基于邻居优先级的区块传播优化方法

贾志龙[1] 张 虹[1]
JIA Zhilong ZHANG Hong

摘 要 在采用 Gossip 算法的区块链网络中，发送节点以一个固定的概率选择邻居传播区块。然而，这种算法会导致接收节点收到重复区块，从而增加数据冗余性和传播延迟。为解决上述问题，文章提出了一种基于邻居优先级的区块传播方法。该方法在网络初始化阶段根据邻居的延迟时间设定节点优先级。在每一轮区块传播过程中，优先向优先级高的邻居发送新区块，并根据接收区块的时间动态更新邻居的优先级。此外，每个节点定期广播区块高度信息给尚未接收新区块的邻居，以避免区块的重复发送。与传统的 Gossip 算法相比，基于邻居优先级的区块传播方法有效降低了区块传播延迟和数据冗余，并减少了全网的数据开销。

关键词 Gossip 算法；区块链；区块传播；优先级；数据冗余

doi：10.3969/j.issn.1672-9528.2025.03.028

0 引言

区块链是融合数学、密码学、计算机技术等领域的新技术 [1]，具有去中心化、不可篡改和公开透明等特点。P2P 是区块链的核心技术之一，而 Gossip 是一种重要的 P2P 通信算法 [2]。该算法被广泛应用于多个区块链项目，如比特币、Hyperledger Fabric 和以太坊，其性能直接影响整个区块链网络的效率 [3]。

原始的 Gossip 算法在区块同步过程会产生数据冗余 [4]，因为邻居节点被选中的概率是相同的，导致节点重复接收相同的区块。这种冗余传播不仅延长了区块同步时间，还增加了数据传输的负担，进而影响区块链系统的性能。

为解决上述问题，本文提出了一种改进的 P-Gossip 算法旨在提高区块链系统的性能和同步效率。

1 相关工作

优化传播路径方面，He 等人 [5] 提出了 HNA-Gossip 算法，该算法在原始算法的基础上新增一个历史节点列表，并将其加入到发送消息中，从而避免向历史节点发送重复数据。成卫青等人 [6] 提出了基于二进制指数退避的 Gossip 算法，以避免向节点重复发送新消息。尽管现有路径优化方法在提高传播效率方面取得了显著成效，但仍未考虑节点实际的物理链路延迟，这可能限制其在各种网络环境中的有效性。

减少数据开销方面，Yang 等人 [7] 提出了基于 Gossip 的 ARI 块传播算法，该算法将区块分割为均匀小块进行传输，利用节点的上下带宽，减少了区块链网络的块同步时间。Berendea 等人 [8] 针对区块链网络中数据传播的带宽开销问题，提出在 push 阶段只转发区块的摘要信息，消除了因多次接收相同的完整块而导致的通信开销。虽然这些数据开销优化方法取得了良好的效果，但它们也可能导致较大的计算和存储压力，影响系统的整体性能。

在优化拓扑结构方面，Kitagawa 等人 [9] 提出了一种基于 Plumtree 算法的区块链网络拓扑结构优化方法，该方法通过构建一个随机拓扑上的松散固定传播路径，有效降低了区块链网络中的通信资源消耗。Xu 等人 [10] 提出了 DC-Gossip 协议，结合密度聚类算法，基于当前网络中节点的分布构造一个确定性结构。在 Fabric 中的实验表明，DC-Gossip 可以显著提高区块传播效率。尽管这些拓扑结构优化方法取得了显著成效，但其适用性仍然受到现有区块链架构的限制。

鉴于以上研究仍存在的问题，本文提出一种基于邻居优先级的 P-Gossip 区块传播方法。该方法首次引入了优先级，通过结合物理延迟和区块到达时间来设定并动态更新邻居优先级，每一轮优先选择高优先级的邻居进行区块传播。此外，算法还定期广播区块高度信息，有效避免了区块的重复发送，从而降低了网络的冗余开销。

2 相关知识

2.1 Gossip 算法

Gossip 协议在信息同步过程中，无法确保所有节点在同一时刻接收到消息，但理论上所有节点最终都会接收到消息，从而实现最终一致性。Gossip 算法的通信模式包括推送模式、拉取模式以及推 / 拉结合模式。

（1）Push 推送方式：持有最新消息的节点 A 随机选择

1. 太原师范学院 山西晋中 030600

节点 B，主动发送最新消息给 B。节点 B 接收到消息后，比较并同步数据。

（2）Pull 拉取方式：节点 A 随机选择节点 B，主动请求并获取 B 的消息。接收到消息后，节点 A 比较并同步数据。

（3）Push/Pull 推拉结合：节点 A 随机选择节点 B，一方面主动发送消息给 B，另一方面请求并接收来自 B 的消息。接收到消息后，节点 A 和 B 分别比较并同步数据。

