软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(1):233−253 [doi: 10.13328/j.cnki.jos.006219]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

链下通道路由算法综述*

贾林鹏[1,2], 裴 奇[1,2], 王 鑫[1,2], 张瀚文[1,2], 于 雷[2], 张 珺[3], 孙 毅[1,2]

[1](中国科学院大学 计算机科学与技术学院, 北京 100049)
[2](中国科学院 计算技术研究所, 北京 100190)
[3](内蒙古大学, 内蒙古 呼和浩特 010021)
通信作者: 孙毅, E-mail: sunyi@ict.ac.cn

摘 要: 链下通道网络是实现区块链系统性能提升的有效手段, 其中路由算法是保证链下通道网络长期高效、稳定运行的关键. 提出了链下通道网络层次化架构及路由算法基础模型; 从单路路由与多路路由两方面, 系统梳理了代表性路由算法及其发展过程; 从有效性、并发性、可扩展性、通道平衡、路由选择中心化、成本效益、隐私保护、吞吐量、处理延迟、成功率、搜索效率 11 个方面设计了链下通道路由算法评价体系, 对代表性路由算法进行了对比分析, 并探讨了未来研究趋势.
关键词: 区块链; 链下通道; 路由算法; 评价体系
中图法分类号: TP393

中文引用格式: 贾林鹏, 裴奇, 王鑫, 张瀚文, 于雷, 张珺, 孙毅. 链下通道路由算法综述. 软件学报, 2022, 33(1): 233–253. http://www.jos.org.cn/1000-9825/6219.htm
英文引用格式: Jia LP, Pei Q, Wang X, Zhang HW, Yu L, Zhang J, Sun Y. Survey on Offchain Channel Routing Algorithm. Ruan Jian Xue Bao/Journal of Software, 2022, 33(1): 233–253 (in Chinese). http://www.jos.org.cn/1000-9825/6219.htm

Survey on Offchain Channel Routing Algorithm

JIA Lin-Peng[1,2], PEI Qi[1,2], WANG Xin[1,2], ZHANG Han-Wen[1,2], YU Lei[2], ZHANG Jun[3], SUN Yi[1,2]
[1](School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)
[3](Inner Mongolia University, Hohhot 010021, China)

Abstract: Offchain channel network (OCN) can effectively improve the performance of blockchain system. The key component for OCN to achieve long-term efficient and stable operation is routing algorithm. This study proposes OCN architecture and the basic model of offchain channel routing algorithm. From perspectives of single-path routing and multi-path routing, typical routing algorithms are systematically reviewed and discussed. Meanwhile, an evaluation system is established for offchain channel routing algorithm, in terms of effectiveness, concurrency, scalability, channel balance, routing centralization, cost-effectiveness, privacy protection, goodput, latency, success rate, and efficiency. Finally, these algorithms are compared, and challenging research issues and technology trends of offchain routing algorithm are discussed.
Key words: blockchain; offchain channel; routing algorithm; evaluation system

区块链起源于 2008 年 11 月提出的比特币[1], 是未来构建价值互联网的重要基础设施. 现有区块链系统存在通量低、延迟高的缺点, 其应用大都局限在存证等低频交易场景, 无法满足实时支付等场景的高频交易需求, 这也限制其更深层次应用的开展. 为了提升区块链系统通量水平, 早期研究工作主要从共识算法[2]、并行化[3]、数据扩容[4]、传输优化[5]、专用硬件[6]等方面展开, 通过多层次协同提升系统通量. 然而, 链上通量提升依旧无法满足小额高频、

* 基金项目: 国家重点研发计划 (2019YFB1404903); 国家自然科学基金 (61972382, 61672499, 61772502); 北京市科技计划课题 (Z181100003218018); 内蒙古自然科学基金 (2020MS06017)
收稿时间: 2020-04-16; 修改时间: 2020-09-28; 采用时间: 2020-12-06; jos 在线出版时间: 2021-01-15

复杂计算等交易处理需求, 链下扩容[7,8]技术应运而生.

