本文讲述以太坊的p2p通信机制以及以太坊p2p网络协议对应的知识点，致力于为用户带来全面可靠的币圈资讯，希望对你有所帮助！对等网络是区块链的核心技术之一。主要关心的是为区块链提供一个稳定的网络结构，用于广播未打包的事务(事务池中的事务)和一致块。一些共识算法还需要对等网络支持(如PBFT)和另一个辅助功能，如以太坊的消息网络。，也需要对等网络的支持。

P2P网络分为结构化和非结构化网络。结构化网络采用类DHT算法构建网络结构；非结构化网络是一个平面网络，每个节点都有一些相邻节点的地址。

对等网络的主要职责包括维护网络结构和发送信息。网络结构要注意新节点的加入和网络更新，发送信息包括广播和单播

如何建立和维护整个点对点网络。？节点如何加入和退出？

网络结构的建立有两个核心参数，一是每个节点对外连接的节点数，二是最大转发数。新节点对整个网络一无所知。，要么通过一个中心服务让网络中的一些节点连接起来，要么把&quotseed&quot网络中的节点。

新节点加入或退出时的网络更新处理，甚至一些原有节点也可以&#039；由于网络不好而无法连接，过一段时间他们又住在一起了。等等这些情况。通常，这些路由表的变化通过节点的现有连接来广播。需要注意的是，由于对等网络的特殊性，每个节点的路由表是不一样的(也叫局部视图)

广播一般采用洪泛协议。也就是说，接收和转发使得消息在网络中传播。一般来说，应该采用一些限制，例如设置消息的最大转发次数，以避免网络负载过大。

单播需要结构化网络结构的支持，一般是DHT，类似于DNS解析。，逐跳查找目标节点的地址，然后传输并更新本地路由表。

为了快速检索信息，有两种数据结构可以使用。一种是树型，如AVL树、红黑树、B树等。另一个是哈希表。

哈希表比树更高效，但需要占用更多内存。

信息用键-值对来表示，即一个键对应一个值，我们要找的是键，值是附加的信息。

哈希表要解决的问题是如何为每个键平均分配一个存储位置。

有两个重点：1。为键分配一个存储位置，这个分配算法是固定的。保证存储和搜

索时使用相同的算法，否则存储后找不到；2.它是均匀分布的，不应该有多存储数据的一点地方，少存储数据的一点地方。

通用语言中的hashtable、map等结构都是用这种技术实现的，hash函数可以直接使用模函数，key%n，这样n代表有多少位，key是整数。如果这个键是其他类型的，那么它需要首先被散列。，将键转换为整数。这种方法可以解决以上两个要求，但是当n不够大(小于要存储的数据量)时，就会产生冲突，一个地方必须要存储两个密钥。这个时候就需要在这个地方放一个链表。，将被分配到相同的地方，不同的键，并按顺序放置。当一个地方的键太多时，链表的查找速度太慢，要转换成树型结构(红黑树或AVL树)。

如上所述，哈希表效率高，但是占用内容。这个限制可以通过使用多台机器来解决。在分布式环境中，上面的地方可以理解为一台计算机(后称为节点)，即如何将一个键映射到一个节点，每个节点都有一个节点ID，即key-nodeid的映射。这个映射算法也应该是固定的。

这个算法还有一个很重要的要求，就是可伸缩性。当新节点加入和退出时，需要迁移的键应该尽可能少。

这种映射算法有两种典型结构。一个是年轮，一个是树；环叫一致哈希算法，典型的树叫kademlia算法。

选点算法是解决key-nodeid的映射算法，形象地说就是为一个key选择她生活中的她(节点)。。

假设我们使用32个哈希，那么可以容纳的密钥的总数据量就是2**32，这就是所谓的哈希空间。节点的ID被映射成一个整数，键也被映射成一个整数。。键哈希和节点哈希之差称为距离(如果是负数，应该是模，而不是绝对值)。比如一个键的hash是100(整数)，一个节点的hash是105，那么两者之间的距离就是105-100=5。。当然，也可以使用其他距离表示，例如反减法，但算法应该是固定的。我们将密钥映射(放置)到最近的节点。如果距离是模的，就好像把节点和钥匙放在一个环上，从顺时针角度看钥匙属于离它最近的节点。

