一种面向元宇宙中物联网数据可靠性的创新双层DAG反应式协议

Changlin Yang、Ying Liu、Kwan-Wu Chin、Jiguang Wang、Huawei Huang、Zibin Zheng

arXiv:2304.05094v1 [cs.DC] 11 Apr 2023

摘要——许多应用（例如数字孪生）依赖来自物联网（IoT）网络的感知数据，用于推断事件并采取行动以影响环境。这引发了对数据完整性和溯源的担忧。区块链是解决这些问题的一种可能方案。然而，区块链的资源需求很高，因此不适合用于资源受限的物联网设备。为此，本文提出了一种名为两层有向无环图（2LDAG）的新方法，物联网设备仅存储其邻居生成的数据的数字指纹。此外，本文还提出了一种新颖的路径证明（PoP）协议，使运营者或数字孪生能够按需验证数据。仿真结果表明，与传统区块链以及采用有向无环图结构的区块链相比，2LDAG的存储成本和通信成本分别低两个和三个数量级。此外，即使49%的节点是恶意节点，2LDAG仍能实现共识。

索引词——数据可靠性、图、哈希、信令、分布式共识。

I. 引言

元宇宙有望通过紧密耦合数字世界与物理世界，彻底改变互联网 [1], ，其中数字孪生体（DTs）用于表示某个物理对象/实体，例如个人、机器或整个工厂。为此，数字孪生体需要来自物联网（IoT）网络中设备的数据。例如，某设备可以采集某个人的健康数据，然后由其数字孪生体使用这些数据 [2]。事实上，物联网网络构成了连接物理世界与元宇宙的桥梁。如图 1 所示，在感知层运行的设备收集信息，并通过网关将其发送至元宇宙层。随后，数字孪生体利用收集到的数据对物理实体/过程进行建模和仿真。此外，这些数字孪生体还可以根据接收到的数据发起智能合约 [3]。

因此，数据可靠性是任何用户或数字孪生体都关注的关键问题。当数字孪生体被用于决策时，这一点尤为重要；例如，运营者可以利用数字孪生体进行模拟，以确定如何在工厂分配资源，或何时对机器进行维护。此外，

本文已被第43届IEEE分布式计算系统国际会议（ICDCS 2023）接收。C. Yang、Y. Liu、H. Huang 和 Z. Zheng 隶属于中国广东珠海中山大学软件工程学院（电子邮件：{yangchlin6, liuy2368, huanghw28, zhzibin}@mail.sysu.edu.cn）。K. Chin 隶属于澳大利亚新南威尔士州伍伦贡大学电气、计算机与电信工程学院（电子邮件：kwanwu@uow.edu.au）。J. Wang 隶属于中国河南郑州中原工学院计算机科学学院（电子邮件：2020107237@zut.edu.cn）。通讯作者：Ying Liu。

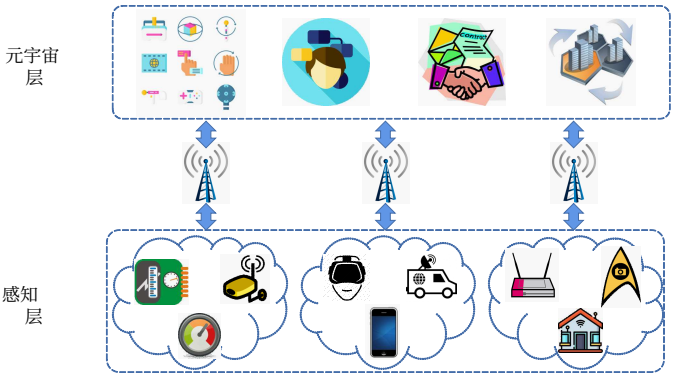


图1。一个元宇宙和物联网示例。设备收集数据，然后通过网关发送到元宇宙应用。

用户/数字孪生体可能希望审计收集到的数据，这就要求数据可追溯、不可篡改且透明 [3]。

一种可能的解决方案是采用区块链，其中数字孪生体利用不可篡改性、可追溯性和完整性等特性 [4]。因此，已有许多研究将区块链应用于安全数据和信息交换 [5]。其中一项研究是 [6], ，其利用区块链在供应链中提供端到端的数据完整性。

遗憾的是，区块链存在许多可扩展性问题，限制了其在物联网设备上的应用 [7]。首先，区块链的存储需求很大，每个参与节点都需要存储一份完整区块链的副本。例如， 2022 [8]比特币区块链的大小已超过 380 GB，而瑞波区块链（又称 XRP账本）的大小超过 14 TB [9]。其次，区块链的交易吞吐量有限，这与数据写入区块的速度有关。例如，比特币和以太坊每秒处理的平均交易数分别为 7 笔和 15 笔 [10]。因此，区块链不适用于会产生频繁交易的应用。第三，区块链的通信成本很高，因为完整区块链需要发送给所有参与节点。例如，比特币节点每月可能上传超过 200 GB 的数据 [11]。最后，我们注意到，区块链资源需求高的一个关键原因是它采用了主动共识协议。这意味着，即使参与者并未使用某笔交易或某些数据，节点仍会通过共识协议耗费资源来验证数据。因此，所有参与者都必须存储足够的信息，以验证其他参与者生成的数据，从而导致较高的存储和通信开销。

为此，我们设计了一种新颖的分布式方法，称为两层有向无环图（2LDAG），以确保数据

在物联网网络中的可靠性。简而言之，每个物联网节点仅存储其生成的数据及对应的哈希值，以及其邻居生成的数据及对应的哈希值。节点仅与邻居互相发送和接收哈希值，并且可以并行生成证明。这些证明构成一个有向无环图，将物联网网络中的所有数据关联起来。此外，我们提出了一种新颖的路径证明（PoP）协议，使任意节点都能追踪并验证源节点的数据。值得一提的是，这是一种反应式共识协议，也就是说，只有当节点需要验证源节点的数据时，数据验证过程才会启动。总而言之，本文的贡献如下：

- 它提出了一种称为 2LDAG 的架构，用于确保物联网数据的完整性和可靠性。与基于区块链的解决方案相比，它具有以下优势：提升存储、通信和吞吐量。
- 它提出了一种新颖的 PoP 协议，以反应式和分布式方式运行，即使存在大量恶意节点，也能正确运行。
- 它表明，2LDAG能够抵御多数攻击、女巫攻击、拒绝服务（DoS）攻击、日蚀攻击和自私攻击。
- 本文首次研究了 2LDAG。仿真结果表明，2LDAG 的存储和通信需求比基于区块链的解决方案分别低两个和三个数量级。

接下来，第二节讨论旨在扩展区块链和有向无环图区块链的解决方案。第三节和第四节分别展示2LDAG架构并概述PoP。第六节给出了数值模拟和结果。第七节总结了本文。

II. 相关工作

物联网或无线传感器网络（WSN）中的数据可靠性数十年来一直备受关注 [5]。传统解决方案利用消息身份验证代码 [12] 或公钥基础设施 [13] 来确保数据完整性。然而，这些解决方案是中心化的，因此推动了区块链 [5]等去中心化解决方案的发展。

为此，我们仅评述考虑资源受限节点的区块链相关研究。具体而言，我们首先评述降低区块链存储和通信需求并提高其吞吐量的解决方案。随后，我们讨论为物联网网络设计的共识协议。

A. 区块链可扩展性

旨在扩展区块链的研究提出了轻节点、修剪节点、分片、编码区块链和有向无环图（DAG） [7]。以下段落评述其主要思想和局限性。

Nakamoto [14] 提出了所谓的轻节点，这些节点仅存储区块头。然而，轻节点依赖全节点存储完整的区块链并验证数据。尽管轻节点的存储需求较低，Nakamoto 的方法会导致数据可用性攻击 [15]。

精简节点会移除生成新区块时不需要的信息。例如，[14] 中的节点会移除未花费的交易输出（UTXO）；这些输出仅在一笔交易中使用，不用于创建区块。然而，两者轻节点和精简节点都会永久删除数据，这意味着无法再追踪区块链数据的来源。

分片 [16] 是一种将参与者划分为所谓分片的方案。每个分片都维护一条独立的子链，因此，与传统区块链相比，每个分片中的参与者所需的存储需求较低，吞吐量较高，尤其是在分片数量增加时。然而，跨分片活动需要更高的信令复杂度。此外，由于每个分片中的参与者数量较少，这种方案会牺牲安全性。

编码区块链 [17] 利用纠错码对区块链数据进行编码，并以分布式方式将其存储在参与者处。然而，编码区块链需要额外的计算来进行编码和解码。此外，编码区块存在无法解码的概率 [17]。

DAG区块链 [18] 将区块链的结构从链式结构改为图结构。因此，交易无需排队等待添加到链的末端，从而显著提高了区块链的吞吐量。IOTA 或 Tangle [19] 就是DAG区块链的例子。IOTA 中的每笔交易都会验证此前生成的两笔交易。网络中没有“矿工”节点，交易可自由且同时生成。然而，DAG区块链节点需要存储整个图，即整个区块链，才能验证其他节点生成的区块。因此，它不适用于存储空间有限的物联网设备。

B. 共识协议

即使某些节点是恶意或不诚实的，共识协议也能帮助节点以分布式方式就信息达成一致。关于共识协议的全面综述，请参阅 [20]。本文仅讨论适用于资源受限设备的共识协议。这些协议要么采用基于证明的机制，要么采用实用拜占庭容错（PBFT）协议。

