YeeZChain v1.1 介绍文档

Introduction to YeeZChain v1.1

熠智科技

https://yeez.tech

2020 年 6 月

目录

1 概述

区块链 (Blockchain) 是一种由多方共同维护，使用密码学保证传输和访问安全，能够实现数据一致存储、难以篡改、防止抵赖的记账技术，在某些领域也被称为分布式账本（Distributed Ledger）。值得注意的是，区块链并不是一种单一的技术，而是多种技术整合的结果，包括但不限于分布式存储、共识机制、智能合约、对称/不对称加密等等。这些技术以新的结构组合在一起，形成了一种新的数据记录、存储和表达的方式。

区块链开创了一种在不可信的竞争环境中低成本建立信任的新型计算范式和协作模式，凭借其独有的信任建立机制，实现了穿透式监管和信任逐级传递。区块链源于加密数字货币，目前正在向垂直领域延伸，蕴含着巨大的变革潜力，有望成为数字经济信息基础设施的重要组件，正在改变诸多行业的发展图景。[1]

YeeZChain 是由熠智科技研发的自主可控区块链底层平台，由模块化架构支撑，并具备极佳的隐私性、灵活性和可扩展性，能有效支撑企业、政府、产业联盟等行业的分布式账本应用。核心研发团队全部为国内外知名高校计算机专业博士和硕士研究生，曾就职于微软、华为、旷视科技等互联网公司。YeeZChain 结合了团队在区块链领域的多年研发经验，基于企业的实际需求和应用场景，致力于通过技术创新推动整个区块链行业的发展。

YeeZChain 平台基于微服务架构，支持每个微服务独立部署和扩展，在此之上提供了可插拔共识模块，高性能的智能合约执行引擎，跨链交互，成员、证书动态管理和可视化部署运维工具等。同时在保证区块链去中心化和数据不可篡改特性的基础上，YeeZChain 提供了对基于密码学的数据隐私保护技术和基于可信执行环境（TEE）隐私计算的支持。

YeeZChain 的设计指导原则：

- **自主研发** 底层完全自主研发，真正掌握核心技术;
- **技术创新** 相关技术在国际顶级会议发表多篇论文，已申请多个技术专利;
- **聚焦行业** 专注可信数据交互、供应链金融、智慧城市等领域;
- **开放生态** 打造开放社区，兼容最新技术标准。

[1] 来源: 中国信息通信研究院和可信区块链推进计划《区块链白皮书（2019）》，中国信通院

2 技术

2.1 基础知识

对于初次接触区块链技术的人员，推荐首先阅读本章的内容，以便于理解区块链的工作原理、YeeZChain 具有的主要特点以及包含的组件。

分布式账本 区块链账本通常被定义为去中心化，这是因为在整个网络中，每个参与者都保存着一个区块链账本的副本，所有参与者通过协作共同维护着账本。除了去中心化与协作，区块链的另一个显著特点是信息在只能以“附加”的方式记录在区块链中，同时使用加密技术保障了交易一旦被添加进账本中，就无法被篡改。区块链的这种不可篡改性使得信息来源的确认变得异常容易，这是由于参与者可以肯定信息一旦被写入区块链中就几乎不可被篡改。

共识机制 共识是分布式系统中的基本性质之一，具体来说是指通过消息传递使得系统中所有节点均以相同顺序执行一个命令序列，而在实际环境中，节点可能出错产生异常行为，或者消息无法被正确传递，使得系统无法达成一致。区块链技术的出现让共识机制重新得到关注，在区块链网络中，共识机制实现了网络中所有账本交易的同步流程，共识保证了账本只会在交易双方都确认后才进行更新。同时在账本更新时，交易双方能够在账本中的相同位置，更新一个相同的交易信息。根据是否能够容忍节点的作恶行为，共识机制通常分为非拜占庭类共识和拜占庭容错共识。

智能合约 为了实现更复杂的逻辑操作，以及对账本进行管理，区块链网络引入了智能合约来实现对账本的访问和控制。智能合约是一套以数字形式定义的承诺，包括合约参与方可以在上面执行这些承诺的协议。一个合约由一组代码（合约的函数）和数据（合约的状态）组成，并且运行在区块链上。根据底层实现机制，区块链智能合约实现方式主要包括 Docker 和轻量级虚拟机两种方式。

加密 区块链是建立与密码学算法之上的一种分布式记账本，从区块链底层数据结构到区块链交易，密码学算法都发挥着它的重要作用。区块链涉及的密码算法包括了哈希函数和非对称加密算法。哈希函数主要用于对传输信息进行完整性校验，保证数据的完整性。而非对称加密技术一般用于对身份识别和认证，主要是为了保证加密通信只有两个人看到并确认消息的发出者。

隐私保护 在区块链中，所有数据 (包括交易类别、交易双方的地址、交易金额等) 都是公开的，这在一定程度上提高了参与者对数据真实可靠的信心。大多数区块链系统

会将所有交易数据记录在公共账本中，任何用户均可查询。但公开所有数据在某些关键应用场景下是不可行的，例如金融、医疗等数据敏感性高的场合。因此，区块链交易数据隐私保护的目的在于，满足区块链系统正常运行的前提下，选择合适的密码技术来实现交易数据的隐私保护。

许可区块链 根据节点的权限，区块链被划分为许可区块链（Permissioned Blockchain）和非许可区块链（Permissionless Blockahcin），一种更为通俗的划分方式是联盟链（Consortium Blockchain）和公链（Public Blockchain）。以比特币、以太坊为代表的非许可区块链允许任何人在其中进行交易，甚至可以作为验证者加入；而许可区块链则是由若干机构或组织共同参与管理的区块链，这些机构组织各自运行着一个或多个节点，之中的数据只允许系统内不同的机构进行读取和发送交易，并且共同记录交易数据。

许可区块链（联盟链）以部分中心化的代价，换取了如下优势：

- 安全的访问控制；
- 更好的扩展性；
- 更高的交易处理性能，和更低的能耗；
- 符合相关法律法规

2.2 YeeZChain 简介

YeeZChain 是一个基于账户模型的许可区块链，由模块化架构支撑，并具备极佳的隐私性、可伸缩性、灵活性和可扩展性。

据统计，目前超过 50% 的联盟链都沿用了以太坊或 Hyperledger Fabric 的架构设计，然而在实际的应用场景中联盟链与公链存在的很大不同，通常面临着十分复杂的定制需求，包括但不限于：

- 一致性、可用性需求：为满足账本数据的一致性和可用性等安全要求，传统分布式系统中的数据校验、数据容灾备份等技术方案正在逐渐被应用到区块链系统中；
- 高性能需求：在某些高频交互场景下，对于底层区块链平台的性能需求更高，直接表现为要求更高的 TPS，这需要共识机制、网络通信、硬件等多方面的优化实现；

(YeeZTech) 熠智科技

- 隐私安全需求：随着区块链技术在供应链金融、司法存证、政务数据共享等对数据安全有强需求的领域推广落地，在签名验签、链上数据授权访问等业务流程中，国密加密逐渐成为区块链应用的主流选择，同时数据隐私保护正在引起重视；
- 软硬件兼容：需要兼容政府企业已有的业务系统（操作系统、数据库）和硬件，以及物联网终端设备；
- 跨链需求：在目前许多行业存在“多链并存”的情况下，应用场景已经从从原有跨链资产互通逐步过渡到多链间全状态的自由流通。类似于“主侧链”、“中继链”的技术尚处于早期阶段，同时“链上-链下”数据协同也有待研究。