2.2 Fabric 区块结构

在 Fabric 架构中，区块结构包含 3 个主要部分：区块头、区块数据和区块元数据。区块头记录了区块序号、前一个区块的哈希值和当前区块的哈希值，但不包含生成时间。区块数据包含按时间顺序排列的多笔交易信息，本文采用最后一个交易的时间戳作为区块生成时间的近似值。区块元数据则包含区块生成者的证书、公钥和签名信息。

3 P-Gossip 算法

3.1 数据结构

本文提出的 P-Gossip 算法包括了 4 种数据结构：广播消息 BH、区块消息 BI、路由表 RT、动态转发表 DT。广播消息 BH 的结构相对简单，仅包含节点的区块高度信息，数据大小为 4 字节。区块消息 BI 则指的是 Hyperledger Fabric 中的区块数据。路由表 RT 和动态转发表 DT 的具体内容详细阐述如下。

3.1.1 路由表 RT

路由表是一种基于邻居节点优先级的排序表，由两部分组成：邻居节点标识和相应的优先级。节点标识采用 IP 地址，每个地址占用 4 字节，以区分不同的邻居节点。优先级则是基于接收区块的时间和邻居节点的延迟动态计算，同样占用 4 字节。具体的格式如图 1 所示。

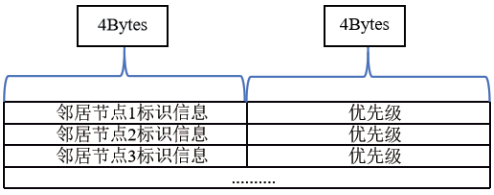

图 1 路由表

3.1.2 动态转发表 DT

动态转发表 DT 用于记录即将接收新区块的邻居节点标识信息。换言之，该列表包含了未来需接收区块的邻居节点。此表会随网络状态的动态变化而更新，主要有 3 种更新方式，具体格式如图 2 所示。

（1）在区块传播过程中，节点将区块发送给选定的邻居节点，并在发送后从动态转发表 DT 中删除这些邻居的标识信息。

（2）在区块传播过程中，节点接收来自特定邻居的区块，并在接收后从动态转发表 DT 中删除这些邻居的标识信息。

（3）在每轮区块传播完成后，根据路由表中的优先级顺序，生成一个与优先级相对应的邻居节点标识信息表，即动态转发表 DT。

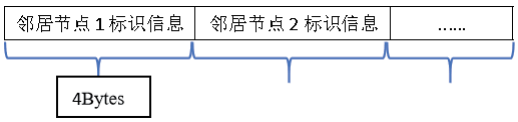

图 2 动态转发表

3.2 具体算法流程

3.2.1 路由表初始化

本文采用主动探测机制来评估与不同邻居节点的延迟，并将这一结果作为设定邻居优先级的参考。初始阶段，每个节点向其所有邻居发送一个问候（hello）消息，邻居节点在接收后返回确认（ack）消息。通过计算 hello 消息与 ack 消息之间的时间差评估与邻居节点的延迟。随后，基于这些延迟信息，计算每个邻居的优先级，并将邻居的 IP 地址及其对应的优先级记录到本地路由表中，以完成初始化过程。

优先级的计算分为两步。首先需要计算延迟的倒数，其计算公式为：

$$d_{i,j}=\frac{1}{r_{i,j}} \tag{1}$$

式中：$r_{i,j}$ 为节点 i 到节点 j 的延迟；$d_{i,j}$ 为延迟倒数。计算延迟倒数是为了确保优先级能够反映节点之间的响应速度，延迟越小，优先级越高。

接着，计算具体的优先级值，其计算公式为：

$$p_{i,j}=\frac{d_{i,j}}{\sum_{n}^{L_i} d_{i,n}} \tag{2}$$

式中：L_i 是节点 i 所有邻居集合；分母是节点 i 到所有邻居节点延迟倒数的总和。采用该方法进行归一化处理，可以确保邻居优先级值落在合理区间内，避免优先级值过大或过小对后续更新计算的影响。

3.2.2 动态转发表初始化

动态转发表是记录待发送区块的邻居节点列表。初始状态下，节点尚未参与区块的接收与转发，因此动态转发表初始化为包含所有邻居节点的标识信息。在每轮区块同步完成后，所有节点的动态转发表将被清空，并在下一轮区块传播开始前重新初始化。

3.2.3 P-Gossip 区块传播流程

P-Gossip 区块传播流程是区块链网络中一种优化数据传播效率的机制。在这一流程中，每个节点“扮演”着双重角色，通过执行两个核心线程：主动线程和被动线程，以确保区块信息的高效和可靠传播。以下将分别介绍这两种线程。

