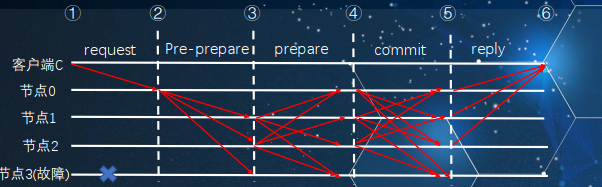
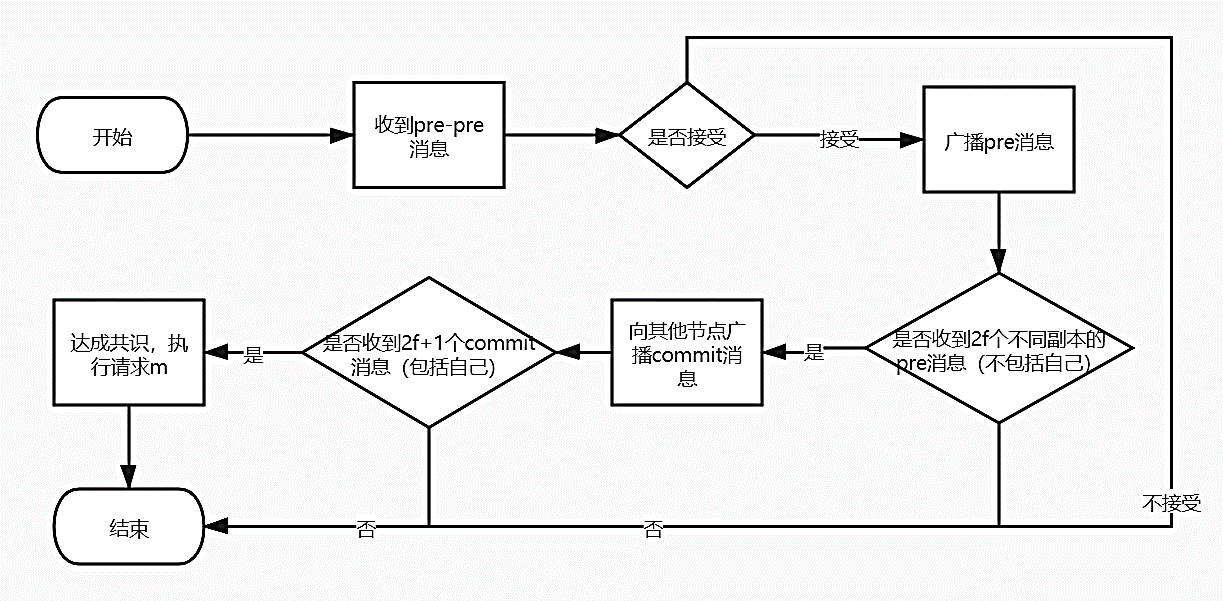
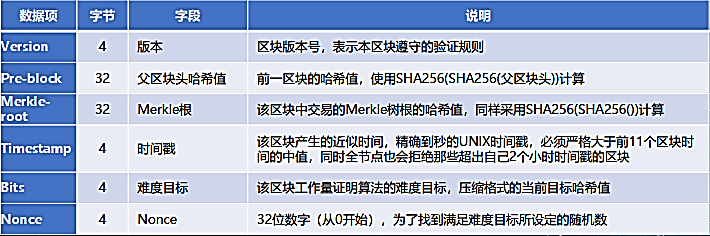
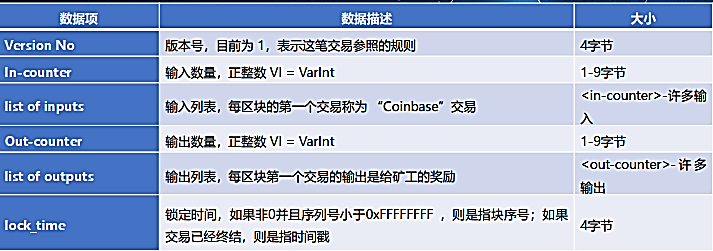
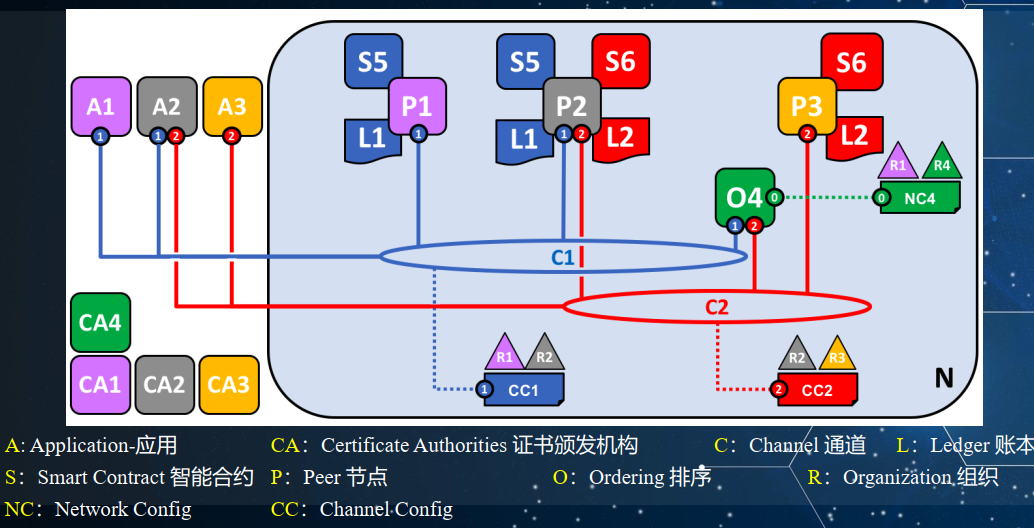
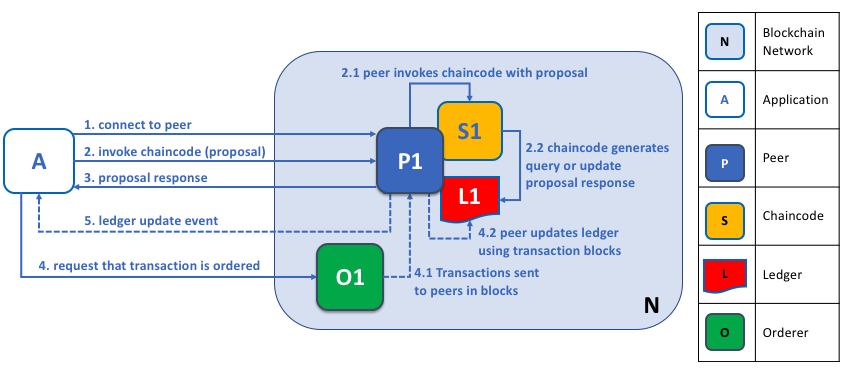
***概述*** BC又称分布式账本,是一种去中心化的分布式数据库.**BC本质**上是一个去中心化的分布式数据库,.***BC定义:***BC技术是在*不完全可信*的环境中,通过构建*点对点网络*,利用*链式数据结构*来验证与存储数据,借助*分布式共识机制*来确定BC结构,利用*密码学*的方式保证数据传输和访问的安全,利用由自动化脚本代码组成的*智能合约*来编程和操作数据.**特点:**开放共识,去中心化,无需信任,无法篡改,可溯源,交易透明,双方匿名.**数据层:**数据层封装了BC的底层数据存储和加密技术.每个节点存储的本地BC副本(数据账本)可以被看成三个级别的分层数据结构:BC、区块、区块体(封装交易).每个级别需要不同的加密功能保证数据的完整性和真实性.**不可变数据:**只能添加、不能篡改,无法篡改.不可变数据+时间刻度→互联网加上了时间轴.**加密基于密码学体系:**PKI公钥体系,零知识体系,数字签名,数字指纹,非对称加密**网络层:**网络层封装了BC系统的**组网方式**、**消息传播机制和验证机制**,组网方式通常采用点对点.网格网络(Mesh),权限对等、数据公开,数据分布式、高冗余存储.**共识层:**共识层主要封装BC系统使用的各类共识算法.BC技术的核心优势之一就是能够在决策权高度分散的去中心化系统中使得各节点高效地针对区块数据的有效性达成共识.*常见共识算法*:PoW、PoS、BFT、DPoS等.早期的btbBC采用高度依赖节点算力的PoW保证btb网络分布式记账的一致性**激励层:**激励层是将经济因素集成到BC技术体系中来,包括经济激励的*发行机制*和*分配机制*等,主要在**公有链**当中出现.在公有链中必须激励遵守规则参与记账的节点,并且惩罚不遵守规则的节点,使得节点最大化自身收益的个体理性行为与保障去中心化的BC系统的安全和有效性的整体目标相吻合.私有链系统不一定需要进行激励,因为参与记账的节点往往是在链外完成了博弈,通过强制力或自愿来要求参与记账.另一激励的来源则是交易费.新创建区块没有系统的奖励时,矿工的收益会由系统奖励变为收取交易手续费.**合约层:**合约层封装BC系统的各类脚本代码、算法以及由此生成的更为复杂的智能合约.数据、网络和共识三个层次作为BC底层’虚拟机’分别承担数据表示、数据传播和数据验证功能的话,合约层是建立在虚拟机之上的商业逻辑和算法,是实现BC系统灵活编程和操作数据的基础.“智能合约”:满足一定条件可以自动执行的合约.**应用层:**BC技术是具有普适性的底层技术框架,除可以应用于数字加密货币外,在经济、金融和社会系统中也存在广泛的应用场景.***BC特征:*去中心,去信任:**BC由众多节点共同组成一个端到端的网络,不存在中心化的设备和管理机构.节点之间数据交换通过数字签名技术进行验证,无需互相信任,只要按照系统既定的规则进行,节点之间不能也无法欺骗其它节点.**开放,共识:**任何人都可以参与到BC网络,每一台设备都能作为一个节点,每个节点都允许获得一份完整的数据库拷贝.节点间基于一套共识机制,通过竞争计算共同维护BC.任一节点失效,其余节点仍能正常工作.**交易透明,双方匿名**:BC的运行规则是公开透明的,所有的数据信息也是公开的,因此每一笔交易都对所有节点可见.由于节点与节点之间是去信任的,因此节点无需公开身份,都是匿名的.**不可篡改,可追溯:**单个甚至多个节点对数据库的修改无法影响其他节点的数据库,除非能控制整个网络中超过51%的节点同时修改,这几乎不可能发生.BC中的每一笔交易都通过密码学方法与相邻两个区块串联,因此可以追溯到任何一笔交易.***BC分类:按照去中心化程度:*公有链:**无官方组织及管理机构,无中心服务器,参与的节点按照系统规则自由接入网络、不受控制,节点间基于共识机制开展工作.**联盟链:**由若干机构联合发起,兼具部分去中心化的特性.**私有链:**建立在某个组织内部,系统的运作规则根据组织要求设定,修改甚至是读取权限仅限于少数节点,真实性和部分去中心化特征.**按照有无许可**:**无许可BC:**一种完全去中心化的分布式账本技术,允许节点自由加入和退出,无需通过中心节点注册、认证和授权,节点地位平等,共享整个账本.**许可BC:**存在一个或多个具有较高权限的节点,可以是可信第三方,也可以是协商制定有关规则,其他节点只有经过相应授权后才可访问数据,参与维护.***BC简史*BC1.0**解决交易速度、挖矿公平性、能源消耗、共识方式以及交易匿名等问题,参照物为btb(BTC).**btb运行机制:**1.产生新交易.2.通过P2P网络被广播到所有的参与节点.3.各节点都会将新交易进行验证(哈希),并各自形成一个等待上链的区块.4.通过共识算法选出拥有记账权的节点.5.获得记账权的矿工通过P2P网络广播它的新区块,全网其它节点核对该区块记账的正确性.6.超过一定数量的节点验证新区块无误后,就可以将这个区块连接到上一个区块上组成BC.**BC2.0**解决数据隐私、数据存储、BC治理、高吞吐量、域名解析、合约形式化验证等问题,参照物为ytf(ETH).优势:加入了**部署智能合约**功能,可以根据需求发布自己的智能合约.ytf网络中的每笔交易都需要支付一定的*手续费*.无论是转账交易还是部署智能合约,所支付的手续费越高,该交易就越快地被打包进区块中,也是以太币最主要的价值.**智能合约:**一种旨在以信息化方式传播、验证或执行合同的计算机协议,它允许在没有第三方的情况下进行可信交易,这些交易可追踪且不可逆转.***数字钱包*:**拥有私钥就拥有对应地址的数字货币,管理密钥的软件为’钱包’.**全节点钱包:**全节点钱包在使用时需要下载所有节点的信息.**轻钱包:**轻钱包在使用时不需要下载节点信息,交易速度低.***BC应用场景:***多源身份认证,分布式声誉系统,数据协同交换(数据发现、数据交易),分布式流程协作**小结BC:定义:**BC是一个*分布式账本*,一种通过去中心化,去信任的方式集体维护一个可靠数据库的技术方案 **数据角度**:BC是一种几乎不可能被更改的*分布式数据库*,分布式不仅体现在对数据的分布式存储,也体现在对数据的分布式记录 **业务角度**:BC是多种技术的整合的结果,通过*新的数据结构、分布式共识机制、哈希加密算法*以及独特的运行机制,使得去中心化的信任构想成为现实.