基于以上现状，YeeZChain 的架构采用了更加方便定制的轻量级微服务架构，以便能够应对繁复的企业定制需求，并最大程度实现模块复用。这也是 YeeZChain 与其他区块链解决方案的显著区别。具体来说，YeeZChain 中各个组件（例如：存储、共识、执行、权限等）之间彼此独立，并通过微服务分发路由进行通信。每个组件可以不同的编程语言实现，完全去耦合、即插即用。

YeeZChain 的最新版本架构如图1所示：

- 物理层：运行 YeeZChain 的基础设施（资源），包括计算资源、存储资源、网络资源以及虚拟化服务;
- 核心层：YeeZChain 的核心部分，包括了共识引擎、智能合约执行引擎、数据存储和加密组件等等；
- ABCI: YeeZChain 将共识逻辑和上层应用做了分离，底层和上层应用基于 ABCI 进行通信；
- 应用层：YeeZChain 的应用，包括转账、智能合约的基础超级账本应用以及其他扩展应用；
- SDK：开发和部署工具，包括 IDE、API、区块链浏览器、监控、安全分析检测工具、自动化部署工具等等；
- 管理工具：负责证书、权限和身份管理

2.3 YeeZChain 特点

YeeZChain 所具有的核心特性如下：

溯信科技（YeeZ Tech）

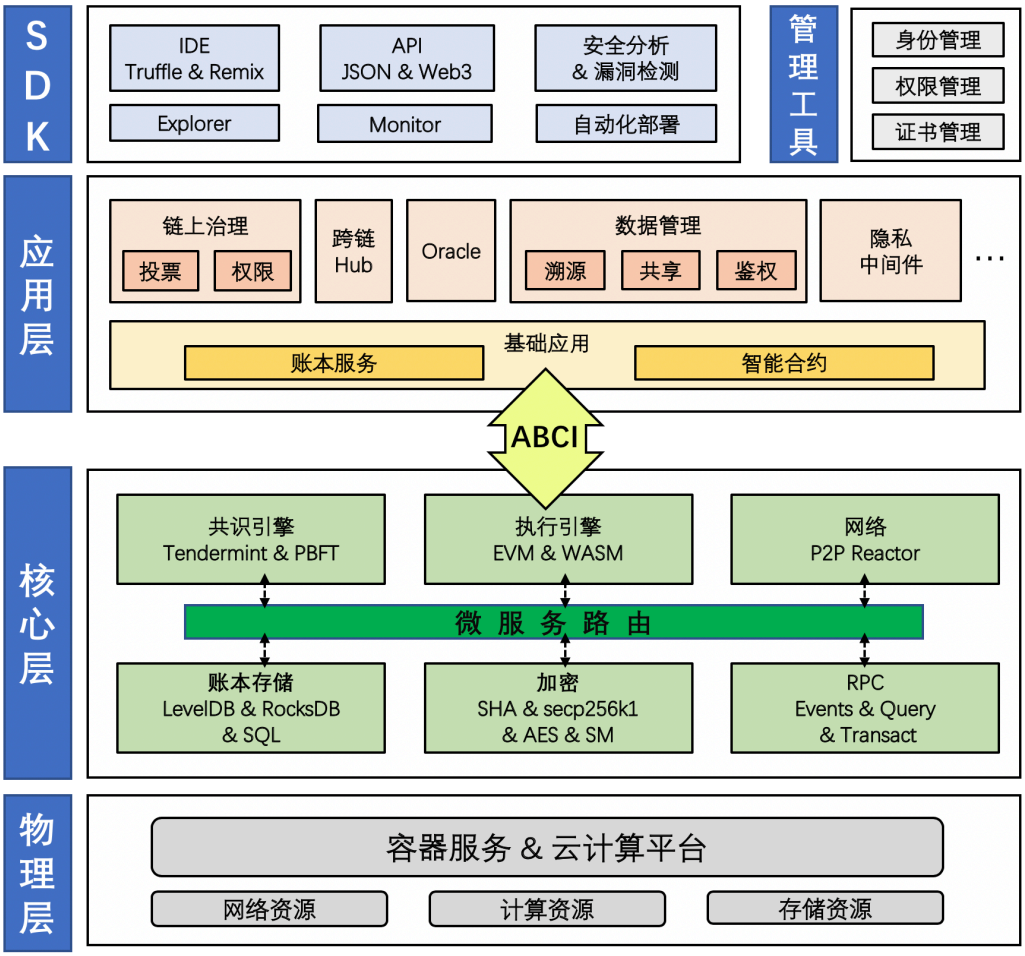

图 1: YeeZChain 架构图

微服务架构 底层基于微服务架构设计，支持每个组件独立部署和扩展，组件可以跨语言开发，即插即用。可以灵活应对繁复的企业定制需求，快速兼容政府企业已有的业务系统 (操作系统、数据库) 和硬件以及物联网终端设备。

独立共识模块 YeeZChain 将共识模块和 P2P 网络封装为共识引擎，上层应用逻辑与之分离，应用开发无需考虑系统一致性问题。共识引擎支持插拔共识，包括 PBFT、Tendermint 等共识算法。共识性能最高可达 3000TPS[2]，低于 300ms 延时，同时支持多达 100 个验证节点网络规模。

高性能执行引擎 YeeZChain 采用深度优化的 EVM 和 WASM 兼容虚拟机，支持 Solidity、Typescript 和 WebAssembly 等智能合约编程语言。同时完美支持 Truffle、Remix 等主流智能合约开发部署工具，提供 RPC JSON & REST 调用接口，兼容 Web3 标准。

全链路安全 支持多级加密机制，针对数据、账户和网络通信各个环节进行加密保护，提供 Secp256k 哈希函数以及 RSA 和椭圆曲线加密算法的支持，除此之外也支持国密 SM2/3/4 等多种加密和签名算法。

自适应升级 YeeZChain 可以通过将系统升级信息封装为特殊交易更新到各个节点从而保证一致性，同时该运行时环境支持多协议版本的并发执行，最后执行引擎通过软件事物内存保证了并发请求下执行结果的正确性。

隐私保护 在保证区块链去中心化和数据不可篡改特性的基础上，YeeZChain 提供支持基于密码学的数据隐私保护技术和基于可信执行环境的隐私计算中间件[3]。从隐私数据的管理到隐私数据的价值传递、挖掘, 满足不同维度的隐私需求。

跨链互操作 支持中继跨链，通过链间通信（IBC）协议，支持不同分区通过共享 Hub 来互相通信与互操作，解决区块链交互操作和可扩展性问题。

简易部署运维 支持内部网络、复杂网络下的节点一键部署。提供完整的管理运维工具，方便实现管理、配置、监控等方面的功能。

[2]包括转账与智能合约调用
[3]具体内容详见《熠智隐私计算中间件产品介绍》

熠智科技 (YeeZ Tech)

项目	YeeZChain	Hyperledger Fabric	优势
共识算法	PBFT、Tendermint，可扩展	Raft、Solo	支持拜占庭容错，共识算法适应不同需求
执行引擎	EVM、WASM 虚拟机	ChainCode（go）	兼容主流智能合约开发框架，便于现有 DApp 移植，支持跨平台设备智能合约执行
加密算法	ECC/ECDSA，RSA，国密 SM2/3/4	ECC/ECDSA	全链路安全，支持国密算法
TPS	≥3000	300~500（Kafka）	满足现有业务场景性能需求
PKI/CA	自建 CA 第三方 CA	自建 CA	支持第三方 CA
隐私保护	支持 TEE 全同态加密	Channel 隔离 私有数据存储	支持复杂数据处理逻辑，不影响节点架构
数据存储	KV 数据库 SQL 数据库 IPFS	KV 数据库	支持多种数据库，支持分布式存储
跨链	中继跨链	不支持	支持跨链互操作
运维部署	支持一键部署，提供可视化监控	第三方适配	满足快速部署运维需求
系统升级	支持自适应升级	手动升级	简化系统升级流程，确保一致性，前后版本兼容

