EducationEKT 修改白皮书 1550633 on 7 May

2 contributors

Executable File 149 lines (82 sloc) 18.8 KB

一、引言

比特币作为第一代加密数字货币，实现了完整的去中心化记账的功能。但由于比特币只提供了非图灵完备的语言以及有限的交易结构，使得比特币只能用于用户价值的存储与交换，无法运行复杂的应用。基于比特币发展出来的区块链技术在最近几年得到了长足的发展。以太坊作为一个智能合约开发平台，不仅引入了图灵完备的开发语言，也把智能合约的应用场景进行了丰富的扩展，ERC20协议（代币发行）以及 ERC721协议（加密猫等不可替代性的数字资产）的广泛应用就是一个典型的案例。以太坊虽然加速推进了区块链的发展，但是由于以太坊早期设计的单一主链架构，在跨链通信、并发能力以及 DApp 的运行环境支持上都有着很大的局限性。为了解决区块链应用的性能，为 DApp 的开发和运行提供更好的支持，我们提出并设计了一套全新的区块链公链架构——EKT，志在打造一个世界级的区块链基础设施，开创一个全新的区块链生态。

二、概述

EKT 是一套全新的区块链架构，定位为一个高性能的 DApp 开发平台。EKT 提供了一套全新的智能合约开发语言 AWM，使得开发者可以很方便的根据自己的业务需求定制自己的智能合约。智能合约的运行环境为 AWM VM， 开发者可以方便的利用该虚拟机进行本地调试，极大的提高开发效率。

EKT 的多链架构提供了发行资产的支持。不同主链的资产通过 EKT 提供的钱包可以自由在整个 EKT 网络里流通。

基于 EKT 的跨链协议，其他公链的资产也可以接入到 EKT 主链并流通。跨链协议打破了目前各个区块链项目之间的隔离，成为了连接各个区块链桥梁。

EKT 主链维护一套统一的用户系统。基于这套用户系统，无论是主链、多链还是 DApp 应用，都可以快速的利用主链已有的用户体系进行开发并获取用户。也极大的降低了用户在不同 DApp 间的切换成本。另外用户可以修改公私钥对以及使用的加密算法，这使得 EKT 的用户体系的安全度会一直跟随时代的发展。即使量子计算机普及，用户也可以相应的把算法更换为抗量子攻击的新算法。

三、共识机制

EKT 采用的共识机制为 DPoS， 在 EKT 主链上共有21个节点维护网络的运转。DPoS 机制的伪代码如下：

```
for round i
    dlist_i = get N delegates sort by votes
    dlist_i = shuffle(dlist_i)
    loop
        slot = global_time_offset / block_interval
        pos = slot % N
        if delegates[pos] exists in this node
            generateBlock(keypair of delegates[pos])
        else
            skip
```

DPoS 共识算法主要分为两个过程：

1. 选举委托人 委托人是可信赖的社区成员，由社区成员通过投票选出。得票率最高的21个委托人成为主链正式委托人，其提供的服务器用户维护 EKT 网络的正常运转。而21名以后的委托人则称为候选委托人，一旦正式委托人发出故障或者作恶，候选委托人可以迅速替换。每个社区的持币用户都可以投票或参与委托人的竞选，投票的票权和持币量成正比。

2. 委托人节点进行区块锻造 在 DPoS 机制里，区块生产的过程称为“锻造”。每一轮选举过程结束后，委托人的服务器节点便会进行区块的锻造。当前网络里未确认的交易会被打包到区块，新锻造的区块会广播给其他节点。每当有一个新的区块添加到主链上，那交易的确认数会增加1。EKT 的区块间隔时间为3s，等到6个确认（18s）以后，便可以认为交易是安全的。

在正常情况下，DPoS 共识机制不会经历任何分叉。区块的生产者之间是合作关系而不是竞争关系。一旦发生区块分叉的情况，主链也会切换到更长的区块链上。如果有节点故意作恶，那该节点也会在下一轮区块生产中被投票出局，不再有锻造区块的权利。

容错

在EKT中，使用公私钥加密和一些路由策略进行容错，每个DPoS节点的公钥都是公开的，具体策略如下：

1. 区块的广播

当一个节点完成打包之后，会对区块进行签名，然后把区块和签名广播给其他的节点。当一个节点收到区块和签名之后会对签名进行校验，确认是从打包节点广播出去的，在确认是打包节点广播出来的区块之后，会判断自己与打包节点在当前轮的距离，如果满足条件 `(currentIndex - miningIndex + len(DPoSNodes)) % len(DPoSNodes) < len(DPoSNodes) / 2` ，则将自己收到的区块和签名广播给其他节点。 当一个节点收到两个不同的打包节点的区块和签名之后，将两个不同的区块和签名发送给所有其他节点，所有节点放弃当前区块，进入下一个区块的打包并对当前打包节点的作恶行为进行记录。