*********BC数据层账户体系:1.资产为核心建模的UTXO (比特币引入)/2.用户为核心建模的账户余额模型(类比银行账户)***BC是一个通过Hash指针将数据区块连接起来的数据结构,用来九路系统中确认的每笔交易 ***区块结构:*数据区块**是BC的基本元素,是一种记录交易的数据结构.区块体只负责记录前一段时间内的所打包交易信息,区块头记录当前区块的元数据.区块类似于账本中的账页,其物理存储形式可以是文件 (如btb),也可以是数据库(如ytf).创世区块:第一个区块***区块头:****btb*系统的区块头主要封装了当前**版本号**、**前一个区块的地址**、 **Merkle根**、**时间戳**、**当前区块的目标哈希值**、当前区块PoW共识过程的解**随机数**等信息.分为三类: **引用父区块**哈希值的数据Pre-block;当前BC所有交易经过哈希运算后得到的**Merkle根**,指向区块体所封装的交易;由**目标哈希值、时间戳与随机数**组成,这些信息都与共识竞争相关,是决定共识难度或者达成共识之后写入区块的信息.**目标哈希值**:定义了矿工需要进行挖矿的工作量证明的难度值.根据新区块挖掘出的速度,目标值会进行调整 **Nonce**:初始为0.当Nonce小于目标值则挖矿成功 **区块的主标识符**:区块头的哈希值,两次SHA256哈希,唯一标识一个区块.**区块高度**,将BC看成一个垂直的栈.也常用来标识一个区块,但可能不唯一.***区块体:***区块体包含了当前区块的交易数量和经过验证的、区块创建过程中生成的所有交易记录.**交易**是BC网络中传输的最基本的数据结构,所有有效的交易最终都会被封装到某个区块中,存于BC上.***btb交易过程:*元数据**:主要存放一些内部处理的信息,包括版本号、交易大小、输入的数量、输出的数量、交易锁定时间,以及标识该交易的哈希值.可以使用该哈希指针指向这个交易.**交易的输入列表**:每笔交易的所有输入排成一个序列,每个输入的格式相同,被序列化成字节流在网上传播.**交易的输出列表**:每笔交易的所有输出也排成一个序列.每个输出的内容分成两部分,一部分是特定数量的btb,以 ‘聪’为单位(最小的btb单位) ;另一部分是锁定脚本,即提出支付输出所必须被满足的条件以 ‘锁住’这笔总额.交易的所有输出金额之和必须小于或等于输入金额之和.当输出的总金额小于输入总金额时,二者的差额部分就作为交易费支付给为这笔交易记账的矿工.***UTXO:***借助前一笔交易的哈希指针,所有交易构成了多条以交易为结点的链表,交易可溯源至Coinbase.输出没有和任何另一笔交易的输入对应称UTXO.通过收集当前所有的UTXO,可以快速验证某交易中的btb是否已被花费.收集某人所有地址的UTXO,可以统计所拥有的btb数.btb系统不存在’账户’,只有’地址’(钱包).可以在btbBC上开设无限多个钱包地址,某人所拥有的btb数量是其所有钱包地址中btb的总和.btb系统并不会把某人的这些地址汇总起来形成其的账户.**btb甲到乙的转账:从甲的一个钱包地址转到乙的一个钱包地址.*一个转账交易过程:***用发起方私钥(从一个输出是发送方地址的交易中上一个UTXO)取出btb,并用私钥对新交易进行签名.一旦交易完成,这些btb就转到接收方的钱包地址中去.接收方钱包中新交易的未使用交易UTXO输出,只有接收方的私钥才可以打开.***使用UTXO优点:*** 1.UTXO设计易于确认btb的所有权,可以让双重花费更容易验证.只要确认上一个交易的确获得了花费的btb即可..2.UTXO设计与BC账本是完全融为一体的.BC账本存储的是状态.每一个BC中的交易都是一个状态转换函数,每一个新区块和它之前的所有区块一起形成了一个新的状态,如此重复持续下去.在确认之后,之前的状态就不可篡改,即不可随意更改***Merkle树:***btb系统采用二叉默克尔树来组织每个区快中的所有交易,*默克尔路径可以*快速校验某个区块中是否有特定的交易log\_2N.**交易类型:生产交易**(coinbase交易)每个区块的第一笔交易.没有输入地址,仅有输出地址,将系统新生成的加密货币奖励给创造当前区块的矿工.**通用地址交易**:BC系统中最常见的交易,由N个输入和M个输出构成,其中N,M>0.根据N和M的不同取值,可以进一步细分为一对一转账交易、一对多分散交易、多对一聚合交易和多对多转账交易.**合成地址交易**:合成地址交易是一类特殊交易,其接收地址不是通常意义的地址,而是一个以3开头的合成地址.合成地址一般M of 模式的多重签名地址,其中1≤N≤3、1M≤N,通常选择N=3.合成地址的交易构造、签名和发送过程与普通交易类似,但其地址创建过程需要三对公钥和私钥,其中公钥用于创建地址、私钥用于签名.M=1 N=3,任意1个都可使用,这种私钥冗余可防止私钥丢失,即使其他2个私钥丢失也不会造成损失M=2 N=3,有2个同时签名才可使用该地址的币,常见于三方中介交易场景.M=N=3:3个私钥同时签名才可使用该地址的币,常见于多方资产管理场景.***BC的运行流程:***①源节点创建交易并验证目的节点的地址 ②源节点对交易进行签名加密 ③源节点将该交易广播至全网其他节点④全网节点接收交易并验证其有效性,直到该交易被全网大多数节点验证和接受⑤交易被暂存于节点内存池,并判断是否为孤立交易⑥交易被打包至节点本地区块中⑦全网共识结束后,获胜节点将其本地区块追加到主链⑧交易在主链上被越来越多的后续区块确认.四个环节:**交易生成、网络传播与验证、共识出块、激励分配 交易生成:**源节点创建交易,将目的节点的公钥作为交易的参数,使用自己的私钥对新交易签名**网络传播与验证:**是P2P网络,使用Gossip协议传播交易.每个节点收到交易后独立对其有效性进行验证,通过后才会中继转发到其他节点.通过验证环节,有效抵御了恶意交易、垃圾信息的传播和拒绝服务攻击.**交易池管理:交易池:**一个内存池用于存放待确认打包的有效交易.**孤立交易池:**暂时存放缺失父交易的子交易.交易池导致的交易拥堵和低手续费交易不能及时确认.**交易费与优先级:** 交易优先级=(sum\_每个输入对应的UTXO UTXO交易额 × UTXO存在时间)/交易字节长度.交易字节长度=148×输入数+34输出数+10.btb采用0.576作为基准优先级,低于该值则会被收费**共识竞争与构建区块:**btb采用**工作量证明(PoW)**共识算法,核心思想:通过引入分布式节点的算力竞争来保证数据一致性和共识的安全性.各矿工节点基于各自的计算机算力相互竞争来共同解决一个*求解复杂*但*验证容易*的SHA256数学难题(挖矿) ,最快解决该难题的节点将获得区块记账权和系统自动生成的btb奖励.该数学难题可表述为:根据当前难度值,通过搜索求解一个合适的随机数(Nonce)使得区块头各元数据的SHA256哈希值小于或等于目标哈希值.btb系统区块的平均生成时间为**10分钟**.PoW共识中每个矿工重复进行以下步骤:1.搜集当前时间段的全网未确认交易,并增加一个用于发行新btb奖励的Coinbase交易,形成当前区块体的交易集合2.计算区块体交易集合的默克尔根记入区块头,并填写区块头的其他元数据,其中随机数Nonce置零3.随机数Nonce加1,计算当前区块头的双SHA256哈希值,若小于或等于目标哈希值,则成功搜索到合适的随机数并获得该区块的记账权;否则继续直到任一节点搜索到合适的随机数为止4.若一定时间内未成功,则更新时间戳和未确认交易集合、重新计算默克尔根后继续搜索.**难度和难度调整机制**:难度是BC系统(特别是PoW类型公有链系统)的重要参数,用来度量矿工成功挖到下一个区块的难易程度.符合要求的区块头哈希值通常由多个前导零构成,前导零越多,难度越大.**目标值计算公式:**(十六进制的前两位为幂).当前区块难度值=创世区块Target/当前区块Target 难度的调整在每个节点中独立自动发生.每2016个区块产生后,所有节点都会有难度调整:**新难度=旧难度×(过去2016个区块的实际时间/20160分钟) [4,1/4]分叉处理与主链判定:**多个矿工节点在同一时间段内成功搜索到符合哈希结果要求的随机数,则这些矿工都将认为自己在共识竞争中获胜并向btb网络中广播其构造的区块,从而产生在同一区块高度出现多个不同的有效区块的情况,即分叉.为保证BC系统中仅有唯一的主链,必须定义合适的主链判定准则来从多个分叉链中选择符合条件的唯一主链.此时不在主链上的区块将成为’孤块’,发现孤块的矿工节点不会得到btb奖励.BC的形状是树状结构,其中每个共识轮次对应的时间点上仅有唯一区块是有效的,因而树有唯一的主链条.**最大工作量原则判断主链(依次进行判断):1**若不同分支的区块高度不同,则选择最长区块高度的分支为主链**2**选择难度系数最大的分支作为主链**3**则选择接受时间最早的分支为主链**4**等待新区块产生并连接到某个或者多个分支、区块高度增加后,重复步骤1-3直至选出主链.此时,生成新区块的节点即可对当前多个分支子链进行’投票’,并链接至最有可能成为主链的分支子链上.***数据层关键技术:*时间戳:**1970.1.1.0.0.0起至现在的总秒数.能够表示一份数据在一个特定时间点已经存在的完整的可验证的数据.**功能:**记录某件事情的发生日期时间,证明事实存在并保证先后关系.基于文档时间戳的数字公证服务以证明各类电子文档的创建时间,由此保证数据的**可追溯与不可篡改**.时间戳服务器对新建文档、当前时间及指向之前文档签名的哈希指针进行签名,后续文档又对当前文档的签名再进行签名,如此形成了一个基于时间戳的证书链,该链反映了文件创建的先后顺序,且**链中的时间戳极难改**.**BC中时间戳**:时间戳服务器:通过把以数据区块形式存在的一组btb交易实施哈希运算并加盖时间戳,并在btb网络中广播该哈希值.这个时间戳证明在该时间这个数据一定是存在的,因为只有数据只有在该时间才能得到相应的哈希值.每个时间戳hash包含了前一个时间戳,对其中一个时间戳的篡改的代价极大,需要同时篡改其后生成的所有时间戳.**btb系统的两个防止节点恶意修改本地时间规则:**①btb节点会与其连接上的所有其他节点进行时间校正,>=5,选择节点的时间中位数作为时间戳,时间差>70min不会更改②大于前11个区块的中位数,小于btb节点的网络调整时间+2小时.**哈希函数:**哈希函数的输出值成为哈希值或数字摘要 **哈希碰撞:**不可避免,加长输出字符串长度降低哈希碰撞的概率.**技术特征:抗原像**:也称单向性,对y,找到任意原像x使得H(x)=y在计算上是不可行的.**抗第二原像**或称**弱抗碰撞性**:即给定x1,寻找x2,使得H(x1)=H(x2)不可行.**强抗碰撞**:寻找输入x1和x2,使得H(x1)=H(x2) 不可行.**迷题友好**:对于任意n位输出y来说,假设k是从具有较高不可预测性的高阶最小熵分布中选取的,无有效方法在<<2^n时间内找到x ,使得H(k|x)=y成立.**雪崩效应**:输入数据发生任何细微变化,会导致输出结果发生明显改变.**定长/定时性**:不同长度输入数据的哈希过程消耗大约相同的时间且固定长度输出.**在BC中的应用**:**完整性校验**:单向性和抗碰撞性 防止出现未经授权的篡改.**数据要素管理**:抗碰撞性使其可以作为任意数据的 ‘**数字指纹**’,从而可以利用数据的哈希值来对其进行高效管理.公钥、私钥、地址交易ID、区块ID,默克尔树、BC系统的数字签名.**共识竞争**:迷题友好性,矿工除了付出大量算力资源执行哈希运算之外,没有其他捷径.**MD5**:(MD:信息摘要)步骤:填充(mod 512=448,填充1一个1和若干9,再补充64表示填充前信息长度)初始化缓冲区,处理分组数据；**比特币公钥-地址转换:SHA-256+RIPEMD-256双哈希,65byte->20byte, SHA256 BASE58 ->33byte**.**默克尔树:**作用:快速归纳和校验区块数据的存在性和完整性.优点:1.区块头包含根哈希值 2.简化支付验证(SPV),不运行完整BC网络节点也能对交易数据进行校验(轻节点)Hash函数采用SHA256算法,二叉hash树(*交易数量为奇数时,最后一个交易重复*)通过**默克尔路径**可快速验证某个区块是否存在指定交易 **SPV节点验证支付**:**具体步骤:**1) SPV节点获得待验证交易信息,向BC发起默克尔区块消息查询请求2)其他有完整BC数据的节点收到请求后:1.定位包含该交易的区块2.检查该区块是否属于整个网络中的最长链3.取出所有交易生成默克尔树,利用getProof获得待验证交易的验证路径4.将验证路径发送回请求源SPV节点3) SPV节点获得验证路径后执行:1.同步BC,确保是整个网络中最长的一条2.先拿默克尔根去BC中查找,确保该默克尔根哈希是在链条中4.利用获得的验证路径,再进行一次默克尔哈希校验,确保验证路径全部合法,则交易真实存在4.根据该交易所在区块头的位置,确定该交易已经得到多少个确认.如无误认为交易可信 **非对称加密:** RSA算法.原理:两个超大素数相乘得到的结果几乎无法因式分解逆运算得到原本的素数.**btbBC网络则采用了椭圆加密算法ECC(相比RSA,密钥更短,加密安全性更高)**y^2=x^3+ax+b,加法 过两点做直线与椭圆曲线的交点关于x对称；取反 关于x对称；特点:方向容易素数相乘容易,反向很难；椭圆曲线签名算法ECDSA**.非对称算法在BC的应用:**私钥证明了用户对于账户的所有权,使用btb=拥有该账户对应的私钥.登录认证中,用户输入私钥信息,客户端使用私钥加密登录信息后发送给服务器,服务器解密认证登录信息.•btb交易中用户使用私钥对交易进行签名,交易广播后,确保信息由A发送.充分利用了非对称加密的特性,使用其中一个密钥加密信息后,只有对应另外一个密钥才能解开;公钥可以向其他人公开,公钥不能逆推出私钥,保证安全性.**数字签名:**一种证明数字消息,文档或者资产的真实性的数学方案,作用 身份认证,不可抵赖,完整性 **定义**:附加在数据单元上的一些数据,或是对数据单元所做的密码变换,这种数据和变换允许数据单元的接收者用以确认数据单元来源和数据单元的完整性,并保护数据,防止被人进行伪造.**数字签名方案的模型**七元组(M,S,SK,PK,Gen,Sign, Verify):①M:明文消息空间;②S可能的签名空间;③SK:签名密钥空间,用于生成签名的私钥集合;S④PK:验证密钥空间,即用于验证签名的公钥集合;⑤Gen:N→SK×PK:密钥生成算法,可生成一对匹配的公钥pk和私钥sk; ⑥Sign:M×SK→S:签名算法,利用私钥sk生成消息m的签名s;⑦Verify:M×S ×PK→{True,False}:验证算法,公钥验证签名s是否正确.数字签名的模型要素.**多重签名:**btb系统一般采用’N选M’的形式,N个私钥,至少需要M个私钥签名***BC网络层P2P网络:*定义:**对等节点拥有同等特权,每个节点将其部分资源(如计算、存储、网络带宽)直接提供给其他节点使用 既是服务器,又是客户端;去中心化:网络中无需中央管理节点的存在.**特点**:•可扩展性强:节点自由加入退出,P2P网络的自组织,自配置和自动负载均衡特性,破解了C/S模式下中心服务器的性能瓶颈问题.•健壮性好:服务和资源分散于各个节点,不经第三方.•高性价比 ­有效利用分散于网络中大量节点上的空闲资源.•私密性:信息传输无需通过集中节点,所有节点具备中继转发能力,提高通信的匿名性,隐私得到保护.•均衡性:资源和处理能力分布于多个节点,避免网络流量过于集中.**P2P网络分类**:**混合式对等网络**•C/S和P2P两种模式的混合,早期网络从C/S到P2P的过渡• Napster•不是完全去中心化的**无结构对等网络**•无固定网络结构图,无中心节点•节点地址没有统一标准•Gnutella,是纯粹意义上的P2P网络**结构化对等网络**•一般采用哈希函数将节点地址规范为标准的标识•内容的存储位置与节点标识之间存在映射关系,可以实现有效的节点地址管理• Chord,Pastry **P2P与覆盖网络(Overlay Network)**•P2P网络构建在更底层的物理网络上,为特定应用提供支持,是典型的**覆盖网络**:建立在另一个网络上、并为更高层应用提供支持的**中间层网络**.­**作用**:使得上层应用无需过多考虑与网络有关的对等实体发现、直接通信、数据安全、资源定位、网络标识及其分配、节点加入与退出、负载均衡等问题,将精力集中在业务功能实现上.Gossip协议(反熵)步骤:节点A周期性地选择相邻的K个节点发送自身存储的数据.节点接收数据后自身没有的数据存储下来,有则丢掉,重复节点A的行为.节点间发送数据的方式1.push模式:将(key,version,value)推送给相邻K个节点,相邻节点更新version新的数据2.pull模式:将(key,version)推送给K个节点,相邻节点将本地version比A新的数据给A.3.push/pull模式:A push模式相邻节点,pull更新A.push通信一次,pull两次,pull/push三次,一致性的收敛速度与通信次数成正比.缺陷:消息延迟,消息冗余.btb的p2p网络:系统的组网方式、消息传播机制和验证机制***btbBC网络:*btb网络节点:**•btb网络由多种类型的节点组成,功能集合:网络路由(N)、完整BC(B)、矿工(M)、钱包(W).每个节点参与全网路由,也可能包含其他功能.**全节点**:拥有完整的、最新BC数据,能独立自主地校验所有交易**SPV节点/轻量级节点**:只保留区块头数据,SPV完成交易验证,没有BC的完整拷贝.**组网方式:**新节点加入BC网络:**地址数据库:**在地址数据库保存.启动时由地址管理器载入(节点第一次启用无法使用)**通过命令行指定:**用户可以通过命令行将指定节点的地址传递给新节点.**DNS种子,硬编码地址,通过其他节点获得**(节点间通过getaddr,addr交换IP)**节点发现过程:**1.用户btb程序启动时,不知道任何活跃节点IP地址2.程序会向DNS地址发出查询请求 获得接入网络和发现地址3.DNS服务器返回的响应包(DNS的A记录)中包含一或多个全功能节点的IP地址***数据传播协议:***节点间TCP,8333.通用**BC网络**的核心场景:①节点入网建立初始连接②节点地址传播发现③矿工、全节点同步区块数据④客户端创建一笔交易⑤矿工、全节点接受交易⑥矿工、全节点挖出新区块,并广播到网络中⑦矿工、全节点接收广播的区块**消息(Message)**是不同节点间信息传输的**基本单位**,协议体现为消息格式的约定和时序.