摘要——区块链的点对点（P2P）网络用于传输其交易和区块，对系统的效率和安全性有很大影响。因此，比特币和以太坊等流行区块链的 P2P 网络拓扑值得我们高度关注。当前的以太坊区块链浏览器（例如，Etherscan）专注于跟踪区块和交易记录，但忽略了底层 P2P 网络的特征。这项工作介绍了以太坊网络分析器（Ethna），这是一种探测和分析以太坊区块链的 P2P 网络的工具。与采用非结构化 P2P 网络的比特币不同，以太坊依靠 Kademlia DHT 来管理其 P2P 网络。因此，现有的类比特币 P2P 网络分析方法不适用于以太坊。 Ethna 实现了一种新方法，可以准确测量以太坊节点的程度。此外，它结合了一种算法，该算法可以推导出以太坊 P2P 网络中消息传播的延迟指标。我们在以太坊主网上运行 Ethna，并进行了广泛的实验来分析其 P2P 网络的拓扑特征。我们的分析表明，以太坊 P2P 网络具有一定的小世界网络效应，节点的度数服从幂律分布，具有无标度网络的特征。

1 引言 首先是中本聪在比特币中引入的，区块链是一种安全、可验证和防篡改的分布式账本，用于支持数字资产交易 [1]。由于能够在许可较少的去中心化网络 [2, 3] 上达成共识，区块链已成为金融科技 [4]、物联网 (IoT) [5] 和供应链 [5] 领域的颠覆性技术6]。 2014 年，以太坊将智能合约引入区块链，以去中心化的方式完成各种图灵完备的计算任务 [7]。借助智能合约，以太坊允许用户开发各种去中心化应用程序（DApps），极大地扩展了区块链的应用。下一代区块链将通过采用新颖的共识协议、跨链方法和分片等尖端技术进一步提升性能。作为通信基础设施，点对点（P2P）网络是区块链系统的重要组成部分[8]。区块链的节点通过 P2P 网络发送和接收包含交易和区块的消息，以实现分布式共识。区块链系统的运行性能和稳定性受消息转发协议、对等发现协议及其底层 P2P 网络拓扑的影响。例如，区块在网络中的传播时间是影响区块链交易处理能力的关键因素[8, 9]——区块传播时间越短，区块链的交易处理能力就越高。此外，恶意节点可以利用底层 P2P 网络的特定缺陷对区块链进行各种攻击，例如 DDoS、Eclipse 攻击 [10, 11]。因此，分析和理解区块链系统的 P2P 网络尤为重要。然而，当前的以太坊区块链浏览器（例如，Etherscan [12]）专注于区块和交易记录的跟踪，而忽略了对底层 P2P 网络的分析和表征。此外，为类比特币 P2P 网络提出的现有分析方法 [9, 13-16] 不适用于以太坊，因为以太坊使用与非结构化 P2P 网络根本不同的 Kademlia DHT 结构管理其 P2P 网络被比特币区块链采用。最后，许多先前关于分析和测量 P2P 网络的工作 [17-21] 都集中在文件共享系统上，例如 BitTorrent 和 Gnutella；这些测量方法不能应用于区块链系统，因为它们利用了特定于相关文件共享系统的协议（参见第 2 节中相关工作的详细讨论）。

这项工作介绍了以太坊网络分析器（Ethna），这是一种探测和分析以太坊区块链的 P2P 网络的工具。与通过直接测量一些宏观网络特征（例如，P2P 网络的规模、节点之间的延迟分布和节点的地理分布）来研究以太坊 P2P 网络的其他工作 [22-25] 不同，我们测量和分析以太坊 P2P 网络的一些精细拓扑特征。我们首先利用以太坊 P2P 网络中消息转发协议的随机选择特性来测量以太坊节点的度分布（即随机选择一些邻居节点转发消息） .由于消息转发的随机性与实际节点度数密切相关，因此我们测得的节点度数足以准确反映以太坊网络拓扑的特点。此外，我们还利用以太坊 P2P 网络的消息转发协议来分析交易广播延迟，并进一步获得将消息传播到整个以太坊 P2P 网络所需的跳数。基于 Ethna 的实测数据和分析结果，我们对以太坊 P2P 网络的拓扑结构得出以下结论：

* 以太坊 P2P 网络中的大多数节点的度数都小于 50，即启动节点时允许与节点连接的默认最大邻居节点数； 有几个度数非常高的超级节点。 所有网络节点的度分布呈现幂律分布，这是无标度网络的特征。
* 向整个以太坊 P2P 网络广播交易的平均延迟约为 200 毫秒。 向全网广播一个新区块或新交易需要 3-4 跳，说明以太坊 P2P 网络具有一定的小世界网络效果。

这项工作的主要贡献如下：

我们设计了一种新颖的方法，可以通过简单的设置（具体来说，我们只需要部署一个节点来探测以太坊 P2P 网络）就可以准确地测量以太坊节点的度数，具有可行的复杂度（具体来说，我们只需要计算 在部署的节点收到消息）。 我们提出了一种有效的算法来分析以太坊网络中消息传播的性能指标，包括交易广播延迟和将消息广播到整个 P2P 网络所需的跳数。我们使用 Go 编程在 Ethna 中实现我们的测量方法 并将此工具部署在以太坊主网中。 我们的实验结果为以太坊的 P2P 网络提供了新的见解，有助于改进区块链系统的网络设计。

本文的其余部分安排如下。 第 2 节回顾了相关工作。 第 3 节介绍了以太坊 P2P 网络的背景。 第 4 节介绍了我们的网络测量方法。 第 5 节分析测量数据以得出网络拓扑特征。 第 6 节总结了这项工作。

**2 RELATED WORK**

比特币的 P2P 网络采用非结构化拓扑、八卦消息广播协议和随机节点发现协议 [8]。工作 [9] 调查了比特币 P2P 网络中的区块和交易传播，发现比特币区块链上的分叉主要由消息传播延迟决定。工作 [13] 测量了比特币 P2P 网络的规模、节点地理分布、稳定性和传播延迟。基于采用的随机节点发现协议，一些工作 [14, 15] 测量了比特币 P2P 网络中节点的度数。这些结果表明，比特币 P2P 网络的节点度数分布遵循幂律，即少数节点度数非常大，大多数节点度数非常小。 [16]中的研究还测量了具有类似于比特币网络协议的门罗币的P2P网络，发现门罗币的节点度也遵循幂律分布。节点度分布遵循幂律的网络通常称为无标度网络[26]。因此，[14-16] 的测量结果可以支持比特币和门罗币 P2P 网络类似于无标度网络的结论。当前的以太坊区块链浏览器（例如，以太扫描 [12]、以太坊区块链浏览器 [27] 和以太坊节点浏览器 [28]）跟踪交易、区块和节点记录，但忽略了底层 P2P 网络的特征。研究了以太坊 P2P 网络的一些特征[22-24]，包括以太坊 P2P 网络的规模、节点间的延迟分布和节点的地理分布。但是，没有对以太坊网络拓扑做出任何结论。迄今为止，缺乏对以太坊P2P网络中节点度数的度量以及基于度数分布分析以太坊P2P网络拓扑的研究。以太坊的对等发现协议与比特币和门罗币的发现协议有很大不同，因为以太坊 P2P 网络采用 Kademlia DHT 协议 [29] 中的 K-bucket 数据结构来发现网络节点并维护节点信息。由此可见，在以太坊中衡量网络节点的度数，并根据节点度的分布来研究以太坊 P2P 网络并非易事。例如，[25] 假设存储在 K 个桶中的对等点数量与对等点的度数相同，测量了以太坊 P2P 网络中的对等点的度数；然而，[25] 中测量的对等度远大于实际的对等度，因为随着时间的推移，实际节点度通常远小于存储在 K-buckets 中的节点数（我们提供实验结果支持第 5 节中的这一点）。