2. 区块的校验与投票

在每个区块头上，都会有区块body的Hash校验值，节点可以向其他节点获取区块body，对body进行处理之后，对当前打包的区块进行投票，所有节点都会把区块的校验结果进行签名，发送给满足 `(currentIndex - miningIndex + len(DPoSNodes)) % len(DPoSNodes) < len(DPoSNodes) / 2` 条件的节点进行唱票。 当任何一个节点收到超过半数对同一个区块的投票之后即可认为当前的区块可写入区块链中，并将区块和投票结果发送给所有的节点，所有节点对区块进行记录。如果投票的数量不足半数则在一定时间内停止唱票，将自己的唱票结果发送给其他节点，所有节点在收到其他节点的投票结果之后对结果进行合并，判断最后的投票结果并执行响应的操作。

3. 节点宕机

当一个节点超过一定时间没有出块，当前轮的下一个节点会在 `3*interval/2` 的时间点开始打包下一个区块，进入下一个区块的打包流程。同理，如果连续节点宕机，判断当前节点是否需要打包的条件是 `currentTime - lastBlockTime > (2*(currentIndex - LastIndex)+1)*interval/2` ，一旦满足当前条件，则当前节点开始打包。如果是最后n个区块连续宕机，则按照当前轮的最后一个区块的hash值判断下一轮的顺序，按照递增每个区块加一个出块interval的算法进行计算，判断当前打包的节点并进行打包。 当超过n/2的节点宕机的时候，所有节点会自动停止出块，直到超过1/2的节点存活。

上面的容错算法可以保障在少于n/2的节点宕机或者叛变的情况下，系统不会出现分叉，是一种用计算资源换容错性的方案。最好情况下：消息复杂度O(n^2), 时间复杂度O(1) 最差情况下：消息复杂度O(n^2), 时间复杂度O(n)

EKT 的主链共识机制为 DPoS，但是其他主链可以通过 EKT 提供的"多链多共识"机制，自由的定制其他共识机制，比如 POW、POS 等。

四、多链架构

EKT 设计了一套独特的多链架构。在这套多链架构中，除了 EKT 的主链外还支持多条并行的主链。每条主链中都会有一个主币。不同的主链可以采用不同的共识机制，默认的共识机制为 DPoS。 开发者可以通过主链提供的 Consensus 接口创建并完成自己节点的部署。

1. 设计思想

在去中心化的网络里，为了提升安全性必然会牺牲系统的整体效率。CAP理论是分布式系统被讨论的最多的理论。其中C代表一致性 (Consistency)，A代表可用性 (Availability)，P代表分区容错性 (Partition tolerance)。CAP理论告诉我们C、A、P三者不能同时满足，最多只能满足其中两个。以数据库系统为例，关系型数据库选择了 C 和 A， NoSQL 选择了 A 和 P。对区块链来说，由于区块链建立在一个不可信的网络节点上，导致在同一时间节点，各个服务器上的数据很难保证完全一致，所以区块链首先放弃了 Consistency。区块链上的数据最终是一致的，比特币通过引入了最长链算法，即所有网络里的节点都只认可最长的那条链。再来看 CAP 理论中的另外两个 Availability 和 Partition tolerance。首先，区块链中的节点通过 P2P 来进行通信，所有的节点地位一致，都可以对外提供服务，所以 Availability 是有保证的。其次，由于引进了共识算法以及最长链算法，最终 Partition tolerance 的问题也得到了解决。所以区块链实现了 AP 优先和数据的最终一致性。

在目前成熟的区块链项目里，比特币系统的 TPS 平均为7. 以太坊的 TPS 平均为10~20。为什么区块链的 TPS 无法达到同样选择 AP 的 NoSQL 一样高的 TPS 呢？对区块链来说，TPS=每个区块包含的交易的数量/区块生产间隔时间。如果想要提高系统的 TPS，就需要提高区块的大小或者缩短区块生产的间隔时间。以比特币来说，由于比特币采用的是 UTXO 模型，使用 Merkle 树把交易信息打包到区块，树的深度是log(n)。如果增加区块大小，会给节点的磁盘带来很大的负担。而如果缩短区块生产间隔时间，在节点数量非常多的情况下，POW 的共识算法也会影响不同节点区块链数据的最终一致性。

EKT 的解决方案是多链多共识机制。对实时性要求比较高的区块链应用，可以采用 DPoS 共识机制。对去中心化要求比较高的区块链应用可以采用 POW 算法。不同的链之间数据是隔离的，互不影响。既增加了安全性，又提高了应用的性能。