权益证明（PoS） [21] 中，矿工根据其权益而非计算能力（与 PoW 不同）竞争生成区块。因此，PoS 适用于使用货币奖励的物联网应用，这些奖励可用作“权益”。在经过时间证明（PoET） [22], 中，节点通过一个最短随机等待计时器来生成区块。然而，PoET 中的验证需要可信执行环境。这类环境通常部署在特定硬件上，因此不适用于广泛的物联网应用 [21]。

基于PBFT的共识协议具有高吞吐量、低延迟和低计算开销，因此适用于物联网网络 [20]。然而，传统 PBFT 由于采用三阶段协商流程，通信开销较高。为此，许多研究致力于降低上述通信开销。例如，文献 [23] 提出了 HoneybadgerBFT，该协议利用纠错码降低使用 PBFT 传播区块时的通信开销。然而，这会带来额外的计算复杂度。

C. 讨论

现有的区块链可扩展性解决方案尚未在降低存储、通信和计算成本的同时

为物联网应用保证足够的吞吐量和安全性。一个关键原因是，这些区块链采用主动共识协议，节点必须就生成的数据达成共识。受区块链的不可篡改性、可靠性和可追溯性特征启发，在以下部分中，我们将介绍 2LDAG 和 PoP。

III. 2LDAG架构

2LDAG具有物理层和逻辑DAG层。在物理层，节点维护邻居列表，使用生成的数据构建数据区块，并向邻居传输/接收哈希值。逻辑层将数据区块互连，其中数据区块的哈希值形成有向无环图。该有向无环图用于验证数据区块。

A. 物理层

我们将物联网网络建模为图 $\mathcal{G}(\mathcal{V},\mathcal{E})$。它包含一组 $\mathcal{V}$静态节点，这些节点通过集合 $\mathcal{E}$中的无向边相互连接。每个节点由 $i \in \mathcal{V}$进行索引，且最多有 γ 个节点是恶意的。请注意，我们不考虑信任管理，并假设存在一种补充方法可将设备注册到网络中。每条边表示为 $e_{ij} \in \mathcal{E}$，其中 i、$j \in \mathcal{V}$。对于节点 i，其邻居集合为

$$\mathcal{N}(i) = \{j \mid e_{ij} \in \mathcal{E}, j \in \mathcal{V}\}. \tag{1}$$

所有节点都具有拓扑信息，即它们知道$\mathcal{G}(\mathcal{V},\mathcal{E})$。每个节点以 r_i bit/s 的速率生成数据。我们强调，节点仅存储自身生成的数据；也就是说，节点不会将其数据转发给 $\mathcal{V}$中的任何其他节点。我们用 m_i 表示节点 i的存储容量。

节点将其生成的数据存储在所谓的数据区块中，每个数据区块包含两个部分：主体和头部。主体部分存储节点收集的数据，大小固定为 C 比特。这意味着节点 i 每 C/r_i 秒生成一个数据区块。头部部分包含用于维护逻辑层的数据；详情请参见第III-B节。

我们使用 $b_{i,t}$ 来表示由节点i生成的第 t个数据区块；我们还使用 t 来表示$b_{i,t}$ 的生成时间。令 $b^h_{i,t}$ 和 $b^d_{i,t}$ 分别表示 $b_{i,t}$ 的头部和主体。节点 i 将其所有数据区块记录在集合 $\mathcal{S}_i$ 中。

节点 i 生成数据区块 $b_{i,t}$后，会将相应的摘要 $H(b^h_{i,t})$ 传输给其在 $\mathcal{N}(i)$ 中的所有邻居，其中 $H(.)$ 表示哈希函数。节点处的数据生成过程将在第 III-D 节中介绍。

B. 数据块结构

节点使用的数据区块结构如图 2 所示。区块头由以下字段组成：

- 版本用于跟踪2LDAG架构和协议的更改与更新。其大小为 f^v。
- 时间记录数据区块的生成时间。我们用f^t 表示其大小。
- 根对应于默克尔树根。其大小为 f^H，也就是哈希大小。
- 摘要记录了节点 i 从其邻居收到的哈希值，即 $H(b^h_{j,t})$ $\forall j \in \mathcal{N}(i)$，以及其哈希值的前一个数据区块，即 $H(b^h_{j,t-1})$。该字段的大小为 $f^H \times (n+1)$，其中 $n = |\mathcal{N}(i)|$；注意，$|.|$ 表示集合的基数。

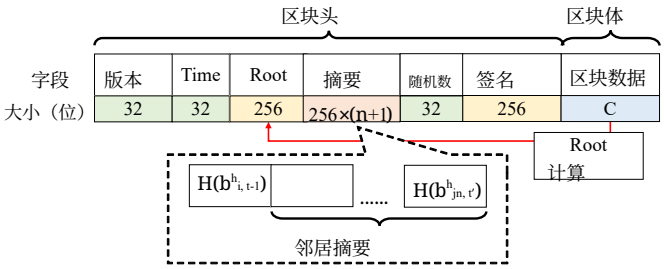


图 2。2LDAG 的数据块结构。版本、时间和随机数的大小为 32位。根和签名的大小为 256 位。摘要字段包含节点生成的最新区块的摘要，以及其邻居的摘要。这意味着其大小为 $256 \times (n+1)$，其中 n 是邻居的数量。所有数据区块的主体大小为 C 位。

- 随机数是一个数字，用于确保版本、时间、根、摘要和随机数的哈希满足给定的难度级别。选择此难度是为了使设备能够快速找到合适的随机数，详情请参见第 III-D 节。其大小为 f^n。
- 签名采用公钥-私钥方案计算。每个节点 i 都有公钥 pk_i 和私钥 sk_i。令 $E(.)$ 和 $D(.)$分别表示加密函数和解密函数。我们可以采用低复杂度加密方案；例如，[24] 中为智能家居应用提出的方案。令 m 为区块头中的字段。签名字段包含 $s_{i,t} = E(m, sk_i)$。$s_{i,t}$ 的大小为 f^s。

由于区块主体仅包含采样数据，且大小为C 位，因此节点 i 处数据区块的总大小为

$$f_i = f_c + f^H(|\mathcal{N}(i)| + 1) + C, \tag{2}$$

其中，f_c 是一个定义如下的常数

$$f_c = f^v + f^t + f^H + f^n + f^s. \tag{3}$$

C. 逻辑层

定义 $\mathcal{G}(\mathcal{B},\mathcal{L})$为一个有向无环图，其中数据区块集合为 $\mathcal{B} = \cup_{\forall i \in \mathcal{V}} \mathcal{S}_i$。集合 $\mathcal{L}$ 包含连接数据区块的有向边。若数据块头$b^h_{j,\,t'}$ 包含区块头$b^h_{i,\,t}$的摘要，则 $\mathcal{L}$ 中存在一条边($b_{i,\,t}$, $b_{j,\,t'}$)。我们称$b_{i,\,t}$ 为$b_{j,\,t'}$的父块，并称$b_{j,\,t'}$ 为$b_{i,\,t}$的子块。本文中，如果上下文明确，我们省略 i、t，并将$b_{i,\,t}$ 写作 b_x。

定义从数据区块 b_x 到 b_y 的路径为

$$\mathcal{P}_{x,y} = \{b_{n_1}, \ldots, b_{n_q}, \ldots, b_{n_Q}\}, \tag{4}$$

其中，$n_1 = x$、$n_Q = y$ 和 Q 是路径长度。对于路径$\mathcal{P}_{h,\,k}$中任意两个相邻的数据区块 b_{n_q} 和 $b_{n_{q+1}}$，数据区块b_{n_q} 是 $b_{n_{q+1}}$的父块。此外，我们称数据区块 b_y 是 b_x的后代块。若节点 i 中的数据区块 b_y 是 b_x的后代块，则称节点 i 指向区块 b_x 。

D. 数据块生成

在网络初始化时，即 $t = 0$，每个节点 $i \in \mathcal{V}$生成一个创世块；$b_{i,0}$。然后，节点 i 计算该

摘要 $H(b_{i,0}^h)$ 并将其发送给 $\mathcal{N}(i)$ 中的每个邻居节点 j。当节点 i 从邻居 j 收到摘要 $H(b_{j,0}^h)$ 后，节点 i 构建集合 $\mathcal{A}_i = \{H(b_{j,0}^h) \mid j \in \mathcal{N}(i)\}$，其中包含 $\mathcal{N}(i)$ 中每个邻居的最新摘要；注意，我们有 $|\mathcal{A}_i| = |\mathcal{N}(i)|$。

给定 C 位数据，节点 i 构建数据区块 $b_{i,t}$。在 $b_{i,t}$ 的头部中，节点 i 记录版本和时间，并计算根 $M(b_{i,t}^d)$，其中 $M(.)$ 是默克尔树根函数。随后，它在摘要字段（记为 Δ_i）中纳入从邻居接收到的摘要，以及其前一个数据区块 $H(b_{i,t-1}^h)$ 的摘要；即有 $\Delta_{i,j} = \mathcal{A}_i \cup \{H(b_{i,t-1}^h)\}$。接下来，节点 i 确定一个随机数$n_{i,t}$，使其满足

$$H(M(b_{i,t}^d), \Delta_{i,j}, n_{i,t}) \leq \rho, \tag{5}$$

其中 ρ 是一个较小的固定难度级别，使节点能够快速找到合适的$n_{i,t}$，例如在几秒内完成。之后，节点i 使用版本 v、时间 t、根$M(b_{i,t}^d)$、摘要 $\Delta_{i,j}$ 和随机数 $n_{i,t}$计算签名字段。形式化地，

$$s_{i,t} = E(H(v, t, M(b_{i,t}^d), \Delta_{i,j}, n_{i,t}), sk_i). \tag{6}$$

最后，节点 i 将摘要 $H(b_{i,t}^h)$ 传输给 $\mathcal{N}(i)$ 中的所有邻居，并将数据区块 $b_{i,t}$ 添加到其存储 $\mathcal{S}_i$ 中。邻居 j 从节点 i 接收到摘要 $H(b_{i,t}^h)$ 后，通过将 $H(b_{i,t-1}^h)$ 替换为 $H(b_{i,t}^h)$ 来更新集合 $\mathcal{A}_j$。