Kademlia算法的距离用密钥hash和节点hash(整数)异或计算后的数值表示。从左到右，越&quot相同的前缀&quot有，距离越近，左边位置不同，距离越远。

树形结构的体现是将节点和键视为树的节点。该算法支持的位数为160位，即20个八位字节。树的高度是160，每边代表一位。

选择点的算法与一致性哈希算法相同，从所有节点开始。，选择与该键距离最小的节点作为该键的目的地。

因为是在分布式环境下，为了保证高可用性，我们假设没有中心路由表，也没有能看到全貌的路由表，这就带来了一些挑战。，比如怎么找节点，找节点？

在P2P网络中，常见的方法是每个节点维护一个部分路由表，其中只包含部分节点的路由信息。在洪泛算法中，这些节点是随机的；在DHT算法中，该路由表是结构化的，并且维护的节点是选择性的。那么如何合理选择需要维护路由信息的节点呢？

一个简单的办法就是每个节点保存比他大的节点的信息，这样就可以形成一个环，但是这样一来，有一个大问题和一个小问题。大问题是每个节点知道的信息太少(只有下一个节点的哈希和地址)。当一把钥匙给了，它不会&#039；我不知道网络中是否有比它短的节点。所以它首先判断这个键是否属于自己和下一个节点。如果是，则该键属于下一个节点。如果没有，它调用下一个节点的相同方法。复杂度是n(节点数)。一种优化方法由每个节点I维护的其他节点是：I21，I22，I2**31。通过观察这些数据，发现节点从近到远越来越稀疏。这可以降低每个节点保存的其他节点的lgN

信息的复杂度。，包括，从左到右，每比特最多选择k个与本节点不同的节点(算法的超参数)。例如，在节点00110上(选取5位数字进行演示)，要保存的节点的路由信息是

。

1****:xxx，...，xxx(k个)

01:xxx，...，xxx(k个)

000:xxx，...，xxx(k个)

0010:XXX，XXX(k)

00111:XXX，XXX(k)

上面这条线叫k桶。从形象上来说，离自己越近。节点越密越远，节点越稀疏。这种路由查找和节点查找的算法也是lgN复杂度。

Let&#039；s先放一个以太坊的架构图：

在学习过程中，我们主要是用单一的模块来学习和理解，包括P2P和密码学。网络、协议等。直接开始总结：

密钥分发问题也是密钥传输问题。如果密钥是对称的，那么密钥只能离线交换。如果密钥在线传输，可能会被截获。所以我们使用非对称加密，两个密钥。要保存的私钥，公钥。公钥可以在互联网上传输。没有线下交易。确保数据的安全性。

如上图，节点A向节点B发送数据，此时采用公钥加密。节点A获得节点B&#039；的公钥来加密明文数据。，得到密文并发送给nodeB，nodeB用自己的私钥解密。

2。消息篡改无法解决。

如上图，节点A使用B&#039；的公钥进行加密，然后将密文传输给节点b，节点b用节点a的公钥解密密文。

1。因为一&#039；的公钥是公开的，一旦网络黑客截获消息，密文就没用了。说白了，这种加密方式只要截获消息就能解锁。

2。还存在无法确定消息来源和消息被篡改的问题。

如上图所示，节点A在发送数据之前，用B&#039加密；的公钥得到密文1，然后用一个&#039；的私钥来获得密文2。节点B得到密文后，用A的公钥解密得到密文1，再用B的私钥解密得到明文。