2. 账户体系

EKT 主链会提供不同的加密算法供用户使用。在 EKT 主链中，每个用户可以提交公钥以及主链支持的加密算法申请个性化的地址。由于EKT 公链中的用户地址不是通过公钥计算出来的，而是通过申请得到的。用户可以更换地址背后所对应的公私钥对。如果用户更换了自己的公私钥对，那主链以外的其他主链该如何校验用户的签名信息呢？有两种方法可以实现：第一种是如果对安全性要求不是太高，那可以同步主链区块，使用最新的区块来校验用户的公钥，即使同步主链区块略微延时几个区块也不影响。如果用户更新了私钥以后，使用新的私钥对消息进行签名时，其他主链校验不通过时该怎么办？这时可以使用第二种方法：其他主链通过 RPC 调用 EKT 主链的校验方法。将地址、消息和签名发送到主链的节点，节点验证后返回处理结果。一般情况下，如果有(1+n/2)个节点校验通过则可以视为整体验证通过。

3. 资产转移

在 EKT 系统中，不同主链资产的交易和转移是非常简单的。假设用户 A 和用户 B 分别拥有Token T1 和 Token T2，其中 T1 和 T2 分别是不同主链的资产。我们将操作 `用户A 转账一定数量的 T1 给用户 B` 定义为 Tx1，将操作 `用户 B 转账一定数量的 T2 给用户 A` 定义为Tx2，这个时候 Tx1 是在Token T1所在的主链上进行的，Tx2是在Token T2所在的主链上进行的，手续费分别为两个链上的主币。由于所有的链共享了同一套用户系统，所以 EKT 天然支持资产的跨链转移。

一个 Token 只能在一条链上进行打包，资产转移更改的是当前 Token 打包的链。一般来说，在 EKT 上进行跨链资产转移主要出于以下几个原因：当前主链网络比较拥堵、想更换其他主链的共识协议、当前区块的交易费用过高、想自己创建一条链成为主币等等。资产转移时需要指定转移的高度和目标链，资产转移之后也就遵循新的主链的共识机制。

五、跨链

1. 跨链共识

用户共识：两条独立的公链如果想要进行跨链的资产交易，首先需要对用户达成共识，即在一个公链上的地址转移到另外一条公链时需要双方都对对方的公链先进行注册。EKT 提出了一种名为“长地址”的解决方案。在 EKT 系统中存在两种地址：“内部地址” 和 “外部地址”。内部地址就是在EKT系统中使用的地址，用于在EKT主链和其他主链之间转账和其他DAPP的开发使用；外部地址的长度为68字节，前4个字节存储外部地址的公链ID的长度和在其公链上内部地址的长度，后n位存储的是公链id和内部地址，中间用0x00填充。

Token 共识：和用户共识类似，Token 的地址也分为“内部地址”和“外部地址”。内部地址和外部地址的含义参考上一部分。

交易共识：跨公链的交易共识是这样运作的：将 EKT 主链定义为 `PC1` ，ETH定义为 `PC2` 。在 `PC1` 上有一个用户 `user1` ，地址位 `address1` ， 在 `PC2` 上拥有一个用户 `user2` 。在 `PC1` 上有一个Token定义为 `T1` 。如果 `user1` 要将自己在 `PC1` 上拥有的 `T1` 转移给 `PC2` 上的 `user2` ，那么在 `PC2` 上必须已经对 `PC1` 这条公链和 `T1` 进行了注册，这时就有了第一个报文：跨链握手报文。当 `PC2` 对上述报文回复正确之后， `user1` 才可以将自己的 `T1` 转移到 `PC2` 。在此跨公链转账打包成功之后， `PC1` 需要将交易信息和区块头信息发送给 `PC2` 进行校验，这就是第二个报文：跨链交易报文。 `PC2` 校验之后存入自己的区块链中之后需要给 `PC1` 回复第三个报文：跨链确认报文，否则 `PC1` 会继续给 `PC2` 发送跨链交易报文，直到自己收到 `PC2` 的第三个报文：跨链确认报文。给 `user1` 将自己的 `T1` 转移到了 `PC2` 上之后，这些Token在 `PC2` 上怎么转移对于 `PC1` 来说是无关紧要的，因为在 `PC1` 上记录的是将 `T1` 转移到了 `PC2` ，而在 `PC2` 向 `PC1` 发送跨链交易报文的时候， `PC1` 也是不会校验在 `PC2` 上每个地址持有 `T1` 的数量，而是校验在 `PC1` 链中当前的 `PC2` 公链id拥有的 `T1` 的数量。通过这种方式完成了跨公链的资产转移。

