**第一课笔记**

**一、课程关键内容回顾**

**（一）序列器，证明器和节点**

**1. 序列器Sequencer**

序列器的角色类似于以太坊的验证者，主要任务是接收用户的transacton，按顺序排列这些transaction，处理这些transaction，然后将它们打包成块。

序列器对transaction处理的方法包括：排序、执行、打包以及生成区块。

序列器需要不断的快速的处理transaction，因此序列器需要强大且可靠的计算基础设施。

Starknet的发展路线包括将序列器的角色去中心化，从而提高网络的鲁棒性，并减少中心化的风险。

**2. 证明器Prover**

证明器也是Starknet一个非常关键的组件，证明器的职责包括：接收区块、处理生成证明、将证明发送到以太坊。

Starknet中的证明器充当二次验证的角色，在序列器处理并批量将transaction打包成区块以后，这些区块会传递给证明器，证明器会重新处理这些区块，以确保所有的StarkNet均以正确执行，然后证明器为每个区块生成证明，并将它们发送到以太坊网络进行验证。

这个步骤的计算量是非常巨大的 ，证明器被设计为处理复杂任务的一种组件，它的工作可以是异步的，证明器的工作可以被分成多个部分，允许并行处理和高效的证明生成，这种灵活性的工作方式使得负载可以在多个证明器之间分配，每个证明器可以处理不同的区块，实现并行性和高效的证明生成。

在这种情况下，以太坊托管了一个能够验证这些Stark证明的智能合约，如果证明有效，stark状态根就会在L1上更新。

**3. 节点Nodes**

StarkNet中的节点与比特币或以太坊区块链的节点相比，有着独特的功能。以太坊中的节点充当网络的审计者，维护着网络的状状态，例如每个参与者拥有多少ETH，或者特定智能合约的当前状态。

节点通过处理transaction，并保存所有的transaction记录来实现这一点，StarkNet节点不一定需要处理所有transaction 来维护这种状态。

通常节点访问网络状态数据的三种主要方法包括：

第一种方法是重放旧的transaction，这种方式类似于以太坊或比特币，节点可以取走所有的transaction，并重新执行它们。尽管这种方法准确，但除非你拥有能够处理这些负载的强大机器，否则它并不可扩展。如果能重放所有的transaction，这就可以作为一个序列器使用了。

第二种方法是依赖L2的共识，所有节点信任序列器能够正确的执行网络，这种方法就涉及到一些信任假设，但是他的资源需求较少。目前Stacknet围绕一个序列器进行，所以这些节点是信任Stackware 不会破坏网络的。

第三种方法是在L1上检查证明并验证，节点呢可以通过观察L1，并确保每次发送证明能收到更新的状态来监控网络的状态，这样他们不必信任任何人，只需要跟踪StarkNet上最新的有效的transac tion就可以。

**4. 节点实现**

**（1）Pathfinder节点**

由Equilibrium开发，Pathfinder是一个用Rust编写的完整节点。它在高性能、可扩展性方面表现出色，并与Starknet的Cairo规范保持一致。

**（2）Juno节点**

Nethermind的Juno，是一个用Golang编写的完整节点，以用户友好、易于部署和与以太坊工具兼容性而闻名。

**（3）Papyrus节点**

StarkWare的Papyrus，一个基于Rust的完整节点，专注于安全性和鲁棒性。它对即将推出的Starknet Sequencer至关重要，预计将提高网络吞吐量。