1。当数据密文2在网络上被截获时，因为A&#039的公钥是公开的，可以用一个&#039；的公钥来获得密文1。所以看起来是双重加密，但是最后一层的私钥签名是无效的。一般来说，我们都希望签名签在最原始的数据上。如果把签名放在后面，因为公钥是公开的，签名缺乏安全性。

2。有性能问题，非对称加密本身效率很低，加密过程进行了两次。

如上图。节点a用一个&#039；的私钥，然后用b&#039的公钥。收到消息后，节点B首先用B&#039解密；的私钥，然后用一个&#039；的公钥。

1。当密文数据2被黑客截获时，密文2只能被B&#039解密；的私钥。，和B&#039；的私钥只有NodeB拥有，其他人无法保密。所以安全性最高。

2。当节点B解密得到密文1时，只能用A的公钥解密.只有被加密的数据才被A&#039的私钥可以被一个&#039；的公钥。只有节点A具有A的私钥，因此可以确定数据是由节点A发送的.

经过两次非对称加密，性能问题严重。

基于上述篡改数据的问题，我们引入了消息认证。。消息认证后的加密过程如下：

节点A发送消息前，明文数据哈希一次。得到一个汇总，然后和原始数据同时发送给nodeB。当节点B接收到该消息时，它解密该消息。。对哈希摘要和原始数据进行分析，然后对原始数据进行同样的哈希计算得到摘要1，并与摘要1进行比较。如果是一样的，说明没有被篡改；如果不一样，说明被篡改了。

在传输过程中，只要密文2被篡改。产生的哈希和哈希1将是不同的。

签名问题解决不了，就是双方互相攻击。a总是否认他发出的信息。比如A给B发错误信息，导致B亏损。但是否认并没有&#039；不要单独发送。

在(3)的过程中，双方的相互攻击是没有办法解决的。你什么意思？可能是因为A发的消息对节点A不利，后来A否认没有发消息。

为了解决这个问题，引入了签名。。这里，我们将(2)-4中的加密方法与消息签名相结合。

上图中，我们用节点A的私钥对其发送的摘要信息进行签名，然后对原文进行签名，再用节点b的公钥进行加密，而b得到密文后，先用B&#039解密；的私钥，然后用一个&#039；的公钥，并且只比较两个摘要的内容是否相同。这样既避免了防篡改问题，又避免了来自两边的攻击。因为A签署了信息，它可以&#039；不可否认。

为了解决数据非对称加密中的性能问题，通常采用混合加密。这里有必要介绍一下对称加密，如下图：

加密数据时，我们使用双方共享的对称密钥进行加密。对称密钥尽量不要在网络上传输。，这样才不会输。这里的共享对称密钥是根据自己的私钥和对方计算的&#039；的公钥，然后应用对称密钥来加密数据。当另一方收到数据时，它也计算对称密钥并解密密文。

上面的对称密钥不安全。因为A的私钥和B的公钥一般在短时间内是固定的，所以共

享的对称密钥也是固定的。为了增强安全性，最好的方法是为每次交互生成一个临时共享对称密钥。那么我们如何在每次交互过程中生成一个随机的对称密钥呢？还有唐&#039；不需要传输？

那么如何生成随机共享密钥进行加密呢？

对于发送方A节点，每次传输都会生成一个临时非对称密钥对。然后，根据节点B的公钥和临时非对称私钥，可以计算出一个对称密钥(KA算法-密钥协商)。然后使用对称密钥加密数据。对于共享密钥，这里的流程如下：nodeB的