下面通过一个示例来说明 2LDAG。参见图 3，共有四个物理节点 A、B、C和 D。它们的邻居集合分别为 $\mathcal{N}(A) = \{B\}$、$\mathcal{N}(B) = \{A, C, D\}$、$\mathcal{N}(C) = \{B, D\}$ 和 $\mathcal{N}(D) = \{B, C\}$。假设节点 D 首先生成一个区块，例如 D_1。它将摘要$H(D_1^h)$发送给节点 B 和 C。当节点 C 生成一个区块，例如 C_1时，其中包含来自节点 D的摘要 $H(D_1^h)$。随后，节点 C将摘要 $H(C_1^h)$发送给节点 B 和 D。同样，在节点 A 生成区块 A_1后，摘要 $H(A_1^h)$会被发送给节点 B。因此，节点 B 生成的区块 B_1 包含摘要 $H(A_1^h)$ $H(C_1^h)$和 $H(D_1^h)$。如图 3 所示，存储在这些区块中的相应摘要构成一个有向无环图。请注意，每个节点只存储自身生成的数据区块，并且只传输区块摘要。例如，节点B 仅在其本地存储中存储数据区块 B_1，而节点 B 的通信成本与其同节点 A、C和 D之间收发的三个摘要有关。这说明了 2LDAG 为何具有较低的存储和通信成本。

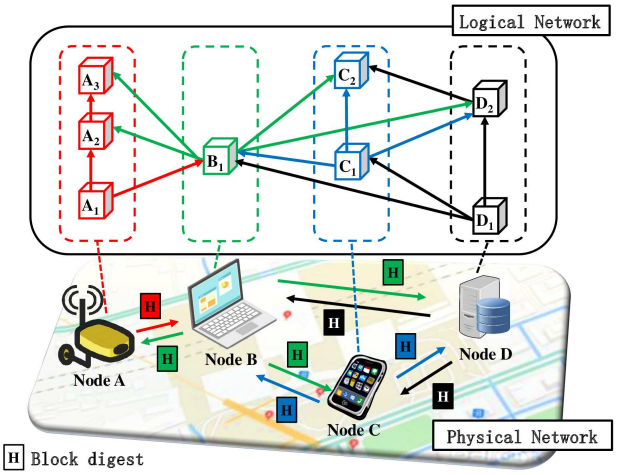


图 3. 一个2LDAG示例。在物理网络中，每个节点仅存储其自身的数据区块；见彩色虚线框。此外，节点仅向邻居传输摘要；见带有“H”的彩色方框。在逻辑网络中，数据区块通过有向边连接并形成有向无环图。

IV. 路径证明协议

当一个节点（例如 i）需要验证存储在某个节点（例如 j）中的区块内容，即$b_x = b_{j,t} \in \mathcal{S}_j$时，PoP过程便会启动。我们将节点 i 称为验证者，将节点 j 称为验证器。节点 i 首先从节点 j获取区块 $b_{j,t}$，其中包括区块头$b_{j,t}^h$ 和区块主体 $b_{j,t}^d$。利用上述有向无环图，节点 i 旨在识别一组随后存储在集合 $\mathcal{R}_i$中的节点；这些节点包含一个指向区块 $b_{j,t}$的区块。当满足以下条件时，区块 $b_{j,t}$ 的共识即达成：$|\mathcal{R}_i| \geq \gamma + 1$，其中 γ 是可容忍的恶意节点数量。换言之，那里是一条路径 $\mathcal{P}_i$，其中至少包含 $\gamma + 1$ 个就区块$b_{i,\ j}$的完整性达成一致的节点。

最初，我们有 $\mathcal{R}_i = \{j\}$ 和 $\mathcal{P}_i = \{b_{j,t}\}$。将路径 $\mathcal{P}_i$ 上最后添加的区块定义为待验证区块，记为 $b_{v,t}$，其初始设置为 $b_{v,\ t} = b_{j,t}$。要更新上述集合，请令 $b_{j',t'}$ 为 $b_{v,t}$的子数据区块。随后，验证者获取相应的区块头 $b_{j',t'}^h$，并检查值 $H(b_{v,t}^h)$是否位于 $b_{j',t'}^h$ 中。如果是，验证者进行以下更新：(1) 将区块 $b_{j',t'}$ 添加到 $\mathcal{P}_i$，(2) 将节点 j' 添加到 $\mathcal{R}_i$，以及 (3) 将 $b_{v,t}$ 替换为 $b_{j',t'}$。

为阐释PoP，请参见图4。考虑 $\gamma = 2$，这意味着如果 $\mathcal{R}_i$ 中有三个节点，则可达成共识。假设节点 B、C 和 D 彼此互为邻居。节点 A 的唯一邻居是节点 B，而节点 E 的唯一邻居是节点 D。假设有请求验证区块 $B1$。那么两条可能的路径是 $\mathcal{P} = \{B1, D1, E2\}$ 和 $\mathcal{P}' = \{B1, A2, B2, C2\}$。两条路径都包含三个不相交节点，这意味着有 γ+1 个节点直接或间接地验证区块$B1$的完整性。因此，如果验证者构建这两条路径中的任意一条，即可达成共识。然而，路径 $\mathcal{P}'$ 更长，这意味着选择路径 $\mathcal{P}'$ 的验证者需要获取额外的数据块头。

接下来，我们提出一种算法，以减少验证者在选择数据区块$b_{v,t}$的下一个验证节点时所检索的区块头总数。

A. 加权路径选择（WPS）

选择验证节点 $b_{v,\ t}$时需要考虑两种情况。在情况 1 中，如果验证节点 i 选择了存储在节点 $j' \in \mathcal{N}(v)$ 中的一个区块，而该节点已被纳入 $\mathcal{R}_i$，那么 j'不会对区块 $b_{j,\ t}$的共识作出贡献。因此，验证节点 i 需要选择一个不在 $\mathcal{R}_i$中的节点 j'。在情况 2 中，节点 v 的所有邻居都可能已在 $\mathcal{R}_i$中，此时选择任何一个邻居的效果都相同。

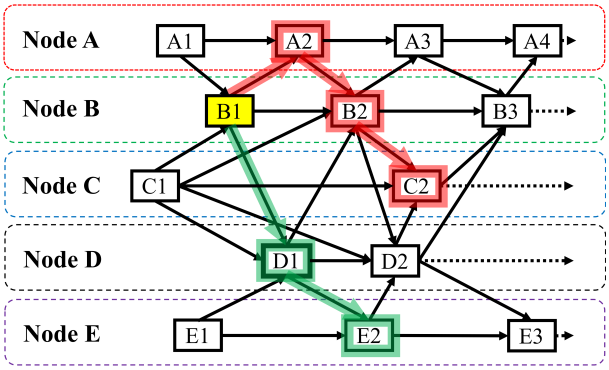


图 4. PoP 示例。验证区块 $B1$ 时，验证者可以获取区块 $D1$ 和 $E2$ 的头部（见绿色路径），也可以获取区块 $A2$、$B2$ 和 $C2$ 的头部（见红色路径）。两条路径都能达成共识，但绿色路径包含的数据区块更少。

为节点 v 引入权重 w_v。其中，w_v 定义为节点 v 的邻居中未包含在 $\mathcal{R}_i$ 中的比例，即，

$$w_v = \frac{|\mathcal{R}_i \cap \{\mathcal{N}(v) \cup \{v\}|}{|\mathcal{N}(v)| + 1}. \tag{7}$$

节点 i 随后选择 $j' \in \mathcal{N}(v)$ 中 $w_{j'}$ 值最小的一个。从数学上来说，

$$j' = \arg\min_{\hat{v} \in \mathcal{N}(v)} w_{\hat{v}}. \tag{8}$$

如果多个邻居具有相同的权重，算法会选择不在 $\mathcal{R}_i$ 中的那个。

WPS 的伪代码如算法 1 所示。其输入包括 $\mathcal{R}_i$ 和 $\mathcal{N}(v)$。其输出是一个节点 j'，该节点存储要纳入路径 $\mathcal{P}_i$ 的 b_v 的子块，t。节点 i 首先使用式（7）计算每个 $j^* \in \mathcal{N}(v)$ 的权重 w_{j^*}；见第 1 至 3 行。随后，它选择权重最小的节点，并将其加入候选节点集合 $\mathcal{Z}$；见第 4 行。如果只有一个节点 $j' \in \mathcal{Z}$，则返回节点 j'；见第 5 至 7 行。如果有 $|\mathcal{Z}| \geq 2$，则有两种情况：1）如果 $\mathcal{Z} \cap \mathcal{R}_i = \emptyset$ 或 $\mathcal{Z} == \mathcal{R}_i$，则可以选择 $\mathcal{Z}$ 中的所有节点；或者 2）如果 $\mathcal{Z} \cap \mathcal{R}_i \neq \emptyset$，则节点 i 从 $\mathcal{Z} \backslash \mathcal{R}_i$ 中选择一个；见第 8 至 13 行。

