清华大学

本科生毕业设计论文

**学院：**新雅书院

**专业：**计算机科学与技术

**班级：**新雅91/计92

**姓名：**狄永正

**指导教师：**向勇

二○二三年 一月 四日

**树状区块链的跨链操作的设计研究**

**摘 要**

本课题首先基于现有项目结构完成对于树状区域搜索区块链的现有功能的性能测试，基于出租车调度系统，部署并完成树状区块链的性能测试实验；最后，在现有的基础上完成树状区域搜索区块链对于跨链跨区域搜索的实践实验。

**目 录**

摘 要 ............................................................................................................................ I

Abstract .........................................................................................................................II

第 1 章 绪论 ............................................................................................................... 1

1.1 研究背景 ........................................................................................................... 1

1.2 相关技术调研 ................................................................................................... 2

1.2.1 区块链技术概述 ........................................................................................ 2

1.2.2 智能合约概述 ............................................................................................ 3

1.2.3 区域索引区块链和树状区块链概述 ........................................................ 3

1.3 本文研究内容及贡献 ....................................................................................... 5 第 2 章 基于区域索引区块链的出租车调度系统复现 ..........................................6

2.1 环境配置 ........................................................................................................... 6

2.2 复现步骤 ........................................................................................................... 7

2.2.1 建立区域索引区块链 ................................................................................ 7

2.2.2 部署合约 .................................................................................................... 8

2.2.3 上传地图数据 ............................................................................................ 9

结 论 ..........................................................................................................................50

**第 1 章 绪论**

**1.1 研究背景**

现阶段，城市中越来越多的流动车辆的增加极大的增大了城市的交通压力，在当下社会中，人们都过着快节奏的生活，对交通的便捷性需求自然日益增长。越来越多的人会选择计程车或网约车出行；计程车，网约车有着针对性强，更为便捷，流动基数更多等特点，也切实的便捷了人们的出行。

随着智能车的发展与普及，车辆之间的交互将变得必不可少，为营造更安全高效的交通环境。决定使用区块链技术来构建车辆与路测节点间的特殊自组网，来进行车辆间或车辆与路侧节点间的数据交互。区块链的去中心化、独立性、匿名性等特点保证了该技术的透明安全性。

使用区块链技术可以去除中心管制，并基于此建立可信任、去中心化的数据库，创造共享式经济[[[1]](#endnote-1)]。同时，在出租车调度系统中应用该技术可以消除中介，允许乘客与司机的直接交流与交易，能够为双方提供更为可信的验证，降低信任成本。在网络出租车服务中使用区块链技术有助于参与的所有利益相关方关系更加紧密[[[2]](#endnote-2)]。

本课题首先基于现有项目结构完成对于树状区域搜索区块链的现有功能的性能测试，基于出租车调度系统，部署并完成树状区块链的性能测试实验；最后，在现有的基础上完成树状区域搜索区块链对于跨链跨区域搜索的实践实验。

**1.2 相关技术调研**

1.2.1区块链概述

区块链技术（Blockchain technology，简称BT）是利用块链式数据结构来验证与存储数据，利用分布式节点共识算法来生成和更新数据，利用密码学的方式保证数据传输和访问的安全，利用由自动化脚本代码组成的智能合约来编程和操作数据的一种全新的分布式基础架构与计算范式。用区块链技术所串接的分布式账本能让两方有效纪录交易，且可永久查验此交易。区块链技术具有去中心化、去信任化、匿名化、可扩展、独立性、安全性等特点，可应用到各种安全交易之中。

区块链可以创建无法篡改且端到端加密的记录，有助于防止欺诈和未经授权的活动。隐私问题也可以在区块链上得以解决，解决方式包括匿名个人数据以及限制访问权限等。信息存储在整个计算机网络上，而不是单个服务器上，这使得黑客很难查看数据。

更大的透明度：由于区块链使用分布式账本，导致交易和数据在多个位置采取完全相同的方式进行记录。所有具有访问权限的网络参与者都能同时查看相同的信息，从而实现信息的完全透明性。所有交易记录均不可篡改，并带有时间和日期戳。这使成员能够查看交易的整个历史记录，并几乎消除了任何欺诈机会。

去中心化：在区块链系统中，没有任何一个机构或个人可以实现对全局数据的控制，而任一节点停止工作都不会影响系统整体运作，这种去中心化特点使得区块链具有更高的可靠性和安全性。

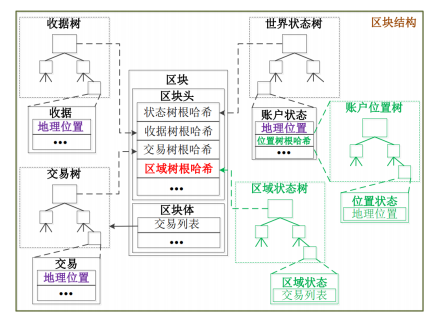
降低成本、提高效率：区块链技术可以通过去除中间商、简化流程、提高效率等方式降低成本、提高效率。