**基本格式**:消息头+消息体•**起始字符串**:奇异数(0xf9beb4d9),标识下一个消息的开始•**命令名**:ASCII,内容在消息体中•**消息体大小**(字节)长度•**校验**值•**消息体** **建立初始连接:**•A通过发送Version消息到远端对等节点B表示连接成功,包括节点的版本消息、区块和当前时间•B收到后检查 兼容则确定连接,返回Verack消息,向Ａ发送B Version消息•节点Ａ收到后检查 兼容则返回Verack消息,连接成功建立.**地址广播及发现**•成功连接后,新Ａ向Ｂ发送包含自身IP地址的Addr消息.相邻节点会将此Addr消息再度转发给各自相邻节点,保证新结点A可被更多节点获知.•Ａ可以向Ｂ发送Getaddr消息,B会回送若干地址信息,如其它节点点IP地址、端口等数据.**同步区块数据(全节点)**•全节点连接到对等节点后,首先构建完整的BC.该节点向相邻节点发送version消息,该消息中包含BestHeight字段(区块高度).互相发送version消息,对等节点就可得知双方区块数量.•对等节点发送getblocks消息,包含了本节点保存的BC顶端区块hash,若收到的hash在自己BC中不属于顶部->自己链更长.•更长BC的节点识别出其他节点需要补充的块,发送库存inv消息,包含块的hash值,从而告知其他节点这些块的存在.收到inv的节点发现缺少块,就会向周围节点(未必是发送节点)发getdata请求具体某些块的数据来补全.收到getdata的节点把请求的块数据发送.**同步区块数据(SPV节点)**•同步的是区块头.•getheaders,headers消息.**初始区块下载IBD**通过下载区块、区块头、交易,btb的所有的区块交易在本地有副本,成为完整的btb网络节点,跟btb网络同步.通常要求下载并验证此时网络中**最长、最正确**的区块序列,从**编号1的区块开始**;**新加入或离线24以上**,均需执行这个操作,方可接入btb网络,才能验证未确认的交易/新近挖出的区块**btb的BC数据下载方法块优先**假设B已完成同步,A刚启动,A除了创世区块以外没有数据.①A向B发送一个’Getblocks"消息:HeaderHashes:区块0的哈希值(大端)(最新区块是创世区块).StoppingHash:填写全0,(同步到最新).②B收到,根据要求返回**’inv’消息**给A.B返回有<=500条内容的inventory清单条目:Type字段:block,Unique Identifier字段:区块哈希值.**哈希的顺序代表区块的顺序**.③IBD节点A接收,发送**getdata消息**向B请求128个块(上限),**按顺序请求**(**每个块头均引用前一个块的头哈希)**.④B回复所请求的每个块.以序列化的块格式放入,以单独的block消息发送.⑤A对接收到的每个块,对其进行验证,并**保持<=128**下载队列.•A收到区块数据,对这个区块的合法性进行验证,等待下一个区块数据.A接收到128个后,A根据之前inv消息继续发送,**如此循环**.这样就完成了一轮区块发送任务.•为避免出错,加入了**Checkpoint**(指定一个区块高度的区块哈希等于某个哈希值).块优先的**优点**简单 **缺点**IBD节点的所有下载依赖于单个同步节点和块发送的**顺序性**.导致:**速度限制**:所有请求均发送到同步节点,若同步节点上传带宽有限,速度很慢**重复下载**:同步节点向IBD节点发送非最佳的BC,迫使IBD从其他节点重新下载.**磁盘空间浪费**:与重复下载相关**高内存使用**:同步节点无序发送块,验证其父节点后才能对其进行验证.孤立块在等待验证时会存储在内存中.**头优先:**•先尝试下载链中区块的**描述头结构**,然后以**并行方式**连接多个网络节点下载区块.•头优先是Ａ使用**GetHeaders消息发起**,B**返回Headers消息**(最多2000) •A收到B的’headers’后,验证区块哈希合法性,同时:1.继续同步blockheaders.2.同步区块数据(’getdata’’data’)•可以同时从(默认8)个节点并行下载区块数据.每次从一个节点最多只请求16个区块,每次最多128个区块.使用最大1024个区块的下载移动窗口,最大化下载速度.**交易广播**交易数据传播协议核心步骤:①新生成的交易数据使用**inv**向全网所有节点广播②节点发送**getdata消息**请求inv消息中所有交易的完整信息,收到的节点使用tx发送交易.接收节点将收集到的交易数据存储到一个区块中③每个节点在区块中找到一个**POW**④当节点找到POW后向全网所有节点广播此区块block消息⑤当在区块中的所有交易都有效的且未存在过的,其他节点认同该区块的有效性⑥其他节点接受该数据区块,并在该区块的末尾制造新的区块以延长该链条.**新块广播**•矿工发现新的区块后,将此区块在全网尽可能大的范围内广播.方式:**1.主动推送**:•向每个完整节点对等发送带有新区块的’block’消息•合理地绕过标准中继方法•不能判断其对等方已经拥有该块.**2.区块中继**•不主动推送•发出**Inv消息**,携带新区块消息•由收到Inv消息的节点把区块取走.**检测节点存活ping消息**确认接收方是否仍处于连接状态,检测节点是否存活•接收方回复pong消息,告诉自己仍然存在.•默认情况,**超过20分钟未响应**ping消息会被认为该节点已经从网络中断开.***数据验证机制***每个接收到新区块的节点都将对区块进行**独立验证**,验证**通过进行转发**,尽早杜绝无效或者恶意数据在网间传播,预防小部分节点串通作恶导致无效区块被网络接受,保证网络中传播区块的正确性.•btb网络,节点接收邻近节点数据后,首要工作是**验证该数据的有效性**.收集和验证P2P网络中广播的尚未确认的交易数据,并对照预定义的标准清单,各方面校验交易数据的有效性,并将有效交易打包到当前区块中.**数据验证清单**①验证**区块大小**在有效范畴②确认区块**数据结构(语法)的有效**性③验证区块**至少含有一条交易**④验证**第一个交易是coinbase**(PreTranshash =0,PrevTxout index=-1),有且仅有一个⑤验证区块**头部有效性**:1.确认区块版本号是本节点可兼容的2.区块引用的前一区块是有效的3.区块包含的所有交易构建的默克尔树是正确的4.时间戳合理5.区块难度与本节点计算的相符6.区块哈希值满足难度要求⑥验证**区块内的交易有效性**,具体检查:•检查交易语法正确性•输入与输出列表不为空•lock\_time小于等于INT\_MAX/nLockTime, nSequence的值满足(之前11个区块时间中位数)•交易的字节大于等于100•交易中签名数量小于签名操作数量上限•解锁脚本(scriptSig)只能够将数字压入栈中,并且锁定脚本必须要符合isStandard的格式(拒绝非标准交易)•coinbase交易,验证签名长度[2,100]•输出值,总量在规定值的范围(不超过总币量,>0)•对于输入,若引用的输出已存在于内存池,该交易被拒绝•验证孤立交易:对于每一个输入,引用的输出交易缺少,该交易被认为孤立交易.若与其匹配的交易还没有出现在内存池中,加入到孤立交易池•若交易费用太低(低于设定值)无法进入一个空的区块,则交易将被拒绝•每一个输入的解锁脚本必须依据相应输出的锁定脚本来验证•不是coinbase交易,确认交易输入有效,对于每一个输入:验证引用的交易存于主链;验证引用的输出存于交易;若引用coinbase交易,确认至少获得COINBASE\_MATURITY(100)个确认;确认引用的输出没有被花费;验证交易签名有效;验证引用的输出金额有效;确认输出金额小于等于输入金额(差额即为手续费)•若是coinbase交易,确认金额小于等于交易手续费与新区块奖励之和.**矿池网络协议** 挖矿本质是执行Hash函数的过程,输入数据是区块头.•CPU挖矿,搜索空间主要由nNonce提供,矿机时代搜索空间转向hashMerkleRoot.比特币挖矿的流程:• 打包交易,检索待确认交易内存池,选择包含进区块的交易.• 构造Coinbase,产出规则,矿工可以计算自己本区块的收益.•构造hashMerkleRoot,对所有交易构造Merkle数.• 填充其他字段,获得完整区块头.• Hash运算,对区块头进行SHA256D运算.• 验证结果符合则向全网广播;Setgenerate协议接口代表了CPU挖矿时代(nNonce自增)Getwork协议代表了GPU挖矿时代,需求主要源于挖矿程序与节点客户 端分离,BC数据与挖矿部件分离.思路是:由节点客户 端构造区块,然后将区块头数据交给外部挖矿程序,挖矿程序遍历 nNonce进行挖矿,验证合格后交付回给节点客户端,节点客户端验证合格后广播到全网.• 拥有完整数据的节点构造区块头,即提供Version,Prev-block,Bits和Merkle-root这4个字段• 挖矿程序主要是递增遍历nNonce• 对于Getwork而言,矿工对区块一无所知,只知道修改nNonce这4个字节,共计232大小的搜索空间• 禁用Getwork协议,转向更高效的Getblocktemplate协议• 矿池的核心工作是给矿工分配任务,统计工作量并分发收益.矿池将区块难度分成很多 难度更小的任务下发给矿工计算,矿工完成一个任务后将工作量提交给矿池.矿池采用 getblocktemplate协议与节点客户端交互,采用stratum 协议与矿工交互,最典型的矿池搭建模式•Getblocktemplate协议让矿工自行构造区块• 因为由矿工构建coinbase交易,这种方式所带来的搜索空 间巨大•扩大了搜索空间,但数据负载过大• 协议是为了扩展支持矿池挖矿而编写 的挖矿协议• 数据json封装• Stratum协议利用Merkle树结 构特性,从coinbase构造merkleroot,无须全 部交易,只要把与coinbase涉及的默克尔路径 上的hash值返回即可.数据规模将压缩至log2(N),大大降低了矿池和矿工交互的数据量.***BC分叉:*自然分叉**,机器共识过程产生的临时分叉.btb去中心化P2P网络.节点地域分布/网络传输延迟,造成节点接收新区块存在的时间差异.不同节点近乎同时发掘出新区块AB并进行广播的时候,就会造成后继BC分叉.区块是**并发产生**的,并且具有广播延迟,**高出块率会导致更多的分叉**.分叉浪费网络和处理资源,**降低安全性**•**最长链规则不再安全 同步账本的时间应该远小于出块时间,否则就会增加分叉的概率**.**人为分叉**,人的共识失败产生的分叉(BIP):**’软分叉’**是**向前兼容**的分叉.新规则下产生的区块可被未升级的旧节点所接受,旧节点无法识别、解析新规则.互相兼容,对整个系统的影响较小.**’硬分叉’不向前兼容**,旧版本节点不接受新版本节点创建的合法区块,于是新旧版本节点开始在不同的BC上运行,新旧节点可能长期并存,有可能会*长期存在*的,**分叉链的存活在于其算力的大小**.**软分叉示例**•基于coinbase字段的随机数扩展,coinbasedata[2,10]字节, BIP-34要求开始必须在开头包含块高度且更新块版本信息过程:1.初始矿工将块版本号设置为’2’ ,准备升级,不 要 求 coinbase data包含块高度2.近1000个块中超75%’2’ , 开始强制要求版本号设置为’2’ ,要求coinbasedata包含块高度, 但此时版本号为’1’ 的区块仍被接受3.超过95%是’2’时,迫使最后一小部分节点进行升级.**硬分叉**•A->B以下情况:1.A版本被广泛支持,B版本算力不足消亡,保留原链.2.B保留新链.3.A、B同时并存,最为符合严格意义上的硬分叉,**(ETH,ETC**).4.A被广泛支持,B调整难度,小算力也存活.(与3)区别:这种分叉币几乎没有门槛).5.B获得支持,A调整代码,小算力也可存活.硬分叉的**过程**:**软件分叉**:新的客户端不兼容,首先客户端出现了分叉 **网络分叉**:新节点在网络上运行,其发现的区块被旧版节点拒绝,旧版节点断开与新版节点的连接,网络出现了分叉 **算力分叉**:不同客户端版本的矿工算力将逐渐出现分叉 **链分叉:**升级的矿工基于新规则,拒绝升级的矿工基于旧规则,导致整个BC出现了分叉 **社区分叉**:每一种BC的背后都有其对应的社区、开发者、矿工等利益、信仰共同体,链的硬分叉同时也会带来对应社区的分裂:对于数字货币持有者来说,硬分叉会让他们额外增加一笔财富(分叉链Token) ***第四章:BC共识层*分布式系统模型与共识分布式系统模型:**分布式系统是组件分布在网络计算机上且通过消息传递进行**通信和行为协调**的系统.对外呈现为一个完美的、可扩展的’**虚拟节点**’,相对单物理节点更优越的**性能和稳定性**.**特征**:•并发性•缺乏全局时钟•组件故障的独立性**系统模型**•**结构模型** 构成系统各部分组件的位置、角色和它们之间的关系,定义了系统的各组件之间相互交互的方式以及它们映射到下面的计算机网络的方式.客户/服务器结构、对等结构.•**基础模型** 对公共属性的一种更形式化的描述,包括:交互模型、故障模型和安全模型.**时序模型**1.进程之间通过消息传递进行交互,实现系统的通信和协作功能.准确性受限•延迟大.•难有相同的时间概念.**同步系统:**•进程执行每一步时间有明确上下限.•每一条消息在已知的时间范围内确定被接收到.•本地与实际时间的漂移也在已知范围内.•超时:检测故障.•实际很少有真正的同步系统,进行理论分析和测试.**异步系统**:•对进程执行速度、消息传递延迟和时钟漂移率都没有限制.•实际大多数是异步系统.**部分同步系统:**•对系统执行时间有信息,不一定准确.•部分同步系统理论还不完善.**故障模型** 故障模型定义可能出现的故障形式,为分析故障带来的影响提供依据.设计系统时,知道应如何考虑容错的需求.**故障类型:崩溃故障**•节点正确运行直至崩溃.•节点崩溃后不可恢复,若其他节点可以检测到这种故障则称为’故障-停止’,否则称为’崩溃’.•崩溃后可以恢复运行,则称为’故障-恢复’.**遗漏故障**•节点或信道未能执行本来的动作导致消息丢失.•根据步骤位置分为发送遗漏故障、信道遗漏故障和接收遗漏故障.**时序故障**•节点过早或过迟的响应,(*同步系统假设*)•时钟故障、节点性能故障和信道性能故障.**拜占庭故障**•(随机故障).**安全模型**•分布式系统的模块特性以及开放性,使得它们暴露在**内部和外部的攻击之下**.•**安全模型的目的**是提供依据,以此分析系统可能收到的侵害,并在设计系统时防止这些侵害的发生.**分布式一致性(Consistency)** 定义:1.多个节点对某一变量的取值达成一致,达成一致则变量的本次取值即被确定.2多个服务节点,给定一系列的操作,在约定协议的保障下,使它们对外界呈现的状态是一致的(**保证集群中所有服务节点中的数据完全相同并且能够对某个提案达成一致)**.一致性要求:可终止性,约同性,合法性•**一致性分类**:•**强一致性**:更新后,都会获得最新的值.•**弱一致性**:不保证后续访问都会返回最新的更新的值.(**最终一致性:**,若经过一段时间后要求能访问到更新后的数据).**共识(Consensus)**•描述了分布式系统中多个节点之间,彼此对某个状态**达成一致结果**的过程.核心过程往往需要通过**共识算法**来达成.•共识和一致性是等价的可互换的.•**共识**侧重的是分布式节点达成一致性的过程和算法,是一种**手段**.而**一致性**侧重于节点共识过程最终达成的稳定状态,描述的是**结果状态**.共识并不意味着一致性(强一致性).**共识机制能够实现某种程度上的一致性**.**FLP定理**:在含有多个确定性进程的异步系统中,只要有一个进程可能发生故障,那么就不存在协议能保证有限时间内使所有进程达成一致**CAP定理**•网络服务不可能同时保证如下三个特点,最多只能保持两个:**一致性**:强一致性,同一时刻必须保持同样的值.**可用性**:(**A**:部分故障,系统仍可以处理用户请求,所有读写请求不会一直等待.**分区容错性**:(**P**:即允许丢失任意多的从一个节点发往另一个节点的消息.**在满足分区容错的前提下,不能同时满足一致性和可用性,只能兼顾.实际应用:**CAP定理:工程实践中,一般会适当放宽对特定性质的假设,例如放宽强一致性要求•弱一致性:用户读到某一操作对系统特定数据的更新需要一段时间.•最终一致性:在数据更新操作完成之后的某个时间点,分布式节点的数据最终达成一致.BC系统的设计遵从CAP定理:大多公有链通常牺牲强一致性,同时满足最终一致性、可用性和分区容错性.•某些联盟链或私有链可能会牺牲可用性来满足强一致性和分区容错性.**拜占庭将军问题:节点不可靠,信道可靠,同步系统(两军问题**节点可靠,信道不可靠•经典情形下两军问题是不可解的•TCP三次握手,两军问题的工程解**)** **问题模型假设:**\*一个网络𝑛个节点\*每个节点监听其他节点发送的消息𝑣1,𝑣2,…,𝑣𝑛\*存在𝑚个*恶意节点***求解问题:**在存在恶意节点的网络中,诚实节点能对决策问题达成一致 **求解条件:一致性**:诚实节点接收到相同的消息集合(所有节点角度)**正确性**:若𝑖诚实,其他诚实节点以它发送的消息作为vi(单个节点角度)⚫一致性转化为如下条件**无论𝑖是否诚实,任意两个诚实节点所保存的其消息均为𝑣𝑖(**单个节点角度).从单个节点角度出发.**口头消息(OM)算法OM(0)**情况:主节点向每个从节点发送消息每个从节点接收消息,若缺失则记为缺省值.**OM(m)**情况:(1)主节点向每个从节点发送消息𝑣(2)对任意从节点𝑖,其接收的消息𝑣记为𝑣i,其作为主节点运行OM(m-1)向剩余从节点发送𝑣𝑖(3)从节点采用majority (𝑣1,𝑣2,…,𝑣(n-1)).**算法容错率小于1/3.诚实节点对于诚实主节点的消息达成一致,当主节点为恶意节点时,诚实节点消息也能达成一致 BC解决拜占庭问题**:规则:1.一个时间段内只能发起一个消息.2.收到进攻消息的将军必须也要在消息上签名,确认各自的身份,并盖上时间戳,然后把这个信息拷贝下来传递给其他将军.3.信息加密.公私钥.4.各自看到信息的传递进度.**BC解决方式优势:**1.信息里每个将军都要签名验证身份能看到是哪些人篡改了消息.2.只要一个消息得到了>=6将军的同意,那么大家就达成了共识.**共识过程的主流模型记账节点**通过共识过程选定的记账节点**代表节点**特定算法选举出代表矿工节点参加共识过程**矿工节点**对数据或交易进行验证、打包、更新上链**数据节点**全体数据节点,生产数据或交易**主流模型:(12为选举共识,34为主链共识)**1.**选主**:核心,通过选举、证明、联盟或混合等方式从全体矿工节点中选出记账节点.2.**造块**:记账节点根据特定的策略将当前时间段内全体节点生成的交易或数据打包到一个区块中,并将新生成的区块广播给全体矿节点3.