现在，我们使用图 4 来说明加权路径选择。在确定 $B1$ 的下一个待验证区块或节点时，计算节点 B 的三个邻居的权重，得到 $w_A = 0.5$、$w_C = 1/3$ 和 $w_D = 1/4$。请注意，节点 C 没有任何子块属于 $B1$ 的区块，而节点 D 的权重最小。因此，我们选择 $D1$ 作为验证者要添加到路径中的第二个区块。接下来，在确定 $D1$ 的待验证区块或节点时，我们得到其邻居的权重为 $w_B = 0.5$、$w_C = 2/3$ 和 $w_E = 0.5$。我们选择 $E2$，因为节点 B 存在于 $\mathcal{R}_i$ 中。

B. 信任路径选择 (信任路径选择)

从图 4 中的示例可以看出，当验证区块 $C1$ 时，可能需要再次获取 $D1$ 和 $E2$。这会浪费计算和通信资源。为此，我们要求节点（例如 i）将其已验证的区块头存储在集合 $\mathcal{H}_i$ 中。然后，节点 i 能够直接构建

算法 1：加权路径选择（加权路径选择）

输入：$\mathcal{R}_i, \mathcal{N}(v)$
输出：j'
对每个 $j^* \in \mathcal{N}(v)$ 执行 1 次 do
2 | 计算 w_{j^*}，使用公式 (7)
3 结束
4 $\mathcal{Z} = \arg\min_{\hat{v} \in \mathcal{N}(v)} w_{\hat{v}}$
5 如果 $|\mathcal{Z}| = 1$，则
6 | 将 j' 设置为 $\mathcal{Z}$中唯一的元素
7 结束
8 如果 $|\mathcal{Z}| \geq 2$ && $\mathcal{Z} \cap \mathcal{R}_i = \emptyset$ 或 $\mathcal{Z} == \mathcal{R}_i$ 那么
9 | 将 j' 设置为 $\mathcal{Z}$ 中的一个随机元素
10 结束
11 如果 $|\mathcal{Z}| \geq 2$ && $\mathcal{Z} \cap \mathcal{R}_i \neq \emptyset$ 那么
12 | 将 j' 设置为 $\mathcal{Z} \backslash \mathcal{R}_i$ 中的一个随机元素
13 end
14 返回 j'

一条路径 $\{C1, D1, E2\}$ 使用 $\mathcal{H}_i$ 中的区块。这通过 TPS算法实现，下面将对此进行详细介绍。

验证节点 i 如果其子区块头在集合 $\mathcal{H}_i$ 中，则可以直接验证区块 $b_{v,t}$，即

$$H(b^h_{v,t}) \in b^h_{j^*,t^*} \in \mathcal{H}_i, \tag{9}$$

其中，b_{j^*,t^*} 是区块 $b_{v,t}$ 的子区块。随后，验证节点 i 可以将待验证区块 $b_{v,t}$ 更新为 b_{j^*,t^*}，将区块b_{j^*,t^*} 添加到集合 $\mathcal{R}_i$ 和路径 $\mathcal{P}_i$ 中。

算法 2 详细说明信任路径选择。它使用 $\mathcal{H}_i$ 中的数据区块头迭代更新$b_{v,t}$。它返回更新后的集合 $\mathcal{R}_i$、$\mathcal{P}_i$ 和 $b_{v,t}$，直到 $\mathcal{H}_i$ 中不再有区块头包含 $H(b_{v,t})$。

算法 2：信任路径选择（TPS）

输入：$\mathcal{H}_i$、$\mathcal{R}_i$、$\mathcal{P}_i$、$b_{v,t}$
输出：$\mathcal{R}_i$、$\mathcal{P}_i$、$b_{v,t}$
1 当 $\exists b^h_{j',t'} \in \mathcal{H}_i$ 使得 $H(b^h_{v,t}) \in b^h_{j',t'}$ 时执行
2 | $\mathcal{R}_i = \mathcal{R}_i \cup \{j'\}$
3 | $\mathcal{P}_i = \mathcal{P}_i \cup \{b_{j',t'}\}$
4 | $b_{v,t} = b_{j',t'}$
5 结束
6 返回 $\mathcal{R}_i$、$\mathcal{P}_i$、$b_{v,t}$

C. 综上所述

现在，我们讨论验证节点和响应节点所执行的流程。在此，响应节点被定义为向验证节点发送信息以验证区块 $b_{v,t}$的节点。

验证者：验证者的伪代码如算法 3 所示。假设验证者 i aims to verify block $b_{j,t}$。它首先从节点 j 获取 $b_{j,t}$，该节点也称为验证器。随后，它计算默克尔树根 $M(b^d_{j,t})$，并将其与头部 $b^h_{j,t}$ 进行核对。如果两者不一致，即 $M(b^d_{j,t})$ 不等于 $GetRoot(b^h_{j,t})$，PoP 就会返回错误。这里，GetRoot($b^h_{j,t}$) 返回 $b^h_{j,t}$ 区块头中 Root 字段的值。否则，在算法 3 第 3 行，验证者 i

进行如下设置： $\mathcal{R}_i = \{j\}$、 $\mathcal{P}_i = \{b_{j,t}\}$ ，并将待验证区块设为 $b_{v,\ t} = b_{j,\ t}$。

接下来，利用区块$b^h_{j,t}$ 的头部$b_{j,t}$，验证者 i 计算摘要 $H(b^h_{j,t})$，并根据信任路径选择，使用集合 $\mathcal{H}_i$ 构建集合 $\mathcal{R}_i$ 和 $\mathcal{P}_i$ 。如果未达成共识，即$|\mathcal{R}_i| \le \gamma$，验证者会为路径 $\mathcal{P}_i$ 寻找下一个区块，并将新节点加入集合 $\mathcal{R}_i$。具体而言，它调用加权路径选择来查找数据区块 $b_{v,t}$的下一个验证节点；加权路径选择找到节点 $j' \in \mathcal{N}(v)$来验证区块 $b_{v,t}$。识别出节点j'后，验证者 i 向节点 j' 发送 REQ CHILD 消息，其中包含 $H(b^h_{v,t})$；参见第 17 行。

一旦验证者 i 在超时 τ内收到来自节点j'的 RPY CHILD 消息（见第 19 行），便从消息中提取 b^h_{j',t^*}（见第 20 行）。随后，它计算 $H(b^h_{j,t})$，并检查存储在 b^h_{j',t^*}中的摘要。函数 GetDigest(b^h_{j',t^*}, v) 返回区块头 b^h_{j',t^*}中节点 v的摘要。如果上述两个摘要一致，即 $H(b^h_{v,t}) == GetDigest(b^h_{j',t^*}, j)$，则节点 j'指向区块 $b_{j,t}$。因此，验证者 i将节点j' 加入集合 $\mathcal{R}_i$ ，并将区块索引 b_{j',t^*} 加入路径 $\mathcal{P}_i$。随后，它将待验证区块 $b_{v,t}$ 设为 b_{j',t^*}。接下来，验证者i 调用 TPS 以继续构建路径（见第 9 行）。否则，如果摘要不一致，或节点 j' 未作出回复，节点 i 会从 $\mathcal{N}(v)$ 中移除 j' ，并选择另一个节点发送 REQ CHILD 消息。如果验证者 i 未能从 $\mathcal{N}'$中的所有节点收到有效的 RPY CHILD 消息（该集合在第 13 行初始化为 $\mathcal{N}(v)$），则回退到前一个待验证区块，并向除 v 之外的节点发送 REQ CHILD 消息（见第26–31 行）。为此，它从 $\mathcal{R}_i$中移除 v，并同时从 $\mathcal{V}'$中移除； $\mathcal{V}'$在第 14 行初始化为 $\mathcal{V}' = \mathcal{V}$。请注意，集合 $\mathcal{P}_i$ 是一个栈，因为有向无环图中的路径没有环。随后，验证者从 $\mathcal{P}_i$ 中弹出 $b_{v,t}$（即栈顶元素），并将待验证区块设为更新后的 $\mathcal{P}_i$栈顶所对应的区块。如果验证者回退到路径上的第一个区块，即验证器区块 $b_{j,t}$，且无法收到有效的 RPY CHILD 消息，即 $\mathcal{R}_i == \emptyset$，算法 3 将返回错误。相反，如果它收到有效的 RPY CHILD 消息，则会设置 $\mathcal{V}' = \mathcal{V}$ ，并继续构建集合 $\mathcal{R}_i$和 $\mathcal{P}_i$。达成共识后，即 $|\mathcal{R}_i| \ge \gamma + 1$，验证者$i$ 会将 $\mathcal{P}_i$ 中所有区块$b_{j,t}$ 的区块头 $b^h_{j,t}$ 存入集合 $\mathcal{H}_i$（见第 39 行）。