2. 报文协议

为了兼容大多数公链，EKT 定义的报文协议以 HTTP 协议为基础，请求和响应的Content-Type必须为application/json，请求参数和响应参数必须与协议中的规定相同。关于跨链协议的具体内容，详见 跨链报文协议。

六、智能合约语言——AWM

为了方便开发者基于 EKT 开发符合自己实际需求的智能合约。我们设计了一门新的智能合约开发语言：AWM。 AWM 是一个事件驱动的语言。事件分为两种类型：用户事件和系统事件。用户事件是指 DApp 客户端的调用。经过 DApp 的共识之后，DApp 客户端会发送一个事件到智能合约。系统事件是指区块完成打包、某个交易处理完成或者智能合约订阅的一些其他事件。另外 AWM 也支持面向对象的编程，开发者可以使用面向对象的思想来进行开发。另外，我们后期还会增加对主流编程语言如Java、nodejs，Python等的支持，方便开发者快速开发自己的DAPP。

1. AWM 的特点

事件驱动。 AWM 是一个事件驱动的语言。它与一般语言(比如C/C++、Java、Golang 等)不同，AWM 没有 main 函数，所有的代码都是围绕事件去执行。

面向对象。AWM 支持面向对象的编程范式。它是一个天然支持模块化设计的语言。开发者可以把一些常用的结构体封装成一个对象，然后在对象里封装一些方法。在事件驱动的函数里可以对对象的属性和方法进行调用。

模块化设计。在 AWM 中，我们鼓励开发者把不同模块的代码放到不同的包中，方便上层调用。

2. AWM VM

我们为 AWM 语言设计了一个执行环境。这个环境就是 AWM VM。 和以太坊的开发语言 Solidity 难以调试不同，AWM VM 可以让开发者方便的在本地运行和调试 AWM。 另外 AWM VM 还会提供 API 来让开发者模拟事件。用户可以很容易的使用 Python、Java 或者 JavaScript 等语言编写测试用例，用来测试智能合约是否可以满足需求。

3. 原生函数

在 AWM 的底层中，有许多基础库可以调用。这些基础库也会不断完善。以 AWM 支持的原生数据库操作函数为例，在 EKT 系统中， 智能合约封装在底层的磁盘操作之上，可以直接操作数据库。AWM 为开发者提供了三种方式的数据库操作：SQL、文档型（类似MongoDB）和KV型。开发者只需要导入响应的包即可完成响应的操作。这些方式可以极大的提高开发者的开发效率，降低开发难度。

七、DApp 开发

基于 EKT 开发的 DApp 分为 DApp 服务端和 DApp 客户端两个部分。DApp 服务器就是智能合约，利用 AWM 语言编写完成，后期也会对主流语言如Java、nodejs、Python等进行支持。DApp 客户端是基于 JavaScript 的一个框架，用户可以根据自己的需求利用这个框架实现客户端的功能。

EKT 在设计智能合约之处就考虑到了现有智能合约的缺陷，比如上手难度大、与业务无关操作较多等。目前大部分公链提供的智能合约都是基于 KV 进行数据的存储，没有对上层的智能合约提供可靠易用的 SDK，门槛很高，上手难度大。通过 EKT 提供的 AWM 智能合约开发语言和 AWM VM 虚拟机，开发者可以只关心业务逻辑，进行基本的学习就可以快速上手。

八、应用场景

1. 电子商务平台

在EKT中，开发者使用AWM可以很方便的开发出一个电子商务平台，如京东、淘宝。与传统电子商务平台不同的是，在EKT公链上开发的DAPP是运行在不同的节点间，而且代码是开源的，任何人都无法篡改数据，所有的数据都是公开透明的，相比较于目前的电商平台对于订单的恶意篡改，相信去中心化的电子商务平台对消费者和一个诚实的商家是一个非常不错的选择。

2. 延迟不敏感的游戏

对于一些延迟不敏感的游戏，EKT可以很方便的提供道具系统和金币系统的支持。DPOS+Paxos的高效可以达到秒级确认，对于一些积分类的游戏相信也会是一个不错的选择。

3. 社交类的软件

相信很多人都想过聊天信息的网络安全问题，这个问题在区块链中是非常容易解决的，因为区块链中本来就是以公私钥加密为基础的一个P2P应用，开发者可以很轻易的对用户的聊天信息进行加密，只有用户本人根据自己的私钥才可以对自己的信息进行解密，得到正确的文本，保证自己的信息安全。

4. 传统的Web应用

EKT的共识机制的秒级确认可以让开发者轻易的开发出一个能支撑很高TPS和DAU的应用，而且相比较于传统的C/S结构，开发者还省去了购买服务器的费用，对于开发者来说，应用上链会是一个潮流，而EKT会是最好的选择。