**验证**:矿工节点或代表节点收到广播的新区块后,将各自验证区块内封装的交易或者数据的正确性和合理性4.**上链**:验证通过后记账节点将新区块添加到主链,形成一条从创世区块到最新区块的完整的、更长的链条.有分叉需根据共识算法中的主链判别标准确定主链 **共识算法的分类:算法共识**•研究在特定的网络模型和故障模型的前提下,如何在缺乏中央控制和协调的分布式网络中确保一致性,实质是’机器共识’..**决策共识**•研究无中心的群体决策中,如何就最优的决策达成一致的问题,实质是’人的共识’ 根据**选主策略**,分为:选举类、证明类、随机类、联盟类、混合类;根据**容错类型**,分为拜占庭容错(BFT)和非~(CFT);根据**部署方式**,分为公有链、私有链、联盟链•**选举类共识**:先获得半数以上选票的矿工节点将会获得记账权.Paxos,Raft.•**证明类共识**ProofofX类共识,在每一轮共识过程中必须证明自己具有某种特定的能力,在竞争中胜出的矿工节点将获得记账权.PoW,PoS.•**随机类共识**根据某种随机方式直接确定每一轮的记账节点.Algorand,PoET.•**联盟类共识**:基于某种特定方式首先选举出*一组代表节点*,而后由代表节点以轮流或者选举的方式依次取得记账权.特点:代议制,DPoS.**通常,公有链基于拜占庭共识实现一致性,联盟链基于非拜占庭共识实现一致性 BC共识协议:**1.出块节点选举:在出块节点选举阶段,某节点成为出块节点,提出新区块.可能存在恶意节点及分叉块,不能直接加入本地2.主链共识对新区块及其构成的主链达成一致.共同保证了BC数据的**正确性和一致性**,从而为分布式环境中的不可信主体间建立信任关系提供技术支撑 **POW**引入了对一个特定值的计算工作,1+0s,若想要对已出现的区块信息进行修改,必须完成该区块外加之后所有区块的工作量赶上和超越诚实节点的工作量Y=SHA256(X,a) **POS-权益证明,**解决POW机制中大量资源被浪费的情况.计算持有占总币数的百分比,你占有币数的时间决定记账权.Y=HASH(X,N),*持币越多则挖矿越容易***分布式一致性算法Paxos算法**解决问题:在一个可能发生消息延迟、丢失、重复的分布式系统中如何就某个值达成一致,保证不论发生以上任何异常,都不会破坏决议的一致性.**(非拜占庭故障)Paxos算法节点分为三种角色**:**提议者(Proposer)**负责向A发起提案**接受者(Acceptor)**负责响应提案,对提案进行回应以表示自己接受提案**学习者(Learner)**只从别人那里学习已经确定的达成一致的提案结果.*一个节点同时拥有多个身份*.**若一个提案被半数以上A接受就被选定了(Chosen),并由Learner负责执行选定的提案**.**提案**由两部分组成:**提案编号id+提案值value** **id**由P自行选择决定,一般是**相互独立、不可重复的递增序列.Value是要等待达成共识的数据值**,原则:安全原则:不能做错；存活原则:多数服务器存活且彼此可以通信**约束条件对Value**:若没有Value被提出,不应该有Value被选定.只有被提出的值Value才可以被选定只有一个值Value可以被选定;不被选定的不会被执行 **对提案**:若只有一个提案被提出的话,那么这个提案应该被最终选定,A必须能够接受*多个不同的Value*.•P1:每个A必须接受它收到的第一个提案•P2:若一个提案(v)已经被选定,对于所有编号更大的被选定的提案,值是v.•P2a:若一个提案(v)被选定,对任何A接受的编号更大的提案,也是v.•P2b:若一个提案(v)已经被选定,那么对于任何P提出的编号更大的提案值也是v.•P2c:提案(n,v)被提出,存在一个由大多数A构成的集合S,满足下述之一:①没有成接受过小于n;②接受过的提案编号最大者为v.(P2<P2a<P2b<P2c,建立在P2c)**算法流程**两个阶段:**Phase1**(1)**准备**:一个P创建一个提案(编号N),向超过半数的A发送包含提案编号的Prepare(N)消息(2)**承诺**:每个A收到消息后,检査提案的编号N是否大于它曾接受过的所有提案的编号.若是,会回应以Promise(Nx,Vx)消息,承诺不会接受任何编号小于N的提案;否则不予回应.其中Nx和Vx是它曾接受过的提案中编号最大的提案的编号与值,若没有接受过提案,Nx和Vx为NULL.**Phase2**(1)**请求接受**:若P收到了超过半数A的Promise消息,它需要先找到这些消息中编号最大的提案的值Vn,然后向这些Acceptors发送Accept(N,Vn)消息;若所有Promise消息中Nx和Vx都为NULL,则Proposer可以选择任意的值作为V.(2)**接受**:当A收到Accept(N,V)消息,它首先检査是否已承诺过编号大于N的提案,若答案是否,它就接受该提案N,并发送Accepted(N,Vn);否则就拒绝.**执行阶段**当获得半数以上Accepted返回后,该提案被选定,并提交Learner执行.L可以通过三种方式获取被选定的值value:**方式一**:A每接受一个提案发送给所有L.L快速获取被选定的value,所需的通信次数等于二者数量的乘积.**方式二**:A每接受一个提案,发送给主L;提案值被最终选定,再由主L发送给其他L.主L可能会发生单点故障问题,降低了系统可靠性.**方式三:**A每接受一个提案发送给一个L集合 **L集合的数量越多,系统可靠性就越好,但通信复杂度也相应地越高**.**一轮Paxos算法的过程**•一轮只对一个值达成共识•**A本地记录**以下几个值•**minProposal** id最大prepare请求的提案id•**acceptedProposal**提案编号最大的提案id•**acceptValue**提案编号最大提案值.考虑单个p **Prepare阶段**:•**1.**P向A发送prepare请求.Prepare消息包含这次提案的id.•**2.**A收到后向p返回Promise消息.Promise消息包含 (acceptedProposal, acceptedVal-ue),若没有则留空.然后A更新minProposal(**Acceptor在对一个编号为n的提案做出响应后,不会再对编号小于n的prepare /accept请求做出响应, Accept阶段3.**P收到过半的A的promise消息后进入propose阶段,向A发送Accept请求.•请求同样要带有之前提到的提案id n•请求还要带有一个值,这个值为对要达成共识的值的一个提案:若acceptedValue空,则这个值为P自己选中的值;若有promise消息中带有aV,则从这些消息中找出acceptedProposal最大的消息对应的aV•**4.**A在收到accept请求之后:•检查accept请求的提案idn,若n小于minProposal则向Proposer返回minProposal值•否则,该A接受accept请求并作为acceptedProposal (minProposal)和aV也会向Propose返回自己的minProposal•P等待收到过半A的响应•**若响应中包含的minProposal值大于自己提案id,放弃本轮**•否则说明已经得到了过半Acceptor的支持,即达成一致•**5.**P将结果发送给l记下结果(也可以让A在每次接受一个值告诉L在收集到过半的id的值后确定结果**活锁**一个P两轮广播+收集就能直接完成.多个P先后发起请求,**后一个prepare请求就会使前一个prepare请求后续的accept请求失效**,除非这个accept请求在后一个prepare发出之前就收到了足够的响应达成共识.每当的请求确认失败后马上用新的请求id发起新一轮prepare,**二者的prepare不断使对方的请求失效**,一直持续从而形成活锁.可**在集群中选出一个节点作为leader,在对每条数据/每条事务达成一致性的时候,只有leader会作为P而其他节点不会发起提案**.对每条事务都运行一次上述的**一轮Paxos** Leader向定时发送一个心跳包来声明自己存活.leader消失,则可以发起竞选来尝试成为leader (一轮Paxos).**Raft算法**•适用于非拜占庭容错环境下的分布式一致性算法.模块化(领导选举、日志复制、安全性、成员变更等)状态空间规约**概念:复制状态机RSM(ReplicatedStateMachine)**指多台机器具有完全相同的状态,并且运行完全相同的确定性状态机,**容忍半数节点故障**.每个服务器节点都有:**•一致性模块•日志•状态机**• 基于日志的复制机制•日志记录了导致状态机中状态转换的命令序列.•状态机的状态可以通过执行日志中的命令序列获得•保证不同节点间的**日志一致**(保存相同顺序命令序列),即最终可以保证状态机之间的状态一致性.**思路和过程**Raft算法采用RSM ,所有的服务器都以同样的顺序响应客户端的请求.问题转换成**保证所有的状态机的日志一致性**的问题.其算法思路:•在集群中通过**领导选举**确定一个领导者,全权负责**复制日志的管理**.•领导者从客户端接收日志条目,将日志条目复制到其他服务器,并且在保证安全性的时候通知其他服务器将日志条目应用到它们的状态机中.•简化了日志复制的管理 •当领导者宕机:启动领导选举过程选出新的领导者.•Raft集群节点处于三种状态之一:**领导者Leader、跟随者Follower、领导候选Candidate**.•一个L,其他节点称为F.由L向各个F同步自己的log,log中每一个entry代表一个事务.**(日志复制)**•L**心跳包(领导选举)(成员变更)领导选举-任期Term**将时间分割成**任意长度**的任期,任期由**任期号**标识,**单调递增**.任期从一次领导选举开始.•一个节点在尝试竞选成为L, **指定任期**.•当任期n的L宕机,等待超时的F将自己的状态变为C,申请成为下一任期的L**领导选举-从每个节点的角度:**所有的服务器**初始**都处于F状态等待心跳包到超时并参与选举.•F未收到心跳包,开启一轮新的选举.并且:•将当前的**任期编号+1**•将自己的状态设为C;•投票给自己并向其余的服务器发送<RequestVote>消息.**领导选举有三种可能**:①赢得选举:•在一个给定任期内一个节点只能给一个C投票.•收到大多数节点的投票,该节点就成为了该任期的Leader,并发送<AppendEntries>心跳包给其余的节点服务器②若收到一条<AE>•若消息中的任期编号大于当前任期,意味着某个节点已当选为Leader,则将自己的状态设为F;否则,丢弃消息.③选举失败•若出现多个节点同时竞选,任一参选节点都无法获得大多数投票,选举失败.•每个C超时后重启新的选举,每个C**超时时间**是**随机值**.**日志格式** •每行代表一节点,每一个方格代表一个logentry.•每一行从左到右logindex递增,每个**logentry包含着自己所在的任期号**(由l生成日志项时生成)•若不同节点上的两条logentry有相同的任期号和index,则其内容一定是相同的.每个节点都会维护一个日志条目索引.**日志复制**•每当L收到了来自client发来的事务请求(1),就将其构造成一条新的logentry加入自己的log中.•Leader会将新entry用<AppendEntries>消息发送给各个F(2),发送失败不断重试.当收到大部分F的响应则确认该entry以及其之前的所有entry都可以commit.•leader在确认一条entry可以被commit之后,会通知各follower节点该entry可以被commit(4.2),并会向client发送响应通知(4.1),表示这条事务确实已经被接收.•Entry被commit,代表其中事务所对应的具体内容可以被执行 **日志复制-故障**•阶段1和阶段2.1之前出故障,不影响一致性•阶段2.1失败(部分或全部),有一致性问题•阶段3.1失败(部分或全部),有一致性问题•阶段4.1失败,客户端重新发送请求后提交即可,无一致性问题•某些情况会出现网络分区,而导致**双领导**,原领导降级为F远程过程调用RPC **Append-Entries-RPC**由Leader发出,用于复制日志.当增加日志条目数为0是用作心跳heartbeat.**RequestVote-RPC**:由Candidate节点发出,请求其他节点给自己投票.**集群成员变更** •信息未及时更新会出现双leader’**相同index,相同任期而内容不同’的一对logentry**出现从而导致系统出现错误.Raft算法使用**两个事务**来完成一次成员变更•当Leader收到成员变更的请求(从C-old到C-new),L会先构造一个介于二者之间的配置C-old,new,这个配置包含的成员是新旧两个配置的内容.Leader会先将**C-old,new**作为一个新的entry按照之前的流程同步到各个Follower上.在确认这个中间配置commit了之后,各个节点就按照这个中间配置来行事,即节点需要同时获得新配置和旧配置中大多数人的同意才能成为l.leader出现了之后会再将C-new作为新entry同步给其他节点.当C-new配置commit了之后,则各个节点开始按照新配置来运行.完成成员变更.**安全性选举安全性**一个leader(term).**领导Append-Only**•日志项不能被覆盖或删除,只能扩展.**日志一致性**•不同节点上有相同的任期号,index,则其内容一定是相同的.•该index之前的所有日志都相同.**领导者完整性**•对于logentry被l commit,任意term l 都有logentry.**状态机安全性**•若一个服务器已经在一个状态机上应用了一条logentry,那么所有的服务器都将应用*同一个logindex的同一entry*.主流BC共识算法**PBFT共识算法**•可以应用于**异步网络**,容忍三分之一数量的拜占庭节点,大幅提高了系统的响应效率,具有较强的实用性.•假设共识过程运行环境是一个**异步分布式网络**.网络中可能发生消息传输失败、延迟、重新发送等问题,且节点有作恶的可能.•PBFT算法使用**数字签名**来防止欺骗、重发和检测消息完整性.消息包括公钥签名、消息类型编码、哈希产生的消息摘要等.后续的算法演示设**<m>σi表示消息m被节点i签名**,**D(m)**表示消息m的摘要.**PBFT容错率**:设节点总数是n,其中作恶节点有f个,那么剩下的正确节点为n–f个,这意味着只要收到n-f个消息就能做出决定.所以**n最少是3f+1个**.采用PBFT算法的系统**并非节点数越多越好**,要根据f来计算.•PBFT中的服务被建模为**状态机**,每个节点上都保存有状态副本包括state,log,view(当前视图编号).所有副本的状态变迁通过视图(view)的配置更换来进行.在每一个视图中,只存在一个主节点.主节点可以简单地由**视图编号mod|节点数量|**来决定.**两个限定条件**:①节点是确定性的,给定状态和参数相同的情况下,执行结果相同;②所有节点必须从相同状态开始执行.即使存在失效副本节点,算法对所有非失效副本节点的请求执行总顺序达成一致.•算法的五个阶段:①请求Request②预准备Pre-prepare③准备Prepare④确认Commit⑤回复Reply(12确保同一视图中请求发送的时序性,345确保不同视图之间的确认请求是严格排序的)如图,n=4,f=1.①客户端C发送请求给主节点0,②主节点0把请求广播给其他副节点1、2、3.③由于节点3失效(故障或者作恶,图中红线仅代表诚实节点的信道),节点1、2广播消息.④节点0、1、2搜集到2f(不包括自己)的消息之后,在本地达成了commit条件,然后将消息广播给全网.⑤节点0、1、2搜集到2f+1(包括自己)个commit的消息,执行请求操作,并给客户端回复消息.⑥客户端搜集到3个消息,大于f,故请求成功•**开始阶段**,主节点由p=vmodn计算得出,随着v(视图编号)的增长可以看到p不断变化.图中,节点0为第一轮视图的主节点.**第一阶段(请求)**:客户端签名发送消息<REQUEST,o,t,c>σc给主节点0,o为操作;t为时间戳,用于保证c请求只被执行一次,也可以用于比较操作执行顺序,例如t可以设置为客户端发送请求时的本地时钟;c为客户端编号**第二阶段(预准备)**:主节点构造消息<<PRE-PREPARE,v,n,d>σ0,m>广播到集群中的其它节点,同时消息追加到消息日志中.•PRE-PREPARE标识当前消息所处的协议阶段.•v标识当前视图编号,n为主节点分配给所广播消息的一个唯一递增序号,m为客户端发来的消息,d为m的数字摘要.**第三阶段(准备)**:副本收到主节点请求后,会对消*息有效性*进行检查,检查通过会追加在消息日志中,并广播消息<PREPARE,v,n,d,i>σi,其中i是本节点的编号.对消息的有效性有如下检查:1.摘要d是否和自己生成的摘要一致,完整性.2.v一致.3.之前是否接收过相同序号n和v,但不同摘要d.4.序号n是否在水线h和H之间(防止作恶节点消耗序号空间)**第四阶段(确认)**:副本收到2f个一致的PREPARE消息后,会进入COMMIT阶段,并且广播消息<COMMIT,v,n,D(m),i>σi给集群中的其它节点.在收到PREPARE消息后,副本同样也会对消息进行i有效性检查,包含上一阶段介绍的124三个检查步骤.**第五阶段(回复)**:副本收到2f+1(包括自己)个一致的COMMIT消息后且已经没有序号小于n的请求,则执行m中包含的操作(保证多个m按照序号n从小到大执行),执行完毕后发送消息<REPLY,v,t,c,i,r>σi给客户i端,t时间戳,c客户端编号,r操作结果.•客户端在收到回复后要进行签名验证、时间戳比较和操作结果r比较.当搜集到f+1个一致结果的回复之后才能确定执行结果.若一定时间没有响应,客户端会广播请求,所有副本会进行响应.若响应结果发现是主节点失效,则会通过 (ViewChange)来切换主节点.**检查点消息(Checkpoint)** •当前节点处理的最新请求序号.前面已经提到主节点收到请求会给请求消息编号.•**stablecheckpoint(稳定检查点)**:**作用:最大的目的是减少内存的占用**.因为每个节点的消息日志记录下之前共识过什么请求,随着系统运行,日志数据会越来越大,所以应该有一个机制来实现对数据的删除.**ViewChange(视图更改)事件**当主节点挂了(超时无响应)或者副节点集体认为主节点是问题节点时,就会触发ViewChange,完成后,视图编号将会+1.