图 2: 技术对比

表 1: YeeZChain 区块定义

字段名	描述	大小
区块头	区块元数据集	不超过 653 字节
交易信息	区块中包含的交易信息	大小可变
验证信息	验证节点的错误行为证明	大小可变
提交信息	所有参与预提交的节点签名	大小可变

表 2: YeeZChain 区块头信息

字段名	描述	大小
协议版本	共识协议版本	8 字节
链编号	区块所属链编号	8 字节
高度	区块在链中高度	8 字节
时间戳	节点构造区块的近似时间	8 字节
交易数量	区块包含交易数量	8 字节
交易 Hash	区块中包含的交易 Hash 集合的 MerkleRoot	32 字节
节点 Hash	当前区块参与共识节点 Hash 集合的 MerkleRoot	32 字节
共识 Hash	共识参数 Hash	32 字节
状态 Hash	所有账户当前状态的 Hash	32 字节
验证 Hash	验证节点错误行为的 MerkleRoot	32 字节
提案地址	打包区块节点地址	20 字节
其他	前置区块的执行 Hash、前置区块验证签名 Hash 等	大小可变

3 基础组件

本章介绍了 YeeZChain 的基础组件。

3.1 数据模型

3.1.1 区块

区块链是区块链账本中的重要数据结构，存储着核心交易信息。区块链由区块组成，区块本质上是一个数据包，可以定义为记录一段时间内发生的交易和状态结果的数据结构，是对当前账本状态的一次共识。类似于其他区块链系统，YeeZChain 的区块主要由区块头和交易列表组成，区块具体包括的内容如表1所示。

区块头封装了前一个区块的 Hash 值、时间戳、Merkle 树根植和当前区块的 Hash 值等信息。区块头具体内容如表2所示。

交易信息则包含了所有的交易详情。关于验证信息和提交信息，在3.2章节会有所介绍。

区块数据主要通过区块的形式进行串联，所有区块被从后向前有序地链接形成

溪科技 (YeeZ Tech)

表 3: 账户包含信息

字段名	描述	大小
地址	账户地址，由公钥生成	20 字节
公钥	由私钥生成	32 字节
序号	账户发起的交易数	8 字节
余额	账户拥有的余额	8 字节
权限	账户所拥有的权限	8 字节
名称	智能合约名字（如果该账户是智能合约）	大小可变
EVM&WASM 字节码	智能合约字节码（如果该账户是智能合约）	大小可变
合约 Hash	智能合约 Hash（如果该账户是智能合约）	32 字节
合约元数据	描述合约信息的数据（如果该账户是智能合约）	大小可变
创建者	智能合约创建者（如果该账户是智能合约）	20 字节

一个“链条”，每一个区块都指向其父区块。

3.1.2 账户

从数据存储的逻辑角度而言，在目前的区块链系统中主要有两种记录保存方式，一种是账户/余额模型，一种是 UTXO 模型。以比特币为代表的早期区块链系统采用了 UTXO 模型，而以太坊、Hyperledger Fabric 等项目采用的是账户/余额模型。

相比 UTXO 模型，账户/余额具有如下优点：

- 可以实现复杂编程逻辑，合约以代码的形式存储在账户中，且账户拥有自己的状态；
- 批量交易更方便，可以非常方便的使用合约进行批量交易；
- 轻节点的验证更简单可行。

类似地，YeeZChain 采用了账户/余额模型，账户以地址为索引，地址由公钥生成。在 YeeZChain 中，账户所包含的信息如表3所示。

对于权限管理，YeeZChain 的账户拥有不同权限，具体包括：

- 共识权限：可以参与区块共识；
- 转账权限：可以发起转账交易；
- 部署权限：可以部署智能合约；
- 调用权限：可以调用智能合约；

溯科科技 (YeeZ Tech)

表 4: YeeZChain 交易基本定义

字段名	描述	大小
交易类型	交易类型定义	4 字节
交易 Hash	根据交易内容生产的 Hash 值	32 字节
发起地址	交易发起者地址	20 字节
金额	交易附带的金额	8 字节
用户签名	发起者对交易内容基于签名生成的签名信息	65 字节
序号	交易序号，通常为交易发起者当前序号 +1	8 字节
扩展字段	不同类型交易包含的扩展信息	大小可变

- 管理权限：可以创建新账户、以及管理账户权限；
- 提案权限：可以发起提案，用于系统协议变更

3.1.3 交易

交易（Transaction）是 YeeZChain 中最为基础的数据结构之一，整个区块链的状态都是由于交易而发生变化。不同于其他区块链系统，YeeZChain 的交易有多种类型，定义也更为灵活，可以包含不同字段，交易包含的基本信息如表4所示：

在 YeeZChain 中，存在多种交易类型：

- 调用交易：最为基础的交易类型，可以调用 EVM，并且所有交易都可以基于调用交易实现；
- 转账交易：不同地址之间进行转账的交易；
- 绑定 & 解绑交易：申请注册为共识节点或者解绑的交易；
- 权限管理交易：基于此交易可以修改指定账户的权限；
- 提案交易：发起一次提案，投票通过后则执行相关操作（调用智能合约或者转账）

3.1.4 数据库

在 YeeZChain 中，上述数据都被存储在 K-V 型数据库中，其中键值都是二进制的。具体而言，有三个数据库，分别是 BlockDB、StateDB 和 EvidenceDB。BlockDB 保存了块的主体内容、包括块头和交易；StateDB 保存了账户的状态数据库；EvidenceDB 记录了每次共识的信息，包括参与验证者和投票信息。

链智科技（YeeZ Tech）

YeeZChain 支持 LevelDB 和 RocksDB 两种 K-V 型数据库，两者都支持十亿级别数据量，并且保证非常高的读写性能，并且具有如下优势：

- 支持双向遍历；
- 支持原子写操作；
- 支持过滤策略（bloomfilter）；
- 支持数据自动压缩（snappy 压缩算法）
- 底层提供了抽象接口，允许用户定制

同时为了扩展数据存储需求，YeeZChain 也提供了 SQL 类型数据库以及对分布式文件存储的支持，具体见扩展组件章节。

3.2 共识引擎

共识算法是保证区块链平台各节点账本数据一致的关键，目前常见的分布式系统一致性算法包括 Paxos、Raft、PBFT 等。

以比特币和以太坊为代表的区块链通常对于出块节点没有准入限制，系统通过竞争方式选择出块节点，其中最为广泛采用的是工作量证明 (Proof-of-Work，PoW) 算法。需要指出的是，PoW 仅为出块节点选择协议，而并不是完整的共识机制[4]。

联盟链通过缩小节点规模，实现了更强的一致性保证。节点针对每个提案进行投票，通常基于 PBFT 或者某些更为宽松的投机 BFT 算法。目前主流的企业级区块链解决方案例如 HyperLedger Fabric 都供了 PBFT 的实现方案。YeeZChain 提供了 PBFT 的支持，此外，针对原生 PBFT 灵活性不够的问题，YeeZChain 默认采用 Tendermint 引擎。

Tendermint 是一个可定制的区块链共识引擎，其共识机制是全球首个应用于大规模节点区块链网络的异步 BFT 算法，并通过了严谨的理论验证 [?]。Tendermint 共识为拜占庭式的容错，可以容忍 1/3 的故障，包括任意行为例如黑客攻击和恶意攻击。