。当接收到传输的数据时，解析节点A的随机公钥，然后使用节点A的随机公钥和节点B的私钥计算对称密钥(KA算法).然后使用对称密钥对数据进行加密。

对于上述加密方法，实际上还有很多问题，比如如何避免重放攻击(给消息添加Nonce)和彩虹表(参考KDF机制)。由于时间和能力有限，暂时忽略。

那么我们应该使用哪种加密方式呢？？

主要是基于要传输的数据的安全级别。其实不重要的数据经过认证和签名就够了，但是非常重要的数据需要用更高的安全级别加密。

密码系统是网络协议的一个概念。。主要包括身份认证、加密、消息认证(MAC)和密钥交换算法。

在全网的传输过程中，根据密码套件，主要有以下几种算法：

密钥交换算法：如ECDHE、RSA等。。主要用于客户端和服务器握手时的认证。

消息认证算法：如SHA1、SHA2、SHA3。主要用于消息摘要。

批量加密算法：比如AES，主要用来加密信息流。

伪随机数算法：比如TLS1.2的伪随机数函数，利用MAC算法的哈希函数创建一个主密钥——来连接一个双方共享的48字节私钥。当创建会话密钥(例如，创建MAC)时，主密钥充当熵源。

在网络中，消息的传输一般需要在以下四个阶段进行加密，以保证消息的安全可靠传输。

握手/网络协商阶段：

在双方握手阶段。，需要链路协商。主要的加密算法有RSA的

认证阶段，DH，ECDH等。

认证阶段，需要确定发送消息的来源。。主要的加密方法有RSA，DSA，ECDSA(ECC加密，DSA签名)等。

消息加密阶段：

消息加密是指对发送的信息流进行加密。。主要的加密方法包括DES、RC4和AES。

消息认证阶段/防篡改阶段：

主要是保证消息在传输过程中没有被篡改。。主要的加密方法包括MD5、SHA1、SHA2、SHA3等。ECC:椭圆曲线加密，椭圆曲线加密。它是根据椭圆上的点的乘积来生成公钥和私钥的算法。用于生成公钥和私钥。

ECDSA:用于数字签名，是一种数字签名算法。有效的数字签名使接收者有理由相信消息是由已知的发送者创建的。这样发送者就不能否认消息已经被发送(验证和不可否认)并且消息在传输过程中没有被改变。ECDSA签名算法是ECC和DSA的结合。整个签名过程类似于DSA，不同的是签名中采用的算法是ECC。，最终的签名值也分为r，s.主要用于身份认证阶段。

ECDH:也是基于ECC算法的霍夫曼树密钥。通过ECDH，双方可以在不共享任何秘密的情况下协商共享秘密。而这个共享密钥是为当前通信临时随机生成的，一旦通信中断就会消失。主要用于握手协商阶段。

种类：是一种集成加密方案，也可以称为混合加密方案。，它针对选定的明文和选定的密码文本攻击提供语义安全性。ECIES可以使用不同类型的函数：密钥协商函数(KA)、密钥导出函数(KDF)和对称加密方案(ENC)。，哈希函数(hash)，H-MAC函数(MAC)。

ECC是一种椭圆加密算法，主要描述如何根据公钥和私钥生成一个椭圆，并且是不可逆的。ECDSA主要使用ECC算法做签名。，ECDH是如何通过ECC算法生成对称密钥的。以上三者都是ECC加密算法的应用。在现实场景中，我们经常使用混合加密(对称加密、非对称加密、签名技术等。)。ECIES是由底层的ECC算法提供的集成

(混合)加密方案。它包括非对称加密、对称加密和签名功能。

metacharset="utf-8"

这个先决条件是确保曲线不包含奇点。

因此，随着曲线参数A和B的不断变化，曲线也呈现出不同的形状。比如：

非对称加密的所有基本原理基本上都是基于一个公式k=kg，其中k代表公钥，k代表私钥，g代表选定的基点。。非对称加密的算法是保证公式不可逆序(即不能计算G/K)。*

ECC如何计算公钥和私钥？这里我根据自己的理解描述一下。我明白。ECC的核心思想是在曲线上选择一个基点G，然后在ECC曲线上随机选择一个点K(作为私钥)，然后根据KG计算出我们的公钥K.并确保公钥k也在曲线上。*