**流程**:从节点向其他节点广播view-change事件<view-change,v+1,n,C,P,i>,判断是否收到2f个不同副本的视图编号为v+1的view-change消息(不包括自己),否结束,是则向其他节点广播new-view消息<new-view,v+1,V,O>,节点验证收到的视图编号为v+1的new-View消息里的O集合是否正确,否结束,是则按照pbft流程逐条执行O集合里的请求消息,视图编号为v+1(每个请求执行后不会重新返回消息给客户端),结束(v:上一个视图编号,n:i节点的stablecheckpoint编号,C:2f+1个节点的有效checkpoint信息的集合,P:i节点中上一个view中编号大于n且到达prepared状态的请求消息的集合,V:新的主节点接收到的有效视图编号为v+1的view-change消息集合,O:pre-prepare消息集合) **Raft与PBFT对比**1Raft 2Pbft:**适用环境:**私有链,联盟链 **通信复杂度**O(n) O(n^2) **最大故障和容错节点**故障节点:2f+1≤N 容错节点: :3f+1≤N **流程对比:Raft:**初始化leader选举,谁快谁当;公识过程;重选leader机制 **Pbft**:初始化leader选举(按编号轮流做主节点;共识过程;重选leader机制) PBFT是在**联盟链共识节点较少**的情况下BFT的一种解决方案.**总结**•PBFT算法由于每个副本节点都需要和其他节点进行P2P的共识同步,因此随着节点的增多,性能下降很快,在较少节点的情况下可以有不错的性能,并且分叉的几率很低.PBFT主要用于*联盟链*,但是若能够结合类似DPoS这样的节点代表选举规则的话也可以应用于公有链,并且可以在一个不可信的网络里解决拜占庭容错问题,TPS(服务器每秒处理的事务数)>>POW**PoW共识算法**工作量证明最早是为**防止服务和资源滥用**,或者拒绝服务攻击等场景而提出的一种经济对策.一般要求证明方在使用服务或资源之前,首先完成具有一定难度或者适当工作量的复杂运算;BC系统中的稀缺资源是’**区块记账权**’以及随区块发行的**btb奖励**.PoW共识机制通过引入分布式节点的算力竞争来作为工作量证明,利用其算力来完成大量的哈希函数计算工作,以便选出每个10分钟时间窗口的唯一’记账人’,从而保证BC账本数据的一致性和共识的安全性;工作量证明的核心技术是**哈希**,btb中选择使用SHA256哈希算法,矿工需要找到区块头中的一个随机数Nonce,使区块头的两次哈希结果满足以n个0开头,穷举16^n次哈希才能找到结果.**btb中的PoW**:**三大要素**:PoW函数、区块信息、难度值;PoW函数定义为:Fdiff(𝐵𝑙𝑜𝑐𝑘𝐻𝑒𝑎𝑑𝑒𝑟) →SHA256 (𝑆𝐻𝐴256 (𝐵𝑙𝑜𝑐𝑘𝐻𝑒𝑎𝑑𝑒𝑟)) <MaxTarget/𝑑𝑖𝑓𝑓.其中MaxTarget为btb系统的最大目标值,难度值diff为正实数,diff越大,不等号右边值越小,0越多,计算难度越大;**diff会被系统动态调整 btb中的PoW共识过程**:1.每个节点搜集当前时间段全网未确认交易,并增加一个用于发行新btb奖励的CoinBase交易,形成当前区块体的交易集合;2.计算区块体交易集合的Merkle根计入区块头,并填写区块头的其他元数据,Nonce=0.作为PoW函数的输入数据;3.Nonce+1;计算当前区块头的两次SHA256哈希值,若小于目标哈希值,则成功并获得该区块的记账权3直到任一节点搜索到随机数为止;4.一定时间内未成功,更新时间戳和未确认交易集合,重新算Merkle根后搜索.**当节点拥有全网n%的算力时,该节点总是有n%的概率首先找到合理的随机数**.保证安全性和不可篡改性.**PoW的优势**1.架构简明扼要、有效可靠;2.实现某种意义上的公平性,投入算力越多等比例地增加越多的获胜概率;3有效抵御51%攻击,攻击者必须拥有超过整个系统51%的算力,可能篡改btb账本.**PoW的缺陷**1.强大的算力造成了极大的资源浪费;2.10分钟的交易确认时间不适合小额交易的商业应用;3.矿机和矿池的出现有违去中心化的初衷**PoS共识算法**PoW耗能巨大,算力中心化,整个系统的安全性逐渐取决于矿工和矿池.因此诞生了PoS(权益证明)共识算法:hash(block\_header) <= target\*coinage(币龄 拥有的币\*剩下的使用时间,抵抗51%).PoS共识中,**具有最高权益的节点**最有可能获得记账权,其权益体现为节点对特定数量货币的所有权.**PoS共识算法**:1.PoS+PoW混合共识(PoS1.0)2.纯PoS共识(PoS2.0)3.PoS共识扩展形式(PoS3.0)**PoS+PoW混合共识(PeerCoin)**:PoW共识用于早期货币发行,随着挖矿难度上升,过渡到PoS共识;*两类区块*:PoW区块和PoS区块;**CoinStake(币权)交易**:为PoS特殊设计的交易:交易输入数量≥1,第一个输入不为空,称**Kernel**;交易输出数量≥2,第一个输出置空;标识是第二笔交易为CoinStake(所有区块第一笔交易都是coinbase);PoS的挖矿,就是找到合适的Kernel交易;**激励机制**:Coinstake全部输入的币龄总和按一定比例转化为矿工收益;**共识流程**:节点首先从自己所有的UTXO中选定一个作为Kernel,构造Coinstake交易,计算两次SHA256;不满足上式则重新构造Coinstake交易,重构时间戳nTime会改变,也可以改变Kernel,得到不同的Coinstake交易.如此重复下去.时间戳以秒为单位,共识的搜索空间大大减少,同时**Kernel的币龄**是影响找到合格区块的最大因素;区块生成后,每个区块中的交易都会将其消耗的币龄提交给区块,获得最高消耗币龄的区块将会被选中成为主链;**权益激励**:stakeReward=(0.01×nCoinAge/365)×COIN其中nCoinAge是Coinstake所有输入的币龄总和,这部分收益奖励给矿工;**纯PoS共识**(Nextcoin未来币):第一个纯粹应用PoS机制运行的数字货币,基于**账户结构**而非UTXO,NXT中不存在挖矿,10亿NXT一次性创建在第一个区块,网络靠交易费用维护;透明锻造机制,指定锻造下个区块的节点,其余节点将交易发到此节点上缩短交易时间;**PoS函数**:hit<𝑏𝑎𝑠𝑒𝑇𝑎𝑟𝑔𝑒𝑡×𝑒𝑓𝑓𝑒𝑐𝑡𝑖𝑣𝑒𝐵𝑎𝑙𝑎𝑛𝑐𝑒 ×𝑒𝑙𝑎𝑝𝑠𝑒𝑇𝑖𝑚𝑒.hit:每个区块有一个生成签名字段,用户使用自己的私钥对上一区块的生成签名进行签名,结果进行SHA256哈希,哈希结果的前8字节作为hit变量;baseTarget:全网难度基准值;effectiveBalance:账户有效余额,账户中具有铸币权利的货币余额;elapseTime:当前时间与上一区块的时间间隔;用户不需要挖矿,只需等待时间推移至不等式成立即可锻造区块;用户有效余额越大,更有可能获得锻造区块的机会;优先选择最早生成的区块;**区块锻造流程**:用户必须实时在线,当网络上有新区块产生时,每个账户立即计算自己的hit,然后根据公式elapseTime=hit/ (baseTarget× effectiveBalance)计算得知自己锻造区块的期望时间值,并将这个时间广播给其他节点.当全网每个节点都知道其他节点的期望时间时,也就知道下个区块优先由哪个节点锻造,并将交易发给该节点.该节点在自己的时间窗口锻造好区块后立即广播全网,其他节点检验新区块是否有效.**优点**:1.有效降低能源消耗;2.避免算力集中问题;**缺陷**:1.存在’富者更富’的马太效应问题;2.存在无利害关系等安全性问题;无利害关系攻击,即由于PoS中挖矿代价较低,当链出现分叉时,对于一个节点来说,所有链上挖矿获得最大利益,会导致链分叉越来越多**DPoS共识算法**(委托权益证明)通过共识节点的权益投票将区块数据的记账权和BC参数配置权赋予特定的少数代表节点,实现公平和民主的共识过程和BC治理,解决PoW的能源消耗和PoS的无利害关系攻击.阶段⚫**见证人选举**:DPoS共识过程中,股东节点可以将其持有的股份权益作为选票授予一个代表(**见证人)**.股东节点的投票权重与其持币数量成正比.DPoS选择得票数最多且有意愿的N个节点进入董事会,轮流对交易进行打包和生成新区块;见证人会获得区块奖励、交易费或系统发行的特定奖励,需要缴纳保证金.见证人对股东负责,若其错过签署对应的区块,股东会撤回选票将其投出董事会,因此,见证人保证99%+在线时间以盈利;通过选举方式产生一组**授权代表**(Delegates),有权配置和调整BC系统的参数.大多数授权代表同意变更提案,所有股东节点有一段时间来审查变更提案,可以罢免授权代表并废止提案⚫**生产区块**:DPoS共识过程中,见证人节点按照预先定义的顺序轮流产生区块.每个区块被签署生产之前,需要验证前一个区块已经被受信任的见证人签署.每个见证人轮流在一个固定时间内生产一个区块,若见证人没有在其时间段中生产区块,那么该时间段后这个见证人将会被跳过,由下一个见证人生产下一个区块,如此循环.**RPCA共识算法**Ripple协议共识算法是Ripple系统及其瑞波币(XRP)所采用的的共识算法.利用网络中的集体可信的子网络来克服所有节点同步通信的延迟问题,具有很低的延迟.⚫**RPCA基本概念**1.服务器:是指运行Ripple服务器软件并参与共识过程的验证节点,是被其他服务器加入到信任列表中的节点.非验证节点不参与;2.最新关闭区块(账本):最近被共识过的区块;3.开放区块:正在被共识的区块,开放区块通过共识后,就成了2;4.可信任节点列表(UNL):每个服务器会维护一个UNL.共识过程中只接受UNL节点投票 ⚫**收集交易,形成交易集**1.收集交易’候选集’:服务器节点从网络中收集客户端发起的新交易以及之前共识过程的旧交易,检查后暂时存放在’交易候选集’中;2.做并集:每个服务器节点对它UNL交易候选集做并集,检查所有交易的正确性,打包成提案并广播到网络;3.投票:从网络中接收到新提案,若发送提案的服务器在UNL中,对既在该提案又在自身提案中的交易投票;验证服务器不断处理和发送提案,达到一定投票比例的交易进入下一轮,在最终轮中,超该服务器UNL中80%节点投票的交易被放入交易集;⚫**区块打包**1.计算哈希:形成交易集后,服务器对这些交易验证,并打包新的区块,对当前区块号,交易集的Merkleroothash、父区块哈希等内容计算哈希2.广播:验证服务器在网络中传播区块哈希值;3.阈值:验证服务器收到它UNL广播过来的区块哈希后,80%节点对同一区块哈希时,认为是有效,区块被关闭;若某服务器的区块哈希与通过共识的哈希不同,需要去其他节点拉取正确区块;若没有区块哈希超过设定阈值,则重新开始共识过程

***btb系统***•btb是基于密码学和经济博弈的一种数字加密货币,历史上第一个大规模的虚拟货币系统.•btb既是构成数字货币生态系统基础概念和技术的总称,又是btb网络中参与者存储和传输的货币单位.**btb四个创新•**去中心化的点对点网络(btb协议)•一本公开的交易总账(BC)•一套可独立校验交易和发行货币的规则(共识规则)•一种通过全球去中心化对有效的BC达成共识的机制(POW) **btb网络**•每个节点都是对等的,在P2P网络中不存在中心化服务,也不存在层级结构.btb去中心化.•扮演不同角色.一个btb节点是功能集合:路由、BC数据库、挖矿和钱包服务1 .**btb中继网络**:由于通过POW创建新块,这个竞争对时间是非常敏感的,btb提出了中继网络,*最小化矿工之间网络时延*,以解决网络时延带来的问题.和矿⼯和矿池直通相当于⾼速主⼲⽹,减少跳数和延迟(FIBRE,快速互联网btb传播引擎)**连接的建立**:当一个新节点启动后,需要发现网络中的其他节点.为连接到一个已知的节点,节点间需要建立TCP(8333),在建立连接时节点发送包含基本版本信息的’握手消息’version:{nVersion- P2P版本协议nLocalServices-节点支持的本地服务列表nTime-当前时间addrYou-远程节点IP addrMe-本地IP subver:-子版本信息,用于展示本节点运行的软件类型BestHeight-子节点区块高度} 双方节点通过version对应发送确认信息verack以表示节点兼容,确认过version后连接正式建立 **节点的发现**:新节点通过’DNS种子’方式来获取已有的btb节点的IP地址列表.’DNS’种子:提供btb节点IP地址列表的DNS服务器.分类1.提供稳定的btb监听节点IP地址的静态列表;2.提供随机的btb节点IP地址,通过爬虫或者运行btb客户端来得到.网络中的节点可能随时加入或者离开,新的节点在接入到网络后不断发现新节点被新的节点发现.(Addr+Getaddr到连接节点).**节点状态的检测**:getpeerinfo指令来列出全部对等节点的连接信息 若一个连接上没有通信,节点会定期发送信息维持连接(>90min断开).**全节点和轻节点**:btb能具有去中心化的特性得益于它的数据是完整备份在P2P网络中的对等节点上.**全节点**:维护了一份完整的、最新的包含全部交易的btbBC副本.这份副本由节点独立构建和检验,从最初的第一个区块(创世区块)开始一直构建到网络中的最新已知区块.可以独立可信地验证任何交易,而无须求助于或者依赖于其他节点或是信息源.依赖网络来接收关于新交易区块的更新,在验证之后合并到它本地.一个全节点连接到网络之后,需要在本地建立一个完整的BC副本:新节点只有创世区块,需要在网络中同步所有的区块;重新上线的全节点,检查当前最新块高度,同步缺失的区块 **轻节点**(SPV):通过SPV在不存储完整的BC数据的情况下去验证交易.无法构建所有的可用UTXO全貌,*依赖于对等节点提供的BC数据局部视图进行验证*.**SPV节点对交易的验证**向网络中的对等节点请求数据.用一个merkle路径在交易和包含它的区块之间建立联系,并等待接收到交易所在区块和后续验证区块之后再进行验证.(merkle路径的证明+工作量证明)来确认一笔交易在区块中是否存在.**btb钱包**只包成含私钥/公钥对的密钥链.用户用私钥签名交易,从而证明他们拥有这笔交易的输出(他们的钱币).钱币则以交易输出的形式存储在BC中.**分类***基于密钥技术*, 1.非确定性钱包2.确定性钱包 *基于运行平台*1.桌面钱包2.手机钱包3.网络钱包4.硬件钱包5.纸钱包 **非确定性钱包**(随机钱包)钱包只是随机生成私钥的集合.每个密钥都是从随机数独立生成的,密钥彼此无关.缺点是钱包丢失,资金丢失,需要不断备份.**确定性钱包** 其中所有的密钥都是从一个主密钥派生出来(种子(seed)).所有密钥都相互关联,若有原始种子,则可以再次生成全部密钥.最常用的派生方法:使用树状结构,称为分层确定性钱包或HD钱包(推荐,只需要记seed).除简单测试,不鼓励非确定性钱包.**公私钥和地址生成**:**1.**生成私钥•钱包地址可以通过私钥运算出来.随机选择,要求•32个字节•正数并且小于(2^256)(签名算法为**ECDSA**,限制私钥范围).**2.**生成公钥•私钥传入ECDSA,得到的结果是一个int-64,(x:int32,y:int32拼接)•在结果的开头加上0x04获得btb的完整公钥•压缩版公钥.Y值的最后一个字节是偶数,开头添加0x02.否则0x03 **3**.生成地址•btb的地址是通过公钥进行了一系列的转换而获得的,多重哈希运算.•对公钥加密.X1=RIPEMD-160(SHA-256(PKey)) •对加密版公钥添加网络标识字节•添加校验值,判定地址是否填错或损坏.校验值=(SHA-256(SHA-256(X2)))前4byte.添加到第二步结果末尾即钱包地址.•’Base58’编码:btb钱包地址.•**btb地址**是一个由数字和字母组成的字符串,’1’开头.•btb地址以’收款方’出现 **交易流程**:钱包软件搜集UTXO→提供正确的解锁脚本→构造支付给接收者的输出(交易的输出会被创建成一个包含这笔数额的脚本形式,只能被引入这个脚本的一个解答后才能兑换)→构造交易完毕,发送→节点收到交易,独立验证,通过则转发(自循环)→被某个挖矿节点验证并加入到区块中 **发起交易**•A从B购买btb, B按发送键转账.B的移动btb钱包构建了一个交易,从B BTC发送给A地址,用B的**私钥**对该交易签名.通过点对点网络传输时,它会在btb网络快速传播.网络中节点接收到这笔交易,首次看到A的地址.• A的钱包不断地’listen’在btb网络上发布的交易,寻找与她的钱包中的地址相匹配的任何内容.A的钱包将显示它正在接收BTC.•起初,A的钱包交易显示为’未确认’(尚未记录在BC).**’确认’就是一个交易必须包含在一个区块中,并被添加到BC,每10分钟一次**.(防止分叉多次’确认’).**btb共识-交易验证**共识是用来实现BC在P2P网络上所有节点数据一致性的关键机制,( BFT,CFT,PoX).**完整的共识过程**:出块节点对收到的交易的验证;矿工计算PoW;网点对区块的验证 **交易的验证** btb交易的基本单位是交易输出,而所有可用和可支出的输出就是UTXO.UTXO不可分割,只能通过花费来生成新的UTXO.余额:所有UTXO值,来计算余额.输入大于输出,差值的部分就是**交易费**(不显示在交易中,它会在打包时包含在Coinbase中)交易费会补偿btb矿工,安全机制,使攻击者在经济上无法通过创造大量垃圾交易来瘫痪网络.交易费不是强制的.交易输入通过引用来标识使用哪个UTXO,并通过解锁脚本提供所有权证明.输入1.引用交易哈希和索引(txid,vout)来指向UTXO的指针.2.**解锁脚本**,钱包会构造该解锁脚本,以满足UTXO中设置的支出条件.**交易输出包含UTXO的值和锁定脚本**.**btb脚本**数字签名的验证过程,通过锁定和解锁脚本来实现.交易在生成输出,会在每个UTXO中构造锁定脚本scriptPubKey,来指定将来要花费它所需要满足的条件,每个交易的输入都包含一个解锁脚本scriptSig,它通常包含由用户私钥产生的签名以及公钥,用于满足锁定脚本的条件.若**解锁脚本中的公钥哈希后与锁定脚本中相同,且私钥产生的签名能被锁定****脚本中的公钥解锁**,说明这个签名是这个UTXO的所有者签署的,合法.每个btb验证节点将通过同时执行锁定和解锁脚本来验证交易.每个输入都包含一个解锁脚本,并引用以前存在的UTXO.验证软件将复制解锁脚本,检索输入引用的UTXO,然后从该UTXO复制锁定脚本,依次执行解锁和锁定脚本.若解锁脚本满足锁定脚本条件,则输入有效.解锁+锁定脚本:完整验证过程,脚本各自单独执行,并在两次执行之间传递堆栈.脚本语言是基于栈的语言 **解锁脚本中的签名主要有三个作用**:身份认证,交易认证,内容校验.