**课程报告**

|  |  |
| --- | --- |
| **课程名称：** | **企业软件项目实训** |
| **学生姓名：** | **贾宇然** |
| **学生学号：** | **201630664581** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2018-2019第二学期** |

**软件学院**

**2019年6月**

正文部分（标题行用小四号字加粗，正文内容用小四号字）

**参考文献（小四号字加粗，参考文献不少于6篇）**

[1] ×××（五号字）

根据下面的几个要点写一篇不少于10000字的课题报告（可以根据自己组内分工，至少包含以下内容中的五项完成，可以搭配成果截图）

注：标红部分每个人必须得有；4-10至少包含1项。格式按一般学校课程报告格式即可

1. 区块链技术原理；
2. 联盟链和公有链的异同；
3. 信任链是如何建立的？
4. 链式存储和MPT存储
5. Gas在智能合约中的作用；
6. EVM中的数据存储结构；
7. 群组架构的好处；
8. 分布式存储有什么优势？
9. 并行计算；
10. 当前区块链实施的难度。

**注：（1）程报告每位同学都需要提交电子版和纸质版，提交的电子版请上传到github，纸质版打印装订后交给各组组长，由各组组长交给任课教师。 电子版和纸质版的提交截止时间为：2019年7月10日（周三）下午5点前。**

**（2）正式上交课程报告时，请删除本页。**

### 区块链技术原理

### 区块链介绍

“我一直致力于创造一个完全点对点的全新电子现金系统，这个系统不需要值得信赖的第三方。”这是比特币的神秘创造者中本聪（Satoshi Nakamoto，可能是化名）于2008年10月给钻研信息加密相关人士的邮件中所附的信息。邮件包括一个指向九页白皮书的链接，该白皮书阐述了一些会颠覆当前金融系统的技术，而该技术就是我们现今所见到的区块链。诚然，科技发展的势头已经彻底验证其预见性。

区块链是一种对一段时间内所有交易或者电子行为进行记录的分布式数据记录技术，即一种按照时间顺 序将数据区块以顺序相连的方式组合成的链式数据结构，并以密码学方式保证的不可篡改和不可伪造的分布式公开账本技术。在区块链上，每笔交易都可以被系统的参与者通过多数节点共识的机制进行审核。一旦被记载到区块链上，相关的交易信息就不能被修改、删除。区块链上的区块记录了一定时间内被审核通过的每笔交易记录。比特币是区块链技术最本质的应用，它也因为在没有政府的干预下构造了上百万美元的匿名交易市场而成为最具争议性的区块链应用。 **[1]**

Don&Alex Tapscoot在他们出版的书籍*Blockchain Revolution*中有这样一段对区块链的总结：“区块链是一个不可摧毁的经济交易的数字账本，不仅可被用于记录金融交易，还适用于几乎所有有价值的交易。”通俗地讲，区块链是一种用几乎不可能伪造的方式存储数据的数学结构，服务于各种有相当价值的数据。**[2]**

我们可以简单地将区块链比作一个在线协作文档。小组的成员都在各自的浏览器上有一份实时更新的最新版文档，成员们可以同时对文档进行操作且任何操作都会及时地反映在所有人的文档中。区块链与在线文档最大的区别在于其会被存储于每一个访问者的本地空间，而非如在线文档一般储存在一个特定集中的云空间上。

此前，电子金融领域建立在可靠的、可信任的第三方授权的基础上。用户网络交易必须依赖于第三方机构对 交易双方进行身份和交易资格的确认。人们生活的电子世界在依靠第三方机构维系安全与个人电子资产隐私的背景中，并不是绝对的安全。在这种背景下，区块链的诞生通过对分布式共识的应用，对依靠第三方信任的电子世界规则提出了巨大改进。**[1]**目前，金融机构（如FISCO BCOS）拥有最广泛的区块链技术应用。但区块链不可篡改、开放性、去中心化等特性让许多行业看到了区块链提供的可能性，例如，区块链用于打造一个完整的、基于区块链的供应链。传统的供应链围绕核心企业与产品构建，是一个从供应商开始、途径制造商、运输商、分销商最终到消费者的一种网链状结构，换而言之，他们之间不是单一的一对一关系，而是会出现交叉。区块链的应用能够让供应链所需的物流、信息流和资金流三流合一、从订单到交付，整个交易的状态都是完全透明的。其他应用领域包括支付、知识产权、深度学习、司法证明等方面都在发掘区块链所拥有的潜力。