3.2.1 共识算法

整个共识网络中的的参与者被称为验证者（validator），验证者轮流提出交易区块，并且所有验证者对其进行投票。区块会被提交到链上，每一个块占据一个高度 h。

[4]通常比特币采用的共识被称之为 Nakamoto Consensus，包括了 PoW 机制和最长链原则

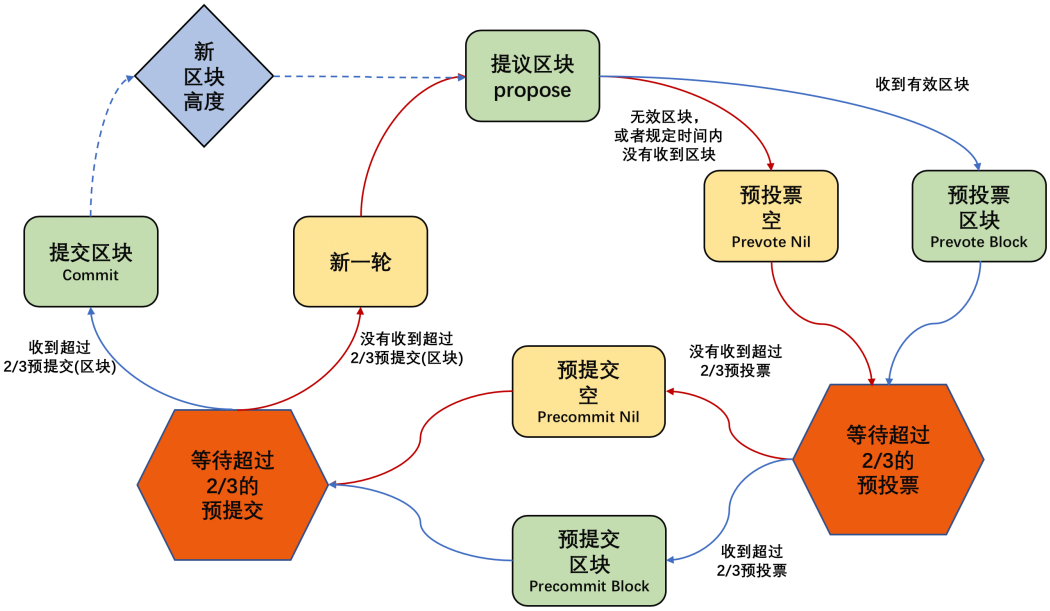

图 3: Tendermint 共识流程

对高度为 h 的块共识的每一轮包括 4 个步骤：提议（Propose），预投票（Prevote），预提交（Precommit）和提交（Commit）。当然提交块可能会失败，如果失败就会开始下一轮的提交。要想成功提交一个块，需要有两个阶段的投票：预投票和预提交。在同一轮提交中，只有超过 2/3 的验证人对同一个块进行了预提交，这个块才能被提交到链上。

每个回合分成若干个步骤：

1. 由本轮被选中的验证者提出区块，每轮只有一个验证者可以提出块，并且用私钥签名从而可以追责。对于如何选择负责出块的验证者，可以采用 round-robin 策略或者类似于 PoS 节点权重计算概率选择。在该阶段允许验证者跳过（轮到某验证者出块的时候此验证者没有出块），其他验证者在移到下一轮投票前会等待一小段时间来接受提议者提出的区块，因此 Tendermint 实际上是一个弱同步协议。

2. 所有验证者对提议进行预投票，通过私钥进行签名。如果验证者当前存在锁定的区块，则优先签名预投票之前锁定的区块；如果收到提议区块，则预投票此区块；如果没有收到提案或者无效提案，则签名一个特殊的空预投票（nil prevote），在本环节不会出现上锁。

3. 在预提交环节，如果验证者收到了超过 2/3 的预投票，则签署并且广播对这个区块的预提交，并且锁定这个区块同时释放前置区块，每个验证者在任一时刻只能锁定一个区块，如果验证者收到了超过 2/3 的空预投票则仅仅解锁前置区

块。如果某个验证者收到了超过 2/3 的预提交投票，则进入提交环节，否则继续一轮预提交。

4. 在提交环节，验证者提交的时候必须保证该区块还没有被提交过，一旦某个验证者成功提交，则将 CommitTime 设置成当前时间，并且进入下一个高度的共识回合。

假设少于 1/3 的验证者是恶意节点（存在拜占庭行为），Tendermint 仍然可以保证系统安全，也就是说，验证者永远不会在同一高度提交冲突的区块（即不会出现分叉）。为此，算法引入了一些锁定规则：

- 验证者一旦预先提交了一个块，就将其锁定在该块上;
- 验证者必须为锁定的块预投票，如果在以后的回合中有一个区块获得超过 2/3 的预投票，它只能解锁并预先提交新区块。

3.2.2 验证节点

用户通过提交一个 bond transaction 成为验证者，这个过程会锁住其部分资产（押金），如果想要解锁，则提交一个 unbonding transaction，经过一定时间返还其资产。

在共识过程中，一旦验证节点出现双重签名的情况，则会被扣除押金。

3.2.3 和 PBFT 比较

不难发现 Tendermint 和 PBFT 有一些相似之处，都包括多个阶段的通信，并且两者都在弱同步的网络环境下可以保证安全性和活性（liveness）。但 Tendermint 没有视图变换过程，并且可以支持更大规模的网络环境（例如超过 100 个节点）。

3.3 P2P 网络

P2P 网络是节点之间共识和信息传递的通道，是 YeeZChain 的网络基础。在 P2P 网络中，节点连接分为出站连接（outbound）和入站连接（inbound）两种模式。

出站连接模式下，节点在启动的时候根据指定的种子（seeds）尝试连接所有的 seed 节点，或者根据初始化的 PEX reactor 定期检测出站节点数量，如果少于阈值（默认为 10），就从本地保存的节点连接。节点连接成功后，PEX reactor 发现本地保存的节点小于阈值（默认 1000）会向对端节点获取节点，然后保存在本地。

YeeZ 科技 (YeeZ Tech)

入站连接模式下，节点启动的时候监听连接，有其他节点接入的时候则会加入网络。连接存在上限（默认为 50），一旦达到上限则会忽略新的连接。

P2P 模块初始化的时候，会初始化不同的 reactor（例如 mempool reactor、blockchain reactor 和 PEX reactor 等，）然后加载到 P2P 模块，上述 reactor 利用 P2P 模块的通道进行消息传输。不同通道之间通过 gRPC 远程调用服务实现通信，其中 gRPC 服务采用 protobuf3 进行数据的序列化和反序列化，能够确保数据的完整和传输的高效和安全。

3.4 ABCI

YeeZChain 将区块链应用状态与底层共识进行了分离，如上一章节介绍，共识算法和 P2P 网络层被封装组成共识引擎，因此整个共识引擎只需要负责：

- 在节点之间共享块和事务
- 建立规范/不变的交易顺序（区块链）

这样的好处是，对于上层的逻辑可以模块化为状态机应用，因此应用逻辑可以采用任何确定性的编程语言编写，而无需考虑底层共识和网络的实现。以最基本的转账应用为例，其只需要负责：

- 维护账户数据库
- 验证交易的加密签名
- 防止交易花费不存在的交易
- 允许客户端查询账户数据库

而应用程序流程和共识流程之间通过非常简单的 API（即 ABCI）来进行交互，ABCI 基于 protobuf 定义，它们从共识引擎传递到上层应用，应用程序则以相应的响应消息进行回复。目前 YeeZChain 中的 ABCI 主要包括了两类通道：

- 共识通道：由共识机制驱动，并且负责区块同步执行，主要传递 `InitBlockchain`、`BeginBlock`、`EndBlock`、`DeliverTx` 和 `Commit` 五种消息；
- 内存池通道：负责验证未打包的交易，传递 `CheckTx` 消息；