1.2.2区域索引的树状区块链

树状区块链是一种多重树状区块结构，以主链为根，通过分叉的方式不断生成支链为枝叶的多重树状区块结构。本课题的技术背景便是实现了支持区域搜索的树状多链区块链。

现阶段实验室前辈实现的树状区块链的主要结构大致为将区块分为创世块、分支区块和叶子区块三种；结合 GeoHash 编码技术，不再采用传统区块链的单链结构，形成类似于字典树的树状结构；同时，为了满足快速查询的需要，在区块中增添了一些辅助数据结构。经过以上改良，树状区块链可以在对地理位置敏感、且网络结构变化较频繁的应用场景中，发挥相较传统区块链更好的理论性能。

如图 1-1 所示，相较以太坊官方实现的传统区块链而言，区域索引区块链在区块头中加入了区域状态树的树根哈希，以支持基于位置的快速信息查询；同时，为追踪每个账户的包括位置信息的完整状态，在账户状态数据结构中还加入了当前的地理位置字段、和账户位置树这一数据结构。不仅如此，在记录交易、收据时，均会记录发起动作的地理位置信息，并以此更新发起人的账户的当前位置及账户位置树。文献[5]中指出，采用3到6位 GeoHash 编码时，区域索引区块链相较传统的无索引区块链，执行相同查询的耗时仅为后者的 5.3%



区域索引区块链仍然保留了传统区块链的单链结构，当区块数量很大时，其查询效率仍然会随之降低。因此，周畅提出并设计了基于区域索引区块链改良的树状区域索引区块链（下简称树状区块链）。其按照 Geohash 编码长度，表示父链和子链的关系，并划分区域子链。树状区块链的区块也分为三种：叶子区块、分支区块和创世块。叶子区块和传统的区块链并无太大差异，叶子区块及其后续区块均采用传统的单链结构加以组织；分支区块则负责将数个叶子区块组织起来，按照叶子区块所代表的地理位置，创世块和分支区块十分相似，但它没有父链指针。上述树状区块链的设计，进一步缩短了区块链网络中的单链长度，进而提升了查询效率。

1.2.3 智能合约

智能合约（Smart Contract）这一概念由 Nick Szabo 于 1994 年提出。在比特币的支付模型中，仅存在一个简单的堆栈计算机。由于其可用的操作方法并非图灵完备，只能执行比较简单的操作，从而限制了区块链的应用场景。智能合约的出现打破了这一局面。在 Szabo 于 1996 年撰写的《Smart Contracts: Building Blocks for Digital Markets》一文[4]中，Nick 设想智能合约就是运行在区块链上的一段程序，当满足某种条件时，相应代码将自动被执行，而该过程人类无需也无法介入。智能合约一定程度上避免了交易双方抵赖的问题，并且其图灵完备的特性也令区块链技术在不同应用场景下的适应性大大增加了。

**1.3 本文研究内容及贡献**

本文将在车联网这一应用场景下，首先基于区域索引区块链，复现实验室已有工作——出租车调度系统的有关工作，证明该系统的可用性。此后，设计并进行树状区域索引区块链在单父链双子链的网络结构下的跨链转账实验，探究树状区块链在不同转账请求压力下的性能表现，验证其功能可用性，并测试跨链操作带来的额外时间开销。作为测试实验的收尾，本文将设计实验，令出租车调度系统分别在网络结构不同的树状区块链上运行，收集并可视化实验数据，评估树状区块链在不同工况的实际应用场景下的性能表现。

1. 基于区域索引区块链，复现实验室已有工作——出租车调度系统，证明该系统的可用性；编写帮助文档和实验日志，以便后人推进研究

2. 设计并进行树状区域索引区块链在单父链双子链的网络结构下的跨链转账实

验，记录其对 10 个、20 个、40 个、80 个、120 个、160 个、200 个账号执行子链之间的跨链转账的耗时、吞吐量等数据，并进行可视化；编写帮助文档和实验日志，以便后人推进研究

3. 在单子链、单父链四子链的网络结构下分别运行出租车调度系统，探究树状区块链在不同工况实际场景中的性能表现，并进行数据可视化；编写帮助文档和实验日志，以便后人推进研究

4.

**第 2 章 go-ethereum树状区块链的部分源码解读**

**2.1 源码整体功能及结构**

**2.1.1区块链实现**

区块链的实现是go-ethereum最重要的一部分，它包含了区块链的数据结构、验证、同步等功能。整个区块链数据结构是由Block、Header、Transaction和Receipt等组成的。其中Block包含了Header和Transaction的列表，而Header则包含了一些关于区块的元数据，如父块的哈希值、时间戳、难度等。Transaction则包含了交易的信息，而Receipt则包含了交易执行的结果。

区块链的验证是由core/blockchain包中的Blockchain结构体实现的。它维护了整个区块链的状态，并提供了一系列的方法，如验证区块的合法性、获取最新的区块等。同时，Blockchain还实现了一个状态机，用于执行交易。 区块链的同步是由p2p包中的peer和server实现的。当一个新的节点加入网络时，它需要同步整个区块链。在同步过程中，节点会向其他节点请求缺失的区块，直到同步完整个区块链。