链下通道[9,10]是目前使用最广泛的链下扩容方案, 其基于哈希时间锁定合约 (hashed timelock contract, HTLC)[11]、原子多路支付协议 (atomic multi-path payments, AMP)[12]等技术, 在节点间建立链上锁定资产或状态的链下流通渠道, 实现锁定资产或状态的链下快速、低成本流通, 有效卸载链上交易处理压力, 提升了系统整体通量水平, 而仅把区块链作为清结算工具与安全性保障.

互相连通的链下通道及其节点构成了链下通道网络[13,14]. 在链下通道网络中, 未直接建立通道的节点间也存在着链下交易需求[15], 若在此类节点间均建立直接通道, 成本过高且利用率低下[16,17]. 为降低交易成本, 提升通道利用率, 故仅在部分节点间建立直接通道, 其他节点间的链下交易请求则基于通道路由, 通过互相连通的多条通道所建立的路由路径间接完成. 以闪电网络[13]为例, 其 5 653 个节点间仅有 31 659 条活跃通道 (动态变化)[18]. 链下通道路由算法的优劣也因此成为了链下通道网络能否长期高效、稳定运行的关键.

与传统网络路由算法相比, 链下通道路由算法在算法目标、通道容量等方面存在较大差异: 其路径搜索与交易执行过程紧耦合, 通道容量也会随交易执行动态变化且不自动恢复. 若直接使用传统网络路由算法, 可能引起严重的通道阻塞、通道失衡[19]等问题, 进而造成网络通量下降以及处理延迟大幅上升. 因此, 如何在通道容量、交易成本、处理延迟等条件受限的情况下, 按照某种原则 (通量高、延迟短、成本低等) 找到交易双方间的可用路径集合, 基于该集合完成链下交易处理, 并达到更优的评价指标 (详见第 4 节) 就成为了链下通道路由算法设计的主要目标与挑战.

国内外研究团队借鉴源路由 (source routing, SR)、LandMark、网络流、包转发等算法思想提出了一系列链下通道路由算法. 本文依据路径搜索与交易处理过程中是否存在拆分交易, 分类讨论现有主流单路路由与多路路由算法, 以期为链下通道技术的应用与创新提供参考.

本文主要贡献如下:

(1) 提出链下通道网路层次化架构, 规范各层次功能与层间交互方式. 在此基础上, 建立链下通道路由算法基础模型, 为不同类型路由算法分析对比, 以及新算法设计打下基础.

(2) 从单路路由、多路路由两个方面, 系统阐述了源路由、蚂蚁路由、类 LandMark、网络流、包转发等类型, 共 10 种代表性链下通道路由算法, 为后续研究与应用提供参考.

(3) 提出链下通道路由算法评价体系, 包括基本目标、扩展目标、性能指标 3 个层次的 11 类指标, 结合现有方案测试结果, 综合全面地分析了现有主流通道路由算法优缺点与提升方向.

(4) 基于路由算法发展现状与对比分析结果, 探讨未来路由算法在网络架构、经济激励、算法性能方面的发展趋势, 分析给出一定研究与发展建议.

本文第 1 节对链下通道网络基本概念进行介绍. 第 2 节建立链下通道网络层次化架构与通道路由算法基础模型. 第 3 节系统阐述各类通道路由算法设计策略、应用场景、相互关系与优缺点. 第 4 节提出链下通道路由算法评价体系, 从 3 个层次对主流路由算法进行对比分析. 第 5 节对本文工作进行总结并探讨未来链下通道路由算法发展趋势.

1 链下通道网络基础

为便于读者理解通道路由算法核心思想, 明确算法间差异. 本节将综合闪电网络[13]、雷电网络[14]等链下通道网络实例, 阐述链下通道与链下通道网络的基本概念, 以及链下通道网络两种核心技术 (哈希时间锁定合约、通道路由算法) 的基本原理.

1.1 链下通道

链下通道是指区块链系统用户 (简称“用户”, 即区块链客户端, 持有区块链系统私钥, 能够使用私钥对应的链上账户在链上发起链下通道创建交易, 此类用户不一定参与区块链共识过程) 依据特定规则, 预先在链上锁定资产或状态, 利用智能合约、数字签名等技术建立的链下资产或状态流通渠道. 链下通道由参与节点 (简称“节点”, 即链下通道客户端, 其持有创建该通道的链上账户对应私钥) 以及节点间的有向链接构成. 例如, 通道 C_{ij} 包含节点 i 与节点 j, 以及链接 $l_{i\to j}$ 与链接 $l_{j\to i}$. 通道在某方向上的可用容量为某一时刻该方向链接的可用资产或状态数量.