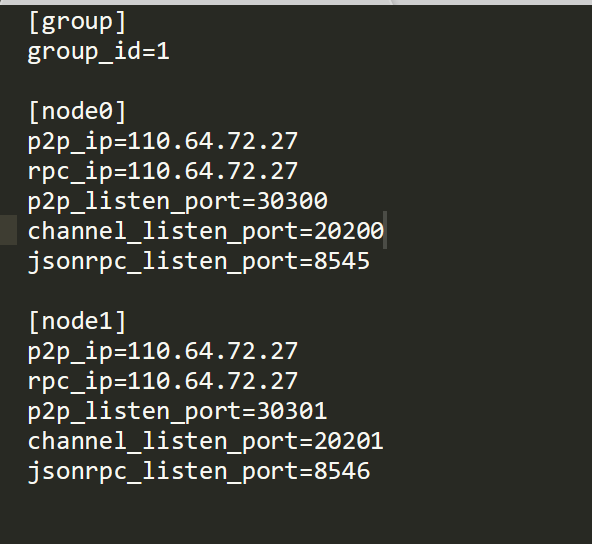
### 区块链的技术原理

正如老师在DAY1-2.区块链技术架构与原理解析中的描述：“区块链这个黑科技其实没有发明什么新的技术，都是成熟技术的组合”，区块链通过融合点对点网络、分布式账本、非对称加密和授权、共识机制和智能合约来实现去中心化、防篡改、匿名性、开放性和自治性的特性。现在我们就来看看区块链背后每一种技术的详解：

### 点对点网络（P2P）

点对点网络又称对等互联网络技术，它依赖于网络中参与者的计算能力和带宽，而不是如传统服务提供者一般将服务使用者产生的性能压力都集中在在相对少数的服务器上。点对点网络能直接将人们联系起来，可类比与我们每天的电话、面对面交流，使得网络上的沟通变得容易、更直接共享和交互，真正地消除中间商。这种网络允许我们直接连接到其他用户的计算机来进行交换文件，而不需要经过传统服务器。 同时，它还提供可靠性的服务，如果网络中的一个或多个节点关闭也不会对系统可用性产生影响。

我们可以在利用FISCO BCOS企业级部署工具建立多群组多机构联盟连的时候找到相关的配置。在生成一个机构的节点证书和P2P连接信息文件前需要修改node\_deployment.ini中的p2p\_ip、rpc\_ip等参数来保证点对点网络的连通性，如图所示：



随后--generate\_all\_certificates指令将会生成节点证书和peers.txt，后者需要被发送给其他同群组节点以便它们生成节点的时候能够正确记录同组节点成员的信息。可以看到，这一部分是点对点网络在FISCO BCOS建链过程中的重要体现。

### 分布式账本

区块链技术使用分布式账本技术来存储数据。与将数据存储在一台服务器上的集中式数据存储不同，分布式账本是一种在网络系统中各成员之间共享、复制和同步的数据库。他会记录网络参与者之间的交易，如智能制造环节中涉及的零部件或数据的交换，可以有效降低因调度不同账本所产生的时间和成本开支。区块链中的每个参与节点都具有区块链的精确副本。**[4]**

与传统的将数据托付给中心化的第三方数据中心方法相比，分布式账本采用去中心化的网络结构，提高了交易的透明度，避免权力过度集中的数据中心结构发生腐败和欺诈的可能性。这种做法的一个关键好处在于每个用户都可以维护自己的副本的总帐，每当新的完整节点加入区块链网络时，他们就会联系其他完整节点并请求区块链网络的完整副本分类帐；当一个节点被攻击并发出异常交易时，其他节点会进行忽略，无法改变账本的有效和合法性。单一网络节点受到攻击并不会对账本本身造成重大影响，有利于保护整个分布式网络稳定性和安全性。这种技术直接杜绝了我们对中心或者第三方信任机构（如银行、公证人、政府）的依赖，我们为可信交易付出的成本大大降低。

### 非对称加密和授权

### 哈希

说到非对称加密，则可以从最基础的哈希算法讲起。哈希算法是区块链中用的最多的一种算法，它被广泛的应用于区块的构建和交易合法性验证上，是一种保证交易的信息不被轻易篡改的单向密码机制。哈希函数有三个基本特性：1.输入为任意大小的字符串 2.产生固定大小的输出 3.能在合理的时间内就能算出输出值。在满足以上三个特性的基础之上，为了保证加密算法的可靠性，它应该还需具备以下几个特质：1.输入的字符串和输出的散列数据应该是一一对应的，任何一个输入信息的变化，都会影响最终输出数据的变化。也就是说，不同的输入必须产出不同的散列序列。如果对于两个不同的输入产生了相同的输出，那么就证明算法的不具有抗碰撞性 2.我们应无法由输出倒推出原来的输入字符串，也就是说加密过程应该是不可逆转的。