（1）主动线程

主动线程负责节点主动向其邻居节点发送数据。在此线程中，节点定期广播其区块高度信息，并根据路由表 RT 和

动态转发表 DT 选择邻居节点以发送区块。具体算法描述如算法 1。

算法 1 主动算法

输入：RT、DT

输出：BI、BH

1: 间隔 timeInterval 广播 BH
2: selectedNodes = RT.nodes[0:*k*] // 选取前 *k* 个节点
3: if DT!=NULL then
4: if selectedNodes ∈ DT then
5: send(BI, selectedNodes) // 发送 BT 给 selectedNodes
6: remove(DT.selectedNodes) // 删除 selectedNodes
7: else
8: *i* = selectedNodes ∩ DT //selectedNodes 与 DT 交集
9: if count(*i*) > 0 then
10: *j*=*k* - count(*i*)
11: if count(*j*) > 0 then
12: additionalNodes=RT.nodes[*k*:*k*+*j*]
13: allNodes = selectedNodes + additionalNodes
14: send(BI, additionalNodes)
15: remove(DT.additionalNodes)
16: else
17: send(BI, selectedNodes)
18: remove(DT.selectedNodes)
19: end if
20: end if
21: if count(*i*) ==0 then
22: selectedNodes = RT.nodes[0:*k*],
23: goto 4// 返回第 4 行
24: end if
25: end if
26: else
27: end the round
28: end if

（2）被动线程

被动线程负责接收并处理来自邻居节点的数据，包括广播消息 BI 和区块信息 BM。该线程根据接收到的信息动态更新动态转发表 DT 和路由表 RT。具体算法描述如算法 2。在该算法中，时间差 timeDiff 表示节点 *i* 接收区块的时间与区块中最后一笔交易的记录时间之间的差异。发送节点记为节点 *j*；接收节点记为节点；maxHeight(*i*) 表示 *i* 节点的最大区块高度；$RT_{i,j}$ 表示 *i* 节点的 RT 表中邻居 *j* 的优先级。

算法 2 被动算法

输入：BI、BM

输出： DT、RT

1: if 收到 BH then
2: if *j* ∈ DT && BH == maxHeight(*i*) then
3: remove(DT,*j*)
4: end if
5: else
6: parse(BI)
7: if 验证通过 then
8: remove(DT,*j*),calculate timeDiff
9: if *j* 首次发送 BI 且 BI > maxHeight(i) then
10: $RT_{i,j}$ += timeDiff
11: UpdateLocalblockchain(*i*) // 更新 *i* 节点本地区块链
12: end if
13: if *j* 非首次发送 BI 或 BI== maxHeight(i) then
14: $RT_{i,j}$ = 0.5*$RT_{i,j}$+0.5*timeDiff
15: end if
16: else
17: discard(BI)
18: end if
19: end if

4 实验与结果分析

本实验电脑配置为 Window10 操作系统，采用 Go 语言对 Gossip 算法和 P-Gossip 算法进行复现。表 1 展示了本次模拟实验的基本参数。所使用的区块来自于 Hyperledger Fabric 的测试链。每次传播 5 个区块，间隔时间为 15 ms，并在所有节点同步后统计结果。为了保证数据的准确性，每次实验都重复 10 次。

表 1 实验参数

基本参数	取值
区块大小	1 MB
节点个数	25 400
每次选取邻居个数	4
同步周期	1 s
广播消息间隔时间	0.5 s
延迟	40~600 ms
区块间隔时间	15 ms

4.1 区块传播时间

本部分统计了从第一个区块开始传播至所有节点区块高度同步为所需时间，实验对比结果如图 3 所示。

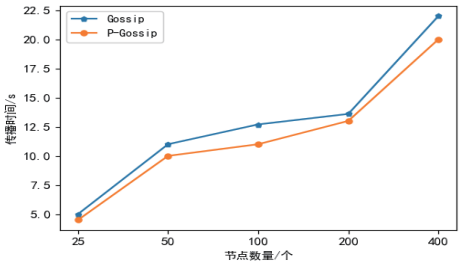

图 3 传播时间对比

图中数据显示，随着节点数量的增加，区块传播时间显著增长。然而，P-Gossip 算法的时间开销始终低于 Gossip 算法，这是因为 P-Gossip 算法优先选择高优先级的邻居进行区块传播，从而优化了区块传播路径并减少了传播延迟。