解锁脚本签名,当前交易哈希的签名.交易在经过节点独立验证后但尚未确认前,会被放入节点的交易池,节点不断监听BC网络,在新的区块产生之后,节点会将新块中包含的交易从交易池中剔除.**交易的验证**独立验证的检查:•交易的语法和数据结构.•输入与输出都不为空• 交易的字节大小范围•输出值,总量范围•散列=0/N=-1输入非法(coinbase)• nLockTime范围•交易的字节大小>= 100• 交易中的签名数量范围•(…数据验证清单.6验证交易有效性)• 每一个输入的解锁脚本必须依据相应输出的锁定脚本来验证**btb共识-PoW**当交易池中的交易达到一定数量后,挖矿节点会将它们打包到一个候选块中,挖矿节点创造一笔特殊的交易Coinbase.Coinbase交易不消耗UTXO支付给矿工自己的btb地址.交易集合生成后,挖矿节点需要构造区块,填充区块头**.**区块的验证挖矿成功,挖矿节点将块发给所有对等节点,对新块进行独立验证并传播,一个节点验证了一个新的区块,尝试将新的区块连接到现存的BC(保证矿工不会作弊).内容•区块的数据结构语法有效•区块头散列值小于目标难度(POW)•区块时间戳早于验证时刻未来两个小时(允许时间误差)•区块大小在限制范围之内•第一个交易(且只有第一个)是coinbase交易•所有交易再次经过之前所示的独立验证有效.一定的**确认数(防止分叉,6)**;**btb激励**•在矿工将交易打包进区块时,其中的第一笔交易是一笔叫做coinbase的特殊交易.与其他常规交易不同,这笔交易不需要UTXO作为输入,其输入中只包含一个叫coinbase的输入;这笔交易只有一个输出,输出支付到矿工自己的btb地址上.**输出**•**交易手续费**•**区块奖励**(+=矿工拿到的奖励)区块奖励值是一个基于区块高度计算出来的值.btb特性-分叉为BTC和BCH **51攻击(**最⻓链拥有记帐权)利用btb网络采用PoW竞争记账权和’最长链共识’的特点 掌握了50%以上的算力,就拥有了获得记账权的绝对优势,可以更快地生成区块,也拥有了篡改BC数据的权利.**双花问题**.假设A拥有51%的算力,在区块高度1127时,A转给B一个btb的记录被矿工打包.待交易确认后,A依靠51%的算力优势在区块高度1126后重新生成了一条’更长的链’,并在区块高度1127处又将该BTC转给C且该交易记录被打包,即该链包含了A将一个btb转给C的记录.根据’最长链共识’,包含给C转账记录的链成为主链,则A转给B的一个btb则为’无效支付’.若掌握了51%的算力,除了可以修改自己的交易记录外,还可以阻止区块确认部分交易,以及阻止部分矿工获得有效的记账权.无法修改其他人的交易记录,也不能阻止交易的发出,更不能凭空产生BTC.**面临的挑战算力中心化**•矿工不得不矿池,按算力比例分配收益,以此来保障挖矿的稳定性.•然而随着越来越多的矿工加入到各个矿池中,btb系统的算力变得越来越中心化.**低吞吐**计算得到一秒钟最多处理7交易,显然这个速度是无法满足正常的交易需求的**资源浪费 监管风险**btb具有匿名性、难追溯、跨境流通便利的特点,使其极易被违法分子用来进行诈骗、赌博、洗钱,成为犯罪的新媒介**BitcoinScript**•btb在交易中使用脚本(Script)系统,简单的、基于堆栈的、从左向右处理,非图灵完备,没有LOOP语句.一个脚本本质上是众多指令的列表,这些指令记录在每个交易中,交易的接收者想花掉发送给他的btb,这些指令就是描述接收者是如何获得这些btb的.•逆波兰表达式语言.•无状态.执行脚本之前没有状态,执行脚本之后也不保存状态.执⾏前后堆栈都是空的.•绝大多数btb交易都是基于P2PKH(pay-to-Public-Key-Hash)脚本的.***ytf与智能合约* 智能合约**•**狭义定义**:运行在分布式账本上预置规则、具有状态、条件响应的,可封装、验证、执行分布式节点复杂行为,完成信息交换、价值转移和资产管理的计算机程序.•**广义定义**:无需中介、自我验证、自动执行合约条款的计算机交易协议.按照其设计目的可分为:旨在作为法律的替代和补充的智能法律合约,旨在作为功能型软件的智能软件合约,以及旨在引入新型合约关系的智能替代合约.**特性**:自动执行、安全透明、自治自足 **ytf概述**解决btb扩展性不足的问题 **定义**(Ethereum)一个开源的有智能合约功能的公共BC平台,通过其专用加密货币以太币(Ether)提供去中心化的以太虚拟机(EVM)来处理点对点合约.(以太币①支付算里,价值因素②任何操作如转帐等需要⽀付的代价)**特点**:具有一个全球范围可访问的单体状态;还有一个执行状态更改的虚拟机.从更加实际的角度来看,ytf是一个开源的、全球去中心化的计算基础架构,可以执行称为智能合约的程序.它使用**BC**同步和保存系统状态,借助**以太币**这种数字货币来计量并控制程序执行的资源开销.**ytf基本原理模型**ytf的本质就是一个基于交易的状态机:起始于一个创世(Genesis)状态,然后随着交易的执行,状态逐步改变一直到最终状态,这个最终状态就是ytf世界的权威版本**.基本组件**•**点对点(P2P)网络**:EthereumMainNetwork •**共识规则**: ytf黄皮书中定义•**交易**:一个网络消息,主要包含交易的发送方、接收方、价值和数据载荷.•**状态机**:ytf的状态转换由EVM处理,基于栈的虚拟机,执行bytecode.被称为’智能合约’的EVM程序采用高级语言编写,并编译为bytecode.•**数据结构**:ytf的BC以数据库的方式保存在每一个节点之上,交易和系统的状态经过哈希处理的数据保存在MerklePatriciaTree.•**共识算法**:ytf使用btb的共识模型Nakamoto Consensus,使用顺序单一签名块,由PoW加权重要性来确定最长链,从而确定当前状态.**ytf账户**取代btb的UTXO模型.账户以地址为索引,地址由公钥衍生而来.**外部账户**(简称’账户’)和**合约账户**.•Nonce:外部账户:从此账户地址发送的交易序号.合约账户:此账户创建的合约序号•Balance:余额,账户拥有的以太币数量•CodeHash:代码哈希,与账户关联的EVM代码的哈希值,创建后不可更改.状态数据库中包含•StorageRoot:存储根节点,账户内容的MerklePatricia树根节点的哈希编码.**外部帐户**由用户实际控制,一对公私钥.可以通过使用其私钥创建和签署交易,将消息发送到其他外部帐户或合约帐户.两个外部账户之间的消息只是一个价值转移.**合约账户**包含合约代码,由合约代码控制.合约账户不能自行发起新的交易,合约帐户只能触发交易以响应其他来自外部帐户或合约帐户的交易.**外部账户可对合约或外部账户发起交易,合约账户间可能相互触发** 两种账户对比**账户私钥**1.一般在ytf的钱包客户端创建通过私钥算法创建生成,私钥由账户所有者自己保管存储2.没有对应的账户私钥 **账户地址**1.账户地址基于账户的公钥采用地址生成算法推导得出2.账户地址采用智能合约发布者的账户相关信息推导得出 **账户链上生成**1.通过账户私钥签名一笔交易发往ytf的节点创建生成2.通过外部账户向ytf发布智能合约时创建生成 **账户存储1.**仅仅存储一个账户对应的交易序号和账户的有效余额2.除了外部账户的存储信息外,还存储该账户对应的智能合约数据.这些智能合约的数据按照特定的编码方式进行组织和存储**账户代码**1.无对应的账户代码2.保存了智能合约的账户代码内容,该内容即为编写的智能合约代码经过编译器编译后字节码信息**UTXO与账户模型的对比UTXO模型好处**:•*可扩展性*-由于可以同时处理多个UTXO,因此可以实现并行交易并鼓励可扩展性创新.•*隐私性*-即使btb不是一个完全匿名的系统,假设用户每笔交易都使用新地址,那么也能够提供更高级别的隐私.**账户/余额模型的好处**:•*简单性*-ytf选择了更直观的模型,以使开发人员在开发那些需要状态信息或涉及多方的智能合约时更简单.•*效率*-除简单性外,帐户/余额模型更有效,因为交易时只需要验证发送帐户是否有足够的余额.•账户/余额模型的缺点是无法避免**双重支付攻击**,但通过递增的随机数可以消除这种类型的攻击,防止同一个交易被多次提交.**ytfBC**ytf区块主要由区块头、交易列表和叔区块头.•**区块头**包含下列信息:父块的散列值(PrevHash)、叔区块的散列值(UnclesHash)、状态树根散列值(stateRoot)、交易树根散列值(TransactionRoot)、收据树根散列值(ReceiptRoot)、时间戳(Timestamp)、随机数(Nonce)等.保存**三棵Merkle树根**,分别是状态树、交易树和收据树.存储三棵树可方便账户做更多査询.•**交易列表**是由矿工从交易池中选择打包进区块中的一系列交易.**ytf区块结构Header成员**•ParentHash:指向父区块的哈希指针.除创世块有且只有一个父.•Coinbase:挖掘出这个区块的矿工地址.在每次执行交易时系统会给与一定补偿的Ether发给这个地.•UncleHash:Block结构体的成员uncles的RLP哈希值.uncles是一个Header数组.•Root:StateDB中的’stateTrie’的根节点的RLP哈希值..•TxHash:Block中’txTrie’的根节点的RLP哈希值’.•ReceiptHash:Block中的"ReceiptTrie’的根节点的RLP哈希值".•Bloom:**Bloom过滤器(Filter)**,用来快速判断一个参数Log对象是否存在于一组已知的Log集合中.•Difficulty:区块的难度.•Number:区块的序号.Block的Number等于其父区块Number+1.•Time:区块’应该’被创建的时间.•GasLimit:区块内所有Gas消耗的上限, ,与父区块有关2/3 •GasUsed:区块内所有Transaction执行时所实际消耗的Gas总和.•Nonce:一个64bit的哈希数,它被应用在区块的"挖掘"阶段,并且在使用中会被修改.**ytf交易与消息**•ytf的’交易’是指一条外部账户发送到BC上另一账户的消息的签名数据包,(发送者的签名、接收者的地址以及发送者转移给接收者的以太币数量)通过ytf网络传播,最终由矿工记录在bc,**唯一可触发bc状态改变或EVM合约执行的是交易**.交易结构:Recipient:接收者的地址或者合约的地址.• Nonce:用来防止重放攻击• GasPrice:计算交易费用•GasLimit:最大gas数量•Value:发送给接收地址的以太币数量•Data•Signature:原始EOA的ECDSA数字签名的三个组成部分(v,r,s).消息:类似于交易,但是由合约账户产生,用于合约账户内部通信,包括发送者,接收者,以太币,可选数据,StartGas,不会被记录到BC.**交易的类型**•**转账交易**:是最简单的一种交易,从一个账户向另一个账户发送以太币.发送转账交易只需要指定交易的发送者、接收者、转币的数量.•**创建合约的交易**:将合约部署到BC上的交易,发送者是合约的创建者,接受者为空,交易数据字段中指定合约的二进制代码.•**执行合约的交易**:调用合约中的方法,需要将交易的接受者指定为要调用的合约的地址,通过交易数据字段指定要调用的方法以及向该方法传递的参数.一段智能合约是被唯一的(合约)地址所标识,该地址有自己的资金余额(以太币),并且一旦有一笔交易发送至该(合约)地址,ytf网络节点就会执行合约逻辑.**ytf使用ytf虚拟机(EVM)来执行智能合约**.**ytf状态转换函数(交易的执行)**ytf的状态转换函数:APPLY(S,TX)->S',定义如下:•检查交易的格式是否正确、签名是否有效和随机数是否与发送者账户的随机数匹配.如否,返回错误.•计算交易费用:fee=GASLimit\*GASPRICE,并从签名中确定发送者的地址.从发送者的账户中减去交易费用和增加发送者的随机数.若账户余额不足,返回错误.•设定初值GAS=GASLimit,并根据交易中的字节数减去一定量的瓦斯值.•从发送者的账户转移价值到接收者账户.若接收账户还不存在,创建此账户.若接收账户是一个合约,运行合约的代码,直到代码运行结束或者瓦斯用完.•若因为发送者账户没有足够的钱或者代码执行耗尽瓦斯导致价值转移失败,恢复原来的状态,但是还需要支付交易费用,交易费用加至矿工账户.•否则,将所有剩余的瓦斯归还给发送者,消耗掉的瓦斯转换成交易费用发送给矿工.**转账或合约调用交易1.发起者发起转账交易或合约调用交易请求**From:发送者地址用户A To:接受账户地址或合约地址Value:转移的金额Gas:用来完成交易(执行合约)的Gas数量 GasPrice:交易中心愿意付出的Gas的单价Data:一段要发送给接受者(合约账户)的消息.经过client交易形式化后,增加一个Nonce:整数,用户发起的每笔交易都有唯一的编号,表示该交易是用户发起的第几笔交易**2.对等节点(可能有多个)检验、存储和转发交易**:检查交易,计算交易费用,从发送方的账户中减去相应费用,并根据交易中的字节数减去一定量的Gas值.将交易请求放在存储池中,并转发给其他节点**3.获得记账权的节点打包交易请求并执行合约代码(使用自己的EVM)** 用户B获得新区块记账权,用户A的交易区块请求和其他的交易一起被打包到区块中**4.生成区块的节点将区块发送到网络中5.交易被所有节点保存在本地的BC**接收到区块的节点运行智能合约,并对结果进行相互验证.该用户同步到用户B挖掘的区块,验证区块的合法性,若合法,将其加到本地的BC**创建合约交易**1.发送者发起创建智能合约的交易请求2.获得记账权的节点打包交易并部署合约3.获得记账权的节点发送新区块至ytf4.智能合约被部署在本地BC上,接上面的5**状态树、交易树和收据树**(组织成MPT树).ytf使用了MerklePatircia树(MPT)作为数据组织形式,用来组织管理用户的账户状态、交易信息等重要数据.**Trie树**(Radix树),通常用来存储(key,value)对.key代表的是从树根到对应value的一条真实路径.从根节点开始,key中的每个字符(从前到后)都代表着从根节点出发寻找相应value所要经过的子节点.value存储在叶节点中,是每条路径的终节点.•假如key中的每个字符都来自一个容量为N且所包含的字母都互不相同的字母表,那么树中的每个节点最多会有N个孩子,最大深度是key的最大长度.**Patricia树**是压缩的Trie树.•非根节点不是只能存储字符,而是可以存储字符串,也就是路径压缩了的Trie,节省内存开销.**MPT树**结合Patricia树和Merkle树的优点,在Patricia树中根节点是空的,而MPT树可以在根节点保存整棵树的哈希校验和,而校验和的生成则是采用了和⼀Merkle树一致的生成方式.ytf采用MPT树来保存交易、交易的收据以及世界状态.十六进制前缀(HP)编码,字母表只有16个字符.节点类型:•空节点(hashNode)•叶节点(valueNode)•分支节点(fullNode)•扩展节点(shortNode)**状态树**每个节点最多有16个孩子,每个叶节点表示一个账户,节点的父节点由叶节点的散列组成,直至根节点.•状态树包含一个键值映射,**键:账户地址,值:账户内容**{nonce,balance,code-Hash ,storageRoot}.•状态树代表发布区块后的整个状态.**交易树**•每个区块都有一棵独立的交易树.交易的顺序由’矿工’决定,根据交易的GasPrice和nonce排序.通过堆(heap)实现.每挖出一个新块,生成一个交易树.•交易树键值对,**键是交易的编号,值是交易内容**.**收据树**•区块有自己的收据树,收据树不需要更新,收据树代表每笔交易相应的收据.**键是索引编号**,相关交易的位置,**值是收据的内容**.•交易的收据是一个RLP编码的数据结构:[medstate,Gas\_used,logbloom,logs]区块头中存在一个布隆过滤器,减少查询的工作量,对轻客户端尽可能的友好.利用存储的三棵MPT树,ytf客户端可以轻松地**查询**:•某笔交易是否被包含在特定的区块中.•查询某个地址在过去的30天中发出某种类型事件的所有实例•目前某个账户的余额.•账户是否存在.•一笔交易,交易的输出是什么.**数据存储LevelDB**•高效的键值对数据库,其中键值都是二进制•ytf存在的LevelDB•**BlockDB**保存块的主体内容,包括块头和交易;•**StateDB**保存账户的状态数据;•**ExtrasDB**保存收据信息和其他辅助信息.•三个MPT树理解为LevelDB所存储数据的索引,便于ytf系统查询和处理这些数据.**ytf的共识机制PoW+PoS**•ytf**四个阶段**,即Frontier(前沿)、Homestead(家园)、Metropolis(大都会)、Serenity(宁静).123采用的是PoW共识机.4自己创建PoS机制(Casper投注共识)增加了惩罚机制.**PoW-Ethash算法**• ytf PoW算法•解决挖矿中心化问题.挖矿的效率基本与CPU无关,而与内存大小、带宽正相关,去除专用硬件的优势,抵抗ASIC.该*算法的基本流程*:①扫描区块头计算出一个种子,根据种子生成伪随机数据集(Cache).②基于Cache生成DAG(有向无环图),它是一个完整的搜索空间.③挖矿的过程就是从DAG中随机选择元素再进行哈希运算,可以从Cache快速计算DAG指定位置的元素,进而哈希验证.挖矿者存储数据集,该数据集周期性更新.④验证者能够基于缓存计算得到DAG中自己需要的指定位置的元素,然后验证这些指定元素的散列是不是小于某个散列值**PoS-Casper**采用’权益’(Stake,即以太币)为记账权背书.•思路:将Casper的应用逻辑通过智能合约来实现,记账权归属于’验证者’.任何拥有以太币的账户都可以成为验证者,前提是抵押一定的以太币.之后通过随机方式,选出一个验证者集合.验证者集合按照一定顺序依次验证区块,若区块没问题,就将其添加到BC中,验证者将会奖励.验证者不遵守规则就会没收抵押的以太币.**ytfGhost协议和叔块(UncleBlock)**•**GHOST**是一种主链选择协议(不是侧链选择协议).以包含子树数目最多为基本原则.•认为孤块也是有价值的,给发现孤块的矿工回报,给引用孤块的矿工奖励.**孤块被称为’叔块’**,为主链的安全作出贡献.•叔块上的交易不会得到执行.•叔块的好处:1.ytf孤块多,鼓励引用叔块,使引用主链获得更多的安全保证(因为孤块本身也是合法的)2.叔块报酬,缓解btb中心化.•区块可以引用0-2个叔块.