当某方向可用容量为 0 时, 便认为对应链接已断开, 需等待其他方向链接转移或发送链上交易的方式增加该方向资产或状态数量以恢复链接.

根据应用范围不同, 链下通道可分为支付通道[9,13,14]与状态通道[10,20,21]两类: 支付通道仅面向支付场景的链下交互; 状态通道面向任意链上状态的链下操作. 两类通道的基本原理相同, 为简化描述后文多以支付通道为例, 其路由算法同样适用于状态通道.

链下通道生命周期包括 3 个阶段: (1) 通道建立; (2) 链下流通; (3) 链上结算. 图 1 为雷电网络[14]中通道生命周期示意图.

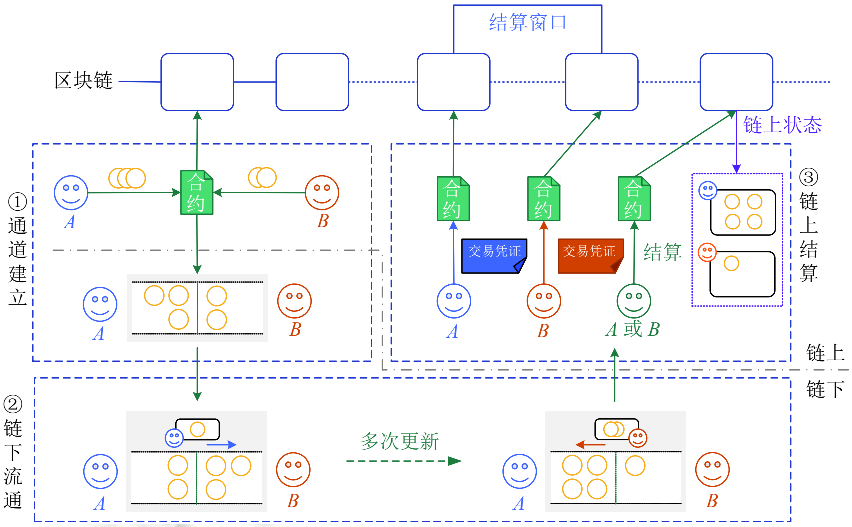


图 1　链下通道生命周期示意图

(1) 通道建立

链下通道所有参与节点向链下通道状态管理类合约发送资产或状态锁定交易. 在锁定交易执行成功后, 通道建立完成, 所有节点在通道内拥有其链上锁定状态的映射, 可以进行链下交易. 例如: 图 1 中节点 A 和 B 分别锁定 3 份和 2 份资产, 通道建立后, A 和 B 也分别在链下持有 3 份和 2 份资产.

(2) 链下流通

链下通道所有节点基于失效树[22,23]等机制, 对链下交易结果达成一致性共识, 完成通道内节点间的链下状态更新, 实现资产或状态的链下流通. 例如: 图 1 中节点 A 首先向节点 B 发送 1 份资产. 此时, 节点 A 和 B 的链下资产余额分别为 2 份和 3 份. 经过多次更新后, 该通道内节点 A 与 B 的资产数量分别为 4 份与 1 份.

(3) 链上结算

链下通道运行过程中, 若通道节点一致同意关闭通道, 则通道节点分别向链下通道状态管理类合约发送最新通道结算凭证, 完成通道结算过程. 例如: 图 1 中节点 A 和 B 分别提交最新状态上链, 在结算窗口关闭后, 他们的链上资产分别更新为 4 份与 1 份. 为了避免节点掉线、提交旧状态等恶意操作造成诚实节点资产或状态损失, 雷电网络设置了"结算窗口"并增加"瞭望塔"服务[14], 基于诚实节点提交或委托瞭望塔服务提交的最新通道结算凭证便可完成通道正确结算, 并对恶意节点采取相应惩罚措施.