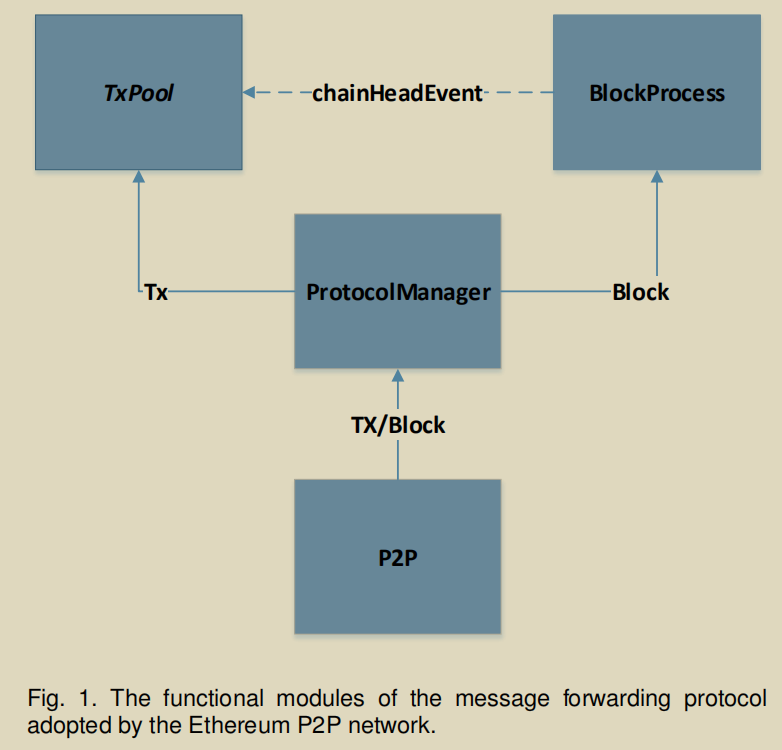
除了区块链系统的 P2P 网络的测量工作外，其他现有的测量 P2P 网络的工作主要集中在文件共享系统上，例如 BitTorrent 和 Gnutella。大多数工作首先捕获 P2P 网络的拓扑，然后从捕获的拓扑中测量网络特征（例如，节点度、消息传播跳数）[17-21]。捕获 BitTorrent 的 P2P 网络拓扑的总体思路是利用其节点发现协议的特性，即对等交换 (PEX) [30]。根据PEX，当一个新节点连接到网络中的某个节点时，新节点可以进一步获取该连接节点的邻居节点列表，然后与列表中的节点建立新的连接。此外，节点将定期与其相邻节点交换有关其新连接和断开节点的信息[30]。 PEX 的这一特性用于测量 BitTorrent 的 P2P 网络拓扑结构，并在 [17-21] 中分析其 P2P 网络。但是，在以太坊区块链的 P2P 网络中，出于安全原因，节点之间不会交换相邻节点的信息，因此 BitTorrent P2P 网络的测量方法不适用于以太坊 P2P 网络

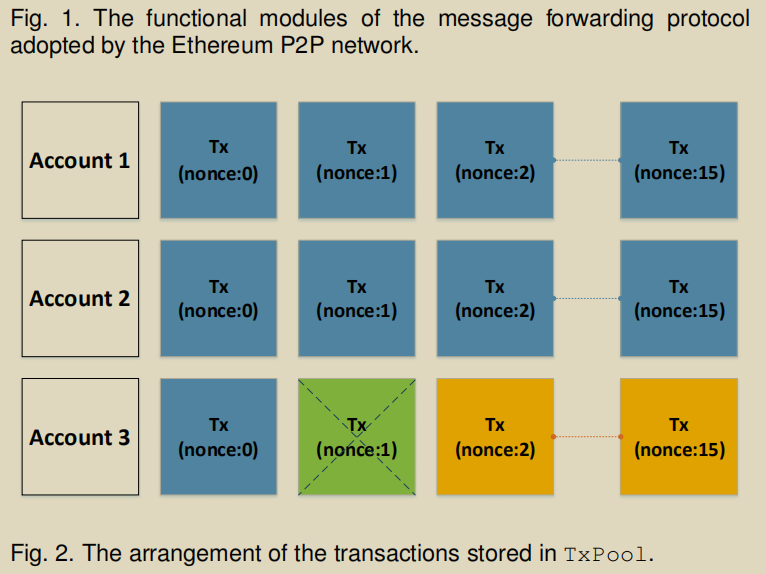
**3 BACKGROUND**

以太坊网络通信在三个协议中定义，RLPx 用于节点发现和安全传输，DEVp2p 用于应用程序会话建立，以及在以太坊的 Geth 参考客户端 [34] 中实现的用于应用程序级通信的以太坊子协议 [31-33] . 本节介绍以太坊子协议采用的消息转发协议，用于在 P2P 网络上传播交易和区块的消息。 Ethna 利用该协议获取有关以太坊 P2P 网络的有用信息。

3.1 消息转发协议功能模块

如图 1 所示，每个以太坊节点的消息转发由几个功能模块处理。这些模块相互交互，完成整个消息转发过程。 P2P 模块负责与底层 P2P 网络通信：它与其他邻居节点交换消息，将包含块和事务的消息传递给协议管理器模块，并将特定于 P2P 网络通信的消息保存在 P2P 模块中进行处理（例如，ping/pong 消息）。协议管理器（ProtocolManager）模块对接收到的区块和交易消息进行处理，并分别交付给交易池（TxPool）和区块处理（BlockProcess）模块。 TxPool 模块用于存储尚未记录到区块链上的交易。 BlockProcess 模块用于处理从网络新接收到的块。 TxPool 中的交易是按照所属账户排列的，如图 2 所示。图 2 中的每一行都排列了同一账户发出的交易，这些交易按照它们的 nonce1 值升序排列。根据它们的 nonce 值是否连续，这些交易分别存储在 TxPool 的两个不同子部分中：





* ? PendingPool：它维护尚未包含在区块链上的块中但准备打包成新块的待处理交易。 如图2所示，蓝色交易为待处理交易； 这些与每个账户关联的交易的 nonce 值是连续的。 对于每个账户，PendingPool 中最多可以存储 16 个未决交易。另外，一旦有交易进入 TxPool 的 PendingPool，TxPool 模块会通知 ProtocolManager 模块该交易可以转发到 其他还没有交易的邻居节点。 矿工节点可以从其 PendingPool 中选择交易，按照以太坊的打包规则打包到一个新的区块中。
* queue：在没有具有连续随机数的交易并且这些未来的交易还没有准备好打包到新区块中之后，它维护“未来”交易。 如图2所示，属于Account 3、nonce值为1的交易（绿色）目前缺失。 因此，后续可用的连续交易（黄色）具有大于 1 的 nonce 值进入 TxPool 的 Queued。 只有当 nonce 值为 1 的交易到达 TxPool 后，这些 nonce 值连续的交易才能进入 PendingPool。 对于每个账户，Queued 最多可以存储 64 笔交易。

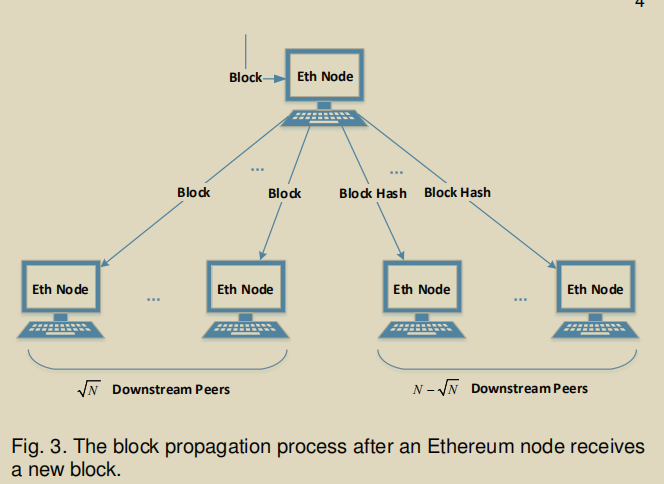
当来自网络的新块到达 ProtocolManager 模块时，它将被发送到 BlockProcess 模块进行进一步处理。 首先，BlockProcess 验证块。 一旦区块通过验证，本地区块链的世界状态将被更新，然后从 BlockProcess 向 TxPool 发送一个链头事件（ChainHeadEvent）。 ChainHeadEvent 包含当前新区块更新的最新世界状态。 收到 ChainHeadEvent 后，TxPool 会根据更新的世界状态进行自我重置。 也就是说，它会移除存储在 PendingPool/Queued 中且包含在新区块中的交易； 此外，它将包含在新块中但不在 PendingPool/Queued 中的交易添加到 PendingPool/Queued 中。 在 TxPool 重置期间，TxPool 暂时被阻塞，ProtocolManager 发送的交易无法进入 TxPool。

通常，当收到一个新块并且在区块链上没有发现分叉时，新块中包含的交易将在 TxPool 重置期间从 PendingPool 中删除并排队，如上所述。另一方面，当依次接收到两个高度相同的区块时，即在区块链上观察到分叉时，TxPool 重置的处理是不同的。这两个块分别用FormerBlock和AfterBlock表示，对应两个不同的世界状态。收到FormerBlock后，TxPool会根据自己的世界状态进行重置，即将包含在FormerBlock中且同时存在于PendingPool/Queued中的交易从PendingPool/Queued中移除，仅包含在FormerBlock中但不存在于PendingPool中的交易/Queued 被添加到 PendingPool/Queued。收到 AfterBlock 后，AfterBlock 中存在的交易将从 PendingPool/Queued 中删除。然后，将FormerBlock和AfterBlock中的交易进行比较，找出未包含在AfterBlock中但包含在FormerBlock中的交易（已从PendingPool/Queued中排除）。这些事务需要重新添加到 PendingPool/Queued。