SHA-256哈希算法目前被广泛应用与区块的加密，对于任何长度的字符串输入，其散列序列都为256位。我们通过SHA-256对区块信息进行哈希，区块信息被散列为一串特定的序列（一般称之为摘要），这个序列能唯一地标识一个区块，其中任何细微数据的改变都会导致区块散列值发生变化。如果根据区块体计算出的哈希值与存放在区块头的摘要相符则表明区块中的数据没有遭到篡改，这个过程也被称为完整性验证。

### 非对称加密

非对称加密会使用一对密钥，公钥和私钥。我们可以由私钥推导得到公钥，反之则不行。数字签名是非对称加密技术提供的一种用于验证交易的完整性和真实性机制，它允许交易保持公开，从而在用户之间实现信任关系。一般情况下，信息发送者会使用接受者的公钥加密信息，接受者则会使用其拥有的私钥对数据进行解密，此时接受者无法确定文件的发送者。在应用数字签名之后，信息传递的流程变为：1.信息发送者使用接受者的公钥加密信息，并用自己的私钥对信息摘要进行签名，发送信息和签名 2.接受者通过私钥解密信息，对信息进行哈希获取摘要，用发送方公钥解密数字签名得到摘要并进行对比。因为只有发送方的公钥能解开其签名，我们可以同时确保信息的完整性和真实性。我们可以从FISCO BCOS的build\_chain.sh脚本文件中发现其使用Secp256k1作为椭圆曲线数字签名算法的参数来进行签名。

### 共识机制

区块链技术的一个关键方面是确定应该由哪个用户发布下一个块。由于点对点网络下存在较高的网络延迟，各个节点所观察到的事务先后顺序不可能完全一致。因此区块链系统需要设计一种机制，对在一定时间范围内内发生的事务的先后顺序进行共识。而这种让所有节点对某个提案达成一致意见、保障系统一致性的机制被称为“共识机制” 。

当用户加入区块链网络时，他们需要同意系统的初始状态。这被记录在唯一的预配置块中，我们称之为创世区块。FISCO BCOS中，每个区块链群组都有自己的一个已发布的创世区块，并且必须根据商定的共识模型将每个区块添加到区块链。无论模型如何，每个块必须是有效的，因此网络用户可以独立地验证每个区块。

在共识算法方面，官方文档有这样一段描述:”FISCO BCOS基于多群组架构实现了插件化的共识算法，不同群组可运行不同的共识算法，组与组之间的共识过程互不影响，FISCO BCOS目前支持PBFT(Practical Byzantine Fault Tolerance)和Raft(Replication and Fault Tolerant)两种共识算法”。PBFT即实用的拜占庭容错算法，是共识算法里面的一个重点，在Miguel Castro 2001年编写的PBFT分析论文中[2]，它给算法的定义如下：“PBFT是一种状态机副本复制算法，即服务作为状态机进行建模，状态机在分布式系统的不同节点进行副本复制。每个状态机的副本都保存了服务的状态，同时也实现了服务的操作。将所有的副本组成的集合使用大写字母R表示，使用0到|R|-1的整数表示每一个副本。为了描述方便，假设|R|=3f+1，这里f是有可能失效的副本的最大个数。尽管可以存在多于3f+1个副本，但是额外的副本除了降低性能之外不能提高可靠性。”PBFT的提出解决了原始拜占庭容错算法效率相对底下的问题，大大降低了算法执行的时间复杂度，增强了BFT在实际应用中的可行性。

### 智能合约

智能合约是是在区块链上达成协议的一种方式，功能上并没有超越传统合约，但它能最小化达成协议所需要的信任成本。我们一般通过Solidity来编写智能合约，它是一种运行在Ethereum虚拟机上的高级语言，其语法接近于Javascript，是一种面向对象的语言。满足某些条件的时候，智能合约当中的规则和条款会被自动地、无需信任地和公正地执行，我们完全不用担心系统在条件被满足时不执行合约；其去中心化、不可篡改的特性完美适应区块链。它相当于一个预先定义好事件的机器人，对接收到的交易进行回应（调用非pure、view函数时返回TransactionReceipt；调用pure、view函数时返回我们定义的数据），接收和储存价值或者是向外发送信息和价值。智能合约实际上借鉴了比特币脚本，并且进一步拓展了其应用范围。与比特币脚本相比，智能合约最大的不同点是支持图灵完备的开发语言，允许开发者在智能合约内执行任意功能并对资产进行精确状态和访问整个区块链的状态。