通道完成链上结算后, 一般处于关闭状态, 节点若想再次进行链下交易需再次锁定链上状态. 为提升通道应用可扩展性, Counterfactual[10]、Sprites[20]等方案, 除完成锁定状态分配外, 还将依据节点共识的交易证明, 执行链上其他合约调用, 完成通道应用状态迁移. 在结算完成后, 部分方案中通道可基于最新链上状态继续运行.

1.2　链下通道网络

链下通道网络由互相连通的链下通道及其节点组成, 可视作一个加权有向图 $G=(V,E)$. 链下通道网络拓扑结

构如图 2 所示, 其中, V 代表链下通道网络中节点集合, 图 G 中节点对应链上锁定通道状态的账户地址, 且每个节点至少与一个非自身节点建立链下通道; E 代表网络中有向链接集合, 每条通道包含一条或两条 (单向或双向) 的有向链接, 链接权重等于源节点能够向目的节点转移资产或状态数量, 其数值随链下交易执行实时更新, 权重为 0 的链接将从图 G 中移除.

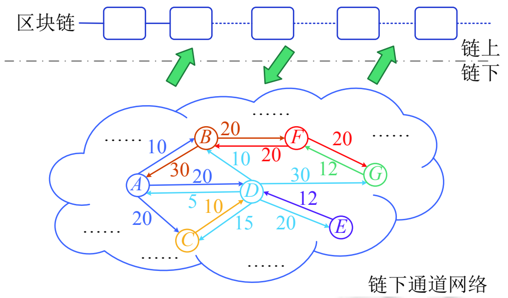
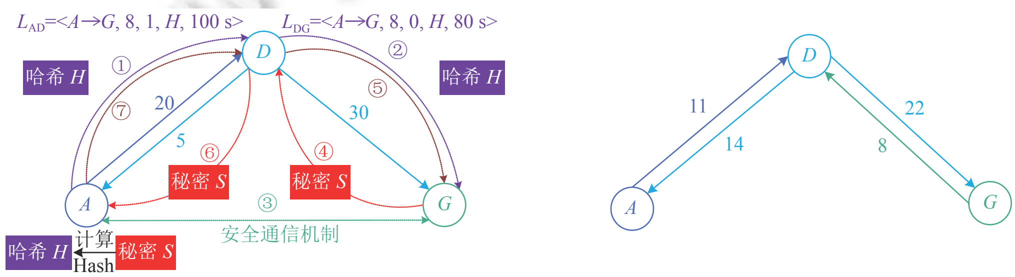


图 2 链下通道网络示意图

未直接建立链下通道的节点间, 可以通过路由算法, 基于全局或局部网络信息 (网络拓扑、通道容量、通道负载等), 计算出节点间可用路径集合 (单路或多路), 使用 HTLC 完成跨通道资金或状态转移. 例如: 图 2 中节点 A 可以通过路径 $A \to D \to G$ 或路径 $A \to B \to F \to G$, 完成一笔向节点 G 转移 8 份资产或状态的链下交易.

1.2.1 哈希时间锁定合约

哈希时间锁定合约是目前链下通道网络通道间协作, 完成跨通道资金或状态转移的主要方式, 其执行流程包括秘密生成、哈希时间锁传递、秘密请求、状态更新 4 个步骤. 由于链下通道网络应用场景不同, 方案间[13,14]哈希时间锁构造、状态更新等操作会有一定差异. 图 3 为简版 HTLC 执行流程示意图.



(a) 状态更新前 (b) 状态更新后

图 3 HTLC 执行流程示意图

(1) 秘密生成

节点 A 作为链下交易发送方, 基于账户私钥或随机函数等为链下交易生成一段唯一的字符串 (秘密 S), 并通过哈希函数得出秘密 S 对应哈希摘要 H. 在生成秘密 S 及摘要 H 后, 节点 A 即可本地构造用于完成此笔路由交易的哈希时间锁 L, 其由〈交易方向, 锁定金额 (转移金额), 交易费, 哈希摘要, 锁定时间〉五元组构成.