响应者：响应者的伪代码如算法 4 所示。它在$b_{j',\ t^*}$ 中找到一个区块，其中其区块头$b^h_{j',\ t^*}$包含 $H(b^h_{v,\ t})$。随后，它向验证者发送一条包含 $b^h_{j',\ t^*}$的消息。形式化地说，节点 j'找到子区块的一个子集 $C_{j'}(b_{v,\ t})$，其中

$$C_{j'}(b_{v,t}) = \{b_{j',t'} \mid H(b^h_{v,t}) \in b_{j',t'}, b_{j',t'} \in \mathcal{S}_{j'}\}. \quad (10)$$

一个或多个区块可能包含摘要 $H(b^h_{v,\ t})$，即$|C_{j'}(b_{v,\ t})| > 1$。这是因为节点生成数据区块的速率各不相同。如果节点 v 的速率较高，其邻居 j' 可能会在多个区块中包含摘要 $H(b^h_{v,\ t})$。例如，如图 3 所示， B_1 的摘要同时包含在 A_2 和A_3中。为此，节点 j' 需要选择最早生成的区块 $b_{j',\ t^*} \in C_{j'}(b_{v,\ t})$，其中

$$t^* = \min\{t' \mid b_{j',t'} \in C_{j'}(b_{v,t})\}. \quad (11)$$

之后，节点 j' 传输一条 RPYCHILD 消息，其中包含$b^h_{j',\ t^*}$ 给验证者 i。

算法 3： 验证者 输入：$b_{j,t}$, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, $\mathcal{H}_i$输出：$\mathcal{H}_i$, ‘成功’或‘错误’ 1 //*** 初始化 ***// 2 $[b^h_{j,t}, b^d_{j,t}] == Request(b_{j,t})$ 3 if $M(b^d_{j,t}) != \text{GetRoot}(b^h_{j,t})$ then 4 return $\mathcal{H}_i$, ‘错误’ 5 end 6 $\mathcal{R}_i = \{j\}$, $\mathcal{P}_i = \{b_{j,t}\}$ 和 $b_{v,t} = \{b_{j,t}\}$7 //*** 构造路径 *** // 8 while True do 9 $(\mathcal{R}_i, \mathcal{P}_i, b_{v,t}) =$ 信任路径选择($\mathcal{H}_i, \mathcal{R}_i, \mathcal{P}_i, b_{v,t}$) 10 if $|\mathcal{R}_i| \ge \gamma + 1$ then 11 break 12 end 13 $\mathcal{N}' = \mathcal{N}(v)$ 14 $\mathcal{V}' = \mathcal{V}$ 15 while $\mathcal{N}' \neq \emptyset$ do 16 $j' =$ 加权路径选择($\mathcal{R}_i$, $\mathcal{N}'$) 17 Send(‘REQ CHILD’, $H(b^h_{v,t})$, j') 18 //*** 等待并接收消息 ***// 19 if Msg = Receive(‘RPY CHILD’) then 20 $b^h_{j',t^*} = GetHeader(Msg)$ 21 if $H(b^h_{v,t}) == GetDigest(b^h_{j',t^*}, v)$ then 22 break 23 end 24 end 25 $\mathcal{N}' \leftarrow \mathcal{N}' \setminus \{j'\}$ 26 if $\mathcal{N}' == \emptyset$ then 27 $\mathcal{R}_i.remove(v)$, $\mathcal{V}'.remove(v)$ 28 $\mathcal{P}_i.pop(b_{v,t})$, 29 $b_{v,t} = \mathcal{P}_i.peak()$ 30 $\mathcal{N}(v) = \{y \mid e_{vy} \in \mathcal{E}, y \in \mathcal{V}'\}$ 31 end 32 if $\mathcal{R}_i == \emptyset$ then 33 return $\mathcal{H}_i$, ‘错误’ 34 end 35 end 36 $\mathcal{R}_i \leftarrow \mathcal{R}_i \cup \{j'\}$, $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \{b_{j',t^*}\}$ 37 $b_{v,t} \leftarrow b_{j',t^*}$ 38 end 39 $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup \{b^h_{v,t} \mid b_{v,t} \in \mathcal{P}_i\}$ 40 return $\mathcal{H}_i$, ‘成功’

算法 4： 响应方

$H(b^h_{v,\ t})$, $\mathcal{S}_{j'}$

Output: b^h_{j',t^*}

1 //*** 监听传入连接 ***// 2 如果 Msg = 接收(‘REQ 子块’) 那么 3 $H(b^h_{v,t}) = GetHash(Msg)$ 4 $C_{j'}(b_{v,t}) = \{b_{j',t'} \mid H(b^h_{v,t}) \in b_{j',t'}, b_{j',t'} \in \mathcal{S}_{j'}\}$ 5 设置 $t^* = \min\{t' \mid b_{j',t'} \in C_{j'}(b_{v,t})\}$ 6 //*** 发送消息 ***// 7 Send(‘RPY 子块’, b^h_{j',t^*}, v) 8 结束

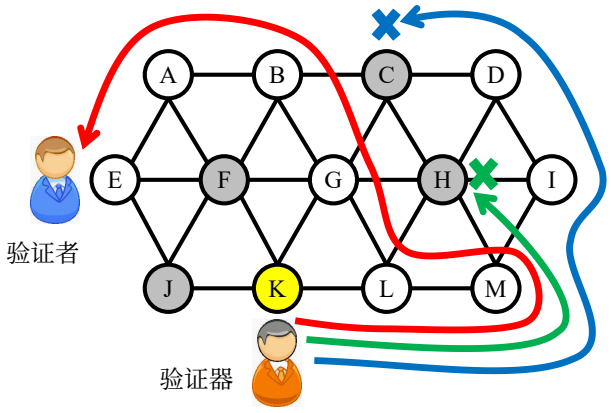


图 5. PoP 绕过恶意节点（灰色圆圈）构建路径的示例。黄色圆圈表示包含待验证区块的节点。验证者首先构建绿色路径，但该路径终止于恶意节点 H。随后，验证者向 I 发送请求，形成蓝色路径。然而，这条蓝色路径终止于恶意节点 C。由于验证者已经用尽了节点 I 和 D的所有邻居，随后向节点 L 的邻居发出请求，形成红色路径。

D. 讨论

这里我们讨论应用层的安全问题。对于协议栈其他层的攻击/问题，请读者参阅文献中广为人知的攻击与防御，例如 [25]。需要指出的是，节点的注册和身份验证不在本文讨论范围内。有关身份验证/安全协议，请参阅诸如 [26] 等研究。最后提醒读者，节点拥有公私钥对，并且彼此知晓拓扑以及对方的公钥。

1）恶意节点：回顾第 IV-C 节，如果验证者 i 未在超时 τ 内从 $j' \in \mathcal{N}$（j）接收到 RPY CHILD 消息，或接收到无效的 RPY CHILD 消息，则会向 $\mathcal{N}$（j）中的另一个节点请求 RPY CHILD 消息。如果 $\mathcal{N}$（j）中的所有节点都未能回复，验证者 i 会创建一条不涉及节点 j'的路径；参见算法 3 第 26—31 行。考虑图 5。假设验证者 E 旨在验证验证器 K处的一个区块。恶意节点以灰色圆圈表示。在此示例中，为确保共识，生成的路径 $\mathcal{P}_E$ 必须至少包含七个节点。在尝试构建绿色和蓝色路径后，验证者最终构建了红色路径。

2）多数攻击：在传统区块链中，假设采用工作量证明（PoW），拥有大多数计算能力（即 51%）的攻击者能够篡改区块链内容。这是因为他们能够构建一条包含无效数据的最长链，而参与者随后会接受这条链。相比之下，在 2LDAG 中，一个节点永远不会用其他节点的区块替换自己的区块。具体而言，节点会维护自己的区块，无须维护全局一致的“链”。

攻击者可能导致 $\mathcal{R}_i$ 中的所有节点或路径 $\mathcal{P}_i$ 中包含许多恶意节点。在这种情况下，一种解决方案是提高 γ 的值，这意味着需要更多节点才能达成共识。

3) 女巫攻击：女巫攻击者会创建虚假节点，例如，这些节点随后可用于发动 51% 攻击。回想一下，验证者（例如 i）会构建一个由唯一节点组成的集合 $\mathcal{R}_i$。这意味着，复制相同恶意节点的女巫攻击者不会对共识过程产生影响。此外，所有节点都会存储拓扑，并且能够使用节点的公钥来验证其身份。这两点都能防止女巫攻击者在不被察觉的情况下添加节点。

4）中间人攻击：我们仅考虑旨在篡改数据区块头和/或主体的攻击者。对于窃听攻击，发送方可以使用其私钥加密消息，或基于其私钥派生一个对称密钥；示例见 [27]。攻击者也可能丢弃源与目标之间的消息。为此，我们可以采用多条路径绕过攻击者 [28]。

在2LDAG 中，数据区块损坏的避免方式如下。由节点（例如 j）发送的数据块头使用其私钥 sk_j 进行签名。验证者可以使用相应的公钥 pk_j验证此数据块。这可以防止中间节点进行篡改。此外，区块在有向无环图中相互连接。这意味着，对某个区块的任何更改都会影响有向无环图中后续区块的摘要。

5）拒绝服务（DoS）攻击：恶意节点可能试图生成大量摘要并向其邻居泛洪。在2LDAG中，节点需要解开一个谜题，而生成一个区块需要花费数秒。因此，恶意节点无法在短时间内生成大量区块。这与IOTA [19] 所采用的策略相同。此外，节点可以禁止生成区块速度快于预期解谜时间的邻居。我们注意到，此类恶意节点只会影响其直接邻居。这是因为来自某个节点的摘要不会在整个网络中泛洪。