**2.1.2 交易执行**

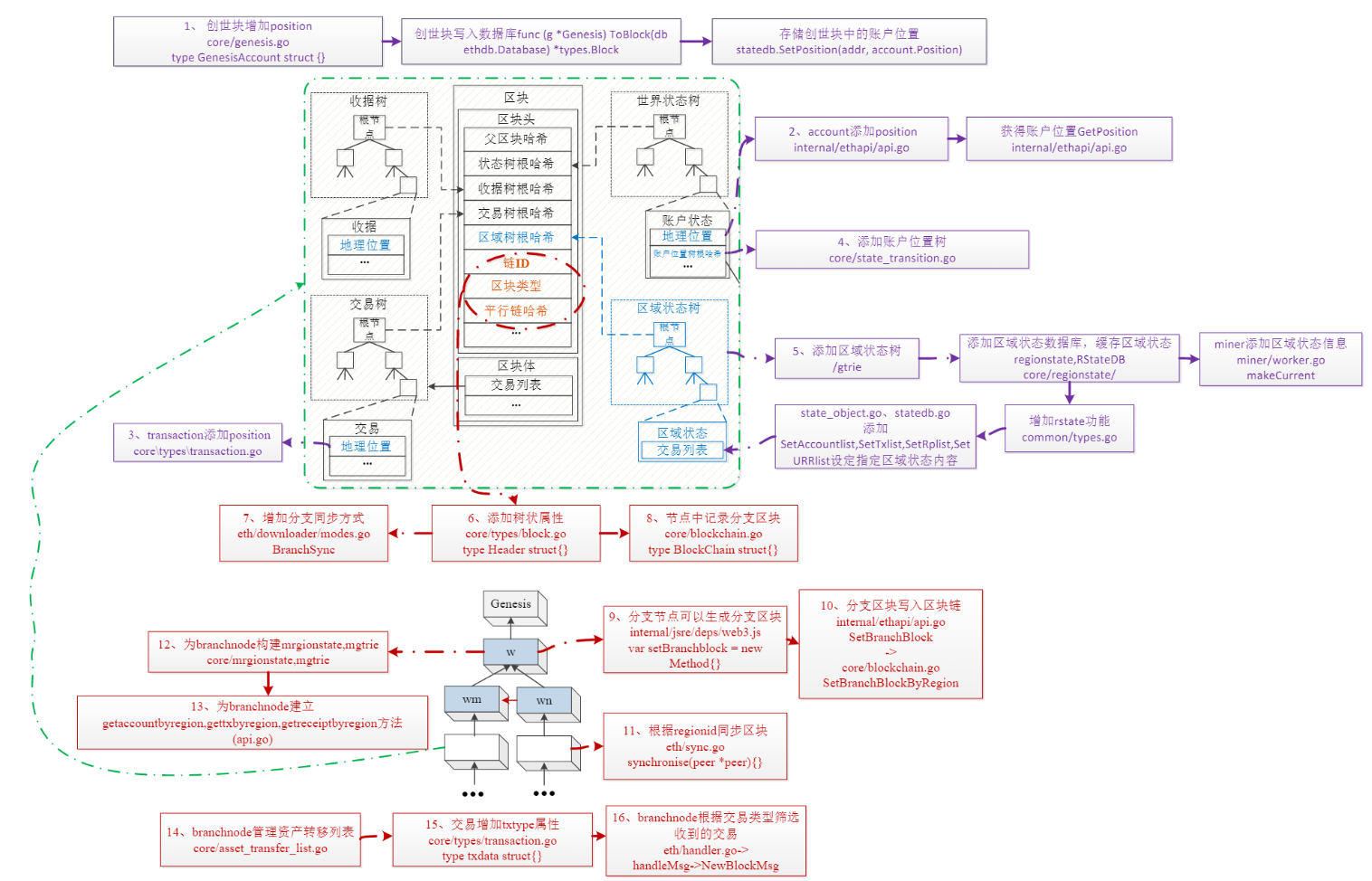
交易执行是go-ethereum的另一个重要部分。它实现了以太坊的EVM虚拟机，用于执行智能合约的字节码。在交易执行的过程中，EVM会执行一系列的操作码，如加法、减法、乘法、除法等，以及读写内存、读写存储等。 交易执行还包括一些复杂的逻辑，如合约的创建、合约的调用等。当一个合约被创建时，它会被编译成字节码，并存储到区块链中。当需要调用合约时，交易中会包含一些输入参数，EVM会根据这些参数执行对应的操作码，并最终返回一个输出结果。

