以太网的目的是基于脚本、竞争货币和链上元协议(on-chain meta-protocol)的概念进行集成和改进，开发人员可以创建基于任何共识的可扩展、标准化、特性完善、易于开发、协作APP。 以太坊构建了最终抽象的底层——区块链——包含图灵完整的编程语言，使每个人都可以创建合同、创建中心化APP，并在其中设置可自由定义的所有权规则、交易方法和状态转换函数。 域名货币的主体框架可以用两行代码实现，货币和信用系统等其他协议可以用少于20行代码实现。 智能约定-有价值，只有在满足特定条件时才能打开的加密箱子-
也可以在我们的平台上创建并且，由于图灵完整性、价值认知(value-awareness
)、区块链认知(blockchain-awareness
)、多状态而增加的力度，比特币脚本能够提供的智能合约

以太网帐户

在以太网系统中，状态由被称为“帐户”的对象和在两个帐户之间移动价值和信息的状态转移构成。 以太坊的帐户包括四个部分：

随机数、用于确定每次交易只能处理一次的计数器

账户的当前以太网货币余额

账户的合同代码，如果有，则为

账户的存储[xy001]以太网(ether
)是以太网内部的主要加密燃料，用于支付交易费用。 一般来说，以太网有两种类型：所有外部帐户(由私钥管理的帐户)和合同帐户(由合同代码管理的帐户)。
外部的所有帐户都没有代码，人们可以通过创建交易并签名从外部帐户发送消息。
每当合同帐户收到消息时，合同内部的代码都会被激活，可以读取和写入内部存储，发送其他消息，或者创建合同。

消息和交易

以太坊的信息在一定程度上类似于比特币交易，但两者之间有三个重要区别。
首先，以太网消息可以由外部实体或合同创建，但只能从外部创建比特币交易。
其次，以太网消息可以包含数据。
第三如果以太网消息的收件人是合同帐户，也可以进行响应。
这意味着以太网消息还包含函数的概念。

以太光的“事务”是存储从外部账户发送来的消息的签名包。 事务包括两个数字：消息收件人、确认发件人的签名、以太网帐户余额、要发送的数据以及开始gas和G

ASPRICE。 为了防止代码的指数爆炸和无限循环，每个事务都必须限制执行包含初始消息和所有执行过程中出现的消息的代码所导致的计算步骤。
STARTGAS是一个限制，GASPRICE在每个计算步骤都需要支付矿工的费用。 交易执行中“煤气用完了”，所有状态都会恢复到原来的状态，但已经支付的交易费用无法回收。 如果交易中止时有气体残留，这些气体将返还给发货人。
创建合同有单独的事务处理类型和相应的消息类型。
合同地址是根据帐户随机数和交易数据的哈希计算的。

消息机制的一个重要结果是以太坊“头等舱公民”的财产，合同与外部账户拥有相同的权利包括发送消息或制作其他合同的权利。
这样，合同可以同时发挥多个不同的作用。 例如，用户可以将一个合同的中心化组织的成员作为中介帐户(另一个合同)另外，为使用偏执地使用基于量子证书定制的朗博尔签名(第三方协议)的个人和使用五个私钥安全帐户(第三方协议)的协同签名实体提供中介服务。 以太网平台的强大之处在于，去中心化组织和代理协议，无需在意协议的每个参与者是什么类型的账户。

以太体的状态转移函数

以太体的状态转移函数： APPLY(S，tx)-s，可以定义为检查

事务的格式是否正确，即是否有正确的数字、签名是否有效以及随机数是否与发件人帐户的随机数匹配。 不，返回错误。

计算交易费用：fee=STARTGAS * GASPRICE，根据签名确定发件人的地址。
从发件人帐户中扣除交易费用或增加发件人的随机数。
如果账户余额不足，则返回错误。

设定初始值GAS=STARTGAS，从交易中的字节数中减去一定量的气体值。

将价值从发件人帐户转移到收件人帐户。 如果接收帐户尚不存在，请创建此帐户。
如果接收账户是合同，执行合同的代码，直到代码执行完毕或没有气体。

如果发件人账户中没有足够的钱，或者执行代码时瓦斯耗尽导致价值转移失败，则恢复到原始状态，但还需要支付交易费用，交易费用将被加到矿工账户中。

否则，将剩余瓦斯全部返还给发货人，将消耗的瓦斯作为交易费用发送给矿工。
例如，假设合同的代码如下所示：

if ！ self.storage [call data load [0] : self.storage [call data load [0]=call data

load [32] [xy 002] [xy001
]需要注意，现实中合同代码是用基本以太网虚拟机(EVM)代码编写的。
上述合同用我们的高级语言Serpent语言编写，可以编译成EVM代码。 假设合同内存最初为空，值为10以太网，燃气为2000，燃气价格为0.001以太网且64字节的数据，前32字节快代表号2和第二代表字CHARLIE。
的事务发送后，状态转移函数的处理如下。 检查