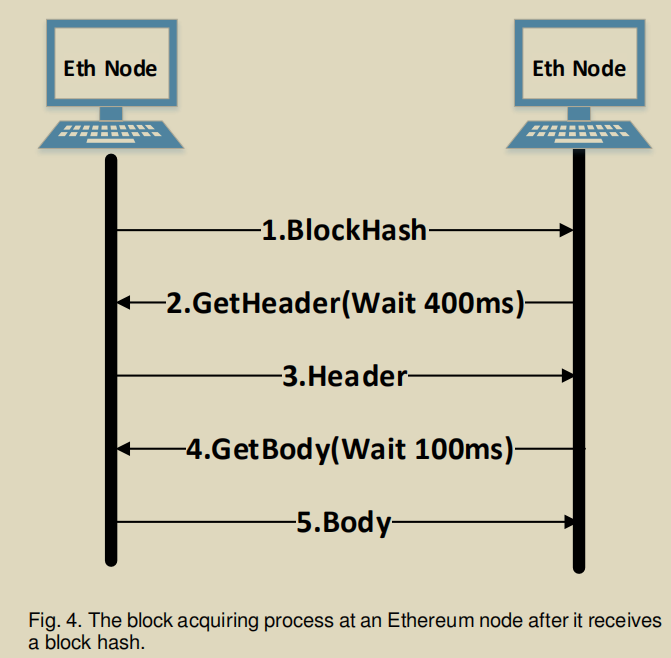
从上面的描述我们可以看出，当 TxPool 重置时，可能会有大量交易同时进入 PendingPool，然后 TxPool 通知 ProtocolManager 这些新来的交易可以转发到网络中的其他节点。

**3.2 Block Propagation Strategy**

在以太坊 P2P 网络中，区块和交易的传播采用八卦策略 [35]。 让我们考虑一个从其相邻节点之一接收新块的以太坊节点。 接收到新块的节点会随机选择一些与其相连的邻居节点来传播接收到的新块。 我们将该节点当前不知道新块的邻居节点称为该节点的下游对等节点。 这些下游节点的数量用N表示。节点收到新区块后，首先验证区块头，然后随机选择√N个下游节点转发新区块。之后，进一步验证整体 块，然后如果块验证通过，则将块的哈希发送给剩余的 N -√N 个下游对等点。 图 3 说明了以太坊节点接收到新区块后的区块传播过程。



我们接下来解释当一个以太坊节点从它的一些邻居节点接收到块哈希时，它是如何获取一个新块的。图 4 说明了在接收块哈希的以太坊节点获取新块的过程。当某个以太坊节点收到新的区块哈希时，该节点首先等待 400 毫秒；然后从已经知道新区块的邻居节点中随机选择一个节点（即已经向该节点发送区块或区块哈希的邻居节点）。之后，该节点发送GetHeader信息，向选中的邻居节点请求新块的头部。节点收到被选中的邻居节点返回的区块头后，会等待100 ms，然后从知道新区块的邻居节点集合中随机选择一个邻居节点，得到新区块的body。最终，接收到的新区块体和新区块头将被组装成一个新区块，并在该区块验证通过后附加到本地区块链的尾部。通常，一个以太坊节点与以太坊 P2P 网络中的多个邻居节点相连。如图 4 所示，节点通过该区块的哈希获得区块是一个漫长的过程。因此，在此过程中，其他邻居节点可能会将新块发送给该节点。一旦节点收到其他邻居节点发来的区块，就会停止通过区块哈希获取区块的过程。



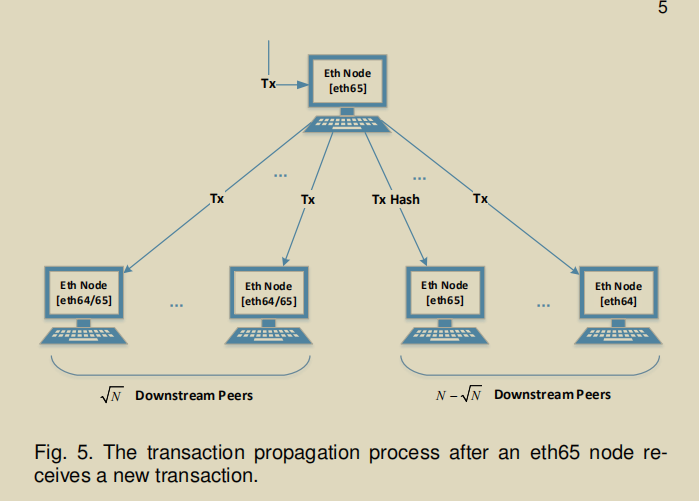
*实际上，传播新消息的邻居节点的数量决定了消息广播复杂性和八卦广播协议的鲁棒性/可靠性之间的权衡。 一般来说，大量的邻居节点传播增加了到达所有节点的概率，但也会在P2P网络中产生更多的冗余流量。 在 [35, 36] 中可以找到基于 gossip 的消息广播协议的理论分析，该协议将它们的可靠性与消息广播复杂性联系起来。*

**3.3 Transaction Propagation Strategy**

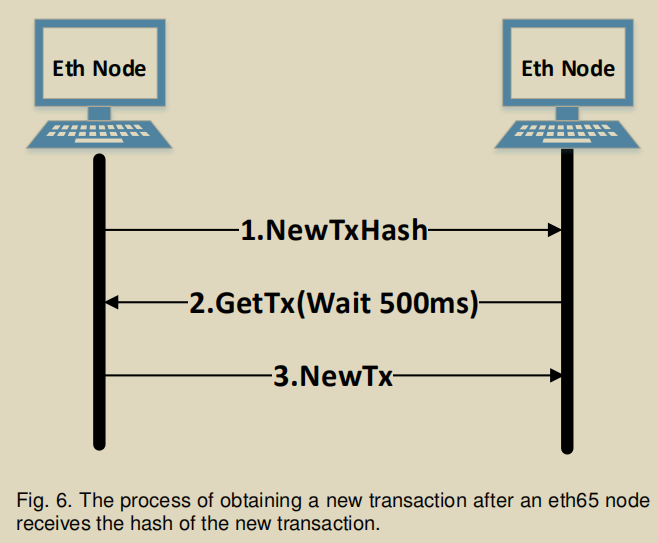
当前以太坊主网中的节点运行不同的协议，例如提供轻节点服务的 les 协议和提供全节点服务的 eth 协议。对于运行 les 协议的节点，只会同步区块头信息；块体中的交易信息会在必要时从其他节点同步。对于运行 eth 协议的节点，包括交易在内的所有区块信息都会一直同步。我们的目标是利用交易传播过程来衡量以太坊 P2P 网络。因此，我们将重点关注运行 eth 协议的节点。

eth 协议有不同的版本，例如 eth62、eth63、eth64 和 eth65 [38]。 运行不同版本 eth 协议的节点具有略微不同的交易传播策略。 注意，eth64 及之前版本的事务传播策略与 eth65 及之后版本完全不同。 而且，目前以太坊主网中的节点使用的 eth 协议大多是 eth64 或 eth65。 因此，eth64 和 eth65 的交易传播策略会影响我们的衡量方法。 下面我们将运行 eth64 协议的节点称为 eth64 节点，将运行 eth65 协议的节点称为 eth65 节点。 下面介绍eth64和eth65的交易传播策略

对于 eth64 节点，接收到新交易，ProtocolManager 模块将新交易发送到 TxPool 进行验证。已验证的新交易按照第 3.1 节中的规则存储到 PendingPool 或 Queued 中。当有新交易进入 TxPool 模块的 PendingPool 时，TxPool 模块会通知 ProtocolManager 模块有新的交易可以转发给其他节点。 ProtocolManager 模块然后将新的交易转发给当前没有交易的邻居节点。对于一个eth65节点，交易传播处理根据转发目标邻居节点采用的eth协议版本，执行两种不同的动作：转发交易本身或转发交易哈希到目标邻居节点。我们在图 5 中说明了 eth65 节点的事务传播过程。当 eth65 节点接收到新事务时，其 ProtocolManager 模块首先将新事务发送到 TxPool 进行验证。 当交易通过验证并存储在 TxPool 的 PendingPool 中时，TxPool 通知 ProtocolManager 模块有一个新的交易可以转发到其他邻居节点。 然后，ProtocolManager 随机选择 √N 个不知道该事务的下游节点作为转发该事务的目标。 对于剩余的 N -√N个下游peer，如果下游peer运行的eth协议版本为eth65，则转发交易hash； 如果下游对端运行的 eth 协议版本是 eth64，则该事务将被转发。



如上所述，eth65 节点将从其邻居 eth65 节点接收交易哈希。当eth65节点接收到新交易的hash时，获取新交易的流程如图6所示。eth65节点收到新交易hash后，等待500ms。在此期间，如果没有其他邻居节点向其发送新交易，则 eth65 节点随机选择一个向其发送新交易哈希的邻居节点，并向选定的邻居节点发送 GetTx 信息请求新交易。请求的邻居节点返回新交易后，eth65 节点对新交易进行验证。如果交易通过验证，则将交易添加到该 eth65 节点的 TxPool 中。



**4 NETWORK MEASUREMENT**

在本节中，我们将介绍 Ethna 设置的以太坊网络节点，用于探测以太坊 P2P 网络。 然后，使用这些探测节点，我们提出了测量节点传播交易的时刻以及其他以太坊节点传播的交易数量的方法。 在我们的测量中，我们假设网络节点遵循第 3 节中描述的特定协议行为，这些行为在接收新事务的散列中实现。 以太坊的 Geth 参考客户端.3