**2.1.3 代码结构**

go-ethereum的代码结构非常清晰，整个代码库主要分为以下几个部分：

* accounts：这是管理以太坊账户的代码，包括账户管理、加密和解密、签名和验证等。
* cmd: 包含所有go-ethereum的命令行工具，如geth、abigen、bootnode等。
* core: 核心代码，包括区块链数据结构、区块链的实现、账户管理、交易执行等。
* crypto: 加密相关的代码，如私钥生成、签名等。
* consensus：这是区块链共识算法的实现，包括PoW（Proof of Work）、PoA、Ethash等。
* eth：以太坊网络协议的实现，包括区块同步、交易广播、状态传播和客户端协议等。
* internal：包含整个项目中使用的内部包。
* miner: 包含了以太坊矿工相关的代码，如挖矿、打包交易、广播区块等。
* node: 包含了以太坊节点相关的代码，如节点的启动、关闭、管理等。
* p2p: 网络层的实现，用于节点之间的通信。
* params: 包含了以太坊的参数配置，如区块链难度、网络ID、区块奖励等。
* storage：包含了以太坊的存储实现，如LevelDB等。
* rpc: 实现了以太坊的JSON-RPC API、WebSocket和IPC等。
* whisper: 实现了以太坊的whisper协议，用于点对点的消息传递，实现安全、私密的通信。
* trie: 包含了以太坊中使用的Merkle Patricia Trie数据结构相关的代码，如节点的添加、删除、查找等。

**2.2 部分核心源码说明**

****

由于笔者毕设的实验目标与跨链与转账有关，故笔者主要重点研究了树状区块链的blockchain结构，以及转账操作中涉及的代码内容。代码说明仓库地址：

2.2.1 针对区域索引的代码说明

创世块增加position core/genesis.go type GenesisAccount struct {}

创世块写入数据库func (g \*Genesis) ToBlock(db ethdb.Database) \*types.Block存储创世块中的账户位置 statedb.SetPosition(addr, account.Position)

account添加position internal/ethapi/api.go获得账户位置GetPosition internal/ethapi/api.go

transaction添加position core\types\transaction.go

添加账户位置树 core/state\_transition.go

添加区域状态树 /gtrie

添加区域状态数据库，缓存区域状态 regionstate,RStateDB core/regionstate/

miner添加区域状态信息 miner/worker.go makeCurrent

增加rstate功能 common/types.go

state\_object.go、statedb.go 添加 SetAccountlist,SetTxlist,SetRplist,Set URRlist设定指定区域状态内容

2.2.2 针对树状多链的代码说明

添加树状属性 core/types/block.go type Header struct{}

增加分支同步方式 eth/downloader/modes.go BranchSync

节点中记录分支区块 core/blockchain.go type BlockChain struct{}

分支节点可以生成分支区块 internal/jsre/deps/web3.js var setBranchblock = new Method{}

分支区块写入区块链 internal/ethapi/api.go SetBranchBlock -> core/blockchain.go SetBranchBlockByRegion

根据regionid同步区块 eth/sync.go synchronise(peer \*peer){}

为branchnode构建mrgionstate,mgtrie core/mrgionstate,mgtrie

为branchnode建立 getaccountbyregion,gettxbyregion,getreceiptbyregion方法 (api.go)

2.2.3 针对资产转移的代码说明

branchnode管理资产转移列表 core/asset\_transfer\_list.go

交易增加txtype属性 core/types/transaction.go type txdata struct{}

branchnode根据交易类型筛选 收到的交易 eth/handler.go-> handleMsg->NewBlockMsg2.2.2 针对树状多链的修改

**2.2 本章小结**

本章对现有系统使用的树状区块链源码中的部分功能所涉及到的源码作了注释，详细说明在仓库中：

主要作用是方便后续工作者可以更好的理清代码逻辑，进而得以对于整个系统的运作流程更为清晰。同时，也便于未来可能的对现有源码的底层的改进工作的进行。

**第 3 章 基于区域索引区块链的出租车调度系统复现**

本章的主要内容是实现基于区域索引的树状区块链的出租车调度系统的复现工作；目前该课题已实现的出租车调度系统，利用geohash编码存储地图数据，实现并部署有路径规划以及车辆匹配的算法合约，并且使用Vue 2.JS 作为前端可以表现出乘客与车辆的操作界面。其中用户可以通过实时地图信息获得自己的位置以及附近在线的乘客、司机等信息。乘客在发出乘车请求后，系统自动匹配附近的符合要求的车辆信息，司机在接到订单信息后，可以选择是否接受订单。司机接受订单后，进行路径规划。车辆在接到乘客后，乘客可以确认上车，司机则是确认到达乘客上车地点。在抵达乘客目标后，乘客需要支付订单，确认订单支付后，调度系统完成。

复现工作大致分为两步：第一步是实现区域索引区块链的出租车调度系统的复现工作，该复现工作主要基于万琦玲前辈的复现手册进行。第二步则是进一步实现基于树状区块链的区域索引出租车调度系统。

实验复现的过程，笔者已形成操作文档放入毕设仓库当中：

供后来者参考。

本章节大致介绍一下复现的流程，主要内容为笔者在这一复现过程中所遇到的问题与解决方案，以及笔者对复现文档做出的改进与调整。当下的复现文档相较于初版大大提高了可理解性与可扩展性。