这里介绍最重要的三种消息：

溯科技（YeeZ Tech）

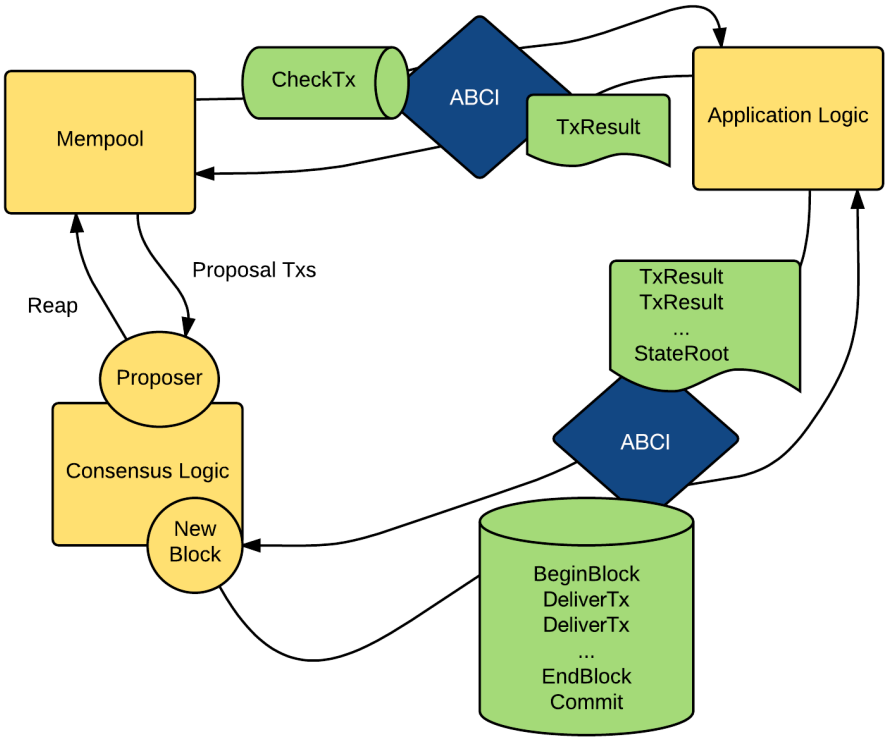

图 4: ABCI 消息流

- `DeliverTx` 消息是上层应用的信息来源，区块链中的每笔交易都带有此消息。应用程序针对当前状态，应用程序逻辑和交易加密凭证基于 `DeliverTx` 消息来验证交易。然后，通过验证的交易会更新上层应用程序状态-例如，更新对应 K-V 数据库中的键值。

- `CheckTx` 消息与 `DeliverTx` 类似，但它只是用于验证内存池中的交易。共识引擎首先基于 `CheckTx` 检查内存池中交易的有效性，然后仅将有效交易转发到其他节点。

- `Commit` 消息包含了当前全局状态的密码学证明，通常会出现在一个区块头之前。因此更新状态的不一致性现在将以区块链分叉的形式出现而直接被底层处理，这样上层应用开发者无需关注状态一致性问题。

确定性的应用逻辑 由于 ABCI 的引入，YeeZChain 能够将共识逻辑和应用逻辑抽分离，从而将整个区块链网络视作一个状态机，因此 YeeZChain 交易处理的逻辑必须是确定性的，如果应用程序逻辑不确定，则节点之间将无法达成共识。理论上而言上层应用可以采用任何确定性的编程语言编写，例如智能合约可以采用 Solidity 这种完全确定性的编程语言，开发者也可以使用现有的流行语言（例如 Java，C ++，Python

或 Go）创建确定性应用程序。但开发人员应该避免诸如以下的非确定性操作构建程序:

- 无确定性种子的随机数生成器
- 多线程
- 未初始化的内存（C 或 C++ 中经常会出现）
- 浮点运算
- 随机的语言功能（例如 Go 中的 map 迭代）

3.5 智能合约

YVM 是 YeeZChain 自主研发的可插拔智能合约引擎通用框架，允许不同智能合约执行引擎接入。为了保证智能合约的确定性，目前 YVM 默认采用 Solidity 兼容的合约虚拟机 yEVM, 之后将继续集成其他虚拟机如 JVM 和 WebAssemblyVM 等。

3.5.1 yEVM

为了最大程度利用开源社区在智能合约技术和经验方面的积累，使得以太坊等区块链上分布式应用可以轻松移植到 YeeZChain 里面，yEVM 重构了 EVM 虚拟机，并且完全兼容 EVM 上开发的智能合约。

在保持 Solidity 开发语言的兼容性基础上，YeeZChain 对 EVM 进行性能优化。首先为了更好的适应联盟链应用场景，yEVM 取消了 gas Limit 限制，因而在 YeezChain 上能够执行更为复杂的应用逻辑。同时 yEVM 增加了对合约部署以及调用权限的管理功能，从而进一步增加了安全性。

同时 YeezChain 支持 Truffle、Remix 等主流智能合约开发部署工具，提供 RPC JSON & REST 调用接口，兼容 Web3 标准。

3.5.2 合约部署升级

YeeZChain 支持基于工厂模式的合约部署，使用工厂模式将合约模版化是指每一个合约都通过调用智能合约相应的方法创建新的实例，用于取代每一个合约的实例都重新部署的方法。该方法的优势在于，每一个合约的实例都是经过验证的，而不需要担心每次部署智能合约可能被篡改；其次，大大降低了每个合约实例的部署难度。

随行科技（YeeZ Tech）

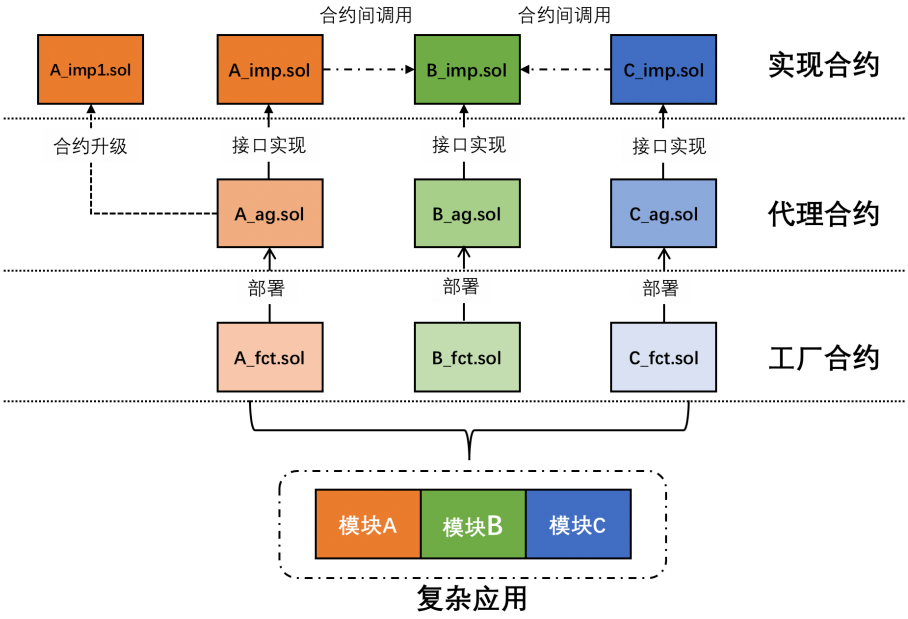

图 5: 智能合约部署升级

同时 YeeZChain 支持合约升级。由于业务的变化或者安全漏洞修复，复杂的智能合约应用必然面对着升级的需求。YeeZChain 基于代理合约可以实现对智能合约进行升级，在面临升级需求时，仅需替换实现合约，而不需要修改相应的代理合约。更进一步的，需要对数据部分和业务逻辑部分进行拆分，以减少数据迁移的开销。