### 联盟链和公有链的异同

### 联盟链

顾名思义，联盟链即联盟区块链，是许多的机构或组织共同经营的一个区块链，每个机构或组织管理一个或多个该联盟链里的节点。节点通过授权后才能加入与退出网络，各机构组织组成利益相关的联盟，共同维护区块链的健康运转。一般由某个群体内部指定多个节点为记账人，每个块生成由所有的预选节点共同决定，其他接入节点可以参与交易，但不过问记账过程。普通节点则可以通过区块链开放的API进行有限制的查询。联盟链适合组织机构间的交易和结算，像是银行间的转账、支付，通过采用联盟链的形式，能够很好地营造一个内部生态系统。

### 公有链

任何个体或团体不需要任何许可即可参与、离开链、发送交易（交易能够获得区块链的有效确认）和参与共识过程。其采用的“共识机制”包括POW（工作量证明）、POS（权益证明）、DPOS（委任权益证明）等，这些机制是经济奖励和数字加密验证的结合，因此也可称为“加密数字经济”。同时，公有链网络中不存在任何中心化的服务端节点，数据的读写不受任何其他人的控制，程序开发者没有权利去干涉用户，这点优于中心化的程序。因此它可以保护公有链应用的用户不受开发者的影响，在程序开发完毕之后，开发者所占的比重越来越小，用户共同参与维护工作，充分的体现了区块链的去中心化和自由。

比特币和以太坊都是一种公有链。

### 区别

1. 准入机制：公有链没有准入机制，所有人共同参与记账。联盟链中节点的加入需要授权，相比而言拥有更强的隐私保护。
2. 交易处理速度：公有链中，一个新的区块是否能够上链需要区块链中所有的节点来决定，一笔交易必须经过每个节点的确认，这导致了公有链低下的交易速度。而对于联盟链，一个新的区块的出块只要预选节点进行确定即可。这意味着一笔交易不需要所有节点的确认就可以进行，成功地降低了交易处理时间。
3. 共识机制：公有链为了保证数据的安全性，采用了POW共识机制：整个系统中每个节点为整个系统提供计算能力，通过一个竞争机制，让计算工作完成得最多的节点获得系统奖励，即完成新生成货币的分配，可以理解为按劳分配（与Prove Of Work相呼应），如比特币这种货币型区块链就应用POW共识机制。这种算法支持完全去中心化、节点自由进出、破坏系统所需的成本巨大，只要网络破坏者的算力不超过网络总算力的50%，网络的交易状态就能达成一致。可靠，就是POW共识算法最大的优点。联盟链如FISCO BCOS则一般采用PBFT、RAFT等算法。PBFT相比POW，去中心化程度会稍微低一点（有领导者节点的存在，支持分级权限），但是相应地需要的算力会比POW算法要少一个量级，它既满足一定去中心化程度，又兼顾了一定的性能。

### Gas在智能合约中的作用

比特币区块链上操作的交易费就是直接支付一定额度的比特币作为手续费，而在智能合约中，我们在调用非pure、view函数的时候也需要根据其复杂程度支付一定的费用。以太坊在区块链上实现了一个运行环境，被称为以太坊虚拟机（EVM），智能合约就是在这个环境中运行的。合约的执行会在所有节点中多次重复，对于每个被执行的命令都会有一个特定的消耗（不同的action就会有不用的消耗）。以太坊将这笔消耗费用命名为Gas(直译为燃气）。

"交易"这个术语在以太坊里被用来指代一个用来存储消息的被签名数据包在区块链上从一个外部拥有账户发送至另一个账户的过程。交易的数据内容包括消息接受者、签名、价值域、GASLIMIT、GASPRICE。每笔交易都被要求包括一个GASLIMIT和GASPRICE。其中GASLIMIT是这笔交易允许的最大的消耗Gas的数量，可以理解为交易服务费消耗上限；而GASPRICE则是单位gas的ETH价格。若交易正常完成后，我们传递的Gas数量大于实际消耗的Gas，那么多余的Gas会以ETH的方式返还。如果我们需要对调用的操作进行加急，那就得付出小费，即附加Gas。如果算上小费，实际消耗的Gas可能超过GASLIMIT。而假使gas在任何时候消耗完或者超过GASLIMIT，合约会抛出out-of-gas 异常且当前交易执行状态全部会被回滚到初始状态，消耗的Gas亦不会全部返还。

gas的作用在于它有助于确保提交给EVM的交易操作会有一定的限制，通过要求操作发起者支付每个操作的执行消耗，保证区块链网络不会因为执行大量对无意义无价值的密集工作而阻塞了后来的请求，防止以太坊网络被蓄意攻击或滥用。第二点则是通过GASPRICE的设定可以对交易的重要性分级，矿工能够根据GASPRICE选择一定时间段内最先被执行的交易。