**4.1 Setting up Probing Nodes**

为了探索以太坊 P2P 网络，我们在以太坊主网上设置了两个不同的节点：

* NetworkObserverNode：这是一个在快速同步模式下运行的以太坊节点。当节点的本地区块链状态远离当前区块链的世界状态时，节点将执行快速同步模式以快速同步到当前世界状态。在快速同步模式下运行的节点只验证从其他节点下载的块中包含的世界状态，并跳过对块中包含的交易的验证和转发。因此，快速同步模式可以减少同步时间[39]。快速同步模式下的 NetworkObserverNode 会随机连接以太坊主网上的其他邻居节点。虽然它的邻居节点会向 NetworkObserverNode 发送新的块和新的交易，但在快速同步模式下的 NetworkObserverNode 收到这些块和交易后不会转发。因此，NetworkObserverNode 在以太坊主网上只扮演观察者的角色。 NetworkObserverNode 运行采用 eth65 协议的 v1.9.15 [34] 版本的以太坊软件。我们使用位于中国深圳的阿里云服务器设置 NetworkObserverNode，其公共 Internet 上的 IP 地址为 8.129.212.167。
* LocalFullNode：这是一个已经同步到区块链最新世界状态的以太坊节点。 LocalFullNode 将接收、验证和转发新区块和新交易。 LocalFullNode 运行采用 eth65 协议的 v1.9.15 版本的以太坊软件。 我们在中国深圳的深圳大学局域网上搭建了LocalFullNode。 LocalFullNode 没有自己的公网 IP 地址，它使用 NAT 协议与以太坊主网上的其他节点通信。

通过这两个以太坊节点，我们可以探测以太坊 P2P 网络来测量交易传播时间，以及每个节点在特定观察时间窗口内转发的交易数量，如下所述。

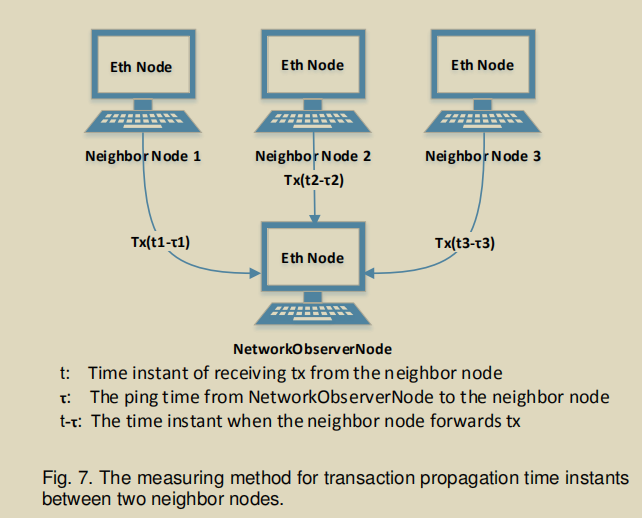
1. *如[24]所示，Geth 客户端约占以太坊主网节点的 76%，因此这不是一个强有力的假设。 此外，我们从以太坊主网的实际数据中获得的测量结果验证了我们测量方法的有效性。*

**4.2 Network Measuring Method**

1）测量交易传播时间：

如第 3 节所述，当以太坊节点加入网络时，它们将选择合适的节点进行连接，然后与这些连接的邻居节点交换块和交易。寻找时间向以太坊 P2P 网络中的大多数节点广播新交易是 Ethna 的主要目标之一。为了实现这一目标，我们首先使用以下测量方法测量 NetworkObserverNode 的相邻节点处的事务传播时间瞬间。我们利用 NetworkObserverNode 收集从其连接的邻居节点发送的交易。我们知道，作为网络观察者，NetworkObserverNode 不会改变其每个相邻节点的 TxPool 状态，因为它只接收交易而不转发交易。 4 通常，交易或交易散列在网络上以仅由该交易或该交易散列组成的数据包的形式传播。在某些情况下，许多事务或事务哈希将被封装到一个数据包中并通过网络传播。5 每当 NetworkObserverNode 从邻居节点接收到一个事务或事务哈希数据包时，我们都会将有关这些事务的有用信息记录到数据库中称为 TxMsgPool。对于每个接收到的数据包中包含的每个事务或事务哈希，我们首先记录一个原始数据记录 tempTxMsg，它由以下形式给出： tempTxMsg { PeerID,TxHash,TimeStamp,GasPrice, PacketSize } 其中字段解释如下： PeerID 是向 NetworkObserverNode 发送交易/交易哈希数据包的邻居节点的对等标识； TxHash 是交易哈希； TimeStamp 是 NetworkObserverNode 收到交易/交易哈希数据包时的本地时间戳； GasPrice 是本次交易支付给矿工的服务费； PacketSize 是接收到的数据包中包含的总交易/交易哈希的数量（例如，当邻居节点向 NetworkObserverNode 发送交易并且该交易被单独封装到数据包中时，PacketSize 的值为 1）。

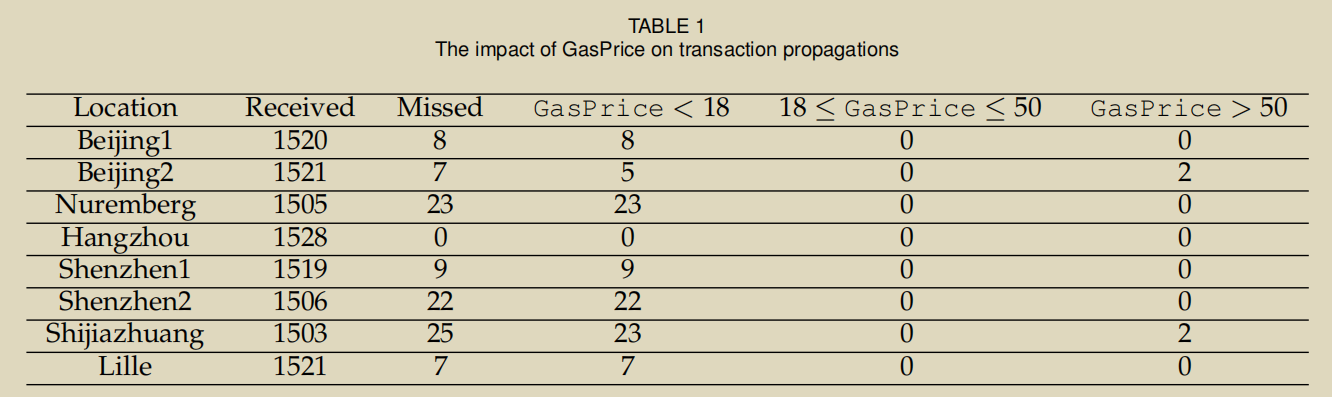
1. *由于配置为执行快速同步的 NetworkObserverNode 的本地区块链状态与区块链当前的世界状态相差甚远，因此以太坊中最近发出的交易不符合本地区块链状态 NetworkObserverNode 和 NetworkObserverNode 将丢弃这些新交易当它在快速同步过程中无法验证事务时。*
2. *一个 eth65 节点会为其每个相邻的 eth65 节点创建两个缓存队列，即 TxQueued 和 TxHashQueued。 TxQueued 用于缓存准备传播到该邻居节点的相应事务，TxHashQueued 用于缓存准备传播到该邻居节点的相应事务哈希。 eth65 节点选择一个邻居 eth65 节点传播交易或交易哈希后，会立即将该交易或交易哈希馈送到该邻居节点的 TxQueued 或 TxHashQueued 中。当处理邻居节点的线程资源空闲时，它将TxQueued中的所有事务打包成一个事务包，并将这个事务包转发给邻居节点。同理，交易hash需要先缓存在邻居节点的TxHashQueued中，然后再转发。当线程资源空闲时，将 TxHashQueued 中的所有事务哈希打包成一个事务哈希包，然后转发给相应的邻居节点。*



所有原始数据记录 tempTxMsg 都存储在数据库 TxMsgPool 中。每个原始数据记录 tempTxMsg 中的时间戳是 NetworkObserverNode 从邻居节点接收到事务/事务哈希时的本地时间戳。我们还想测量这个邻居节点将此事务/事务哈希转发到 NetworkObserverNode 的时间。图 7 说明了测量方法。我们用 Delay 表示从邻居节点向 NetworkObserverNode 传播消息的时间成本，NetworkObserverNode 可以通过向对应的邻居节点发送基于 ICMP 的 ping 数据包并在 NetworkObserverNode 之间占用一半的 ping 时间来衡量 Delay 的值和邻居节点。6 从TimeStamp 中减去Delay，即TimeStamp -√Delay，给出了邻居节点将此事务/事务散列转发给NetworkObserverNode 的时刻。但是，当我们从 tempTxMsg 记录中测量交易传播时间时，我们需要根据以下两个标准过滤 tempTxMsg{PeerID, TxHash,TimeStamp, GasPrice, PacketSize}：