3.5.3 权限管理

在 YeeZChain 中，节点操作智能的权限分为创建（create）和调用（call），只有拥有创建权限的节点才能部署智能合约，而拥有调用权限的节点则可以调用智能合约的方法。

更细粒度地，利用 Solidity 的装饰器语法，YeeZChain 提供了智能合约函数的权限管理模板，开发者可以进一步地对智能合约中不同函数的调用进行权限管理。

4 扩展组件

在基础组件外，YeeZChain 还提供了一些扩展组件和功能。

表 5: JSON-RPC API

方法	功能
yeez_protocolVersion	获取当前 YeeZChain 版本
yeez_syncing	获取当前节点同步状态
yeez_accounts	获取当前节点存储的账户列表
yeez_blockNumber	获取最新的区块数量
yeez_getBalance	获取指定账户的余额
yeez_getStorageAt	获取当前 YeeZChain 版本
yeez_getTransactionCount	获取指定位置上存储的数值
yeez_getBlockTransactionCountByHash	根据区块 Hash 获取指定区块包含的交易数量
yeez_getBlockTransactionCountByNumber	根据区块编号获取指定区块包含的交易数量
yeez_getCode	获取指定合约地址的字节码
yeez_sign	对指定信息进行签名
yeez_sendTransaction	发送交易（未签名），获取交易 Hash
yeez_sendRawTransaction	发送交易（已签名），获取交易 Hash
yeez_call	立即执行新的消息调用，获取调用结果
yeez_getBlockByHash	根据区块 Hash 获取指定区块内容
yeez_getBlockByNumber	根据区块编号获取指定区块内容
yeez_getTransactionByHash	根据交易 Hash 获取指定交易内容
yeez_compileSolidity	编译 Solidity 语言代码，获取编译后的字节码

4.1 API

YeeZChain 提供了若干的接口，方便开发人员和用户访问区块链以及基本应用，用户通过调用 API 可以直接获得特定服务，而无需了解内部工作细节。YeeZChain 中包含了两类 API：JSON-RPC API 和 Web3.JavaScript API。

4.1.1 JSON-RPC API

JSON-RPC 是无状态、轻量级的远程过程调用协议，规范定义了数据结构及其处理规则，数据格式为 JSON，YeeZChain 在 HTTP 和 IPC 上提供了 JSON-RPC 通信服务，基于 JSON-RPC 2.0 接口，可以与节点进行交互，表5列举了比较重要的 JSON-RPC API。

例如，假设要在 YeeZChain 客户端上调用 API 查询账户地址 BE1B4DFE27DAF98178569280847CB867F800A90C 的余额，命令如下：

```
>> curl -X POST --data
    {"jsonrpc":"2.0",
      "method":"yeez_getBalance",
        "params":  ["BE1B4DFE27DAF98178569280847CB867F800A90C",
                    "latest"],
        "id":1}
```

返回结果如下所示：

```
{
 "id":1,
 "jsonrpc":"2.0",
 "result": "0x0234c8a3397aab56"
}
```

4.1.2 Web3.js API

对于 YeeZChain 的开发者，可以使用 Web3.js 库提供的 Web3 对象使应用程序和底层区块链进行交互。通过 RPC 调用 Web3.js 可以与本地节点或者 RPC 层次的任何一个远程节点进行通信。Web3.js API 采用回调函数的形式 `function(object[, callback])`，并且基本都能与 JSON RPC API 对应。

这里给出一个在 JavaScript 中来发起转账的例子：

```
let senderAddress = 'D1293FE65A071A9DB539D64858F0A3D4BCAA2EBA'
let recipientAddress = 'D2349E78DA123790ABCD7620AF390678924DE78B'
let amount = 1000

yeez.transact.SendTxSync({
    Inputs: [{
        Address: Buffer.from(senderAddress, 'hex'),
        Amount: amount
    }],
    Outputs: [{
        Address: Buffer.from(recipientAddress, 'hex'),
        Amount: amount
    }]
})
```

执行上述代码后会返回 `TxExecution`，也就是上述 `SendTx` 交易执行的结果。更多详细的用法参考官方文档。

4.2 yBRE

目前大部分区块链系统（尤其以比特币为代表的公链）的升级往往依赖于节点自发达成共识，而在升级过程中由于节点行为不可控等因素而产生分叉问题，会导致系

熠智科技（YeeZ Tech）

统停止服务、社区分裂以及经济损失等严重后果。尽管 YeeZChain 的共识机制保证了在同一高度不会出现多个区块，但在面临版本升级的过程中，仍然需要解决前后版本兼容、一致性等问题。

前后版本兼容 不同于传统分布式系统，区块链由于保存了所有历史数据，因此底层执行引擎需要兼容所有版本协议，才能保证对区块的同步和验证。举例来说，在某个区块高度 h，由于升级或者安全修复区块链系统的协议表示发生了变化（系统从 A 版本更新到 B 版本），新加入的节点如果直接更新到最新版本 B，则无法同步 h 高度之前的区块。

一致性保证 尽管区块链的共识机制可以保证交易状态的一致性，但在面临系统升级的过程中，却很难保证一致性。在一个出块间隔内，无法保证所有节点同时完成升级，则也会由于协议版本不一致而可能出现分叉的情况。

基于上述问题，YeeZChain 自研了 yBRE, 即自适应升级的区块链运行时环境，通过将系统升级信息封装为特殊交易更新到各个节点从而保证一致性而避免硬分叉和软分叉，同时该运行时环境支持多协议版本的并发执行，最后执行引擎通过软件事物内存保证了并发请求下执行结果的正确性。

同时 yBRE 支持即时编译技术，编译引擎在执行时无需编译、链接过程，可动态地解释。同时将协议转化为可被即时编译引擎直接执行的语言从而大幅提升系统升级效率。此外 yBRE 也支持多协议版本的并发执行，执行引擎通过软件事物内存保证了并发请求下执行结果的正确性。

yBRE 包括了如下模块：

- 协议表示管理模块：负责协议的解析以及协议表示的版本管理，其会解析每一个区块中的交易。当交易类型为系统协议时，协议表示管理模块会解析交易中的数据字段，得到协议源代码以及元数据。协议源代码会交给编译器前端进行处理得到协议表示，后者将交由数据库进行持久化操作。另一方面，当外部向区块链运行时环境请求某个协议的执行结果时，请求会首先通知到协议表示模块。协议表示模块从数据库中拿到对应的协议表示，交由中间表示模块做进一步处理，直到返回协议表示的执行结果，最终，完成外部请求的响应。
- 编译器前端：从协议表示管理模块接收协议源代码，进行语法解析、验证和错误诊断，并将协议源代码翻译成协议表示，最后返回。
- 协议表示数据库：负责持久化不同类型、不同版本的协议表示数据，其中协议元数据以及协议表示以键值对的形式持久化。当协议表示管理模块向数据库请求某协议表示时，协议表示本身以及其依赖的协议表示集合将一起返回。