### EVM中的数据存储结构

智能合约是部署在区块链的代码，区块链本身不能执行代码，代码的执行是每个节点在本地通过以太坊虚拟机（Ethereum Virtual Machine），即EVM来实现的。智能合约可类比做Java程序，Java程序需要通过Java虚拟机（JVM）解释成字节，那么智能合约也需要通过EVM来将代码解释成字节从而执行。EVM是一个沙箱环境，运行在其内部的代码无法接触到网络、文件系统或者其他进行，智能合约之间的调用也是有限制。[5]

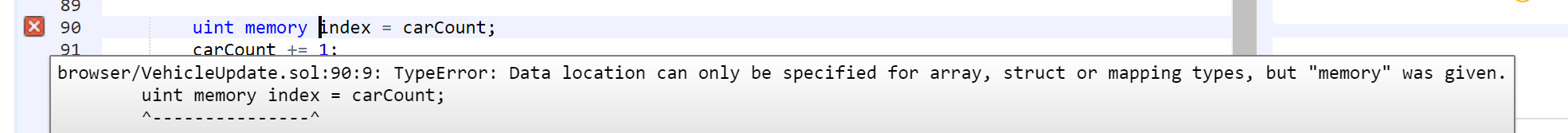
在EVM中，除了几个全局参数之外，大多数状态都存储在accounts中.EVM具有从地址（160位字）到帐户状态的部分映射。帐户状态包含代码，存储，随机数和余额。代码是一个字节序列。存储是从机器字（EVM机器字为256位）到机器字的映射。随机数是一个不断增加的机器字。余额也是关键字，表示可以作为运行EVM的费用支付的一些可转移价值。当代码不为空时，代码控制帐户，这样的帐户称为合同；否则，帐户由对应于该地址的私钥的持有者控制，这样的帐户称为外部帐户。[6]

为了更好地理解EVM中的数据存储结构，我们可以先来看看智能合约的数据类型。Solidity中的基本数据类型可以分为两类：值类型(Value Type) 及 引用类型(Reference Types)，以下对此进行分析：

        1）值类型：

        包括整型(uint)，地址型(address)，布尔(Booleans)，定长字节数组(fixed byte arrays)，枚举类型(Enums)等在内的基本数据类型，在合约中都是以值的类型进行传递

       2）引用类型：



        引用类型是一个复杂类型，占用的空间通常超过256位，拷贝时会产生很大开销，因此我们需要考虑将它们存储在什么位置，memory亦或是是storage。如图所示，只有的复杂类型如数组(arrays)、数据结构(struct)和mapping有一个额外的数据的存储位置属性。值得注意的是mapping只能被声明为storage。

EVM拥有三种数据存储区域：

栈：EVM的执行模型是基于栈结构的，与JVM是类似的，这里提供的栈是用来存储代码执行过程中的中间值变量、局部值变量、函数参数。栈中每个元素大小为256bit、最大数量为1024个，相当于有数量限制的免费使用的内存。

内存（Memory）：用于临时存储的一个链表、在ALU执行时用来存放智能合约和智能合约处理的数据。这个关键字只能用于函数内部，它的声明告知EVM在运行时创建一块固定大小的内存区域给变量使用。链表的每个结点对应一个内存块，而一个内存块为1024字节，链块内存是可扩展的，其扩展以块为单位。每一个内存快是一个线性的整数数组，数组索引以256bit为移动单位。当外部函数对某合约调用完成时，内存型变量即被移除，生命周期仅为整个方法执行期间，仅作临时保存，因此GAS开销很小。

存储（Storage）：账户拥有的持久的储存空间，也被称为库（Repository），为256位二进制的key-value结构。它包含三个缓存：1.账户缓存（key为地址，value为账户）2.代码缓存（key为代码哈希+地址，value为代码）3. 存储缓存（key为地址，value为状态map），不借助外力的情况下不可遍历。这种存储结构是在合约创建的时候就确定好了的，它取决于合约所声明状态变量。但是内容可以被交易改变。智能合约只能读写当前区块链账户的Storage而无法修改其他账户的Storage。Storage变量会永久存储在区块链中，GAS开销由于其永久保存性是最大的。