那么kG怎么算呢？？如何计算kG才能保证最终结果不可逆？这就是ECC算法要解决的问题。

首先，我们随机选取一条ECC曲线，a=-3，b=7，得到如下曲线：这条曲线上的

。我随机选取两个点，如何计算这两个点的乘积？我们可以把问题简单化，乘法可以用加法来表示，比如2^2=2^2，3^5=5^5^5。那么只要能算出曲线上的加法，理论上就能算出乘法。因此只要在这条曲线上可以计算加法，那么乘法就可以从理论上计算，表达式k*G的值也可以从理论上计算。在

曲线上加两个点怎么样？这里为了保证不可逆，ECC定制了曲线上的加法系统。

现实中1^1=2，2^2=4，但是在ECC算法中，我们是不可能理解这个加法系统的。所以需要定制一套适合这条曲线的加法系统。

ECC定义，在图形中随机找一条直线。，与ECC曲线相交于三个点(也可能是两个点)，分别是P、Q和R。

那么PQR=0。其中0不是坐标轴上的一个点，而是ECC中的一个无穷远的点。也就是说，无穷远点定义为0点。

同样，我们可以得到PQ=-Rr，由于R和-
R关于X轴对称，所以我们可以求出它们在曲线上的坐标。

PRQ=0，所以PR=-Q，如上图。

以上描述了在ECC曲线世界中如何执行加法运算。

从上图可以看出，直线和曲线只有两个交点，也就是说直线就是曲线的切线。在这一点上，p和r重合。

即P=R，根据上述ECC加法系统，PRQ=0，所以我们可以得到PRQ=2PQ=2RQ=0

。

所以我们得到2P=-Q(是不是越来越接近我们非对称算法的公式K=kG)。

于是我们得出一个结论，我们可以计算乘法，但是我们只能在接触到点的时候计算乘法，而且只能计算2的乘法。。

如果2可以变成任意数进行乘法运算，那么它就可以在ECC曲线中表示乘法运算，那么ECC算法就可以满足非对称加密算法的要求。

那么我们可以随机计算任意数的乘法吗？答案是肯定的。即点积计算法。

选一个随机数k，那么k*P是什么？

我们知道在计算机世界里，一切都是二进制的。既然ECC可以计算2的乘法，我们就可以把随机数k描述成二进制，然后再计算。假设k=151=10010111

由于2P=-Q，因此计算出kP。这就是点乘算法。因此，在ECC的曲线系统下可以计算乘法，因此认为这种非对称加密方法是可行的。

至于为什么这个计算是不可逆的。这需要大量的推导，而我不&#039；我不明白。但我觉得可以这样理解：

在我们的手表上，一般都会有一个时间刻度。现在如果我们以1990年1月1日0:00:00为起点，如果我们告诉你，起点已经过去了整整一年，那么我们就可以计算出出现的时间，也就是我们可以将时针、分针、秒针指向手表上的00:00:00。但是反过来。我说我手表上的时针现在指向00:00:00。你能告诉我从起点到现在已经过去多少年了吗？

ECDSA签名算法和其他DSA、RSA基本相似，都采用私钥签名和公钥验证。。只是

算法系统采用了ECC算法。交互的双方应该采用同一套参数系统。签名原理如下：

选择曲线上的一个无穷远点作为基点G=(x，y)。随机取曲线上的一个点k作为私钥。，K=k*G计算公钥。

签名流程：

生成随机数R，计算RG。

根据随机数r，消息M的哈希值H，私钥k，签名S=(Hkx)/r

消息M，RG，S发送给接收方。

签名验证流程：

收到的消息M，RG，S

计算哈希值H

根据发送方计算HG/SxK/S&#039；s公钥k，并将计算结果与RG进行比较。如果它们相等，则验证成功。

公式推导：

Hg/sxk/s=Hg/sxk(kg)/s=(hxk)/GS=rg

在介绍原理之前，让&#039；说明ECC满足结合律和交换律。也就是说ABC=ABC=(AC)B.