事务是否有效，格式是否正确。

检查交易的发送者至少有2000*0.001=2个以太网货币。
在某些情况下，从发件人帐户中扣除两个以太网货币。

初始设定为gas=2000，假设交易长度为170字节，则每字节的费用为5，扣除850所以，还剩1150。

从发件人帐户中减去10个以太网货币，向合同帐户添加10个以太网货币。

执行代码。 在这个合同中，执行代码很简单。 检查是否有两个地方使用了合同内存索引，并注意到没有使用然后将其值设置为CHARLIE。
假设这消耗了187单位的气体。 于是剩下的气体变成1150 - 187=963。 6
.将963*0.001=0.963个以太网货币添加到发件人帐户，然后返回到最终状态。 如果没有接收交易的合同，则所有交易费用都等于GASPRICE乘以交易的字节长度，交易数据与交易费用无关。 另外，需要注意的是，合同开始的消息可以为它们发生的计算分配气体限额子计算的气体用完后，只需恢复到出现消息时的状态即可。 因此，合同也可以通过对生成的子计算设置严格的限制来保护计算资源，就像交易一样。

代码执行

以太网协议的代码用低级基于堆栈的字节码语言编写，称为“以太网虚拟机代码”或“EVM代码”。 代码由一系列字节组成，每个字节表示一个操作。
一般来说，代码执行是无限循环的中选择所需的族。 每增加一个程序计数器，就运行STOP或RETURN命令，直到代码执行完成或出现错误。
操作可以访问存储三种类型数据的区域：

堆栈、后进先出数据存储32字节的数值进入堆栈，可以从堆栈中出来。

内存，可无限扩展的字节队列。

合同的长期保存。 私钥/数字的存储，私钥和数字都是32字节的大小，与计算结束后重置的堆栈和内存不同存储内容将长期保留。

代码提供对数字的访问权限，就像访问块标题数据一样，并且发送者和接收消息中的数据以及代码还可以返回数据的字节队列作为输出。

EVM代码的正式执行模型简单得惊人。 以太网虚拟机运行时，其完整计算状态可以用元组(block_state、transaction、message、code、memory、stack、pc、gas)定义其中block_state是包含所有帐户余额和存储的全局状态。
在每个循环中，通过调用代码的第pc (程序计数器)字节来找到当前指令。
每个指令都定义了如何影响元组。
例如另外，ADD将两个要素进行堆栈，将它们的和进行堆栈，减少一个gas (气体)而增加一个pc，SSTORE将最上面的两个要素进行堆栈，将第二个要素插入到由第一个要素定义的合同存储位置，同样最多减少200个gas值，有很多方法可以通过即时编译优化以太体，但是以太体的基础实现可以通过数百行代码来实现。

区块链与采掘

存在一些差异，但以太体区块链在许多方面类似于比特币区块链。 这些区块链架构的区别在于以太网区块不仅包含交易记录和最近的状态，还包含区块号和难度值。
以太坊的块确认算法如下。 检查

块参照之前的块是否存在和有效。

检查

块的时间戳是否大于参考的上一个块且小于15分钟。

检查区块号、难度值、交易路线、叔路线和燃气限额(许多以太体特有的基础概念)是否有效。

检查

块的工作量证明是否有效。

将S[0]代入前一块的STATE_ROOT。

将TX分配给区块的交易列表，共有n笔交易。 0
.关于属于n-1的I进行状态转移s[I1]=apply(s[I]，TX[i])。

如果其中一个转换发生错误，或者程序运行到此为止所需的气体(gas)超过了GASLIMIT，则会返回错误。

在

s[n]中为S_FINAL赋值，向矿工支付区块奖金。 检查S-FINAL是否与STATE_ROOT相同。 如果是，则阻止有效。 否则，阻止无效。

此确认方法看起来效率很低，因为必须保存每个块的所有状态，但实际上以太体的确认效率可以与比特币进行比较。 原因是状态保存在树结构中(tree structure)中选择所需的族。 每次添加块时，只需更改树结构的一小部分。 因此，一般来说，两个相邻块的树结构需要相同的大部分，因此可以存储一次数据，并使用指针(即子树散列)进行两次参考。 要实现这一点，有一种叫做“Patricia Tree”的树结构。
它包括修改默克尔树的概念，不仅可以更改节点，还可以插入和删除节点。 另外，其他的，所有状态信息都是最后一个块的一部分，因此不需要保存所有块历史记录。 如果该方法适用于比特币系统，通过计算可以节省10-20倍的存储空间。