被引用过的叔块不能重复引用.**Ytf钱包**管理EOA密钥对(私钥随机产生,用于签名,公钥由私钥导出,用于生成EOA地址)Privare Key; Keystor &passwd;Mnemonic code**以太币(ETH)**ytf发行的数字货币,在公有链上发起任何一笔交易都需要支付以太币.•来源包括’矿前＋区块奖励＋叔块奖励＋叔块引用奖励’.**单位:Wei智能合约**定义•从用户角度看,’智能合约’是根据事先任意制订的规则来自动转移数字资产的系统.•从开发者角度看,’智能合约’就是存储在BC上的代码,用以实现执行特定的功能.**BC上的智能合约**•是存储在BC上的代码,用以实现执行特定的功能.•**优势**:1.逻辑明确,不容易产生歧义.2.不可篡改.合约执行记录,可以作为永久凭证.3.合约执行的强制力可以保证**智能合约运行机制:** •智能合约具有值和状态属性,代码中预置了合约条款的相应触发场景和响应规则.•智能合约经多方共同协定、各自签署后随用户发起的交易提交,经P2P网络传播,矿工验证后存储在BC特定区块中,用户得到返回的合约地址及合约接口等信息后即可通过外部数据发起交易来调用合约.•矿工受系统预设的激励机制激励,将贡献自身算力来验证交易,矿工收到合约创建或调用交易后在本地沙箱执行环境EVM中创建合约或执行合约代码:•合约代码根据可信外部数据源和世界状态的检查信息,自动判断当前所处场景是否满足合约触发条件,以严格执行响应规则并更新世界状态.•交易验证有效后被打包进新的数据区块,新区块经共识算法认证后链接到BC主链,更新生效.**智能合约的基础架构模型**•***运维层*:**封装了一系列对合约层中静态合约数据的动态操作.•***合约层***:封装静态的合约数据,可看作是智能合约的静态数据库,封装所有智能合约调用、执行、通信规则.•***基础设施层***:封装了支持智能合约及其衍生应用实现的所有基础设施.**智能合约结构**合约包含状态变量、函数、函数修改器、事件、结构和枚举.支持*继承*.**创建和调用智能合约的流程**1.编写创建智能合约,编译为字节码,打包后部署到ytfBC网络上2.调用合约,发起交易 **ytf上的智能合约相关流程**•编译合约:•编写,再编译成EVM的字节码形式.•部署合约:需要一个外部账户通过一个部署合约的交易来将合约的代码部署到链上.合约在部署后,会以**合约账户**的形式存在,拥有自己的合约地址.•调用合约:用户需要构造一笔交易/消息.该消息的接受者是目标合约的合约地址,消息中Data部分需要加上对应的数据.•监听:(编写合约时)可以在合约中定义各种event.•在合约的函数中,emit<eventname>(args..)触发事件.•通过web3.js中的函数监听.•销毁•selfdestruct函数.只有拥有者才能销毁合约 **合约使用场景**金融类场景:对冲合约、储蓄钱包、遗嘱等 非金融类场景:在线投票、去中心化治理、域名注册 **智能合约-安全性**•智能合约都被公开的执行,源码很容易获得, Bug和漏洞也更容易被恶意攻击者发现.•TheDAO事件 **ytf虚拟机**是ytf协议和具体操作的核心.•基于栈的架构,ytf中的堆栈指令解释执行器,图灵完备的执行环境.基本数据单元256位的’字’.• EVM的所有指令执行必须具有*确定性的输出*.•*单线程*.•EVM为了保证交易的可终止性,引入了执行器燃料Gas,按照执行指令的复杂度、所需要的内存空间等对每个步骤进行计价,消耗指定数量的Gas, Gas=0时停止指令的执行,退出执行环境.**执行费用**1.维护ytf网络的稳定运行,对提供节点算力的用户支付一定的奖励,来源:单纯的区块打包奖励;打包交易的手续费.2.交易手续费定价取决于执行交易需要消耗多少算力.保证了智能合约不会出现执行后无法终止的情况3.执行指令,存储空间的使用都按Gas进行计价.**堆栈执行器**•ytf的EVM虚拟机是一种堆栈指令解释执行的执行器,从BC的数据库中加载代码,从头开始执行指令,结合交易请求的相关参数和相关的跳转指令来实现智能合约执行流程.**数据位置**内存或存储,三种类型:memory(函数参数默认),stroage(存到bc),calldata(只读 不会持久化).**Solidity**是一种用于编写ytf智能合约的面向对象高级语言1.Solidity中的合约与类相似,在一个合约中可以声明多种成员,包括状态变量、函数、函数修改器、事件等.继承.2.状态变量是永久存储在合约账户存储中的值,用于保存合约的状态.3.函数是合约的执行单位,共同组成合约的工作逻辑.4.函数修改器可用于改变函数的行为,在函数执行前插入其他逻辑.5.事件记录合约执行过程中发生的各种事件和状态变化.**事件Event**通过Logs实现事件功能.日志内容是交易收据的一部分,内容,包括Receipts的其它内容会生成一个ReceiptsRoot存储在区块的头部.日志永远存在.通过设置一些过滤条件,来获取某些特定的事件.可以用于追溯合约的历史状态,设置检查点等.**预言机Oracle**定义•BC外信息写入BC内的机制.**功能**将外界信息写入到BC内,完成BC与现实世界的数据互通.•是*智能合约与外部进行数据交互的唯一途径*,也是*BC与现实世界进行数据交互的接口*.•Oracle可以视作SmartContracts与外部交互的API.**为什么需要**？•BC是确定性的环境,智能合约不管何时何地运行都必须是一致的结果,所以虚拟机(VM)不能让智能合约有直接的networkcall(网络调用),否则结果就可能是不确定的.•智能合约无法主动去获取链外的数据,只能被动接受数据.**工作流程**•用户的智能合约把请求给链上Oracle合约,通过链下的API接口获得外部数据.外部把数据给链上的Oracle合约,然后Oracle合约再把数据给用户的智能合约.**应用场景**•一切需要与链下进行数据交互的DApp.**价值**1.打破了BC封闭的运行环境提高了BC对于外部 数据的感知度2.在打通交互通道的基础上维护了BC的一致性、安全性,真正实并建立了数据获取、组织、管理、存储与交互这—条完整的生态链3.增强BC应用的可移植性,实现智能合约的利益价值最大化.**架构模型**1.用户调用智能合约等方式发起预言机服务请求,调用内置的**预言机服务**合约接口,通过BC执行引擎请求预言机调用2.请求消息封装了对外部数据源的请求信息3.预言机向外部数据源请求外部数据,获得返回数据后生成一笔回调交易,签名后发回BC执行引擎**当前的问题**•中心化预言机非常易受攻击,去中心化预言机在未来将成为更好的方案.•链上是无法生成随机数的,或者说在链上的随机数是可以被预测和破解的.需要预言机从外部给智能合约安全的、不可预测的随机数.**Oracle分类**•**基于DATASOURCE:**•Software oracles.在互联网上处理,通过浏览这些数据找信息•Hardware oracles.扫描仪和传感器•Humanoracles.依靠人的行为•**基于TRUSTMODEL**:•Centralizedtrustmodel.集中式信任模型依赖于来自单一来源的数据.可能出现单点故障.•Decentralizedtrustmodel.解决了单点失效问题.高的延迟,效率低.•**基于DESIGNPATTERN**•Request-response.数据空间太大,只需要整个数据集的一小部分.•Publish-subscribe.有效地为预期会发生变化的数据(如价格和天气信息)提供广播服务.•Immediate-read.用于提供即时决策所需的数据的oracle,如学位认证和拨号代码.•**基于INTERACTION**:Inboundoracles.将外部世界的数据插入到BC中.•Outboundoracles.允许智能合同向外部世界传送数据.(智能锁).**目前预言机项目和解决方案**•Oraclize:为ytf提供中心化预言机服务.是中心化的,目前只能在ytf网络使用,gas费较高.•Chainlink:ytf上第一个去中心化预言机解决方案,通过在链上的智能合约和链下的数据节点,通过奖惩机制和聚合模型的方式,进行数据的请求和馈送,链式聚合成本较高,拓展性差,容易集中化.•欧链OracleChain:EOS上的第一个去中心化预言机解决方案.•欧链采用自主的PoRD机制,抵押代币奖惩机制的声誉系统,奖励数据节点惩罚作恶节点,可以实现Augur、Gnosis等预测市场应用的功能,还能支撑对链外数据有更高频率访问需求的智能合约业务.•DOSNetwork:支持多条主流公链的去中心化预言机服务网络•DOSNetwork是一个Layer-2的预言机解决方案,它通过在链上部署一个轻量级智能合约,p2p,保证了去中心化和数据安全,并达到快速反应.•可以适配所有主流公链**去中心化应用**DApp是一个大部分或者全部去中心化的应用.•设计时考:1.后端软件(⼀般去中⼼化,放在智能合约上)2.前端软件3.数据存储4.消息通信5.域名解析,所有这些可能是中心化的,也可能是去中心化的.•例如,可以将前端开发为运行在中心化的Web服务器上,或者开发为在个人设备上运行的移动端应用.后端和数据存储:部分运行在中心化的服务器上/都交给智能合约和P2P.**DApp优点**•弹性:•用智能合约管理后端逻辑.•透明:•上链特征使得任何人都能对代码进行检查,审核功能.交互历史都保存在BC上.•审查阻力:•只要用户能够访问ytf节点,就可以一直访问DApp.一旦代码被部署到BC网络,任何服务无法对代码进行修改.**DApp后端(智能合约)开发**•考虑因素:1.智能合约的大小:•若计算太过复杂,用户需要提供更多的交易费,降低了参与热情.设计者应当尽量将最关键的需要去中心化的逻辑和关键数据存储使用智能合约实现.2.合约代码被部署后无法修改:•更新合约的一种方法是使用版本控制系统.**数据存储**.•链下存储的理想方案是使用去中心化的P2P存储系统.**DApp消息通信和域名解析**•**消息通信**:•DApp的通信环节既可以使用传统的中心化服务器,也可以用基于P2P网络的去中心化的通信协议.•**域名解析**:•在ytf中,提供了**ENS**(ytf名称服务)来提供去中心化的解决方案.ENS本质上是一组实现在ytf上的智能合约,为其他智能合约提供服务.通过使用ENS(hash值转换为域名),用户可以通过易读的域名访问DApp或者智能合约.**ytf客户端**•软件应用程序,实现了ytf协议规范,并通过点对点网络与其他ytf客户端通信.•ytf协议有六种主要实现 •可以使用ytf客户端**同步**到各个ytf网络(主网或者测试网络),同步时可以选择全同步模式或者快速同步模式.也可以借助客户端**搭建自己的ytf网络**•不建议直接同步为ytf主网络的全节点,需要很大的存储空间和带宽资源.•先借助Ganacheytf节点仿真器进行部署测试,再使用ytf客户端同步到测试网络进行部署测试,最后部署到主网.**Ganache**•Ganache是用于个人开发的ytf节点仿真器, Ganache搭建的BC节点是运行在内存中的,对于每次交易的返回是实时的,适合开发人员在开发过程中用来测试和调试.**Truffle**•基于js,是用来快速开发ytf智能合约的开发框架.使用Truffle,可以快速地进行环境的配置、solidity智能合约的编译、脚本化部署合约、脚本化测试框架、得到合约抽象接口,从而方便地进行整个应用的开发.•Truffle官方提供了多个DApp应用开发模板,通过使用truffleunbox命令,可以快速下载、构建并初始化对应的开发模板.**案例分析**:去中心化评课社区**整体架构**•表示层主要提供用户与平台的智能合约交互以及与自己部署的账户智能合约交互的可视化图形界面.•业务逻辑层与BC层的交互通过调用web3.js库中封装好的ytfapi来实现.•BC层即ytf网络,可以是本地的模拟测试网络,也可以是真实地ytf主网络.**智能合约**•在去中心化评课社区中,主要用到两个智能合约,CommentRegistry(平台合约)和CommentAccount(账户合约).***超级账本HyperledgerFabric*联盟链**•由多个机构共同参与管理的BC,每个组织或机构管理一个或多个节点,其数据只允许系统内不同的机构进行读写和发送.联盟链的各个节点通常有与之对应的实体机构组织,通过授权才能加入/退出网络.各机构组织组成利益相关的联盟,共同维护BC的健康运转.•指定多个预选的节点为**记账人**,每个块的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,其他第三方可以通过该BC开放的API进行限定查询.**联盟链的特点**•部分去中心化.达成共识容易,交易速度自然也就快很多.•可控性较强.只要所有机构中的大部分达成共识,即可更改区块数据.•数据不默认公开,可以提供更好的隐私保护.不同于公有链,联盟链的数据只限于联盟里的机构及其用户才有权限进行访问.•节点是(基本)可信任的.公有链:匿名的,以出块为激励的BC.联盟链:实名的,以其上运行的商业业务价值为激励的BC.**联盟链平台-超级账本(Hyperledger)**为透明、公开、去中心化的企业级分布式账本技术提供开源参考实现,并推动BC和分布式账本相关协议、规范和标准的发展.遵守Apachev2许可,基本原则:•重视模块化设计:包括交易、合同、一致性、身份、存储等技术场景;•重视代码可读性:保障新功能和模块都可以很容易添加和扩展;•可持续的演化路线:随着需求的深入和更多的应用场景,不断增加和演化新的项目.**Hyperledger管理结构**•TechnicalSteeringCommittee-技术委员会•Governi-ngBoard-管理董事会•LinuxFoundation-Linux基金会**Hyperledger项目管理**•推动BC跨行业应用的开源项目的总称,组织成员可以发起新的BC项目,加入到超级账本项目中,但需要遵循Hyperledger的生命周期.•**生命周期:**Proposal(提案)、Incubation(孵化)、Active(活跃)、Deprecated(过时)、EndofLife(结束).提案:成员发起新项目时,首先发起者撰写草案(实现的目标、开发过程、代码维护等信息),提交给技术委员进行审核; 孵化:当技术委员会有三分之二通过,对项目进行开发、测试,直到项目完成; 活跃:项目参与者对该项目没有疑问.经过几年以上时间后,将进入过时阶段,最后结束整个生命周期.•**HyperledgerFabric**是Hyperledger中的BC项目之一.有一个账本,使用智能合约,是一个参与者管理交易的系统.•HyperledgerFabric是一个许可BC平台,成员需要从可信赖的**成员服务提供者(MSP)**注册.•**许可BC**在一组已知、已识别且经常经过审查的参与者之间运作,这些参与者在产生一定程度信任的治理模式下运作.许可BC提供了一种方法来保护具有共同目标但可能不完全信任彼此的一组实体之间的交互.通过依赖参与者的身份,许可BC可以使用更传统的崩溃容错(CFT)或拜占庭容错(BFT)共识协议,这些协议不需挖掘.•HyperledgerFabric具有模**块化的体系结构**.可插拔共识、可插拔身份管理协议、密钥管理协议或加密库.•HyperledgerFabric还提供**创建通道**的功能,允许一组参与者创建各自的交易账本.对于某些网络而言,这是一个特别重要的选择.这些网络中,一些参与者可能是竞争对手,并且不希望他们做出的每笔交易都被每个参与者知晓•Fabric支持通用编程语言编写的智能合约**Fabric系统结构Fabric模型**•资产(Assets):资产定义为可以在网络上交换的几乎任何有货币价值的东西,可以从有形资产到无形资产.•链码(Chaincode):链码的执行与交易排序分开,限制了跨节点类型所需的信任和验证级别,优化网络的可伸缩性和性能.•账本功能(LedgerFeatures):不可变的共享账本对每个通道的整个交易历史进行编码,高效的审计和争议解决.•隐私(Privacy):通道和私人数据收集使私人和保密的多边交易成为可能,在公共网络上交换资产所必需的.•安全和成员服务(Security&Membership-Services):许可成员资格提供了一个可信的BC网络,参与者知道所有交易都可以由授权的监管机构和审计师检测和跟踪.•共识(Consensus):独特的一致性方法,实现企业所需的灵活性和可伸缩性.**Fabric应用层**提供API、事件、SDK• gRPC接口,是一个跨语言的RPC框架 •在API的基础上,Fabric官方给出了针对不同语言SDK开发基于BC的应用.•*异步通信*的模式.通过在链码中定义事件,然后使用应用程序去监听这些事件.事件被触发,执行预先定义的回调函数.**Fabric核心层**•包括三大组件:•BC服务(Blockchain)•链码服务(Chaincode)•成员权限管理(Membership)•BC提供一个分布式账本平台,BC代表的是账本状态机发生变更的历史过程.•链码包含所有的处理逻辑:通过调用链码接口来改变世界状态.•**世界状态**是一个键值数据库,用于存放链码执行过程中涉及到的状态变量.•成员权限管理基于PKI(公开密钥基础设施),平台可以对接入的节点和客户端的能力进行限制.**安全和成员服务**•超级账本交易网络中的所有参与者都知道身份.•**公钥基础设施**用于生成加密证书.•**数据访问控制**可以在更广泛的网络和通道级别上进行操作和管理.•’许可’概念,再加上通道的存在和功能,解决隐私和保密性.**分布式账本**•账本是Fabric中所有状态转换的有序、防篡改的记录.状态转换是由参与方提交的链码调用(称作’交易’)的结果.每个交易都会产生一组资产键值对,这些资产键值对在创建、更新或删除时提交到账本.•账本组织成一个BC和一个状态数据库.BC用于将不可变的、按顺序排列的记录存储在区块中,状态数据库来维护当前Fabric的状态.*每个通道有一个账本.每个节点为其所属的每个通道保留一份账本副本*.**Fabric账本功能**•使用基于键的查找、范围查询和组合键查询进行账本的查询和更新•使用富查询语言进行只读查询•只读历史查询-查询某个关键字的账本历史,数据溯源•交易包含链码中读取的键/值的版本(readset)和链码中写入的键/值的版本(writeset)•交易包含每个背书节点的签名,并提交给排序服务•交易被按序装入区块,并从排序服务上被’传递’到通道上的节点•节点根据背书策略验证交易,并执行这些策略•在扩展块之前,执行版本检查,以确保所读取的资产状态在链码执行期间未发生变化•一旦交易被验证和提交,就不可改变•通道的账本包含配置区块的定义策略、访问控制列表和其他相关信息•通道包含成员服务提供程序实例,允许从不同的证书颁发机构派生加密材料 **共识**•针对账本和交易提供的对应BC服务.