在2LDAG 中，节点处理以下消息：1) REQ CHILD，2) RPY CHILD，以及 3) 摘要。这些消 息和摘要可以包含随机数，以避免重放攻击。此外，验证器可以在回复REQCHILD 消息之前对验证者进行身份验证。类似地，节点只接收、转发和处理来自已通过身份验证的节点的摘要、REQ CHILD 和/或 RPY CHILD消息。否则，它们可以安全地丢弃这些摘要/消息。这确保有向无环图和 PoP 仅涉及诚实节点。

最后，如果一个节点被捕获并因此变得恶意，其数据区块的任何更改都将被 PoP 检测到。这是因为其数据区块的摘要将与其邻居的数据区块摘要不一致。

6) 自私攻击：在 PoP 中，自私节点可能永远不会回复 REQ CHILD 消息。为防止这种情况，节点可以配备惩罚机制。每个节点都会维护一个黑名单，其中包含因自私行为、断开连接或恶意行为而未回复 REQ CHILD 消息的节点。随后，其他节点将不会验证自私节点生成的区块。在黑名单中的节点帮助传输一定数量的区块后，便会从黑名单中移除。因此，为了表明重新连接后参与的意愿，节点会主动传输其数据区块。

V. 绩效分析

在本节中，我们分析 2LDAG 和 PoP 的存储与消息开销。简而言之，我们的目标是量化

节点 i 在时间 t 存储的数据最大量，以及验证者在验证期间发送和接收的消息数量。

命题 1。在时间 t 时，数据区块总数为 $\sum_{j\in\mathcal{V}}\lfloor\frac{tr_j}{C}\rfloor$。

证明。回顾一下，节点 j每 $\frac{C}{r_j}$秒生成一个数据区块。因此，节点 j在时间t时已生成$\lceil\frac{tr_j}{C}\rceil$个数据区块。因此，数据区块总数为 $\sum_{i\in\mathcal{V}}\lfloor\frac{tr_j}{C}\rfloor$。 □

请注意，节点，例如 i，维护集合 $\mathcal{H}_i$ 以降低通信成本。以下命题界定了 $\mathcal{H}_i$的大小。

命题 2. 在时间 t， $\mathcal{H}_i$ 的大小上界为 $\frac{t(f_c+f^H|\mathcal{V}|)}{C}\sum_{j\in\mathcal{V}\setminus\{i\}}r_j$ 比特。

证明。在最坏情况下，每个节点 i 需要存储 $\mathcal{H}_i$ 中所有区块的头部。令 $f(\mathcal{H}_i)$ 表示 $\mathcal{H}_i$ 的大小，根据式 (2)-(3) 和命题 1，我们有

$$\begin{aligned} f(\mathcal{H}_i) &= \sum_{j\in\mathcal{V}\setminus\{i\}} \lfloor\frac{tr_j}{C}\rfloor \times (f_j - C) \\ &= \sum_{j\in\mathcal{V}\setminus\{i\}} \lfloor\frac{tr_j}{C}\rfloor \times (f_c + f^H(|\mathcal{N}(j)|+1)). \end{aligned} \tag{12}$$

注意，我们有 $|\mathcal{N}(j)| \le |\mathcal{V}| - 1$和 $\lfloor\frac{tr_j}{C}\rfloor \le \frac{tr_j}{C}$。因此，上述方程可以重写为

$$\begin{aligned} f(\mathcal{H}_i) &\le \sum_{j\in\mathcal{V}\setminus\{i\}} \frac{tr_j}{C} \times (f_c + f^H|\mathcal{V}|) \\ &= \frac{t(f_c+f^H|\mathcal{V}|)}{C} \sum_{j\in\mathcal{V}\setminus\{i\}} r_j. \end{aligned} \tag{13}$$

证明至此结束。 □

因此，我们得到以下关于节点存储的命题。

命题 3. 在时间 t， 节点 i 的总存储量上限为 $tr_i + \frac{t(f_c+f^H|\mathcal{V}|)}{C}\sum_{j\in\mathcal{V}}r_j$ 位。

证明。根据命题-1，我们可以看到，节点 i 存储的区块数量为$|\mathcal{S}_i| = \lfloor\frac{tr_j}{C}\rfloor$。因此， $\mathcal{S}_i$的位大小记为 $f(\mathcal{S}_i)$，为

$$\begin{aligned} f(\mathcal{S}_i) &= \lfloor\frac{tr_j}{C}\rfloor \times f_i \\ &= \lfloor\frac{tr_i}{C}\rfloor \times (f_c + f^H(|\mathcal{N}(j)|+1) + C) \\ &\le \frac{tr_i}{C}(f_c + f^H|\mathcal{V}| + C) \\ &= tr_i + \frac{t(f_c+f^H|\mathcal{V}|)}{C} \times r_i. \end{aligned} \tag{14}$$

根据命题 2 中的公式 (13) 和公式 (14)，节点 i 的总存储量界定如下：

$$\begin{aligned} f(\mathcal{S}_i) + f(\mathcal{H}_i) &\le tr_i + \frac{t(f_c+f^H|\mathcal{V}|)}{C}\Big(r_i + \sum_{j\in\mathcal{V}\setminus\{i\}} r_j\Big) \\ &= tr_i + \frac{t(f_c+f^H|\mathcal{V}|)}{C}\sum_{j\in\mathcal{V}} r_j. \end{aligned} \tag{15}$$

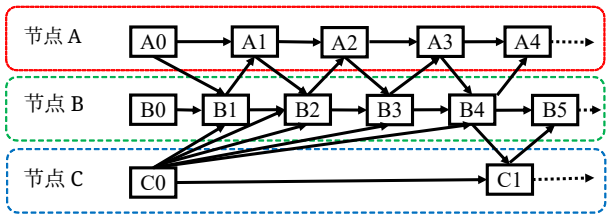


图 6。PoP 中微循环的示例。当 $r_B \gg r_c$和 $\gamma = 2$时。PoP 遍历微循环 $\{B2, A2, B3, A3, B4\}$，然后到达区块 $C1$。

证明到此结束。 □

根据算法 3，我们分析其消息开销，即验证者发送/接收的消息数量。首先，我们给出以下命题，用于量化消息开销下界。

命题 4。当 $\mathcal{H}_i = \emptyset$时，验证节点 i 至少发送和接收 $2(\gamma+1)$ 条消息以达成共识。

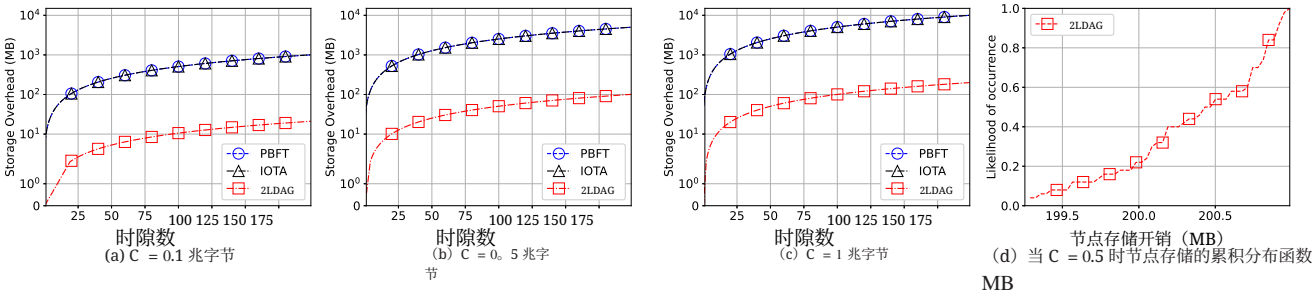
证明。当 $\mathcal{H}_i = \emptyset$时，验证者或节点 i 无法利用其存储的信息达成共识。这意味着它至少需要进行一次 REQ CHILD 和 RPY CHILD 消息交换，以验证区块 $b_{v,\ t_0}$ 此外，它需要验证至少 $\gamma+1$ 个待验证区块，并将不相交节点添加到 $\mathcal{R}_i$ 中，以达成共识，即 $|\mathcal{R}_i| \ge \gamma+1$。因此，当 $\mathcal{H}_i = \emptyset$时，验证者至少发送和接收 $2(\gamma+1)$ 条消息以达成共识。证明完毕。 □

在讨论验证者所产生的消息开销上界之前，我们先指出一个有趣的发现。请看图 6。假设节点 B 的数据生成速率远高于节点 C。因此，区块$C0$ 的摘要存在于节点 B 的多个区块中，而区块 $C1$ 仅包含区块 $C0$ 和 $B4$的摘要。此外，假设 $\gamma = 2$。当验证者需要验证一个区块（例如 $B1$）时，需要构建一条路径 $\{B1, A1, B2, A2, B3, A3, B4, C1\}$，遍历所有三个节点。请注意，将集合 $\{B2, A2, B3, A3, B4\}$中的区块加入路径后，集合 $\mathcal{R}_i$ 的大小并未增加。这是因为这些区块涉及的节点已存在于 $\mathcal{R}_i$中。还要注意，路径中存在一个微循环，即$\{B1, A1, B2, A2, B3, A3, B4\}$。我们用 $\mathcal{M}$ 表示微循环所遍历的节点集合。微循环中的区块数量受限于不在 $\mathcal{M}$ 中的某个节点的区块生成时间间隔，例如图 6 中的节点 C。由此，我们得到以下关于 $\mathcal{P}_i^l$中区块最大数量的命题。