* ? tempTxMsg 的 GasPrice 字段应不小于 18 Gwei。 GasPrice 的值是指示此交易是否在网络上适当传播的指标。当以太坊节点启动 Geth 客户端时，可以设置决定交易是否可以进入其 TxPool 的 GasPrice 的最小值。只有 GasPrice 值不小于设定值的交易才能进入 TxPool 并稍后转发。如果新收到的交易的 GasPrice 太小，会被节点立即丢弃。因此，该交易的传播时间会更长；更糟糕的是，一些节点可能无法接收到交易。当前版本的以太坊软件中 GasPrice 的默认最小值为 1 Gwei，而在旧版本中则设置为 18 Gwei。我们进行了一个简单的实验来研究 GasPrice 对交易传播的影响。我们于 2019 年 10 月 14 日首次从网站 [12] 爬取了 1528 笔交易。这些交易通过 Etheruem P2P 网络传播。然后，我们在 NetworkObserverNode 收集这些事务的传播结果。有八个邻居节点（在不同的地理位置）连接到 NetworkObserverNode。这些邻居节点将它们收到的交易转发给 NetworkObserverNode。我们可以在 NetworkObserverNode 上查看是否所有 1528 个事务都从八个邻居节点传播到 NetworkObserverNode。 1528个交易的传播结果如表一所示。我们可以看到NetworkObserverNode从它的每个邻居节点接收到的这些交易的数量都小于1528。这表明一些交易丢失了。为了调查这些丢失的交易，我们发现大多数丢失的交易的 GasPrice 低于 18 Gwei。因此，我们可以得出结论，这些交易由于 GasPrice 较小而被丢弃。因此，为了调查适当传播的交易，我们只选择原始数据记录 tempTxMsg，其 GasPrice 字段不小于 18 Gwei 进行分析。
* tempTxMsg 的 PacketSize 字段的值应为 1。PacketSize 的值是一个关键指标，用于确定向 NetworkObserverNode 发送事务/事务哈希的邻居节点的 TxPool 在此事务处理过程中是否重置/transaction 散列传播。根据第 3 节，如果邻居节点没有发生 TxPool 重置，则邻居节点可以将该交易存储到其 TxPool 中，并在该邻居节点收到交易后立即将交易本身或该交易的哈希转发给 NetworkObserverNode，并且完成其验证。在这种情况下，用于验证和存储此交易的时间成本非常短（大约 1 毫秒），可以忽略不计。但是，如果在邻居节点有 TxPool 重置，许多交易会同时进入 TxPool，然后这些交易或这些交易的哈希在一个数据包中一起转发。因此，事务的传播时间被 TxPool 重置过程严重放大，导致 PacketSize > 1，我们只选择 PacketSize 字段为 1 的原始数据记录 tempTxMsg 进行分析。

*6. 基于 TCP 的 ping 或协议自身的 ping/pong 消息可能会受到网络拥塞的严重影响，而基于 ICMP 的 ping 通常会绕过完整队列，因此往往更可靠。*



TxHash, TimeStamp, GasPrice, PacketSize} 其中PacketSize = 1 且GasPrice > 18 和对应的Delay，我们可以获得一个交易传播记录，TxMsg，用于该交易传播。 交易传播记录 TxMsg 的形式由 TxMsg { PeerID,TxHash,ForwardTime } 给出，其中 ForwardTime = TimeStamp ？ 延迟。 所有的事务传播记录，TxMsg，都存储在数据库 TxMsgPool 中，用于以下分析。

1. 测量每个eth65节点转发的交易数量：

根据第 3 节，当一个 eth65 节点传播一个新交易时，它会将交易和交易哈希转发到它的一些邻居节点。根据 NetworkObserverNode 在某个测量周期内收到的交易和交易哈希，我们可以为每个与 NetworkObserverNode 连接的节点构建一个节点传播记录，记为 PeerPacketMsg。节点传播记录 PeerPacketMsg 是为与 NetworkObserverNode 连接的节点构建的，存储了 NetworkObserverNode 从该节点接收到的事务数和事务哈希。该节点传播记录写为 PeerPacketMsg {PeerID,TxHashPacketCount, TxPacketCount , StartTime,CurrentTime } 其中 PeerID 为节点标识，TxHashPacketCount 为本节点发送的事务哈希包，TxPacketCount 为本节点发送的事务包数，StartTime 为第一个事务或事务哈希包发送的时刻节点在 NetworkObserverNode 接收到，CurrentTime 是在 NetworkObserverNode 接收到节点发送的最新事务或事务哈希包的本地时间。到目前为止，我们已经在以太坊主网上设置了探测节点，以收集来自以太坊 P2P 网络的交易传播记录 TxMsg 和每个节点转发的交易记录 PeerPacketMsg。在下一节中，我们利用这些传播记录来分析以太坊 P2P 网络的拓扑特征。

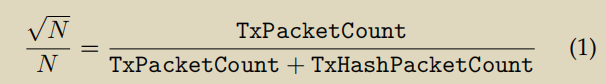
**5 ANALYSIS OF NETWORK TOPOLOGICAL FEATURES**

本节介绍 Ethna 用于分析以太坊 P2P 网络的拓扑特征的算法，即节点度的分布、将交易广播到大多数节点的延迟成本以及广播所需的跳数 以太坊 P2P 网络上的消息。 我们用 Go 编程语言实现了 Ethna，并将其部署在当前的以太坊主网上，以实验验证 Ethna。

* 1. 节点度分布的解析方法

我们首先提出了一种新颖而简单的方法来分析以太坊节点的度分布。如第 3 节所述，每当 eth65 节点收到新交易时，它会将交易转发给从没有该交易的 N 个下游对等节点中随机选择的 √N 个邻居节点，并将交易哈希转发给剩余的 N- √ N 个下游对等体。当一个节点从其相邻节点之一接收到新交易时，它需要在将其转发到其他相邻节点之前验证该交易。在验证到转发期间，可能会发生其其他邻居节点也将相同的事务转发到该节点的情况。因此，这些也转发事务的邻居节点在该节点转发事务时不会被视为下游对等节点。因此，严格来说，该节点的下游对等体的数量应该小于该节点的度数。此外，对于不同的交易，下游节点和下游节点的数量可能不同。在以太坊 P2P 网络中，交易从节点接收到节点转发之前的平均处理时间约为 1 毫秒。与 200 ms 的两个相邻节点之间的平均事务传播时间相比（参见我们在 5.2 节中提供的测量结果），这个处理 9 时间可以忽略不计。对于每个节点，我们将其所有转发事务的下游对等点的数量视为相同的数字N，我们将其节点度近似为N + 1，其中加1是由于对于每个事务，都有一个邻居节点将交易传输到该节点。

在第 4 节中，NetworkObserverNode 对于其每个连接的节点，都获得了一个节点传播记录 PeerPacketMsg，该记录存储了该节点转发的事务包的数量和该节点转发的事务哈希包的数量。 注意，在转发每笔交易时，一个节点会从 N 个下游对等节点中随机选择 √N 个邻居节点转发该交易，其余 N N √N 个下游对等节点转发该交易哈希7。 因此，NetworkObserverNode 从一个节点接收到的事务数与该事务数加上 NetworkObserverNode 从同一节点接收到的事务哈希之比为 √NN 。 基于以上分析，我们可以利用每个节点对应的PeerPacketMsg记录中包含的TxPacketCount和TxHashPacketCount数据建立如下公式：



当该节点转发的交易和交易散列的数量很大时，它在统计上成立。 使用 PeerPacketMsg 中包含的每个节点的数据，我们可以解决 (1) 找到下游对等点的数量 N 并使用 N + 1 作为该节点度数的近似值。 由（1）计算 N 是相当可行的，即我们只需要收集有关接收到的交易数量和交易哈希的数据，并有一个简单的计算...公式。 用于计算的收集数据的设置也很简单，即我们只需要 NetworkObserverNode 的一个节点作为以太坊主网上的网络观察者。 接下来，我们将通过实验验证我们Ethna中节点度的这种估计方法是否准确。