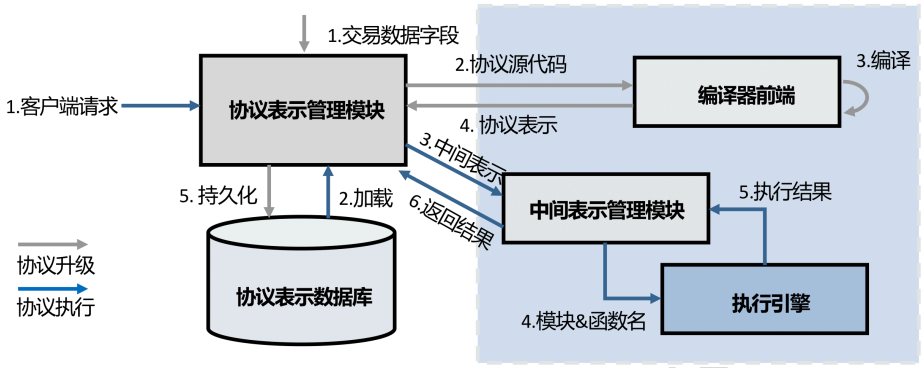

图 6: yBRE 执行流程图

- 中间表示管理模块：负责接收待执行的协议表示集合，对协议表示集合进一步解析以适配执行引擎。首先，中间表示管理模块会插入处理并发的代码到协议表示集合中; 接着，协议表示集合被解析成执行引擎的模块集合，并从模块集合中提取执行的函数入口，交由执行引擎。中间表示管理模块具有管理上述信息的功能，对于已经处理过的协议表示集合，可以快速检索模块集合，避免重复解析处理。
- 执行引擎：负责对即时编译引擎的封装，接收模块集合以及执行函数入口，执行并返回接口。

yBRE 运行流程如图6所示：

4.3 区块链浏览器

区块链浏览器将区块链中的数据可视化，并进行实时展示。方便用户以 Web 页面的方式，获取当前区块链中的信息。区块链浏览器主要显示内容如下：

- 首页：显示内容主要包括最近区块信息，最近交易信息；
- 账户：显示内容包括链上所有账户地址信息；
- 待提交交易：目前正在等待确认上链交易信息；
- 工具：可以发起新的交易以及查看智能合约信息等。

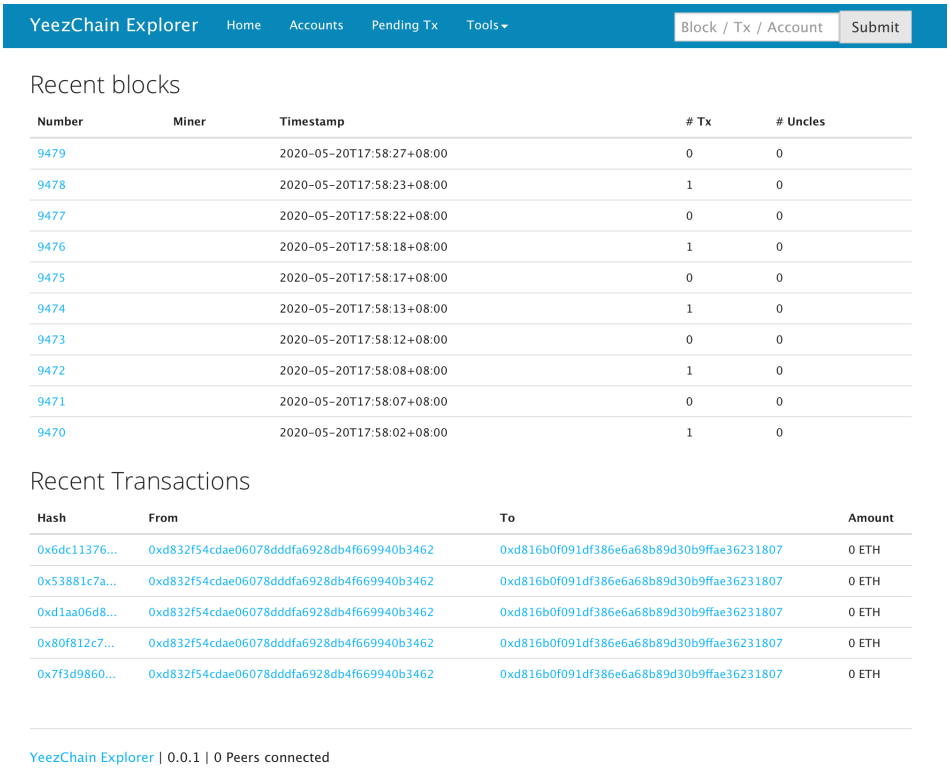

图 7: 区块链浏览器

5 部署及案例

本章介绍 YeeZChain 所需的必要安装和配置。本章通过在部署单节点联盟链以及多节点联盟链帮助用户掌握 YeeZChain 部署流程。

5.1 安装

YeeZChain 提供多种安装方式，包括 Docker 和二进制安装以及一键部署等，这里详细介绍部署流程二进制安装过程，更简单的安装模式可以在官网查看。

5.1.1 环境准备

首先需要安装 Golang，运行 YeeZChain 需要 Go 1.13 及以上版本，并配置环境变量，建议操作系统为 Linux Ubuntu 16.04 及以上或者 MacOS 10.13 及以上。

MacOS 安装 GO

```
# 基于Homebrew安装go
```

```
>> brew install go
# 配置环境变量
>> export GOPATH=/path/to/workspace
```

Linux 安装 GO

```
# 下载
>> wget https://dl.google.com/go/go1.13.linux-amd64.tar.gz
# 解压
>> tar -C /usr/local -xzf go1.13.linux-amd64.tar.gz
# 配置环境变量
>> export PATH=$PATH:/usr/local/go/bin
>> export GOPATH=/path/to/workspace
```

5.1.2 下载安装

在页面下载 YeeZChain 二进制包以及相关工具。将下载好的二进制包 yeez 拷贝至 GO 环境变量目录 $GOPATH 下，或者执行：

```
>> export GOPATH=$GOPATH:<当前yeez路径>
```

之后在命令行执行：

```
>> yeez --version
```

如果返回版本号，表明安装成功。

5.2 部署

YeeZChain 中的节点本质上仍然是一个链上账户，在3.1.2节已经介绍了账户可以具有多个权限，并且可以动态调整。YeeZChain 可以部署的节点类型包括根节点、全节点、验证节点、共识节点和开发节点，所有节点默认都具有转账权限和调用权限，此外每类节点特有的权限如下：

- 验证节点（Validators）：其账户具有共识权限，可以参与区块共识；
- 开发节点（Developers）：其账户具有部署权限，即可以创建智能合约；
- 参与节点（Participant Node）：除转账和调用权限之外没有其他权限；

溯源科技（YeeZ Tech）

- 全节点（Full Node）：具有所有权限；
- 根节点（Root Node）：备选共识节点，即创建新账户以及管理账户权限，可以通过绑定交易成为验证节点；

5.2.1 单机部署

可以直接使用如下命令配置并在单节点上启动 YeeZChain：

```
>> yeez spec -f1 | yeez configure -s- | yeez start -c-
```

上述命令可以拆分为：

```
# 初始化一个全节点
>> yeez spec --full-accounts=1 > genesis-spec.json
# 生成配置文件
yeez configure --genesis-spec=genesis-spec.json > yeez.toml
# 启动
yeez start --config=yeez.toml
```

spec 负责初始化链的描述，configure 则根据描述生成配置文件，YeeZChain 的配置文件文件采用 toml 格式，主要包括了如下部分：

- GenesisDoc：创世区块信息，包括链名称参数、初始账户的信息（账户地址、公钥、初始金额、权限）以及验证节点信息等；
- Consensus Core：共识引擎配置，包括监听地址、端口、Seed 等；
- API 配置：RPC 相关配置，包括 GRPC 和 Web3 接口的地址、端口等；
- 日志配置：日志输出设置

5.2.2 多节点部署

在满足拜占庭容错的情况下，最少需要 3 个节点参与共识，这里给出三个节点部署的示例。在节点 1 上运行一下命令：

```
>> yeez spec -f3 | yeez configure -s- --pool
```

此时在节点 1 的 yeezchain/目录下会生成三个配置文件 yeez000.toml、yeez001.toml 和 yeez002.toml。将 yeez001.toml 和 yeez002.toml 分别放置在节点 2 和节点 3 的对应目录下，在不同节点下分别执行：