这里&#039；这是一个来自WIKI的示例，用来说明如何生成共享密钥。也可以参考爱丽丝和鲍勃的例子。

如果Alice和Bob要通信，双方的前提是ECC基于相同的参数体系生成的公钥和私钥。因此，所有ECC都有一个公共基点g.

密钥生成阶段：

Alice使用公钥算法ka=ka*G生成公钥ka和私钥ka，并将公钥KA公之于众。

Bob使用公钥算法kb=kb*g。，生成公钥kb和私钥KB，并将公钥KB公开。

ECDH阶段的计算：

Alice使用计算公式Q=ka*KB计算一个密钥Q

Bob使用公式Q&#039=kb*KA来计算密钥Q&#039。

共享密钥验证：

Q=KAKB=KA*KB*G=KA*G*KB=KA*KB=KB*KA=Q&#039；

因此，双方计算的共享密钥可以被Q加密而不被泄露。我们称q为共享密钥。

以太坊采用的ECIEC加密套件中的其他内容：

1。采用最安全的SHA3算法Keccak作为哈希算法。

2。签名算法是ECDSA

3。认证方法是H-MAC

。

4。ECC的参数系统采用secp256k1。其他参数系统请参考此处的

。

ThewholeprocessofH-MACiscalledmessageverificationcodebasedonhash.Itsmodelisasfollows:

在以太坊的UDP通信中(RPC通信的加密方式不同)，采用并扩展了上述实现方式。

首先以太坊中UDP通信的结构如下：

其中，sig是用私钥加密的签名信息。Mac可以理解为整个报文的总结，ptype是报文的事件类型，data是RLP编码的传输数据。

其UDP的整个加密、认证和签名模型如下：

节点发现功能主要涉及服务器表udp等几种数据结构，它们有各自的事件响应周期，节点发现功能是由它们协作完成的。在...之中启动后，每个以太网客户端都会在本地运行一个服务器，将网络拓扑中的相邻节点视为节点，而Table是节点的容器，udp负责维护底层连接。。下面重点介绍事件循环处理的重要字段和关键部分。

私钥该节点的私钥，用于与其他节点建立连接时的握手协商

协议所有支持的上层协议

静态节点默认静态对等体。当节点启动时，它们将首先发起连接以建立邻居关系

。

新传输较低的传输层实现，它定义了握手过程中的数据加密和解密方法。默认的传输层实现是用newRLPX()创建的rlpx，这不是本文

的重点。

ntab典型的实现是Table，所有的peer都以nodes的形式存储在Table

ourhandshake中与其他节点建立连接时的握手信息。，包含本地节点的版本号和支持的上层协议

addpeer——连接握手完成后，连接进程通过此通道通知服务器

监听周期。，启动底层监听套接字，收到连接请求时，接受后调用setupConn()启动连接建立过程

服务器的主要事件处理和功能实现循环

。

节点唯一表示网络上的一个节点

IP-IP地址

UDP/TCP-UDP/TCP端口号用于连接

。

ID在以太网中唯一标识的节点，本质上是一个椭圆曲线公钥，对应于服务器的私钥。节点的IP地址不一定是固定的，但是ID是唯一的。

[XY001]SHA-用于计算节点间的距离

表主要用于管理与本节点和其他节点

的连接的建立、更新和删除。

桶-所有对等体根据它们与该节点距离被放置在不同的桶中。详见节点维护

刷新请求-更新表请求通道

。

Table的主事件循环主要负责控制刷新和重新验证过程。

refresh.c以固定间隔(30秒)启动对等刷新进程的计时器

refreshreq接收其他线程向表提交的刷新对等连接的通知，收到通知后开始更新。有关详细信息，请参见稍后更新邻居关系

。

重新验证。c-定期重新检查连接节点有效性的计时器。具体请参考下面的探针检测

udp负责节点间通信的底层消息控制。，也就是表运行的Kademlia协议的底层组件

conn——底层监听端口

的连接。

addpending-UDP用于接收挂起的通道。使用场景是，当我们向其他节点发送数据包时，我们可能希望收到它的回复。，pending用来记录这个还没有到达的回复。例如，当我们发送ping数据包时，我们总是希望对方回复pong数据包。此时，您可以构造一个挂起的结构。，它包含预期的pong包和相应的回调函数的信息，并将该丁鹏传递给udp的该通道。Udp在收到匹配的pong后执行预设的回调。