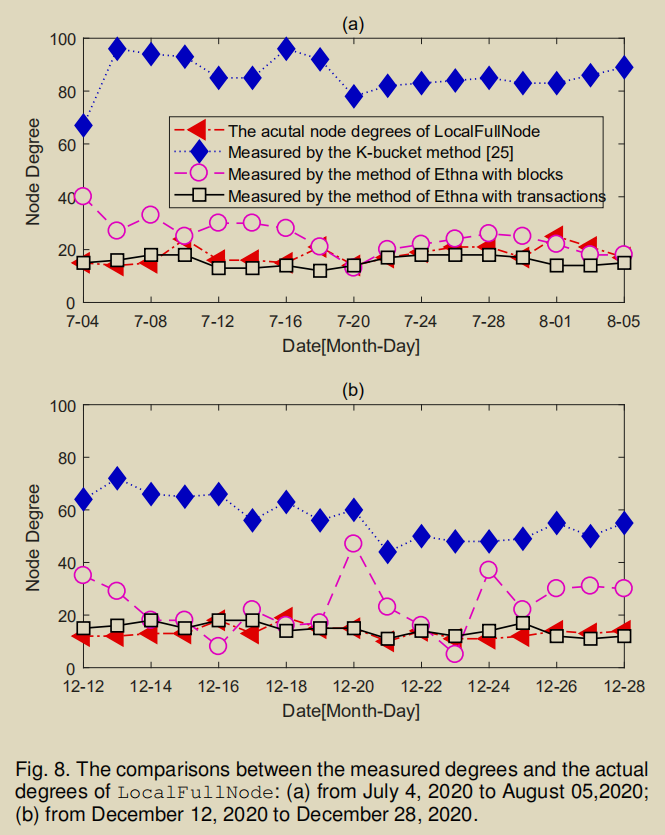
1. 估计节点度的实验验证：

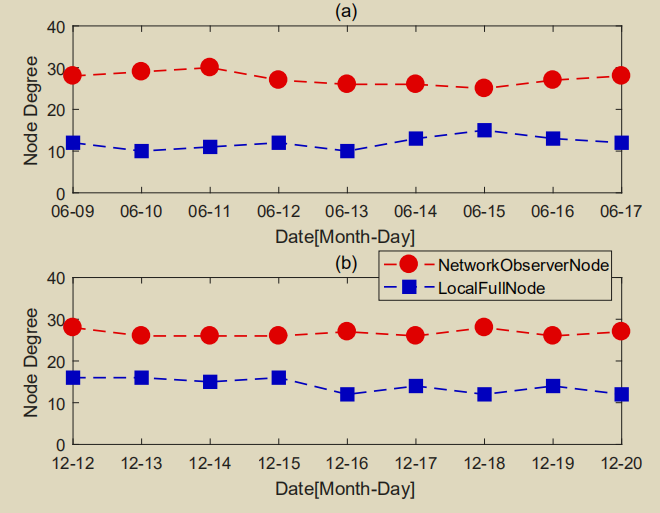
我们在以太坊主网上运行 NetworkObserverNode 和 LocalFullNode 进行以太坊节点度数测量实验。我们只使用 eth65 节点作为目标来衡量以太坊 P2P 网络的节点度，而没有使用 eth64 节点。附录中解释了不使用 eth64 节点来衡量节点度的原因。我们采用第 4 节中提出的网络测量方法来统计每个与 NetworkObserverNode 稳定连接的 eth65 节点发送的事务包和事务哈希，以获得 PeerPacketMsg 记录。通过PeerPacketMsg记录中包含的数据，我们可以求解（1）中的公式得到N，并使用N+1来近似每个eth65节点的节点度。之后，我们可以推导出以太坊节点的度数分布。我们首先进行了实验，以验证 Ethna 中节点度的测量方法。我们的 NetworkObserverNode 和 LocalFullNode 在两个不同的测量期间在以太坊主网上相互连接，即 2020 年 7 月 4 日至 2020 年 8 月 5 日期间和 2020 年 12 月 12 日至 2020 年 12 月 28 日期间。完成区块链同步后，会将交易和交易哈希传播到 NetworkObserverNode。因此，NetworkObserverNode用于收集LocalFullNode转发的事务包和事务hash包，得到LocalFullNode的PeerPacketMsg记录。然后，使用式(1)中的公式计算LocalFullNode的下游对等体的数量N。我们每天测量网络并计算 LocalFullNode 的 N，即每天，我们重新计算事务/事务哈希数据包的数量以获得 PeerPacketMsg 并重新计算 LocalFullNode 的 N。此外，我们还统计了块包和块散列包的数量，并用它们来代替（1）中的交易包和交易散列包的数量来计算N。然后，我们将 N + 1 视为 Ethna 中 LocalFullNode 的测量程度。我们还使用 [25] 中提出的基于 K-bucket 的方案测量了 LocalFullNode 的度数。图 8 显示了 LocalFullNode 的测量度及其实际度，用于比较两个不同的测量周期。从图 8 可以看出，使用我们的 Ethna 方法测得的节点度数与实际节点度数非常接近。使用基于 K-bucket 的方案 [25] 测量的节点度数远大于实际节点度数。使用我们的Ethna方法对交易包数量的测量结果比使用我们的Ethna方法对块数据包数量的测量结果更接近实际节点度（原因在附录中解释）。我们可以看到，使用我们的 Ethna 方法对事务数据包测量的节点度数与实际节点度数之间的不匹配范围超过 [2, 4]。因此，用我们的Ethna测得的N+1的值来近似节点度，被认为是相当准确的。根据测量的与我们的 NetworkObserverNode 稳定连接的节点的度数，我们可以推断出以太坊 P2P 网络中所有节点的度数分布。

*7. 是否将交易或交易哈希转发给对等点由以下过程确定。 有一个名为 peerSet 的表，它列出了 eth65 节点的不知道此交易的邻居节点（即，此 eth65 节点的下游节点关于此交易的节点）。 eth65 节点从其 peerSet 中选择前 √N 个节点来转发交易。 并且对于每个接收到的交易，其 peerSet 中的邻居节点的顺序是随机的。 因此，即使是同一组邻居节点，它们在 peerSet 中的顺序对于不同的事务也会有所不同。 这意味着每次都会随机选择eth65节点转发交易的√N个邻居节点。*

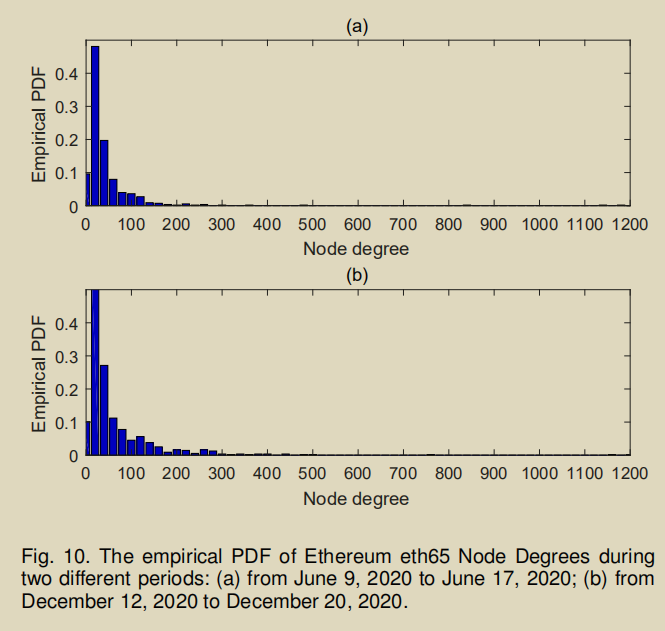
2) 度分布推导实验：

通过测得的节点度，我们可以分析以太坊 P2P 网络中节点的度分布。分析P2P网络拓扑的正确方法，例如节点的度分布，应该通过同时对整个P2P网络中所有节点的状态进行快照来进行。当以太坊 P2P 网络稳定运行时，其节点的度数在一个狭窄的范围内变化，没有明显的波动。而变化范围与MaximumPeerCount（启动节点时允许连接的最大邻居节点数）和本地网络配置有关。我们可以观察我们的 NetworkObserverNode 和 LocalFullNode 节点的度数，以验证以太坊节点的度数是否随时间稳定在可接受的范围内。图 9 展示了 2020 年 6 月 9 日至 2020 年 6 月 17 日和 2020 年 12 月 12 日至 2020 年 12 月 20 日期间 NetworkObserverNode 和 LocalFullNode 的观测度数。NetworkObserverNode 和 LocalFullNode 的 MaximumPeerCount 设置为 50。从图 9 的结果可以看出，在观察期间，NetworkObserverNode 的度数在 [25, 30] 范围内波动，LocalFullNode 的度数在 [10, 14] 范围内波动。这表明 NetworkObserverNode 和 LocalFullNode 的节点度在不同的时间里都是相当稳定的。因此，我们认为在分析以太坊节点的度分布时，可以通过在不同时刻测得的eth65节点的度来推导出以太坊P2P网络中节点的度分布。我们接下来运行 NetworkObserverNode 来测量 eth65 节点与 NetworkObserverNode 建立稳定连接的程度（如果一个节点向 NetworkObserverNode 传播超过 1000 个交易包或交易哈希，则认为该节点建立了一个稳定的连接）。我们发现有 555 个 eth65 节点具有稳定的连接。 9. NetworkObserverNode 和 LocalFullNode 在两个不同时期观察到的节点度数： (a) 2020 年 6 月 9 日至 2020 年 6 月 17 日； (b) 2020 年 12 月 12 日至 2020 年 12 月 20 日，2020 年 6 月 9 日至 2020 年 6 月 17 日测量期间与 NetworkObserverNode 的连接，从2020 年 12 月 12 日至 2020 年 12 月 20 日。然后我们使用 Ethna 的方法分析他们的学位。图 10 显示了经验概率密度函数 (pdf)，图 11 显示了节点度的经验累积分布函数 (cdf)。从图 10 和图 11 可以看出，网络中有少量的超级节点（节点度数非常高）； 2020 年 6 月 9 日至 6 月 17 日期间 555 个 eth65 节点的平均度数为 47，2020 年 12 月 12 日至 2020 年 12 月 20 日期间 769 个 eth65 节点的平均度数为 64。软件于2020年6月12日更新到v1.9.0，默认最大节点度数为50。从图10和图11的结果可以看出，2020年6月9日至6月期间17、78%的节点度数小于MaximumPeerCount，22%的节点度数大于MaximumPeerCount； 2020年12月12日至2020年12月20日期间，70%的节点度数小于MaximumPeerCount，30%的节点度数大于MaximumPeerCount。它揭示了大多数节点都是从MaximumPeerCount开始的，只有少数节点修改MaximumPeerCount来充当超级节点。



