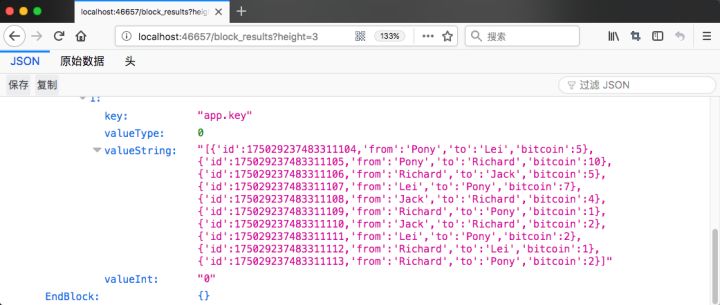


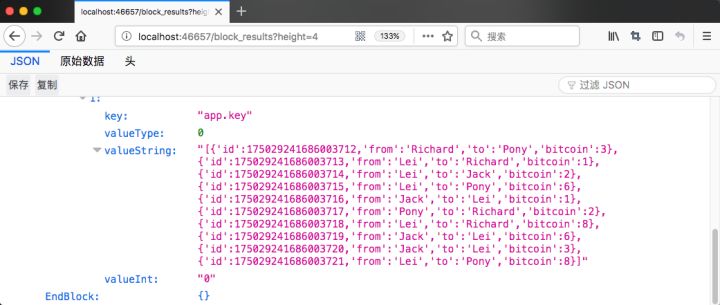
最终我们创造了一条区块链，一共有 7 个区块，2~6 号区块各自写有 10 条账本记录，第 1 和第 7 号区块是系统创建的空区块。

*空区块是怎么回事？原先 PO 主也以为是 BUG，去了官方的开发* [*issue*](http://link.zhihu.com/?target=https%3A//github.com/tendermint/ethermint/issues/308) *里了解到 Tendermint 每次收到新的 tx 或者区块链 hash 值变化的时候（就是区块链状态变了）会产生一个新区块去接受 mempool 里未被提交的新 tx，这是 Tendermint 用于正常自检工作产生的。*

来看看这条区块链长什么样子，这里展示了 2, 3, 4 号区块的账本记录







**十四 tendermint管理发布工具Mininet 安装介绍**

1） Mininet源码安装方式

下载源码：git clone http://github.com/mininet/mininet

安装：cd mininet/util

./install.sh -a (全部安装）

安装报错： Mininet 不支持 CentOS ，于是漫长的流程走下去：

**a. yum -y install openssh-server**

**b, chkconfig sshd on**

**c, service sshd start**

– Disable SELinux to get OVSDB to stasrt  
sudo setenforce Permissive

– Modify sudoers secure\_path to add /usr/local/bin so the ‘controller’

which be found. 设置环境变量 : /etc/profile

secure\_path = /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin

**D，修改 install.sh 文件如下：**

修改mininet/util/install.sh：  
\*\*\*ADD the following before the line ‘test -e /etc/fedora-release && DIST=”Fedora”‘. Somewhere around line 47.  May differ.

**增加如下内容**   
test -e /etc/centos-release && DIST="CentOS"  
if [ "$DIST" = "CentOS" ]; then  
    install='sudo yum -y install'  
    remove='sudo yum -y erase'  
    pkginst='sudo rpm -ivh'  
    # Prereqs for this script  
    if ! which lsb\_release &> /dev/null; then  
        $install redhat-lsb-core  
    fi  
fi  
\*\*\*修改如下

if ! echo $DIST | egrep 'Ubuntu|Debian|Fedora|CentOS'; then  
    echo "Install.sh currently only supports Ubuntu, Debian and Fedora."  
    exit 1  
fi

**./install.sh -a** 执行安装 mininet 成功

但在运行 **mn** 是，部分报错，那是因为需要安装 **openvswitch . 见下面的步骤。**

**E，还需要安装 openvswitch**

**1, 安装依赖包** yum -y install make gcc openssl-devel autoconf automake rpm-build redhat-rpm-config

yum -y install python-devel openssl-devel kernel-devel kernel-debug-devel libtool wget

**2.预处理：**

**[plain]** [view plain](http://blog.csdn.net/xinxing__8185/article/details/51900444) [copy](http://blog.csdn.net/xinxing__8185/article/details/51900444)

1. mkdir -p ~/rpmbuild/SOURCES
2. wget http://openvswitch.org/releases/openvswitch-2.5.0.tar.gz
3. cp openvswitch-2.5.0.tar.gz ~/rpmbuild/SOURCES/
4. tar xfz openvswitch-2.5.0.tar.gz
5. sed 's/openvswitch-kmod, //g' openvswitch-2.5.0/rhel/openvswitch.spec > openvswitch-2.5.0/rhel/openvswitch\_no\_kmod.spec

**3.构建RPM包：**

**[plain]** [view plain](http://blog.csdn.net/xinxing__8185/article/details/51900444) [copy](http://blog.csdn.net/xinxing__8185/article/details/51900444)

1. rpmbuild -bb --nocheck ~/rpmbuild/SOURCES /openvswitch-2.5.0/rhel/openvswitch\_no\_kmod.spec

**4.安装：**

**[plain]** [view plain](http://blog.csdn.net/xinxing__8185/article/details/51900444) [copy](http://blog.csdn.net/xinxing__8185/article/details/51900444)

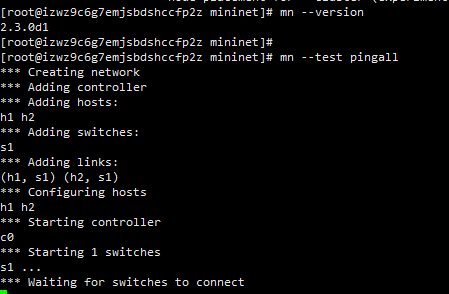
1. yum localinstall ~/rpmbuild/RPMS/x86\_64/openvswitch-2.5.0-1.x86\_64.rpm

**5.启动相关服务：**

**[plain]** [view plain](http://blog.csdn.net/xinxing__8185/article/details/51900444) [copy](http://blog.csdn.net/xinxing__8185/article/details/51900444)

1. systemctl start openvswitch.service

**最后： 运行 mn --test pingall 成功**

****

**十五 tendermint管理发布介绍（测试网络）**

**1、如何手工部署 Tendermint 验证节点（验证节点拥有区块的所有交易数据，全能节点）**

1、 选择一个云服务提供商（例如阿里）。

2、在所有的验证节点上，安装Tendermin 和区块链应用。

3、在每个节点上用 tendermint gen\_validator 命令生成私钥钥，私钥存放在

root/.tendermin/ priv\_validator.json 文件中，将该文件中的

4、将每个验证节点的公钥全部加入 genesis.json中。

5、再每个节点上运行如下：（默认的端口号为 46656，IP和端口可设定 ）

tendermint node --p2p.persistent\_peers=192.168.0.1:46656,192.168.0.2:46656,192.168.0.3:46656,192.168.0.4:46656

6、为了防止共识失败，所以至少需要四个验证节点，这四个节点互相连接，开始生成区块链数据。

**2**, **增加一个非验证节点，非常简单**

只需要将原始的  genesis.json  文件拷贝到 ~/.tendermint/config 下，然后启动节点就可以了。

**附录：Git常用命令**