gotreplyUDP用于接收来自其他节点的回复。有了上面的addpending，在收到回

复后，遍历已有的pending链表，看看是否有匹配的pending。

服务器中的

Table和ntab是Table

udp的同一个处理周期，负责控制消息的向上提交和收发，控制

udp的底层接受包周期。，其负责从其他节点接收分组

。以太坊使用Kademlia分布式路由存储协议来维护网络拓扑。建议先阅读并理解分布式协议。更权威的信息可以在维基上找到。。一般来说，协议：

源代码以表结构保存所有桶，桶结构如下：

节点在条目和替换中可以相互转化。如果Validate失败，entries节点将被替换数组中原来的节点替换。

有效性检测是使用ping消息来探测活动。。Table.loop()启动定时器(0~10s)，定时随机选择一个桶，向其条目中的最后一个节点发送ping消息。如果对方响应pong，则探测成功。

Table.loop()会定期(定时器到期)或不定期(收到refreshReq)更新邻居关系(发现新邻居)，两者都调用doRefresh()方法。这种方法对于在网络上找到几个离自己最近的节点和三个随机节点很有用。

表格&#039；slookup()方法用于查找目标节点，其实现是Kademlia协议，通过节点间的中继。一步一步，接近目标。

当节点启动时，它将首先启动与已配置静态节点的连接。发起连接的过程称为拨号，这个过程通过在源代码

中创建dialTask来跟踪。

dialtask是指服务器启动时，任务

一次性发起与其他节点的连接。，调用newDialState()根据预先配置的StaticNodes初始化一批dialTask，这些任务在Server.run()方法中启动。

拨号进程需要知道目的节点(dest)的IP地址。如果你不&#039；如果不知道，您必须首先使用recolve()来解析目的地的IP地址。如何化解？？就是先用Kademlia协议在网络中找到目标节点。

当获得目标节点的IP后，下一步就是建立连接，就是通过dialTask.dial()

建立连接。

连接建立的握手过程分为两个阶段。第一阶段是在SetupConn()中实现

时的ECDH密钥建立，第二阶段是协议握手和相互交换支持的上层协议

。

如果两次握手都通过，dialTask会将对方的信息发送到服务器的addpeer通道

首先，它会回答你什么是以太坊，以太坊是一种编程语言，是一个平台。但投资者/投机者关注的以太坊，其实就是以太坊，也就是一种由以太坊(Ethereum)衍生出来的数字代币

以太坊是否有投资前景，我们只需要看它的价值，从几十元到几百元不等。且不说未来泡沫与否，但现阶段的升值空间还是值得大家&#039；注意！

以太坊交易平台。目前推荐btctrade平台(比特币交易网)，这是国内比较靠谱的大型交易平台！2016年在以太坊上线。币价涨得惊人！

以太坊是将比特币中的一些技术和概念应用到计算领域的创新。比特币被认为是一个维护共享账本的系统，共享账本安全记录所有比特币账单。。以太坊使用许多类似比特币的机制(如区块链技术和P2P网络)来维护一个共享的计算平台，可以灵活安全地运行用户想要的任何程序(当然也包括类似比特币的区块链程序)。

那&#039；这一切都是为了在以太坊中引入p2p通信机制。感谢您花时间阅读本网站的内容。唐&#039；别忘了在以太坊中查找更多关于p2p网络协议和p2p通信机制的信息。