### 链式存储和MPT存储

### 链式存储

二叉树的链式存储就是通过链表的形式储存二叉树的机构，是数组存储的一种泛化形式。它不需要一块连续的储存空间，优势在于插入、删除和删除等修改性操作，但无法如数组一样直接通过下标访问某个元素。我们可以根据二叉树的定义来构造二叉树节点：二叉树的每一个结点至少应该包括三个域：左子节点、当前节点数据、右子节点。这种链表结构一般被叫做二叉链表，使用这种链表可以很方便的表示和找到它的子节点。当然，要想找到它的父节点却有一定的困难。为此，我们还可以增加一个父节点指针域，这种结构被称为三叉链表。

### MPT存储

MPT 即Merkle Patricia Tries，译为默克尔帕特里夏树，是以Trie树为基础的、用于以太坊存储数据的核心数据结构。它是由默克尔和帕特里夏结合形成的一种树形结构，提供了一个基于区块链密码学的、自校验防篡改的数据结构。在解析 MPT 数据结构之前，我应该先来看起一下基础的Trie、默克尔树和帕特里夏树的结构。

### Trie

Trie，又被称为前缀树或字典树，是用于保存关联数组的一种有序树，其中的键通常是字符串。与Binary Search Tree不同，键不是直接保存在节点中，而是由节点在树中的位置决定。一个节点的所有子孙节点拥有相同的前缀，也就是这个节点对应的字符串，而根节点则相当于空字符串。一般情况下，不是所有的节点都有对应的值，只有叶子节点和部分内部节点所对应的键才有相关的值。普遍的来说，若索引键值中的每一个字符都来自一个容量为N且包含的字符都互不相同的字母表，这个树就为N叉树，树的深度就是key的最大长度。

Trie树的优势在于查询拥有共同前缀键的数据时会拥有十分高的效率，它不需要像哈希表一样遍历整个表，而是遍历以对应共同前缀的节点作为根节点的子树即可。第二个优点则是不存在hash键冲突的情况，key与value是直接一对一的关系。其劣势在于随机查找和储存空间的不平衡。假设现在我们需要储存一个key值内容很长的节点但树中没有相同前缀的分支。那么为了存储该节点，我们需要创建许多中间节点来构建从根节点到该叶子节点间的路径，这会导致整棵树的不平衡现象并造成存储空间的浪费。

### 默克尔树

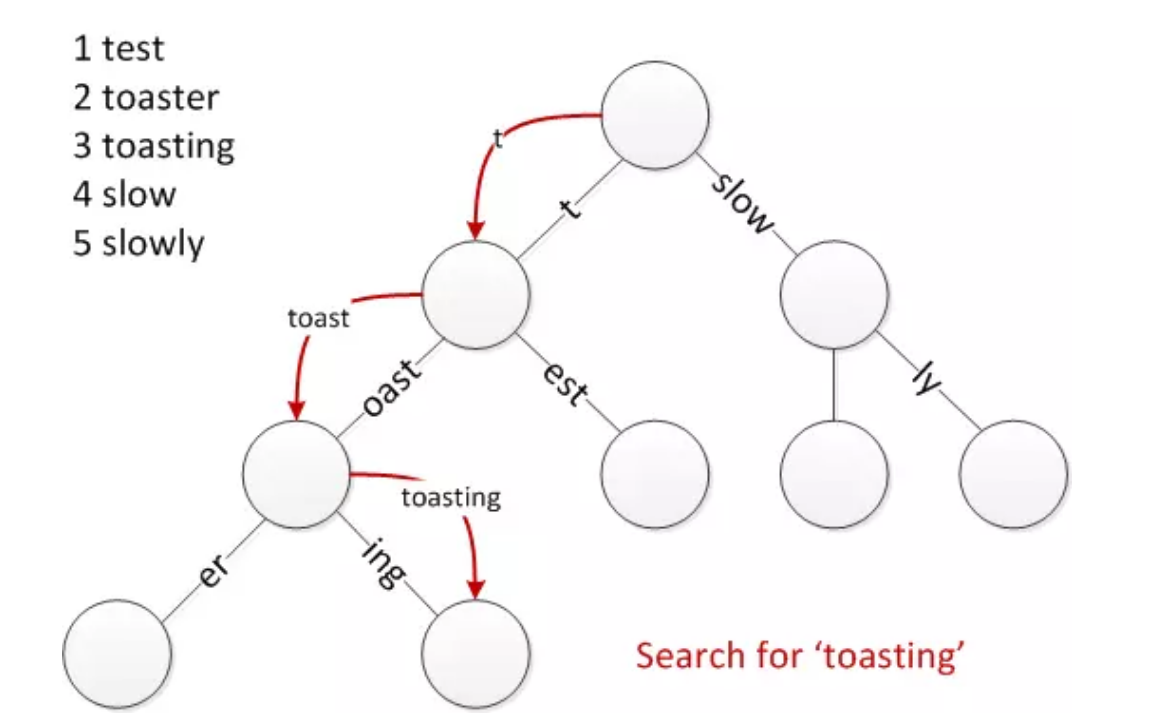
Merkle Tree，一般也被称作Hash Tree。顾名思义，这是用于存储hash值的树。Merkle树的叶子是数据块(如区块链状态、交易列表、交易回执等)的hash值。非叶节点的值是其子节点值连接形成的字符串的hash（如节点的左节点的值为“science”，右节点的值为“deocracy”，那么该节点的值为hash（“sciencedeocracy”）运算得到的字符串散列）。