4.2 总体数据开销

总体数据开销统计了区块传播过程中所有节点发送的数据总量，实验结果如图 4 所示。图中结果显示，P-Gossip 的数据开销远远低于 Gossip，并且随着节点数量的增加，这一优势愈加明显。尽管 P-Gossip 会定期发送广播消息，但这部分开销相对于区块来说非常小。得益于新增的数据结构——动态转发表 DT，P-Gossip 优先向未同步的邻居发送区块，从而有效减少冗余消息，降低了数据开销。

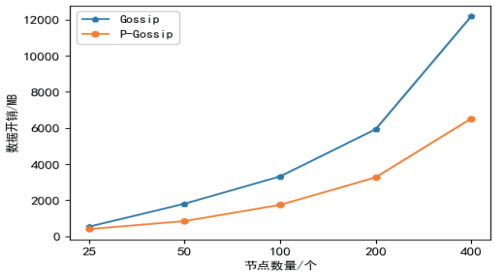

图 4 数据开销对比

4.3 冗余消息条数

冗余消息条数统计了所有节点接收到的重复区块总数，实验对比结果如图 5 所示。与总体数据开销的结果相似，随着节点数量的增加，冗余消息条数的减少趋势更为显著。例如，当节点数为 200 和 400 时，Gossip 算法产生的冗余消息数量非常庞大，而 P-Gossip 算法能够将冗余消息减少超过一半。这表明 P-Gossip 算法更适用于大规模区块链网络。

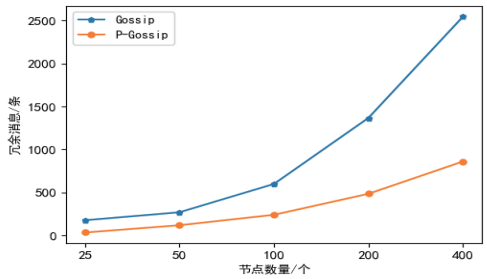

图 5 冗余消息数对比

5 结语

P-Gossip 算法引入了广播消息、动态转发表和路由表 3 种数据结构，节点定期广播自己的区块高度，并根据区块的收发情况和邻居的广播消息更新动态转发表。同时，节点根据接收区块的时间计算邻居的优先级，并更新路由表。在每一轮传播中，节点根据路由表和动态转发表转发区块，从而实现最低延迟路径的传播，减少数据冗余和全网数据开销。未来的研究将计划将数据压缩技术融入本文，以进一步降低数据传输的开销。

参考文献：

[1] 何蒲，于戈，张岩峰，等．区块链技术与应用前瞻综述 [J]. 计算机科学，2017,44(4):1-7.

[2] 武岳，李军祥．区块链 P2P 网络协议演进过程 [J]. 计算机应用研究，2019, 36(10):2881-2886.

[3] ANTWI R, GADZE J D, TCHAO E T, et al. A survey on network optimization techniques for blockchain systems[J]. Algorithms, 2022, 15(6): 193.

[4] 司冰茹，肖江，刘存扬，等．区块链网络综述 [J]. 软件学报，2024, 35(2): 773-799.

[5] HE X W, CUI Y J, JIANG Y C. An improved Gossip algorithm based on semi-distributed blockchain network[C/OL]//2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. Piscataway: IEEE, 2019[2024-05-11].https://ieeexplore.ieee.org/document/8946047.

[6] 成卫青，张蕾．二进制指数退避的 Gossip 算法研究 [J]. 电子与信息学报，2021,43(12):3486-3495.

[7] YANG X, SHI L. Ari: a P2P optimization for blockchain systems[C/OL]//2019 17th International Conference on Privacy, Security and Trust (PST). Piscataway: IEEE,2019[2024-06-09]. https://ieeexplore.ieee.org/document/8949064.

[8] BERENDEA N, MERCIER H, ONICA E, et al. Fair and efficient Gossip in hyperledger fabric[C/OL]//2020 IEEE 40th International Conference on Distributed Computing Systems. Piscataway:IEEE,2020[2024-06-19].https://ieeexplore.ieee.org/document/9355725.

[9] KITAGAWA Y, SHUDO K, MIZUNO O, et al. A study of using plumtree algorithm in blockchain networks[J]. Journal of information processing, 2023, 31: 387-391.

[10] XU Z G, YE K Z, DONG X H, et al. DC-Gossip: an enhanced broadcast protocol in hyperledger fabric based on density clustering[J].Wireless algorithms, systems, and applications, 2022, 13473: 3-19.

【作者简介】

贾志龙（2000—），男，山西吕梁人，硕士研究生，研究方向：区块链技术、P2P 网络。

张虹（1977—），通信作者（email:zhanghong8130@163.com），女，山西太原人，博士，副教授、硕士生导师，研究方向：区块链与智能技术、人工智能。

（收稿日期：2024-10-30）