**（二）Transactions**

**1. Transaction类型**

（1）声明事务（Declare Transactions）：能够引入新的类别，从而可能引入新的智能合约。

（2）调用事务（Invoke Transactions）：可以发起动作，但不能引入新动作。

（3）部署账户事务（Deploy Account Transactions）：专为设置智能钱包合约而设计。

**2. Declare Transactions 版本**

V0版，适用于Cairo 0合约，在引入随机数之前。

V1版，适用于引入了随机数的Cairo 0。

V2版（当前版本），为现代Cairo合约优化。

**3. UDC(Universal Deployer Contract)**



4. **Transaction生命周期**

（1）transaction 首先被发送到一个称为（Mempool）的网关节点。

（2）目前由单一服务的序列器（Sequencer）首先验证并依序执行Transaction。

（3）在证明者（Prover）阶段，系统对新区块进行操作，计算其证明，并将其发送到L1。

（4）在第一层（L1）上的最终确认。

**5. Nonces**

图形用户界面, 应用程序

描述已自动生成

**6. Transaction 创建**

（1）查询他们账户的nonce。

（2）对 Transaction 进行签名。

（3）将其发送至节点。

**7. Rejected Transactions**

（1）检查账户余额是否至少为最高费用。

（2）检查nonce。如果Transaction的nonce与账户预期的下一个nonce不匹配，将导致失败。

（3）此时重复的合约声明会导致Transaction被拒绝。

（4）在网关中限制每个账户的Transaction数量。

**（三）证明者和验证者**

**1. 证明者Prover**

**（1）证明者的角色**

StarkNet中的证明者负责为每一组计算出具证明，这些计算被称为作业。这些证明对于验证网络上的计算至关重要。

**（2）协作处理**

StarkNet中的证明者会聚合多个Cairo程序。这种协作方式大大降低了成本并提高了效率。

**（3）无状态设计**

StarkNet的证明者以无状态的方式运行，从而提供了更大的灵活性和可扩展性。

**2. SHARP（Shared Prover）**

（1）SHARP在整个网络层面上运作，它汇集多个证明以提高效率。

（2）SHARP使用递归证明引专注优化和效率层面。

（3）SHARP对网络的效率和可扩展性有更广泛的影响。

**（四）L3 / Appchain**

**1. Appchain**

Appchain是为特定应用需求量身定制的专业区块链。它们在诸如哈希函数和共识算法等方面提供定制化选项，同时继承了它们的父级第一层或第二层区块链的安全性。

**2. StarkNet的L3的优势**

（1）快速协议迭代；

（2）治理独立性；

（3）经济效率；

（4）增强的安全性和隐私性；

（5）避免拥堵和稳定性；

（6）创新和测试平台。

**4. Madara**

Madara是L3应用链的序列器。

**（五）数据可用性**

**1. 可选模式**

（1）Validity Rollup

在Rollup模式下，账本的状态及其证明都存储在以太坊上。这样做的好处很明显：只通过与以太坊区块链的交互，就可以重建账本的状态。这意味着作为终端用户，即使第二层系统停止运行，你仍然可以无需信任地与以太坊上的相关智能合约进行交互，并取回你的资金。

（2）Validium

在Validium模式下，账本信息不会发送到以太坊，而是会在链下存储。以太坊存储这些账本信息的Root。

（3）Volition

Volition是一种数据可用性架构，它在交易级别上提供了在Validium和Rollup模式之间选择的可能性。开发者可以为每个独立的交易选择Validium模式或Rollup模式，将数据存储在以太坊（L1）或StarkNet（L2）。

**（六）L1-L2消息传递**

**1. 从L2到L1的消息传递**

在StarkNet上，合约可以通过使用send\_message\_to\_L1系统调用向L1发送消息。

在 L1 上，重要的部分是构建与 L2 上相同的有效负载。然后，您通过传递 L2 合约地址和有效负载来调用 ConsumerMessageFromL2。请注意，consumeMessageFromL2 期望的 L2 合约地址是在 L2 上发送交易的账户的合约地址，而不是执行 send\_message\_to\_l1\_syscall 的合约地址。

**2. 从L1到L2的消息传递**

从以太坊向 Starknet 发送消息，Solidity 合约必须调用 StarknetMessaging 合约的 sendMessageToL2 函数。

接收者合约需要实现特定的方法，对于 L1->L2 消息，Starknet 序列器不断监听以太坊上 StarknetMessaging 合约发出的日志。一旦在日志中检测到消息，序列器就会准备并执行 L1HandlerTransaction 以调用目标 L2 合约上的函数。

**3. 消息取消机制**

存在一种机制来减轻风险，比如L2消息消费的故障，通过在延迟后允许取消L1到L2的消息。

**4. 费用机制**

由于L1到L2的消息会在L2产生交易，这个交易费用在L1上支付，并由序列器收取以处理该消息。

**二、作业分析**

**1. StarkNet是什么类型的网络？**

A) 第一层 (L1)

**B) 第二层 (L2)**

C) 第三层 (L3)

D) 独立网络

答案：B

**2. StarkNet中用于聚合多个证明的系统叫做什么？**

A) Stone

**B) SHARP**

C) Madara

D) Volition

答案：B

**3. 在StarkNet中, 负责生成密码学证明的是什么？**

A) Sequencer

**B) Prover**

C) Validator

D) Dispatcher

答案：B

**4. StarkNet的数据可用性解决方案叫做什么？**

A) SHARP

B) Stone

**C) Volition**

D) Madara

答案：C

**5. 在StarkNet中, L1到L2的消息传递是通过哪个合约来实现的？**

**A) StarkNet Core Contract**

B) L1 Messenger Contract

C) L2 Dispatcher Contract

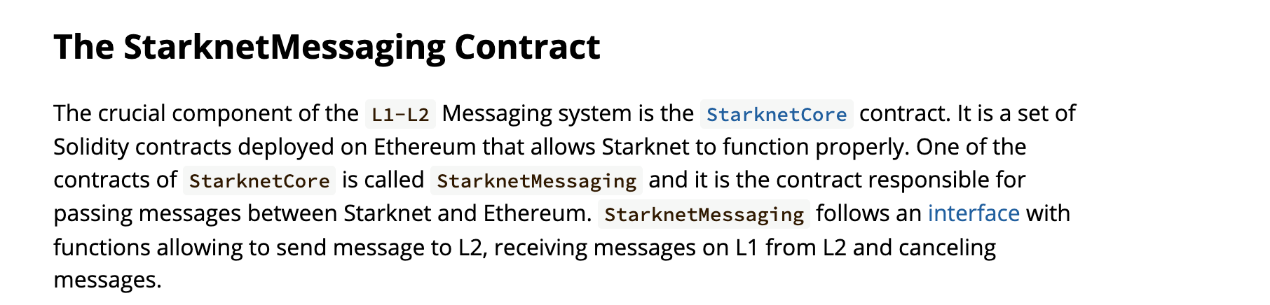
D) StarkNet Bridge Contract

答案：A

参考：

<https://docs.starknet.io/documentation/tools/important_addresses/>

<https://book.cairo-lang.org/ch99-04-00-L1-L2-messaging.html>





**三、学习资料**

**1. L1到L2消息传递的示例合约**

https://book.cairo-lang.org/ch99-04-00-L1-L2-messaging.html

**2. StarkNet社区团队**

https://github.com/keep-starknet-strangehttps://github.com/kkrt-labshttps://github.com/dojoenginehttps://github.com/NethermindEthhttps://github.com/lambdaclass

**3. StarkNet Bridge Contract**

可以理解为一类合约，例如用于桥接同质化或非同质化代币的合约。可以片面的理解为构建在消息传递合约之上的一类合约。