我们在P2P网络中进行数据传输的时候，往往会同时从多个机器上下载数据，而很多机器可以被认为是unstable或者untrusted。为了校验下载数据的完整性，一般选择把大的文件分割成小的数据块来进行传输。如果小块数据在传输过程中损坏了，那么只要重新下载这一块数据而不用重新下载整个文件。在确定小的数据块的完整性的时候Merkle树就派上用场了。

在p2p网络下载网络之前，我们应先从可信的源获得文件的默克尔树（即树和数据可以从不同的节点获取）。随后，从其他从不可信的源获取数据并以Bottom-Up的方式形成一颗默克尔树。小块数据默克尔树的构成：在下载完成后、数据合并前，我们将每一个小数据块的hash值作为默克尔树的叶子结点，并一级一级往上计算哈希最终得到树的根节点，也就是根哈希值。我们现在可以比较可信的根哈希值和当前默克尔树的哈希值来确认下载是否有出现异常。只要某个小数据块出现了一点点的变化，其哈希值的改变会一层层地影响到最终的根哈希。默克尔树拥有容易验证的特性，只要不断对比节点的哈希值就能不断追踪错误的数据区块的位置并最终重新下载（因为我们的哈希验证可以从任意一个节点开始，即使有多个数据块同时出现错误也能够经过对比快速地锁定错误位置）。

### 帕特里夏树

帕特里夏树，即Patricia trie，也成为压缩前缀树，是一种节省空间、解决了普通Trie树存储不平衡情况的变种Trie。它的核心思想在于：如果存在一个父节点只有一个子节点，那么这个父节点将与其子节点合并。这样可以缩短 Trie 中不必要的深度，大大加快搜索节点速度。如图所示，t之后的前缀oast与est因为没有对应的分支，因此帕特里夏树会将他们合并成toast和test节点，而toaster和toasting会被整合成toast节点的左右子节点。



### MPT

MPT（Merkle Patricia Trie）可以理解为在Patricia树的基础上，添加了加密安全功能。二则是引入了更多的节点类型，包括Null Node（空节点），Extension Node（拓展节点），Leaf Node（叶子节点），Branch Node（分支节点）来提高效率。MPT的确定性是指同样内容的键值，将被保证找到同样的结果，有同样的根哈希（这些特性来源于默克尔树）。关于效率方面，对树的插入，查找，删除的时间复杂度控制在O(log(n))。它的数据存储在LevelDB数据库中，Key节点为RLP编码的SHA3值，Value则是节点的RLP编码。

接下来看一下MPT各个节点类型与默克尔、帕特里夏特性的联系：

1. 空节点(Null): 在代码中就是一个空串，如上图slow节点的左子节点
2. 叶子节点(Leaf): 叶子节点包含两个字段， 第一个字段是剩下的Key的半字节编码,而且半字节编码方法的第二个参数为true， 第二个字段是Value
3. 扩展节点(Extention): 扩展节点包含两个字段，第一个字段是剩下的Key的可以至少被两个剩下节点共享的部分的半字节编码，第二个字段是下一个node的hash值，设计体现了帕特里夏树的特点，通过合并只有一个子节点的父节点和其子节点来缩短 trie 的深度。
4. 分支节点(Branch): 分支节点可以携带多个子节点。它有一个容量为 17 的 node 数组成员变量 Children。数组中前 16 个空位分别对应 16 进制 (hex) 下的 0-9a-f，这样对于每个子节点，根据其 key 值 16 进制形式下的第一位的值，就可挂载到 Children 数组的某个位置，fullNode 本身不再需要额外 key 变量。Children 数组的第 17 位，留给该分支节点的数据部分。分支节点继承了原生 trie 的特点，而每个父节点最多拥有 16 个分支是基于总体效率的考量。