(YeeZ Tech)

```
# 节点1
>> yeez start --config=yeez000.toml
# 节点2
>> yeez start --config=yeez001.toml
# 节点3
>> yeez start --config=yeez002.toml
```

5.2.3 共识节点加入 & 离开

通常在初始化链时，会预先配置若干根节点。当需要动态加入共识节点时，只需要发送绑定交易，即可将当前备选根节点绑定成为新的验证节点。

```
>> yeez tx --config yeez001.toml formulate bond --amount 10000 | yeez tx commit
```

或者指定节点账户地址进行绑定：

```
>> yeez tx --config yeez001.toml formulate bond --source <address> --amount 10000
```

当验证节点离开时，同样发送解绑交易即可：

```
>> yeez tx formulate unbond | yeez tx commit
```

5.3 案例

本小节将会介绍一个基于 YeeZChain 的业务应用场景开发全过程，从业务场景分析，到合约的设计实现，然后介绍合约编译以及如何部署到区块链，最后介绍一个应用模块的实现，通过我们提供的 Web3.js API 实现对区块链上合约的调用访问。

5.3.1 节点架构

在本案例中，我们给定有三个机构参与构建的联盟链场景，如图8所示：

企业 A、企业 B 和监管机构分别负责管理一个区块链节点，节点 A 为全节点，具有共识、部署合约、调用合约、管理等权限；节点 B 为验证节点，具有共识、部署合约、调用合约权限；节点 C 为验证节点，具有调用合约的权限。这里假定节点部署网络和核心企业 A 位于同一个网络中，而企业 B 和监管机构的业务网络和节点部署网络不属于同一网络，此时为了便于和机构原有业务系统对接，需要部署前置接入服务器便于跨网络访问区块链节点网络。

溪链科技（YeeZ Tech）

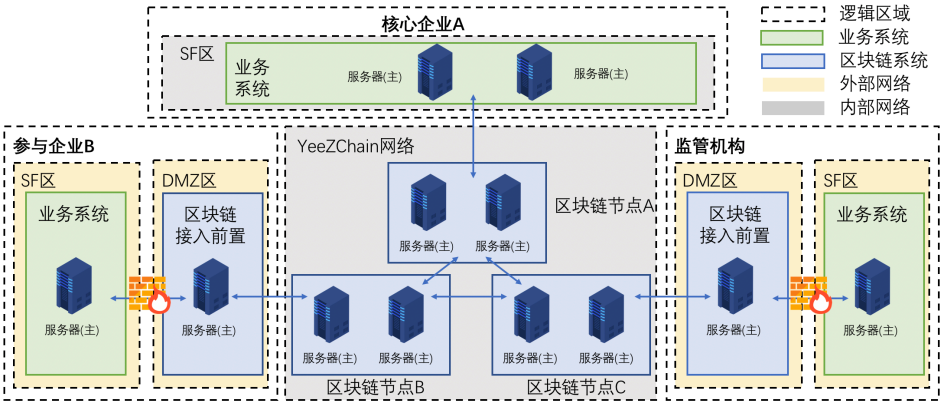

图 8: 案例节点部署

首先每个机构基于本身的公钥和私钥对生成证书，用于互相进行身份认证。节点在启动、运行过程中，需要使用私钥对数据包进行签名，从而完成身份认证过程。假设私钥泄露，则任何人都可以伪装成对应的机构，在不经过该机构授权行使该机构的权利。

在初始化时，机构之间协商创世区块中包含的信息（即 GenesisDoc），可以由机构生成自己的信息，再加入其它机构的节点；或是由核心机构（例如企业 A）直接生成所有机构内的节点信息，并将节点配置文件发送给各机构，相关部署教程可以参考5.2小节。

5.3.2 示例应用开发部署

区块链天然具有防篡改，可追溯等特性，使其非常适用于金融领域，本小节将会提供一个简易的薪资合约管理的开发示例，并最终实现以下功能：

- 在区块链上记录员工账户余额和薪资信息；
- 基于智能合约实现自动批量薪资发放；
- 员工可以查询自己账户的资产；
- 员工（账户拥有者）可以提取不大于余额的金额

数据结构 根据上述需求，这里需要两个数据表，分别用于存储账户的余额和薪资，在智能合约（以 Solidity 为例）中的表示如下：

```
mapping (address => uint256) public balances;
mapping (address => uint256) public salary;
```

接口设计 按照业务的设计目标，需要实现薪资发放、查询、取款等功能，对应功能的接口如下：

```
# 查询余额
function checkBalance(address _addr) public view returns(uint256)
# 查询薪资
function checkSalary(address _addr) public view returns(uint256)

# 修改薪资
function setSalary(address _addr,uint256 _salary)
public returns (bool)
# 发放薪资
function paySalary(address _addr) public returns (bool)
# 取款
function takeMoney(address _addr,uint256 _amount)
public returns (bool)
```

完整的合约代码 `salary.sol` 可以在此处查看。

合约部署 在合约编写完成后，需要一个部署脚本，在 YeeZChain 中，部署脚本基于 yaml 实现，这里给出 `salary.yaml` 的内容：

```
jobs:
        - name: salary_demo
          deploy:
        contract: salary.sol
```

然后通过一下命令可以完成智能合约的部署上链：

```
>> yeez deploy --address $ADDRESS salary.yaml
```

其中 `$ADDRESS` 是合约部署者的链上地址，该命令本质上是发送了一条部署交易，交易执行成功后会生成文件 `salary.out.json`，其中包含了智能合约的地址。

合约部署成功后可通过 YeeZChain 提供的 API 与智能合约进行交互。在企业实际应用场景中，可能存在更为复杂的交互需求，例如基于其他原有业务程序与之进行交互，这里给出基于 Go 工程项目与上述智能合约进行交互的示例。

合约编译 为了使 Go 程序直接调用 Solidity 合约，需要先将 Solidity 合约文件编译为 Go 文件。

(YeeZ Tech)

```
# 编译生成对应的ABI
>> solcjs salary.sol -o $Path --abi
# 编译生成对应的二进制
>> solcjs salary.sol -o $Path --bin
# 基于ABI和BIN生成Go接口
>> abigen --abi salary_sol_salary.abi --bin salary_sol_salary.bin
   --pkg contract --out $Path/salary.go
```

salary.go 提供了与智能合约交互的接口，业务开发者可以方便的在原有业务逻辑中调用。

隐私保护 由于区块链本身的特性，链上所有数据对于任何节点都公开可见，这一定程度上增加了数据的可信度但却丧失了隐私性。许多联盟链（例如 Hyperledger Fabric）通过引入通道（channel）机制尝试解决隐私保护问题。但一方面构建通道的开销会随着节点规模增加而成平方级别增加；另一方面，通道仍然无法满足对于智能合约中数据的细粒度隐私保护（例如对于同一个智能合约中的部分字段限制可见而对于另一部分字段公开化）。

基于上述问题 YeeZChain 与 ASResearch 团队[5]共同研究了“链上隐私数据加密机制”，可以实现对于智能合约中的敏感字段进行加密，并且满足在加密状态下的同态操作，同时可以在不泄露数据的情况下实现单次交互的零知识证明。

以薪资合约应用为例，在原有特性的基础上，隐私保护版本还提供了如下特性：

- 每个员工的薪资和余额在链上处于加密状态，仅自己可见；
- 在薪水和余额均加密的状态下，仍然可以自动完成薪水发放（即实现加密余额和薪水的加同态操作）；
- 员工可以在不泄露私钥的情况下完成取款（即提供范围证明）；

目前已经基于该机制实现了上述薪资合约的隐私保护版本，在此处可以查看，关于更多算法详细介绍可见论文 [?]。

[5]https://asresearch.io/#/index