**3.1区域索引区块链的出租车调度系统**

**3.1.1 环境配置**

操作系统 Ubuntu 22.04.1 LTS

虚拟机 VMWare Workstation Pro 17

**一些**JavaScript库：npm、truffle、node.js、ethereum、web3.js等

区域索引区块链的二进制可执行文件（代码仓库中的 geth1 二进制可执

行文件）存放到/usr/local/bin 文件夹下

**3.1.2 实验步骤**

实验复现的过程，笔者已形成操作文档放入毕设仓库当中：

具体可以参考从《1 传统区块链初始化和启动》到《7 调度系统复现实验》的这一过程，其中本节重点说一下《7 调度系统复现实验》的过程，其余实验均为调度系统复现工作的前置测试实验，用于理解并测试系统的相关功能所用。

**3.1.2.1 初始化并启动区域索引区块链**

首先配置genesis.json，配置创世块文件，之后，命令行运行如下指令实现初始化区块链：

```

geth1 --identity "MyEth" --rpc --rpcaddr 127.0.0.1 --rpcport "8545" --rpccorsdomain "\*" --datadir gethdata --port "30303" --nodiscover --rpcapi "eth,net,personal,web3" --networkid 91036 init genesis.json

```

指令参数解读

–identity "MyEth"： 设置节点的标识。

–rpc： 启用RPC服务器。

–rpcaddr 127.0.0.1： 指定RPC服务器的IP地址。

–rpcport “8545”： 指定RPC服务器的端口号。

（外部程序可以使用该端口接入区块链，进而借助 JSON-RPC API 或者 web3.js 库和区块链进行交互。）

–datadir gethdata： 指定链上数据存储目录。

–port “30303”： 指定节点端口号。

–rpcapi “eth,net,personal,web3”：启用RPC API。

–networkid 91036： 指定网络ID。

init genesis.json： 使用genesis.json文件初始化区块链。

命令行运行如下指令启动区块链：

```

geth1 --datadir ./gethdata --networkid 91036 --port 30303 --rpc --rpcaddr 127.0.0.1 --rpcport 8545 --rpcapi 'personal,net,eth,web3,admin' --rpccorsdomain='\*' --ws --wsaddr='localhost' --wsport 8546 --wsorigins='\*' --wsapi 'personal,net,eth,web3,admin' --nodiscover --allow-insecure-unlock --dev.period 1 --syncmode='full' console

```

启动区块链后，终端中将出现 JavaScript 控制台，可以使用 JavaScript 编程语言的一个子集和链进行交互。

此时，区块链已经启动完毕，可以在控制台创建账户并进行测试；需要注意的操作有：1. 创建账号后，需要将账户信息添加到genesis.json的alloc字段中，同时赋予创建的账号初始余额，以方便后续的调度实验的进行；2. 每次启动区块链都要解锁账户，否则账户无法部署合约，进而系统无法工作

在打开的控制台中输入exit退出控制台，然后删除目录./gethdata/geth。随后，再运行一次初始化区块链和启动区块链的代码，此操作是为了强制重新加载创世块文件。此时，所有账户应该都有余额了。可以用eth.geBalance(账户地址)来检查余额，余额显示正常则表明区块链已成功建立。

**3.1.2.2 部署合约**

这一步，笔者并未遇到什么问题，主要介绍一下系统所用到的两个合约：

StoreMap.sol：主要有存储各种地图数据的数据结构以及查询方法，此外还提供了A-Star寻路等算法的实现

StoreTraffic.sol：主要提供了司机和乘客的各项信息的管理，导航结果，以及基于geohash的对于调度车辆的查询算法

在部署合约的这一步操作中，源复现文档仅给出了合约编译后的abi以及bytecode内容，笔者在文档中添加了Remix Desktop编译合约并记录编译结果中的ABI字段并进行进行压缩转义，同时记录bytecode（下称字节码）字段的这一过程。

将二者复制到部署合约的代码模板中，并将两份编辑好的模板复制到正在运行 geth1 的 JavaScript命令行后，合约部署的请求就已经提交至交易池。开始挖矿并密切观察控制台输出，直至观察到形似以下格式的输出。此即为两份合约各自的合约地址。

null [object Object]

Contract mined! Address: 0x23b98f92ceac005e570b6768da377b3abd11012e

[object Object]

null [object Object]

Contract mined! Address: 0xfa6b8f0b92b323c28557faf69da028e33856f6ca

[object Object]

**3.1.2.3 上传地图**

这一步是直接调用了现有系统的上传地图的JavaScript 脚本文件，注意需要修改StoreMap的合约地址，初始的执行数据提交的账号的公钥地址，以及要上传的地图数据文件（可选，在后续实验中修改了地图数据文件）运行该脚本，直至终端输出“地图数据上传完成”字样后，结束挖矿。此时，地图数据便已成功的上传到了区块链上。