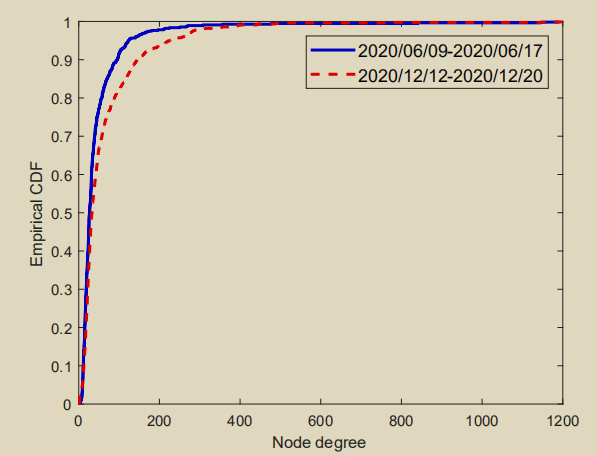


对于两个测量周期，以太坊网络节点的度数均服从幂律分布，其 pdf 由 P (k) ～ kkγ 给出，其中 k 是节点度数，γ 是函数的幂律指数.通过将我们的测量数据拟合到这个幂律分布的 pdf 中，我们发现以太坊 P2P 网络的 γ 值在 2020 年 6 月 9 日至 2020 年 6 月 17 日期间为 2.3363，在 2020 年 6 月 17 日期间为 2.3765 2020 年 12 月 12 日至 2020 年 12 月 20 日。这些值非常接近比特币 P2P 网络的 γ 值，在 [15] 中测量为 2.3。工作 [40] 建议使用零假设显着性检验的 p 值来验证收集数据的经验分布是否为幂律。 p 值是反对零假设的证据； p 值越小，您应该拒绝原假设的证据就越强。通常，p值小于0.05的阈值被认为是原假设无效的条件。在我们的测试中，零假设是收集数据的经验分布是幂律。在[40]中，p值阈值从0.05提高到0.1，即当数据样本的p值小于0.1时，认为幂律分布假设不成立。基于 p 值方法，我们使用从以太坊 P2P 网络收集的数据来拟合节点的度分布。我们发现，2020 年 6 月 9 日至 2020 年 6 月 17 日期间收集的数据的 p 值为 0.15，2020 年 12 月 12 日至 2020 年 12 月 20 日期间收集的数据的 p 值为 0.15。因此，使用 p 值验证节点度分布支持了以太坊 P2P 网络的节点度符合幂律分布的结论。由于无标度网络的节点度分布遵循幂律 [26, 41]，我们可以得出结论，以太坊 P2P 网络类似于无标度网络。



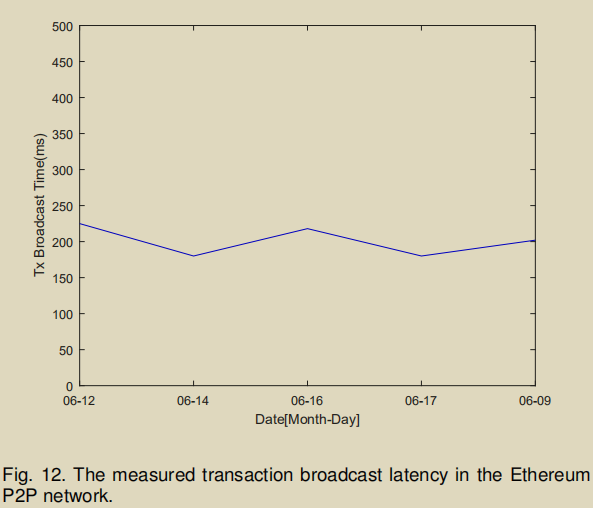
* 1. 交易广播延迟的分析方法

这部分介绍了 Ethna 如何分析将交易广播到以太坊 P2P 网络中的大多数节点的延迟。根据 4.2 节，我们将 NetworkObserverNode 接收到的每个数据包中包含的每个事务的传播信息写入一个事务传播记录 TxMsg{PeerID, TxHash, ForwardTime}，所有 TxMsg 记录都存储在 TxMsgPool 中。由于 NetworkObserverNode 同时与多个邻居节点相连，因此多个节点会将同一事务转发给 NetworkObserverNode。因此，TxMsgPool 中会有多个 TxMsg 对应同一个交易，即这些 TxMsg 记录的 TxHash 字段是相同的。因此，我们可以从 TxMsgPool 中提取同一笔交易对应的所有 TxMsg 记录，并选择它们来构造同一笔交易的一组交易传播记录： TxMsgSet = { TxMsg[1],TxMsg[2],... , TxMsg[n]} 其中 TxMsg[i]{PeerID[i], TxHash[i], ForwardTime[i]} 是该交易的第 i 个交易传播记录。在集合 TxMsgSet 中，所有 TxMsg[i] 的 TxHash[i] 相同，每个 TxMsg[i] 的 ForwardTime[i] 是对应的邻居节点转发这个事务到 NetworkObserverNode 的时刻。根据 TxMsgPool 中的交易传播记录 TxMsg，可以为每个交易建立 TxMsgSet 集合。有了记录交易的所有 TxMsgSet 集，我们提出了一种算法来计算将交易广播到以太坊 P2P 网络中的大多数节点的延迟。计算交易广播延迟的算法解释如下：



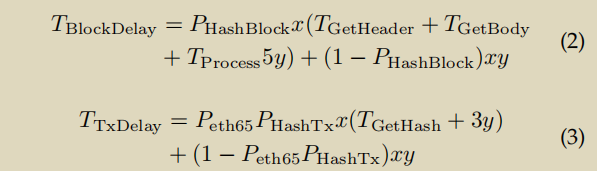
1. 首先，对于每一个向NetworkObserverNode转发事务的邻居节点，我们建立一个空集，称为PeerTimeDiffSet[PeerID]，其中PeerID是邻居节点的网络对等体标识。
2. 2) 我们选择一个事务并为这个选定的事务获取事务传播记录集 TxMsgSet。然后，我们计算 TxMsgSet 中所有事务传播记录 TxMsg 的 ForwardTime 字段的最小值，并将该最小值命名为 minTime，这是该事务被邻居节点转发到 NetworkObserverNode 的最早时刻。
3. 3) 对于TxMsgSet中的每一个TxMsg[i]{PeerID[i],TxHash[i],ForwardTime[i]}，我们首先计算TxMsg[i]中的ForwardTime[i]和minTime的差，即ForwardTime [i] ] 分钟时间；然后，我们把时间差 ForwardTime[i] ? minTime，根据节点的对等标识PeerID[i]，放入对应的集合PeerTimeDiffSet[PeerID[i]]。
4. 4) 我们对每个记录的事务重复步骤 2) 和步骤 3)，以获得所有将事务转发到 NetworkObserverNode 的邻居节点的时间差集 PeerTimeDiffSet[PeerID]。
5. 5) 我们计算 PeerTimeDiffSet[PeerID] 集合中所有条目的平均值，并用 PeerTimeDiffMean[PeerID] 表示每个具有对等标识 PeerID 的邻居节点的平均值。
6. 6) 通过对所有 PeerTimeDiffMean[PeerID] 进行平均，我们可以得到将交易广播到以太坊 P2P 网络中大多数节点的估计平均延迟。
7. 寻找交易广播延迟的实验：

我们使用上述算法和收集到的交易传播记录来分析平均交易广播延迟。但是，在这样做的时候，我们需要确保我们只使用在网络上首次广播的新发行交易的传播记录，而不是由于一些问题而在网络上重复广播的一些以前的交易。因此，有必要在我们的分析中识别以太坊的新交易。该网站 [12] 根据其历史记录实时发布从以太坊主网观察到的新交易。因此，在运行 NetworkObserverNode 测量交易传播记录时，我们使用爬虫来收集网站发布的新交易信息。当我们分析交易广播延迟时，我们只检查新交易的传播记录。为保证 NetworkObserverNode 收集的交易传播记录能够尽可能反映以太坊网络的特点，在每次进行数据分析时，我们保证 NetworkObserverNode 与分布的 20 多个邻居节点相连世界各地的。我们在 2020 年 6 月 9 日至 2020 年 6 月 17 日期间每天在以太坊主网上进行网络测量，以获取交易传播记录并估计每天的平均交易广播延迟。图 12 展示了分析的平均交易广播延迟的结果。我们可以看到，以太坊 P2P 网络的交易广播延迟在测量期间相对稳定，在 200 毫秒左右略有波动。因此，我们在后面的分析中将以太坊 P2P 网络的平均交易广播延迟视为 200 毫秒。



* 1. 广播消息所需跳数的分析方法

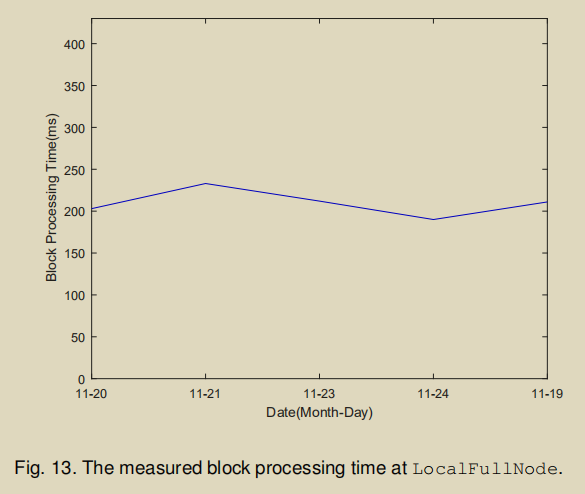
在这一部分中，我们提出了以太坊 P2P 网络中交易和区块广播延迟的模型，并使用所提出的模型和消息广播延迟的测量结果来分析在以太坊 P2P 上广播交易和区块所需的跳数 网络。 向大多数以太坊节点广播交易和区块的延迟可以在数学上建模为