为了管理键值对，MPT引入了编码算法。MPT中涉及到了三种编码，分别为keybytes编码、Hex编码和Hex-Prefix Encoding 16进制前缀编码。

1. keybytes编码：这种编码格式就是原生的key字节数组，大部分的Trie的API都是使用这种编码格式。
2. Hex编码:对传统的编码十六进制字符串进行编码，将他们转为十进制存储。比如0f1248表示的是三个字节的[15，18，72]。具体操作方法为：将 keybytes 中的 1byte 信息，将高 4bit 和低 4bit 分别放到两个 byte 里，最后在尾部加 1byte 标记当前属于 Hex 格式。然而，这种编码方式无法处理16进制的字符长度为奇数的情况。
3. Hex-Prefix 编码：这种编码首先将 Hex 尾部标记 byte 去掉，然后将原本每 2 nibble 的数据合并到 1byte如果输入 Hex 格式字符串有效长度为偶数，增添 1byte 在输出数据头部以放置 Compact 格式标记位00100000(叶子节点偶数)/00000000(扩展节点偶数)；如果输入 Hex 格式字符串有效长度为奇数，将 Hex 字符串的第一个 nibble 放置在标记位 byte 里的低 4bit,并增加奇数位标志 0011xxxx(叶子节点奇数)/0001xxxx(扩展节点奇数)。

### 当前区块链的的实施难度

我们目前的DAPP为汽车维修保养记录与管理平台，目标为通过记录汽车维修保养情况，反映车辆的当前车况，并且可以延伸出车辆零部件寿命纪录等其他功能。拥有可信赖的维修保养记录的汽车，在二手车售卖过程中更加值得车主信任，有益于促进市场诚信的建立，也降低了二手车交易的门槛。

DAPP的区块链底层搭建计划采用星形拓扑的结构，维修保养平台需要有一个最高权限的中心管理者来负责维修点、4S店等机构的授权，它属于多个群组并可以运行多家机构的应用。而其他的附属机构则属于不同的群组并运行各自的应用以更新汽车维修信息、转让汽车所有权等。

区块链的实施重点、难点首先在于需要选出一个维修点与4S点都信服的中心机构来作为维修授权的发放和管理机构。所有的维修信息需要有一个聚集点才能使得整个记录系统有应用意义，也就是说不应该出现多个记录系统合约同时部署并被使用的情况。第二点则需要大量的机构参与才能使得DAPP的信息记录系统有实际意义。说服各种维修点与4S店都采用这套DAPP系统进行维修记录的记录有一定的难度，需要从他们的角度找到应用本区块链项目之后获取的价值和潜在的利益。第三，中心机构需要出台一定的标准和规范来确保信息的可靠性和价值。在没有可信度的情况下，再多的数据记录也不过是废纸一张。

### 课程收获

本次区块链课程让我接触到了非常多的新知识。

1. 首先是了解了区块链方面的基础知识，包括区块链的特点、优势、技术原理、三种链之间的区别和各自的应用场景等。
2. 第二点则是对由金链盟在BCOS开源平台基础上进行模块升级与功能重塑而成的FISCO BCOS区块链底层平台有了深入的了解。我在课程中实践了build\_chain建链脚本-l方式建链、-f方式通过ipconf文件建链和通过FISCO BCOS generator企业级部署工具部署多机构多群组联盟链等三种建链方式。
3. 课程让我们阅读了LAGCredit积分智能合约并通过CryptoZombies教程学习了Solidity语言的数据结构、函数定义

### 参考文献

1. 李董, & 魏进武. (2016). 区块链技术原理、应用领域及挑战. 电信科学(12).
2. Castro, O. T. D., & Miguel. (1999). Practical Byzantine fault tolerance /. Symposium on Operating Systems Design & Implementation.
3. <https://www.linkedin.com/pulse/whats-next-generation-internet-surprise-its-all-don-tapscott>
4. 张正, 杨睿哲, 金凯, 杨兆鑫, 林波, & 张延华, et al. (2018). 面向工业互联网场景的新型分布式账本技术. 情报工程, 4(3), 021-028.
5. 李赫, 孙继飞, 杨泳, & 汪松. (2017). 基于区块链2.0的以太坊初探. 中国金融电脑(6), 57-60.
6. Hirai, Y. . (2017). Defining the Ethereum Virtual Machine for Interactive Theorem Provers. International Conference on Financial Cryptography & Data Security. Springer, Cham.