比如针对接入层交易提供担保(背书)、针对接入层多个交易做排序、交易排序结果打包成区块后通知给组织的锚节点,然后锚节点再通过goosip协议广播给组织内的其他节点、节点收到区块验证通过后再同步到分布式账本中.**交易排序**•Solo模式指整个Fabric网络依赖于一个Order节点.•Kafka模式依赖于一个KafkaOrder集群.•BFT(拜占庭容错)模式则是去中心化的Qrder集群.**Fabric链码** 超级账本支持基于主流编程语言的智能合约(链码)设计 **BC应用**,由若干部署在BC网络中的智能合约,以及调用这些智能合约的应用程序组成.✓用户专注于与业务本身相关的应用程序;✓智能合约则封装了与区块账本直接交互的相关过程,被应用程序调用 **智能合约直接与账本结构打交道**.✓本质上是为了对上层业务逻辑进行支持;✓最终会部署在BC网络中与账本进行交互.应用程序通过调用智能合约提供的方法接口来实现业务逻辑.**Fabric核心概念**•**身份和MSP•节点•Fabric网络•账本•通道•链码•策略认证和身份管理**•不同参与者都需要确定对资源的确切权限以及对参与者在BC网络中拥有的信息的访问权限.•与参与者身份相关联的主体信息,也需要可被验证.•Fabric中采用(**MSP**)来定义管理组织内的有效身份规则,X.509证书作为身份,采用(PKI)分层模型实现:◼公钥加密,私钥解密;◼私钥签名,公钥验证;◼数字签名证书的机构(CA),负责发放和管理数字证书的权威机构;◼CA通过中间CA组建信任链,规避风险;◼Fabric提供了内置了FabricCA组件帮助用户在BC网络中创建CA.•根CA和中间CA构成的认证链,提供了身份证明;Fabric提供了一个内置的CA组件,允许在所形成的BC网络中创建CA.•(**MSP**)为BC网络中的参与者提供一个允许的身份列表;•在BC网络中,MSP主要在本地节点(localMSP)和通道(channelMSP)内生效;•每个组织(节点)都必须定义一个本地身份管理服务,定义该节点的管理权和参与权.组织、组织的管理员、节点管理员和节点本身都必须有相同的信任根;•**channelMSP**定义了通道级别的管理权和参与权,识别谁拥有通道级别的权限•**localMSP**只在本地节点或用户的文件系统实例化,**channelMSP**会在通道的每个节点实例化,并通过共识保持同步;•若组织需要加入通道,则需要在通道配置中包含包含组织成员信任链的MSP,否则,来自该组织身份的交易将被拒绝;•建议每个参与通道的组织都定义其MSP,更好的保证隐私;•**系统级别的通道**包含参与服务的所有组织的MSP **MSP和BC网络Fabric网络**为应用程序提供账本及智能合约(Chaincode)服务的技术基础设施.首先智能合约被用来生成交易,这些交易会被分发给网络中的每个节点,这些交易会被记录在它们的账本副本上并且不可篡改.用户可能是使用客户端应用的终端用户,或者是一个BC网络的管理员**.多个组织会聚集到一起作为一个联盟来形成一个网络,它们的权限是由一套在网络最初配置的时候联盟成员都同意的规则来决定的.并且,网络的规则可以在联盟中的组织同意的情况下随时地被改变**.•**通道Channel**实现BC网络中业务的隔离,一个联盟多个通道,每个通道可代表一项业务,一套账本,通道内的成员为业务参与方(组织),一个组织可以加入多个通道.通道分为:系统通道、应用通道•**排序节点**通过系统通道来管理应用通道,用户的交易信息通过应用通道传递,对一般用户来说,通道是指应用通道.•可以理解为子链•1个通道+1个账本+N个成员**FabricNetwork** Application-应用CA Channel通道Ledger账本SmartContract智能合约P:Peer节点O:Ordering排序R:Organization组织NetworkConfig ChannelConfig背景设定• R1-R4共同搭建HyperledgerFabric网络;•R4为网络的初始创建者,负责配置网络的初始版本(网络配置NC和排序O),不参与网络中的数据交易;• R4指派R1为网络的管理员,并共同管理网络配置•R1和R2因’..’项目A1有私有通讯C1的需求,并由两个组织协商制定了网络规范(CC1);• R2和R3因’…’项目A3有私有通讯C2的需求,并由两个组织协商制定了网络规范(CC2);•R123使用’..’项目A2,A2同时拥有在C1 C2通信的权限;•各通道C1C2遵循排序规则并使用排序规则(O);•R123分别部署了一个节点(P1-P3),其中P1同步了C1的账本、P2同步了C1和C2的账本、P3同步了C2的账本;•应用(A1-A3)通过智能合约(S)访问账本(L1-L2);•组织之间通过CA来辨识用户、应用等的合法性**网络演进**1.(R4)配置本机构的证书颁发服务CA4,为机构节点和管理员签发证书,并初始化Fabric网络,完成了网络配置NC4和排序服务O4 2.(R1)配置本机构的证书颁发服务CA1 3.(R4)指定(R1)为该网络的管理员4.R12组成联盟X1,由R14在NC中进完成配置5.R12小组内达成共识,并形成通道配置CC1和通道C1,C1通道的数据服从排序服务的调度6.(R1)的节点P1进入网络,并获得了通道C1的账本拷贝L1,账本逻辑上存于C1,但物理上还是存在各节点的拷贝L上的7.同样的,R2节点P2加入网络,并同步了账本L1的拷贝8.智能合约S5分别被部署在R12的BC结点P1、P2 9.客户端应用A1A2通过智能合约与通道C1进行交互,智能合约S5提供了完整的定义来实现对账本L1的查询和更新 10.类似地构建C2 **策略-Policies**•认证和身份管理解决了BC网络上的身份认证和管理权、参与权确权问题;•策略是一组定义了如何制定决策和达到特定结果的规则,是Fabric基础设施上的管理机制,负责管理网络成员在修改网络、通道或智能合约时如何达成一致;Fabric网络上的任意操作都受策略控制;•策略是Fabric与ytf和btb等其他BC有所不同的原因之一:•在ytf,btbBC中,交易可以由任意节点生成和验证,管理策略在任何时间固定;Fabric中用户被基础设施看到,用户有能力决定网络的治理.**策略域**:系统通道配置,应用通道配置,ACL和智能合约**Fabric节点**•Peers:表示组织中的节点,以区块的形式从Orderer节点接收有序的状态更新,维护状态和账本.种类•**确认节点**Comitter: •**背书节点**Endorser:对结果进行背书,返回提案响应给客户端•**锚节点**Anchor:通道中每个组织有一个.•**Leader节点**:节点代表,连接到Orderer节点,广播给其他节点.•**排序节点**Orderer:为网络中所有合法交易进行全局排序,并将一批排序后的交易组合生成区块结构.•**证书颁发机构CA节点**:负责组织内部成员用户生成和颁发数字证书**.排序节点和排序服务Ordering**•**排序服务**所有交易在发送给Committer节点进行验证接受之前,先全局排序.•Fabric的设计依赖于确定性的共识算法,任何由普通节点验证的块都是最终的和顺序一致的•将链码执行的背书与排序分离,使Fabric在性能和可伸缩性方面有优势.•节点是Fabric的基本元素,可以被创建,启动,停止,重新配置,删除.暴露了一系列的API;•节点维护了一份或多份账本,并且每个账本都有一个或多个智能合约/链码使用•账本数量和链码数量之间没有固定的关系.•节点维护的账本和链码具有冗余性,避免单点失效,去中心化和分布式 **节点和应用程序** **节点和通道**•通道允许一组特定的节点和应用程序在BC网络中彼此通信.•通道:由物理节点集合构成的逻辑结构,节点:提供对通道的访问和管理的控制点.**节点和组织**•Fabric网络由多个组织管理.节点是构建这种分布式网络的核心.**节点和身份**•网络中的每个节点都由其所属组织的管理员分配一个数字证书,当节点连接到通道时,其数字证书通过通道MSP标识其所属组织.**节点、应用程序和排序节点**通过交易的三个阶段和之前所讲述的模块.**一次交易的流程**•第一阶段:**提案阶段**1.应用程序A1生产了交易T1和提案P 2.T1和P被发送给了通道C上的节点P1和P2 3.P1 使用交易 T1 和 提案 P 来执行链码 S1,生成对交易 T1 的响应 R1,提供背书 E1 4.P2 使用交易 T1 提案 P 执行了链码 S1,生成对于交易 T1 的响应 R2,提供背书 E2 5.提案阶段当应用程序从足够多的有效的 Peer 节点那里收到了签过名的提案响应的时候就结束了,进入下一阶段•第二阶段:**排序和将交易打包到区块** 1.应用程序 A1 向排序节点 O1 发送由 E1 和 E2 背书的交易 T1 2.同时,应用程序 A2 将 E1 背书的交易 T2 发送给排序节点 O1;3.O1 将来自应用程序 A1 的交易 T1 和来自应用程序 A2 的交易 T2 以及来自网络中其他应用程序的交易打包到区块 B2 中; (在 Fabric 中,排序服务生成的区块是最终的,永远不会被重写或删除,确保了没有**账本分叉**;排序节点只做排序,不判断交易内容)•第三阶段:**验证和提交** 1.排序节点将区块分发给连接到它的所有 Peer 节点(并非每个 Peer 节点都需要连接到一个排序节点,Peer 节点可以使用 gossip 协议将区块关联到其他节点)2.每个节点将独立地以确定的方式验证区块,以确保账本保持一致(每个节点都验证区块中的每笔交易,确保得到了所需组织的背书和背书策略相匹配)3.验证完成后,节点 P1 处理区块 B2,在 P1 上的账本 L1 中添加一个新区块.同时,节点 P2 处理区块 B2,从而将一个新区块添加到 P2 上的账本 L1中(节点 P1 和 P2 上的账本 L1 就会保持一致的更新,并且每个节点都可以通知与之连接的应用程序交易已经被处理)**账本ledger**•Fabric中,分类帐由**世界状态和BC**组成•BC被构造为互连块的顺序日志,其中每个块包含一系列交易,每个交易代表对世界状态的查询或更新.•世界状态被实现为数据库,保存一组账本状态的当前值,账本状态表示为键值对.世界状态可更改世界,每次账本变更,相应的版本号递增.•只有获得所需背书签名的交易才会导致对世界状态的更新,若没有得到足够的背书人的签名,则不会导致世界状态的改变.•世界状态将**业务对象属性的当前值**保存为唯一的账本状态.•BC则是世界状态中的业务对象如何达到其当前状态的历史记录,记录了每个账本状态的所有以前版本及其更改方式.**区块**•**区块头**:包含三个字段,区块编号、当前块哈希、上一个块头哈希.•**区块数据**:包含按顺序排列的交易列表,排序服务创建块时写入.•**区块元数据**:包含区块创建者的证书和签名,用于被网络节点验证区块,区块确认节点将每个交易的有效/无效指示符添加到位图中,以便检测状态分叉.*此部分不是区块哈希计算的输入*.**交易**•块的区块数据中的交易由交易头、交易签名、交易提案、交易响应和背书列表组成.•**交易头**保存有关交易的一些基本元数据.**智能合约和链码**•智能合约定义了被添加到账本中的新事实的可执行逻辑;•**链码是智能合约的集合**;•一个Method和支持的Action相当于一个智能合约,Method的组合部署类似于链码•智能合约以编程方式访问 •主要在**世界状态中将状态写入(put)、读取(get)和删除(delete)**,还可以**查询**不可篡改的BC交易记录.•读取(get)操作查询,目的是获取关于交易对象当前状态的信息.•写入(put)操作通常生成一个新的业务对象或者对账本世界状态中现有的业务对象进行修改.•删除(delete)操作代表的是将一个业务对象从账本的当前状态中移除,但不从账本的历史中移除.•**链码和背书策略**•每个链码都有一个背书策略与之相关联,该背书策略适用于此链码中定义的所有智能合约,指明了BC网络中哪些组织必须对一个既定智能合约所生成的交易进行签名,以此来宣布该交易有效.•所有的交易都会被添加到分布式账本中,但仅有效交易会更新世界状态.•**背书策略是HyperledgerFabric与ytf或btb等其他BC的区别所在**.**有效交易** **私有数据-PrivateData**•若一个通道上的一个组织需要对该通道上的其他组织保持数据私有,则可以选择创建一个新通道,其中只包含需要访问数据的组织.但是,在每种情况下创建单独的通道会产生额外的管理开销,在保留一部分数据私有同时,所有通道参与者看到事务的情况是不允许的,创建私有数据集合的功能,允许在通道上定义的组织子集能够背书、提交或查询私有数据,无需创建单独的通道.Fabric使用P2P,gossip协议.**Fabric架构** Fabric联盟结构-联盟-通道(组织(节点(orderer,peer)链码(智能合约)账本(状态模型)(BC KV数据库)))

**BC激励机制**BC不仅是一种全新的分布式计算技术,同时也是一类新兴的交易 模式和成功的商业逻辑.BC系统的核心驱动力之一是经济激励.• BC经济学亦称加密经济学.利用激励和密 码学来设计的新型的系统、应用程序和网络.激励与博弈-**BC中的博弈论** “公共地悲剧” .**代币发行机制** BC经济激励的典型载体和表现形式.BC系统内生代币的发行机制决定了BC资产的分配机制 • 发行代币的BC项目一般可以分为**公链和去中心化应用(**DApp)两类,公链中往往包含多种属性的Token；应用型Token积分+股份相结合的方式,代表了项目的所有权.•**两种代币发行机制.单Token发行机制**,**总量有上限** 这种发行机制要设定Token的总量上限,三部分:项目团队持有部分、用于ICO或私募的部分,奖励给矿工的部分.这种应用型Token具有积分和股份的特性,可以用于兑换服务或抵扣费用,并通过免费发放来增加社区活跃度.持有Token的人可以通过分红和回购销毁等方式分享项目发展的红利.可能会出现挖矿的“公地悲剧”. .**总量无上限的Token**, Token数并不是固定的,随着区块的不断挖出而逐渐增加.目前主流的公链体系,ETH和EOS.• 在PoW共识机制下,持Token的人权利逐渐弱化.• 在PoS/DPoS共识机制下,每个持币人都能不断获得新发行的Token,从而保证持币人Token数量 占比的稳定性.• 在PoW/PoS共识机制下,是客观的,而在DpoS共识机制下具有主观性..**总量有上限+总量无上限相结合的Token发行机制**,主要采用混合挖矿方式来 发行新Token,并奖励给矿工或Token持有人,新发行Token来自总量有上限的Token(不断减少)总量无上限的Token具有一定的通胀率.**双Token 发行机制** • 代表BC系统所有权并且具备激励特性 • 是价格稳定的Token作为生态内的“货币”来使用 • 为了实现Token的稳定性,可以采用发行借据锚定美元等稳定资产、以其它Token作为基础资产抵 押和算法央行这三种方式来发行Token.• Tether,• SteemDollars(SBD•有上限的单币发行方式更具优势,有利于生态体系的建立.Token的分配机制需要权衡各方利益, 从而激励各利益群体参与项目的积极性.**代币分配机制** • 新发行Token的分配方式取决于其所在链上的共识算法: • 采用PoW共识算法的BC项目,往往会把新产生的Token全部分配给矿工.• 采用工作量证明+服务量证明混合模式的项目DASHBC,把区块奖励在矿 工、主节点和基金会这三者之间进行分配,其比例分别为45%、45%和10%.• 采用DPoS共识机制的项目SteemBC,将新发行的Token分配给潜在消费者 (即代币持有者)和矿工,其比例分别为90%和10%**EOS 代币分配机制** • EOS代币分配是运行在以太坊BC上EOS代币分配机制可以保证广泛接受性、 逻辑一致性和公平性.**Filecoin代币FIL的分配机制** • 采用线性释放,即随着每个区块的挖出,逐步分发Token,这样的分配过程可以确保代币的发放过程平滑,可以有效避免突然间的大量代币解禁而对币价造成波动的情况.**交易费**•基础区块奖励和交易费两部分.**矿池与分配方式** • **Solo挖矿**:矿工利用其自身算力进行独立挖矿的过程.**矿池挖矿**:矿池根据BC网络中的挖矿难度给矿工分配适 当难度的任务,通过收集矿工 提交的满足该难度的解来统计矿工的工作量,并根据矿工的 工作量分配奖励.**按比例分配(Proportional)**是激励不相容的.跳矿:矿工可以策略性地选择何时为矿池挖矿以及何时将算力转移到其他矿池,从而获得高于其实际算力的收益,这将损害他人收益.**PPS(Pay Per Share)**将按比例分配模式中矿工所面临的风险转移到矿池,从而使矿工可以获得稳定的收益.•对于矿工优点• 可以将矿工每个部分解的奖励风险降为零,当矿工提交部分解时可以立刻获得.•精确地知道其应获得的奖励数量,容易地验证是否获得了应得的奖励•矿工也不会因为发生跳矿行为而损失奖励.• 对于矿池缺点•风险非常大,所有矿工面临的风险都被转移到矿池.**PPLNS(Pay Per Last N Shares)**运气成分,具有一定的滞后性,矿工挖矿的奖励会有一定的延迟.• PPLNS模式是激励不相容的,矿工可以通过一些策略性行为如延迟汇报等来提高其收益.**策略性行为-自私挖矿** •由矿池发起,新挖到的区块暂不公开,并等待合适的时机广播其全部区块.•目的不是破坏加密货币的BC网络,而是获得更大利润.**扣块攻击** •攻击者在矿池中进行挖矿时,只向矿池汇报部分解, 从而降低矿池及矿池中每个矿工的收益..**跨链套利和跨矿池套利**•矿工提高收益的主要策略性行为,而是根据各链或矿池的当前收益进行切换,不断转到收益较高的分叉链或矿池中进行挖矿,以获取更高收益.损害诚实矿工的收益 和矿池的稳定性.**挖空块** •空块:新区块只含区块头,区块体为空.挖空块时,只能获得coinbase奖励,无交易费奖励.解决:提前获得这些交易信息.BTC/BCH网络中采用布隆过滤器和致密区块或瘦区块解决,无法真正解决• 彻底解决挖空块,矿池需要找到一些不可能在H高度块中包含的交易 **AsicBoost挖矿**•提升挖矿速度.破坏了挖矿的公平性.采用Segwit来阻止AsicBoost并不一定有效.**小结** •BC系统的核心驱动力之一是经济激励.• 加密经济学是保证BC生态系统自洽运行的先决条件.核心要素:保证项目参与方价值分配的合理性和激励相容性,通过这种价值分配方式,实现各参与方的利益最大化,并能够有效激励更多加入.•经济激励是保证各参与方进行分布式协作挖矿的重要前提,是维护BC挖矿可持续性的重要保证.BC技术中所蕴含的经济激励使区 块链技术具有巨大的发展和应用潜力.