其中变量的含义和使用值描述如下：

* ? T 是以太坊 P2P 网络的平均交易广播延迟 TxDelay。在 5.2 节中，我们已经发现在 2020 年 6 月 9 日和 2020 年 6 月 17 日的测量时间内，它的值在 200 毫秒左右。
* ? T 是以太坊 P2P 网络 BlockDelay 的平均区块广播延迟。目前，一些机构和团队已经测量了平均区块广播时间，并在网站上实时发布结果[42]。我们可以从网站 [42] 中发现，从 2020 年 6 月 9 日到 2020 年 6 月 17 日，平均区块传播时间为 477 毫秒。 x 是从 NetworkObserverNode 的邻居节点向以太坊 P2P 网络中的大多数节点广播交易和块所需的跳数。
* ? y 是通过一跳传播事务或块的平均时间。 （2）中，5y表示节点收到区块哈希后需要5次消息传播才能获得区块，如图4所示。同理，（3）中的3y表示节点需要3次消息传播收到区块哈希后获取区块，如图6所示。
* ? TGetHeader 是节点收到区块哈希后等待获取区块头的时间。其值为 400 毫秒。
* ? TGetBody 是节点收到区块哈希后等待获取区块体的时间。其值为 100 毫秒。
* ? TGetHash是eth65节点收到交易hash后等待获取交易的时间。其值为 500 毫秒。
* ? TProcess 是在节点处处理块的平均时间。图 13 显示了 LocalFullNode 在测量期间的块处理时间，它在 200 毫秒左右波动。因此，我们在分析中将变量 TProcess 的值视为 200 ms。
* ? Peth65 是当前网络中 eth65 节点的比例。从 2020 年 6 月 9 日到 2020 年 6 月 16 日，我们观察到共有 1380 个节点连接到 NetworkObserverNode 和 LocalFullNode，其中 40% 是 eth65 节点。因此，Peth65 设置为 0.4。
* ? PHashBlock(PHashTx) 是节点先接收到这些块（交易）的哈希，然后请求这些块（交易）后，收到的块（交易）占该节点接收到的所有块（交易）的比例。我们在第 3 节中讨论过，节点可以直接从其邻居节点接收块/交易，也可以在从其邻居节点接收到块/交易的哈希后请求块/交易。 PHashBlock 和 PHashTx 的值由两个因素决定： i) 节点在接收到块/交易的哈希之后和请求块/交易之前将等待多长时间； ii) 选择多少个节点来转发块/交易的哈希。如第 3 节所述，这两个因素对于块和交易的传播是相同的。因此，我们可以假设 PHashBlock 和 PHashTx 的值在以太坊 P2P 网络中是相同的。我们在 2019 年 11 月的测量期内进行了以太坊主网 PHashBlock 值的实验测量。我们统计了每天 LocalFullNode 收到的块数，以及 LocalFullNode 收到块哈希后请求的块数.图 14 显示了测量结果。我们可以看到，确实有一部分区块是从区块哈希中获得的。在测量期间，从块哈希中获得的块数为 2723，所有接收到的块数为 20042，因此 PHashBlock = 0.135。因此，我们在分析中将 PHashBlock 和 PHashTx 的值都视为 0.135。

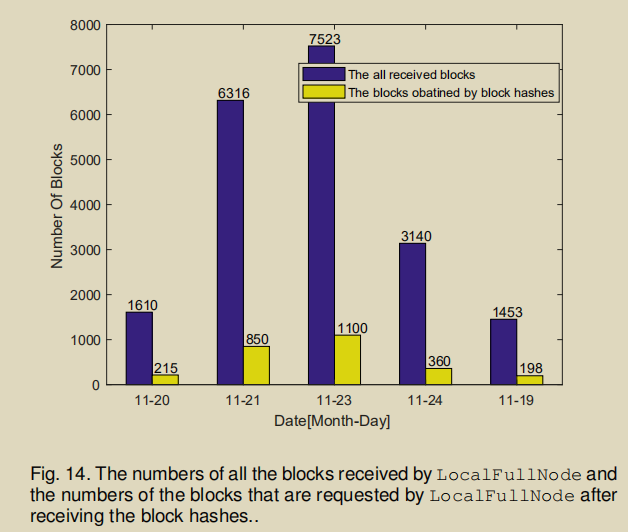
至此，我们已经确定了（2）和（3）中除x和y之外的所有变量的值。 因此，将变量值代入（2）和（3）后，我们可以求解得到 x ≈ 3.7，这意味着平均而言，一个块/交易从 NetworkObserverNode 的邻居节点之一广播到 整个以太坊网络需要 3.7 跳。 这一结果表明，以太坊 P2P 网络具有一定的小世界网络效果 [43, 44]，即网络中的一个节点到达另一个节点需要不超过 6 跳。8



**6 CONCLUSION**

我们提出了以太坊网络分析仪 Ethna 来探测和分析以太坊区块链的 P2P 网络。 Ethna 在以太坊主网上设置探测节点收集消息传播记录，利用以太坊消息转发协议的随机特性分析以太坊 P2P 网络的拓扑特征。 我们测得以太坊节点的平均度数为47，有少数超级节点度数大于1000； 与比特币、门罗币等区块链系统的P2P网络类似，以太坊P2P网络的度分布遵循幂律，具有无标度网络的特点。

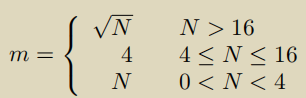
1. *分析P2P网络是否为小世界网络的严格方法是构建整个P2P网络的拓扑，然后用拓扑比较网络的度分布、最短路径和聚类系数这个 P2P 网络与一个随机网络，如 [45] 中的方法。由于以太坊区块链是一个承载着极高经济价值的系统，难免会受到各种攻击。一旦一些恶意节点可以完全推断出网络拓扑，它们就可以对以太坊 P2P 网络进行各种攻击，就像对比特币 P2P 网络的一些识别攻击一样 [10, 11]。为了防止这些攻击，以太坊 P2P 网络的设计故意使其难以推断其 P2P 网络的拓扑结构。例如，以太坊 P2P 网络协议没有散布节点间通信的时间戳；而以太坊 P2P 网络协议使用 K 个桶来维护节点信息，并从这 K 个桶中随机建立节点之间的连接。因此，[45] 中使用的分析方法不能应用于以太坊 P2P 网络。在这项工作中，测量了在以太坊 P2P 网络中广播消息所需的跳数，表明以太坊 P2P 网络具有一定的小世界网络效应。严格验证以太坊 P2P 网络是否是一个小世界网络具有挑战性，并且是一个悬而未决的问题*



此外，我们对消息广播延迟进行建模，并使用收集的消息传播记录分析通过网络广播消息所需的跳数。 我们发现消息可以在 6 跳内广播到大多数以太坊节点。 这一结果表明，以太坊 P2P 网络存在小世界效应。

**APPENDIX**

1. 为什么不能用eth64节点转发的事务来分析eth64节点的度数：本附录先说明为什么不能用eth64节点转发的事务来分析eth64节点的度数。 当一个eth64节点的下游邻居节点数N小于16时，(1)中的公式不适用。 每次，一个 eth64 节点会从其 N 个下游邻居节点中随机选择 m 个节点来转发该块，m 的值由下式给出



之后，eth64 将区块哈希转发给剩余的 N Nm 个下游邻居节点。 如（4）所示，当 eth64 节点 4 ≤ N ≤ 16 时，对于这个 N 范围，m 等于 4。因此，我们无法从 m 中知道 N。 由于 eth64 节点的块转发策略的这种限制，我们无法使用 eth64 节点转发的事务来衡量 eth64 节点的度数较低的度数。

1. 对于eth65节点或者eth64节点我们不能使用区块转发来分析节点度的原因：

然后这个附录解释了为什么我们不能使用块包数（BlockPacketCount）和块哈希包数（BlockHashPacketCount）来分析eth65节点或eth64节点的节点度，即为什么我们不能将（1）中的TxPacketCount替换为BlockPacketCount 并将（1）中的 TxHashPacketCount 替换为 BlockHashPacketCount 以计算每个 eth65 或 eth64 节点的相邻下游节点的数量。 eth64节点或eth65节点在测量期间发送的block包和block hash包的数量太少。根据网站 [12]，目前以太坊每秒产生 30-50 笔新交易，每 15 秒产生一个区块。如果 NetworkObserverNode 或 LocalFullNode 与某个节点保持稳定连接 1 小时，NetworkObserverNode 或 LocalFullNode 可以从该节点接收大约 240 个块和块哈希包，而 NetworkObserverNode 或 LocalFullNode 可以从该节点接收大约 108000 个事务和事务哈希包。此外，NetworkObserverNode 或 LocalFullNode 与每个节点的连接时间通常小于 1 小时，因此每个节点的 BlockPacketCount 值和 BlockHashPacketCount 值甚至小于 240。由于（1）中的公式仅在一个大量消息随机转发以保证统计属性，使用 BlockPacketCount 和 BlockHashPacketCount 来衡量节点的度数是不准确的。