**3.1.2.4 更改文件以加入账户信息**按

照文档说明，在仓库的investigation-cjzhuang2020/cjz\_underg\_2021\_09路径下，找到如下文件：

乘客账户文件，车辆账户文件，StoreMap合约地址文件StoreTraffic合约地址文件，乘客的位置信息文件（包括乘客的账户信息，初始位置，调度起点，调度终点），以及车辆的位置信息文件（包括车辆的账户信息，车辆的初始位置）

将上述文件进行修改后，即可启动测试

**3.1.2.4 启动实验**

启动挖矿，新建两个终端，准备启动车辆客户端和乘客端的测试脚本。

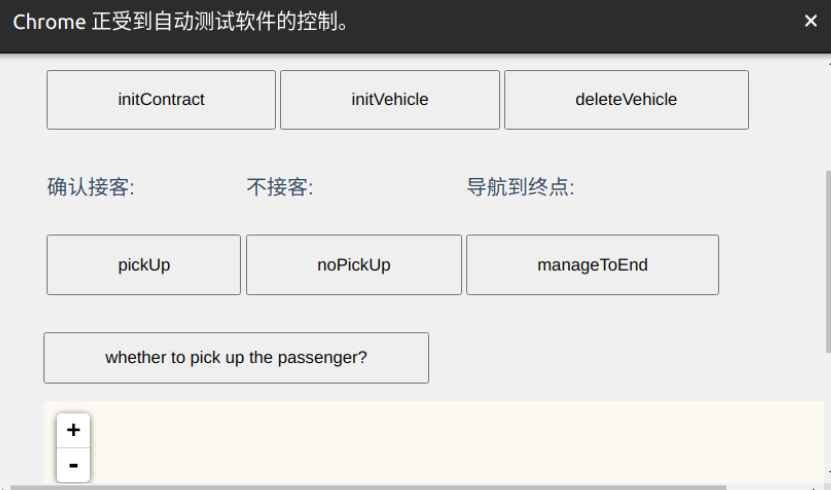
python3 vehicle\_test.py

python3 passenger\_test.py

看到如下提示，说明车辆位置上传成功：

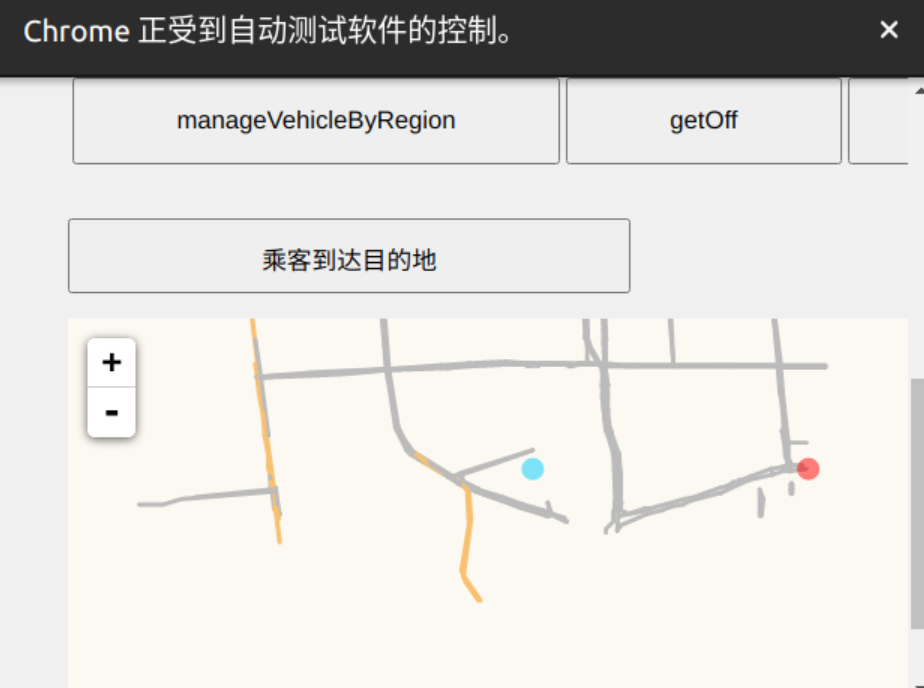


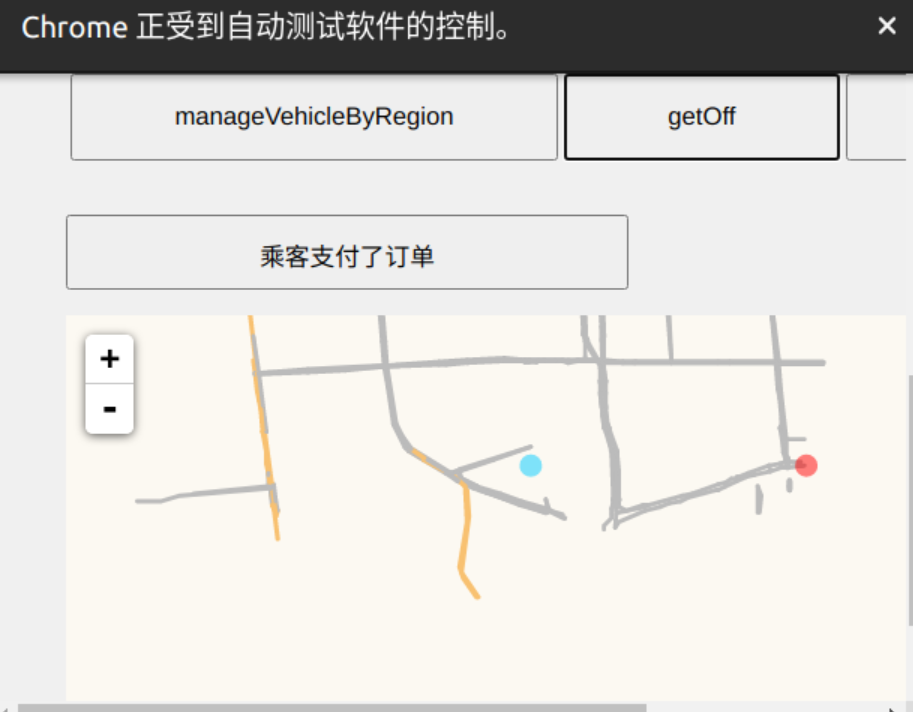
被selenium控制的浏览器会进行一系列的操作，当司机端询问：Whether to pick up the passenger时，点按下图的pickUp按钮接起乘客，即可完成后续的调度步骤：





最终，乘客被送达目的地，并在支付订单之后乘客端的测试程序结束运行：





至此，区域索引区块链的出租车调度系统的复现实验圆满结束。

**3.2区域索引的树状区块链的出租车调度系统**

上面的实验是在一条链上进行的复现实验，主要基于geth1进行，而本实验是实现区域索引的树状区块链的复现实验基于geth-tree完成。

大体操作基本同上，详情参见仓库里的《10 部署在geth-tree上的出租车调度系统复现实验》文档，这里主要介绍一下不同于上面实验的操作。

启动过程和原来的一致，只是启动文件根据区块链的不同有所变化，主要是regionid和position的变化。对于不同的账户而言，所处不同的子链，其对应的regionid和position也应不同，此外，为了支持对应的实验进行，还上传了全新的地图数据文件，参见仓库：

**3.3 问题及解决方案**

原版的复现手册中有着许多未说明清楚的部分，此外，操作步骤也并不完整。本小节主要介绍笔者在复现过程中遇到的问题以及解决方案。

3.3.2 节点连接的问题

笔者在进行多节点连接时，起初报错信息为终端报错，即无法同时运转两个账户，且两个账户之间，经过观察net.peerCount为0发现二者之间并未连接。