命题 5. 设 $\mathcal{M}$ 是一组由微循环遍历的节点，则微循环内区块的数量上限为 $\sum_{i\in\mathcal{M}}\lfloor\frac{r_i}{\min\{r_j \mid j\in\mathcal{V}\setminus\mathcal{M}\}}\rfloor$。

证明。设 t' 为 $\mathcal{V}\setminus\mathcal{M}$ 中节点的最大区块生成时间间隔，其中

$$t' = \frac{C}{\min\{r_j \mid j\in\mathcal{V}\setminus\mathcal{M}\}}. \tag{16}$$



(a) C = 0.1 兆字节 (b) C = 0。5 兆字节 (c) C = 1 兆字节 (d) 当 C = 0.5 时节点存储的累积分布函数

图 7. （a）–（c）当区块体大小为 $C = \{0.1, 0.5, 1\}$ MB 时，2LDAG 相较于实用拜占庭容错和 IOTA 的平均节点存储开销（对数刻度）；（d）在 200 个时隙下，区块体大小为 $C = 0.5$ MB 时节点存储开销的累积分布函数。所有节点在每个时隙生成一个区块，即 $C/r_i = 1, \forall i \in \mathcal{V}$。

令 $S_i(t)$ 表示节点 i 在时间段 t 内生成的区块数量，其中

$$S_i(t) = \lfloor \frac{t}{C/r_i} \rfloor = \lfloor \frac{r_i t}{C} \rfloor. \tag{17}$$

因此，微循环内区块的最大数量为

$$\sum_{i \in \mathcal{M}} S_i(t') = \sum_{i \in \mathcal{M}} \lfloor \frac{r_i}{\min\{r_j \mid j \in \mathcal{V} \backslash \mathcal{M}\}} \rfloor. \tag{18}$$

□

关于消息开销上界，我们有以下命题。

命题 6。设节点按照其数据生成速率的降序编号，即 $r_1 \geq r_2$、...、$\geq r_{|\mathcal{V}|}$和 $r_\gamma > r_{\gamma+1}$。假设不存在恶意节点。验证者发送和接收的消息开销上界为 $(|\mathcal{V}(v)| + \gamma)(\sum_{j=1}^{j=\gamma} \frac{r_j}{r_{|\mathcal{V}|}} + \gamma + 1)$。

证明。回想一下，如果$|\mathcal{R}_i| \geq \gamma + 1$，验证节点 i 就达成共识。它需要构造一条路径 $\mathcal{P}_i$，其中可能包含一个最多由 γ 个节点遍历的微循环。利用命题-5，微循环内的区块最大数量为 $\sum_{j=1}^{j=\gamma} \lfloor \frac{r_j}{r_{|\mathcal{V}|}} \rfloor - \gamma$。然后，路径 $\mathcal{P}_i$ 的长度上界为

$$|\mathcal{P}_i| \leq \sum_{j=1}^{j=\gamma} \lfloor \frac{r_j}{r_{|\mathcal{V}|}} \rfloor + \gamma + 1. \tag{19}$$

当验证者在路径 $\mathcal{P}_i$ 上验证区块$b_{v,\ t}$ 时，它最多会发送 $|\mathcal{N}(v)|$ 条请求子节点消息，其中 $|\mathcal{N}(v)| \leq |\mathcal{V}| - 1$。此外，它最多会接收 $\gamma + 1$ 条子节点回复消息，其中 γ 条无效，因为这些消息由恶意节点发送。因此，验证者的消息开销总计满足

$$\begin{aligned}(|\mathcal{N}(v)| + \gamma + 1)|\mathcal{P}_i| &\leq (|\mathcal{V}(v)| + \gamma)|\mathcal{P}_i| \\ &\leq (|\mathcal{V}(v)| + \gamma)(\sum_{j=1}^{j=\gamma} \frac{r_j}{r_{|\mathcal{V}|}} + \gamma + 1).\end{aligned} \tag{20}$$

证明到此结束。

□

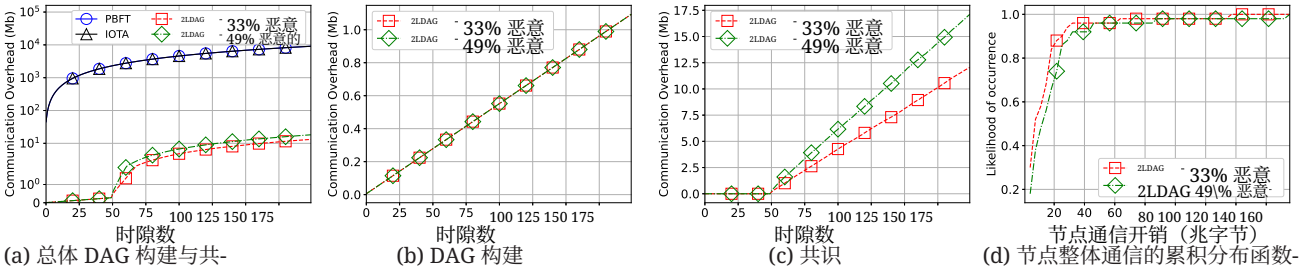
VI. 数值评估

仿真在一台配备 i7-12700 处理器和 32 GB 内存的台式机上进行。我们评估三个关键指标：（1）存储开销，即节点占用的磁盘空间总量；（2）通信开销，即节点在区块生成过程中传输的数据总量；（3）达成共识所需时间，即 PoP 找到足够路径以达成共识所用的时间。我们比较 2LDAG 与 PBFT 区块链 [29], 和无代币 IOTA 区块链[19]的存储和通信开销。我们将摘要和签名字段的大小分别设为 $f^H = f^s = 256$ 位。版本、时间和随机数字段设为 $f^v = f^t = f^n = 32$ 位。物理网络由部署在 1000 平方米区域内的 50 个无线物联网节点组成。所有节点的通信范围均为 50 米。为确保网络连通，我们逐个放置节点。也就是说，我们先在上述区域的中心随机放置一个节点。然后向区域内添加新节点，并确保每个新节点都随机放置在一个已部署节点的通信范围内。

我们将时间划分为时隙。每个节点在每个时隙中最多生成一个区块。当节点生成一个区块时，必须使用PoP 验证过去生成的另一个区块。这意味着，节点在生成数据区块时就是验证者。需要注意的是，PoP 只能验证在 $|\mathcal{V}|$ 个时隙之前生成的区块。这是因为，为了对一个区块达成共识，验证者需要找到一条遍历超过 $\gamma+1$ 个物理节点的有向路径。这也意味着，该路径经过的时隙数超过 $\gamma + 1$ 个。

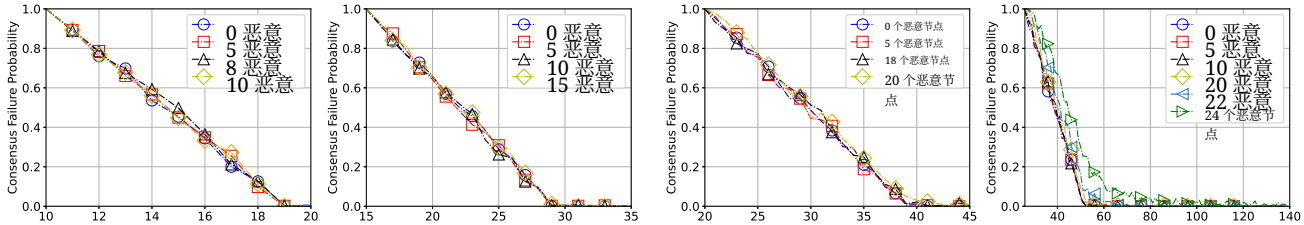
A. 存储开销

我们首先比较 2LDAG与 PBFT和IOTA中节点的平均存储开销。我们评估以下区块大小（单位为兆字节）：$C = \{0.1, 0.5, 1\}$。从图7(a) 到图 7(c) 可以看出，2LDAG的节点平均存储开销几乎比 PBFT和IOTA低两个数量级。需要注意的是，由于节点拥有不同数量的邻居，其区块头大小也各不相同。因此，我们在图7(d) 中展示了第 200 个时间槽时每个节点存储量的累积分布函数（CDF）。可以看到，节点的存储量在199至 201 兆字节之间变化。因此，邻居数量不会对节点存储产生显著影响。



(a) 总体 DAG 构建与共-识 (b) DAG 构建 (c) 共识 (d) 节点整体通信的累积分布函数-tion 开销

图8。当可容忍的恶意节点分别为 γ 33%$|\mathcal{V}|$ 和49%$|\mathcal{V}|$时，2LDAG的 (a) 整体 DAG构建和共识、(b)DAG 构建、(c)共识的平均节点通信开销，以及(d)整体节点通信开销的累积分布函数。区块主体大小为 $C = 0.5$ 兆字节。



(a) $\gamma = 10$ (b) $\gamma = 15$ (c) $\gamma = 20$ (d) $\gamma = 24$

图 9. 当最大可容忍恶意节点数为 $\gamma = \{10, 15, 20, 24\}$ 时，2LDAG 在存在恶意节点的情况下达成共识所需的时隙数。区块主体大小为 $C = 0.5$ 兆字节。每个节点的随机区块生成速率为每 $\{1, 2\}$ 个时隙生成一个区块。