(2) 哈希时间锁传递

本阶段完成哈希时间锁从交易发送方至接收方的逐跳传递过程, 对应图 3 中步骤①与步骤②. 基于通道路由算法得到的可用路径集合, 节点 A 选择路径 $A \to D \to G$ 完成向节点 G 转移 8 份资产的链下交易. 首先, 节点 A 构造哈希时间锁 $L_{AD} = \langle A \to G, 8, 1, H, 100\ \mathrm{s} \rangle$, 表示节点 A 将锁定 9 份资产 100 s 用于本交易执行, 若节点 D 在 100 s 内能够提供哈希摘要 H 对应秘密, 就能得到 1 份资产的交易费. 节点 D 收到 L_{AD} 后, 构造哈希时间锁 $L_{DG} = \langle A \to G, 8, 0, H, 80\ \mathrm{s} \rangle$, L_{DG} 锁定时间小于 100 s, 是为了避免因交易处理延迟而可能造成的 L_{DG} 解锁成功但 L_{AD} 解锁失败的情况.

(3) 秘密请求

节点 G 收到 L_{DG} 后, 通过安全通信机制向节点 A 请求哈希摘要 H 对应的秘密 S, 对应图 3 中步骤③.

(4) 状态更新

状态更新阶段将按照与哈希时间锁传递相反的顺序依次解锁对应通道上哈希时间锁, 对应图 3 中步骤④至步骤⑦. 节点 G 收到 L_{DG} 的 80 s 内, 与节点 A 通信获得秘密 S, 在发送给节点 D 之后即可获得 L_{DG} 内的 8 份资产. 节点 D 与节点 A 之间的操作相同, 最终完成节点 A 向节点 G 转移 8 份资产的链下交易, 路由节点 D 也因提供服务而获得相应收益.

在实际运行过程中, 为避免因路由节点掉线或向上一跳节点提供秘密后拒绝解锁等问题造成路由路径上诚实节点资金损失, 哈希时间锁的锁定时间通常大于链上交易共识时间. 当此类情况发生时, 持有秘密 S 的节点立即将秘密公布到链上, 经链上共识后相关通道哈希时间锁便会自动解锁.

1.2.2 链下通道路由算法

链下通道路由算法的目标是根据链下通道网络全局或局部信息, 在通道容量、交易成本、处理延迟等条件受限的情况下, 按照某种原则 (通量高、延迟短、成本低等) 找到交易双方间的可用路径集合, 基于该集合完成链下交易处理, 并达到更优的评价指标. 尽管传统网络路由算法与链下通道路由算法存在一定相似性, 但传统网络路由算法并不适用于链下通道网络, 主要原因有:

(1) 算法目标: 链下通道路由算法的路径搜索与交易执行过程紧耦合, 其目标是为每笔链下交易搜索满足条件的可用路径集合, 并完成交易执行, 相同节点间不同时刻交易的路由路径可能存在较大差异; 传统网络路由算法多为前置式路由算法, 其路径搜索与数据传输可分离, 旨在维护本节点到目的节点间稳定、延时短的传输路径, 对于目的地址相同的待转发数据包则采用相同策略处理.

(2) 容量 (带宽) 状态: 链下通道可用容量具有方向性, 不同方向容量随链下交易执行实时更新、互不干扰, 且链下交易执行完成后不自动恢复; 数据链路可用带宽与待转发数据包大小无直接关联, 且在数据传输过程完成后便自动恢复.

根据路径搜索与交易处理过程中是否存在拆分交易, 链下通道路由算法可分为单路路由与多路路由两类. 图 4 展示了多路路由示意图 (此处省略交易费).

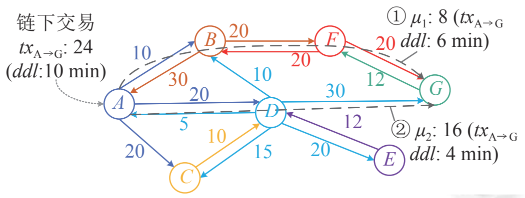


图 4 多路路由示意图

节点 A 收到链下交易请求 $tx_{A\to G}$: 10 min 内, 由节点 A 至节点 G 转移 24 份资产. 节点 A 首先获取其所在通道最新信息, 单路路由无可用路径 ($A\to G$ 任意路径容量均小于 24), 故采用多路路由方式计算可用路径集合. 节点 A 在综合考虑延迟、成本、并发性等因素后, 选择某一路径组合完成该链下交易请求. 图 4 仅为一种可用路径的解, 将链下交易请求拆分为 2 笔实际链下交易, 分别由路径① $A\to B\to F\to G$ 与路径② $A\to D\to G$ 执行. 为保证链下交易请求处理的原子性 (即路径①与②的两笔交易同时成功或失败), 交易处理过程可遵循原子多路支付协议进行[12].