解决方案：复现手册中所说的需要把把Node1中的Key文件复制到Node2中。实际并不明确，事实上，将gethdata/keystore文件夹下的内容复制过去，则表明二者之间有着相同的账户信息，因此，初始的创世块中也应初始化相同的账户余额。笔者将两个文件夹中的genesis.json信息同步后，将Node1文件夹下的keystore文件夹复制到Node2中。

此时节点相互连通，问题解决。

3.3.3 部署合约过程

原版复现手册中仅仅给出了部署合约的模版代码，并未对其进行说明，此外也未对合约编译后的abi以及bytecode内容进行说明，笔者在文档中添加了Remix Desktop编译合约并记录编译结果中的ABI字段并进行进行压缩转义，同时记录bytecode（下称字节码）字段的这一过程。

3.3.4.1 编译合约

笔者使用 Remix Desktop 进行合约编译。编译完成后，切换到编译选项界面，点击“Compilation Details”按钮，即可观察编译结果的详细信息，如下图所示：图中，更靠近中央部分的红框即为部署合约时需要使用的应用程序二进制接口（ABI）和以太坊虚拟机字节码（bytecode）信息，需要妥善记录保存。另外，在合约源代码所在的目录下的 artifacts 目录中，有一份与合约同名的 json 文件，该文件亦需要妥善保存备用。

3.3.4.2 部署合约

本文使用附录 B 中的合约部署模板进行合约部署工作，具体步骤如下：

1. 将代码模板中的“经过压缩转义后的 ABI”替换为经过压缩转义处理（即：去除多余空格，且为双引号等特殊字符进行转义处理）的上一步骤中获得之应用程序二进制接口

2. 将“获得的字节码字符串”替换为上一步骤中获得之以太坊虚拟机字节码

3. 修改变量名 Contract、contractInstance 为合适的名字，避免多份合约重名

4. 将经过以上修改的模板代码复制到正在运行 geth1 的控制台中，单击回车提交部署请求

5. 使用miner.start()开始挖矿，合约地址将随后在终端中显示 需要注意的是，在模板中，可见一名为 position 的字段。若该字段指代的 Geohash 在区块链的管辖范围之外（即：区块链管辖的 Geohash 并非 position 字段的前缀），将获得 out of the blockchain 错误，无法继续部署。因此，每次部署前，均需检查 position 字段指示的范围是否在区块链管辖范围以内