B. 通信开销

区块大小为 $C = 0.5$ 兆字节，每个节点在每个时隙生成一个区块。我们考虑两种可能的恶意节点数量，即恶意节点分别占节点总数的 33%和49%。这些数值分别对应于实用拜占庭容错和IOTA 或比特币等传统区块链所能容忍的恶意节点数量。这意味着，2LDAG 通过包含17个和 26 个物理节点的路径达成共识。从图8(a)中可以看出，2LDAG 的节点总体平均通信开销几乎比实用拜占庭容错和IOTA 低三个数量级。需要注意的是，2LDAG的通信开销在最初的 50 个时隙中几乎为零。这是因为节点在 $|\mathcal{V}| = 50$ 个时隙后才开始验证其他区块。此外，2LDAG 用于共识的通信开销远高于 DAG构建，见图8(b) 和8(c)。这是因为2LDAG 在区块生成过程中只传输摘要，但在共识过程中需要传输区块头。我们可以看到，能够容忍 49%恶意节点的 2LDAG具有更高的共识通信开销，因为它需要构建更长的路径才能实现 PoP共识。图8(d) 展示了每个节点在第200 个时隙的通信开销累积分布函数。我们可以看到，超过 90%的节点传输的数据少于 40 兆字节，而其他节点传输的数据量可能高达 160兆字节。这是因为少数节点对于数据转发至关重要，因此容易受到攻击。

C. 达成共识所需时间

最后，我们研究存在恶意节点时，PoP 在 2LDAG 中达成共识所需的时隙数。每个节点每一个或两个时隙生成一个区块。我们评估 2LDAG 验证在前 γ 个时隙内生成的区块时的共识失败概率。当共识失败概率为零时，我们称共识达成。我们考虑图 9 中的四种情形 $\gamma \in \{10, 15, 20, 24\}$。请注意，对于拥有 50 个节点的网络，它最多只能容忍 24 个恶意节点。从图 9 可以看出，达成共识所需的时隙数会随着 γ 增加。这是因为 PoP 需要更长的路径。此外，对于以下 γ 个取值：$\{10, 15, 20\}$，恶意节点的数量对共识时间没有显著影响。然而，当 $\gamma = 24$ 时，共识最多需要 120 个时隙。这是因为验证者必须找到一条包含所有诚实节点的路径来验证区块。

VII. 结论

本文提出了一种基于有向无环图和 PoP 协议的新型 2LDAG 架构，以确保物联网数据完整性。与采用实用拜占庭容错和有向无环图区块链的方案相比，2LDAG 的存储成本和通信成本更低。因此，它非常适合应用于包含资源受限节点的物联网网络。此外，2LDAG 具有较高的吞吐量。这是因为所有节点都能够独立生成数据区块并传输哈希值。此外，节点的区块生成速率不受共识过程限制。仿真结果表明，与实用拜占庭容错和 IOTA 区块链相比，2LDAG 的存储成本降低了两个数量级，通信成本降低了三个数量级。2LDAG 的一个潜在未来方向是，在运行 PoP 协议时，于物理层构建从验证者到验证器的最短路径。这将进一步降低传输数据区块头的通信开销。另一个未来

工作是动态场景，其中节点会随着时间推移加入和离开网络。

参考文献

[1] L.-H. Lee、T. Braud、P. Zhou、L. Wang、D. Xu、Z. Lin、A. Kumar、C. Bermejo 和 P. Hui，《关于元宇宙你需要了解的一切：技术奇点、虚拟生态系统及研究议程的全面综述》，arXiv，第 abs/2110.05352 卷，2021年。[2] G. Coorey、G. A. Figtree、D. F. Fletcher 和 J. Redfern，《健康数字孪生：推进精准心血管医学》，《自然心血管研究评论》，第18卷，第12期，第803–804页，2021年。[3] I. Yaqoob、K. Salah、M. Uddin、R. Jayaraman、M. Omar 和 M. Imran，《数字孪生的区块链：最新进展与未来研究挑战》，《IEEE网络》，第34卷，第290–298页，2020年9月。[4] S. Suhail、R. Hussain、R. Jurdak、A. Oracevic、K. Salah、C. S. Hong 和 R. Matulevi ˇcius，《基于区块链的数字孪生：研究趋势、问题与未来挑战》，《ACM计算调查》，第54卷，2022年9月。[5] A. Sharma、E. S. Pilli、A. P. Mazumdar 和 P. Gera，《迈向可信物联网：信任管理应用与方案综述》，《计算机通信》，第160卷，第475–493页，2020年。[6] S. Malik、V. Dedeoglu、S. S. Kanhere 和 R. Jurdak，《Trustchain：区块链和物联网支持的供应链中的信任管理》，载于《IEEE国际区块链会议（Blockchain）》，（韩国首尔），第184–193页，2019年5月。[7] Q. Zhou、H. Huang、Z. Zheng 和 J. Bian，《区块链可扩展性解决方案：综述》，《IEEE Access》，第8卷，第16440–16455页，2020年。[8] 《比特币大小》。[在线]。网址：https://ycharts.com/indicators/bitcoin blockchain size。[9] X. Ledger，《容量规划》，2022年。[在线]。网址：https://xrpl.org/capacity-planning.html。[10] L. M. Bach、B. Mihaljevic 和 M. Zagar，《区块链共识算法的比较分析》，载于《第41届信息与通信技术、电子与微电子国际大会论文集》，（克罗地亚奥帕蒂亚），第1545 1550页，2018 – 。[11] BitcoinCore，《运行全节点》，2022 年。[在线]。网址：https://bitcoin.org/en/full-node。[12] M. N. Aman、B. Sikdar、K. C. Chua 和 A. Ali，《物联网系统中的低功耗数据完整性》，《IEEE物联网期刊》，第5卷，第3102–3113页，2018年4月。[13] C. Doukas、I. Maglogiannis、V. Koufi、F. Malamateniou 和 G. Vassilacopoulos，《通过公钥基础设施加密实现物联网移动健康设备的数据保护》，载于《IEEE第12届生物信息学与生物工程国际会议（BIBE）》，（塞浦路斯拉纳卡），第25–29页，2012年11月。[14] S. Nakamoto，《比特币：一种点对点电子现金系统》，2008年。[在线]。网址：https://bitcoin.org/bitcoin.pdf。[15] M. Yu、S. Sahraei、S. Li、S. Avestimehr、S. Kannan 和 P. Viswanath，《编码默克尔树：解决区块链中的数据可用性攻击》，载于《金融密码学与数据安全国际会议》，（马来西亚哥打基纳巴卢），第114–134页，Springer，2020年2月。[16] G. Wang、Z. J. Shi、M. Nixon 和 S. Han，《Sok：区块链分片》，载于《第1届ACM金融技术进展会议论文集》，第41–61页，2019年。[17] C. Yang、K.-W. Chin、J. Wang、X. Wang、Y. Liu 和 Z. Zheng，《利用纠错码扩展区块链：编码区块链综述》，arXiv预印本 arXiv:2208.09255，2022年。[18] Q. Wang、J. Yu、S. Chen 和 Y. Xiang，《Sok：深入探究基于有向无环图的区块链系统》，arXiv预印本 arXiv:2012.06128v2，2020年。[19] S. Popov，《缠结》，2018年。[在线]。网址：
htt
ps://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae3
3637ca92f85dd9f4a3a218e1ec/iota1 4 3.pdf.

[20] M. S. Ferdous、M. J. M. Chowdhury、M. A. Hoque 和 A. Colman，《区块链共识算法：综述》，arXiv 预印本 arXiv:2001.07091，2020年。[21] M. Salimitari、M. Chatterjee 和 Y. P. Fallah，《资源受限物联网网络中区块链共识方法综述》，Elsevier Internet of Things，第 11 卷，第 100212 页，2020年。[22] M. Bowman、D. Das、A. Mandal 和 H. Montgomery，《关于经过时间共识协议》，载于印度国际密码学会议，第 559–583 页，2021年 12 月。[23] A. Miller、Y. Xia、K. Croman、E. Shi 和 D. Song，《拜占庭容错协议中的蜜獾》，载于 ACM SIGSAC计算机与通信安全会议（维也纳，奥地利），第 31–42 页，2016年 10 月。

[24] S. Al Salami、J.Baek、K. Salah 和 E.Damiani，“智能家居的轻量级加密”，载于第11 届国际可用性、可靠性与安全会议（ARES），（萨尔茨堡，奥地利），第 382–388页，2016 年 8 月。[25] I. Butun、P. ¨ Osterberg 和 H. Song，“物联网的安全性：漏洞、攻击与对策”，IEEE 通信调查与教程，第22卷，第 1期，第 616–644 页，2020年。[26] G. Han、J. Jiang、L.Shu、J.Niu 和 H. Chao，“无线传感器网络中的信任管理与应用：综述”，计算机与系统科学杂志，第80卷，第 3 期，第 602–617 页，2014 年。[27] A.Perrig、R.Szewczyk、J. D. Tygar、V. Wen 和 D. E. Culler，“SPINS：传感器网络安全协议”，Springer Wireless Networks，第 8 卷，第 521–534 页，2002年。[28] C. Karlof和 D. Wagner，“无线传感器网-络中的安全路由：攻击与对策”，Elsevier Ad-HocNetworks，第 1卷，第 293=315 页，2003年。[29] M. Castro、B. Liskov 等，“实用拜占庭容错”，载于 USENIXOSDI，第 99 卷，186，1999年 173（新奥尔良，路易斯安那州，美国），第 – 页。