小结：合约问题：

3.3.4.3 合约交互

部署完成后的合约可以使用合约地址与其建立交互渠道，并调用合约内定义的方法。在 web3.js 库的辅助下，与合约进行交互仅需短短数行代码。最小示例如下：

在进行实验时，若出现难以排查的错误，应首先考虑合约相关错误。综合上述 部署步骤，合约相关错误可以从以下几点进行排查：

• 从编译详情处获得的各项数据是否正确？例如，以太坊虚拟机字节码是否与应用程序二进制接口出自同一次编译过程？

• 部署合约时，是否正确配置各选项，例如 position 字段？

• 部署完成之后，是否正确记录合约地址？

• 进行合约交互前，是否确认区块链允许 WebSocket 协议连接？部分功能，例如订阅事件，仅能在 WebSocket 连接下进行。若无，则这些功能可能失效，与合约的交互可能失败

**第 5 章 基于树状区块链的跨链转账测试**

3.1 树状区块链的跨链转账

树状区块链，乃是以区域索引区块链为基础，旨在解决传统区块链单链结构面对大量区块时产生高昂性能代价的痛点。其借助 Geohash 技术，编码地理位置，并借此对区块链进行划分，形成类似于字典树的树状结构。

然而，这样的结构带来了一个问题：若某个账户的位置发生了较大改变，以于它离开了目前所在的子链的管辖范围（即账户所在子链表示的 Geohash 范围不再是账户实际位置的 Geohash 编码表示的前缀），那么账户在新地理位置上发送的所有交易将失败，因为在逻辑上账户并不由管辖新地理位置所在片区的区块链直接管辖，后者可能根本没有关于该账号的任何记录，或者其上与该账户关联的信息并非切实。

对此，树状区块链的解决方案是：账户需要向一个特殊的管理账号发送一种特殊的交易，该交易及其携带的信息足以令管理账号从账户原先所在的区块中，将账户的各项信息转移到新区块链上。此后，该账户可以认为由新区块链接手管辖，可以正常进行诸如发送交易等的区块链交互操作。由于上述转账过程与同一区块链内两不同账户间的转账不同，乃是横跨两个区块链的、同一账户之间的转账，故称该特殊转账过程为“跨链转账”。显然，在尝试解决单链结构的效率问题的同时，树状区块链引入了跨链转账所需的额外时间开销。跨链转账操作的效率，将对用户的使用体验有较大的影响。因此，本章将围绕跨链转账这一主题进行探究。

首先，设计并进行数组跨链转账测试

实验；其中，以较小规模的 10 账号跨链转账实验测试跨链转账的正确性；验证正确性后，再开展更大规模的跨链转账测试，检验跨链转账操作在不同压力情况下的运行效率波动；

最后，给出一个简单的数学模型，探究树状区块链跨链转账代价与其链上数据查询复杂度降低带来的性能优势基本持平时的临界条件，进而给出适用树状区块链代替传统单链结构区块链的应用场景建议。

**3.2 设计测试**

跨链转账测试实验将在如图 3-1 所示的区块链网络中进行。 该区块链网络共由三个链组成，分别以链 w1，链 w11 和链 w12 指代。链 w1 为树状区块链中的分支区块链，其管辖的地理位置范围为 Geohash 编码为 w1 的所有区域；由链 w1 负责串联的子链 w11、子链 w12 均为叶子区块链，分别管辖 Geohash编码前缀为 w11、w12 的所有区域，其下分别运行有一个节点，可以同传统区块链节点一样进行挖矿、交易等操作。初始时，两子链中存在相同的数个账号，但 w11 中的账号均有 5 × 1040 单位的余额，w12 中的账号的余额均为 0，模拟账号进行了大范围物理位置变动后。管辖权仍未移交给管辖新地点的子链的情况。跨链转账开始后，待所有链 w11 中的账号同时发起转账申请后，树状区块链将开始串行地遍历所有子链 w11 中的账号，并将它们在链 w11 的余额全部转移至链 w12 的对应账号之下。待转账完成之后，检查链 w12 中各账户的余额，即可证明跨链转账的功能正确性。

**3.4 测试步骤**

**3.5 测试结果分析**

**四、 参考文献**

1. [] Q. Zhou, H. Huang, Z. Zheng and J. Bian, "Solutions to Scalability of Blockchain: A Survey," in IEEE Access, vol. 8, pp. 16440-16455, 2020, doi: 10.1109/ACCESS.2020.2967218. [↑](#endnote-ref-1)
2. [] Dorri A, Steger M, Kanhere S, et al. Blockchain: A distributed solution to automotive security and privacy[J/OL]. IEEE Communications Magazine, 2017, 55(12): 119-125. DOI: 10.1109/MCOM.2017.1700879. [↑](#endnote-ref-2)