2 链下通道网络架构

本节综合代表性链下通道网络方案, 提出了链下通道网络层次化架构, 确定链下通道网络各层功能以及层间交互方式, 有助于未来开展局部或全局优化工作. 基于链下通道网络层次化架构, 进一步建立链下通道路由算法基

础模型, 便于开展路由算法研究与对比工作.

2.1 链下通道网络架构

基于链下通道网络交易请求处理过程的不同功能, 本文将链下通道网络划分为 4 个层次: (1) 链下通道层; (2) 通道网络层; (3) 通道路由层; (4) 通道应用层. 各层基本功能如图 5 所示.

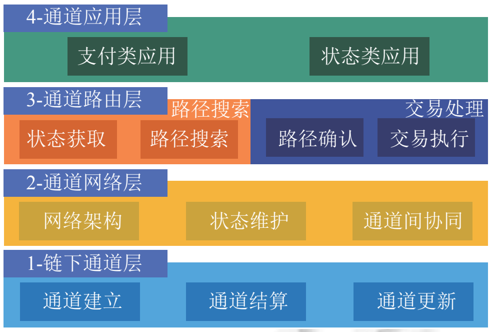


图 5 链下通道网络架构

(1) 链下通道层

链下通道层与区块链直接交互, 用于节点所在链下通道的全生命周期管理, 为通道网络层实现通道间协同提供交易凭证构造、状态锁定等基础服务. 通道建立模块通过发起链上交易锁定状态, 并建立链下通道内的资产或状态映射; 通道结算与通道更新模块引入奖惩机制、时间锁、版本号、结算窗口等设计, 确保能够实时获取通道最新状态, 满足通道网络层、应用层需求.

(2) 通道网络层

通道网络层基于链下通道状态、节点通道分布等信息构建链下通道网络, 为上层应用提供网络状态获取、通道间协同等服务. 网络架构即网络中节点与链接组织、交互的框架, 为节点状态维护与通道间协同提供基础, 现有链下通道网络架构主要有点对点架构[13]、多中心架构与层次化架构[24]3 类. 状态维护模块中, 节点基于网络架构与自身分类维护网络拓扑、通道容量、历史交易量等信息, 不同方案在状态维护方式 (本地/第三方)、维护范围 (全局/局部)、更新频率 (定时/实时/按需) 有所不同. 不同路由状态维护方式将很大程度影响算法的可扩展性、有效性、搜索效率等指标. 通道间协同模块为路由路径上链下交易请求处理提供交易原子性[11,12]、容量锁[25]、拥塞控制[7]等服务, 从而提升网络交易处理通量水平, 减少交易回滚、通道阻塞等情况发生.

(3) 通道路由层

通道路由层路径搜索模块接收来自通道应用层发来的链下交易请求, 搜索符合链下交易请求处理需求的可用路径集合并传递给交易处理模块使用. 状态获取模块基于通道网络层服务与节点设置, 获取最新网络状态. 路径搜索模块基于网络最新状态与链下交易请求, 选择单路路由或多路路由算法, 通过本地计算、节点间通信或访问第三方服务的方式 (例如: 雷电网络中寻路服务[14]) 得到可用路径集合.

交易处理模块基于路径搜索模块传来的链下交易请求与对应可用路径集合, 完成链下交易处理过程并将状态更新通知各层. 路径确认模块综合各项指标 (第 4 节将详细阐述), 选择一条“最优”(节点视角) 路径. 交易执行模块调用通道间协同模块功能, 完成链下交易请求处理, 更新通道状态. 若所选路径执行失败, 则通道状态回滚至交易处理前, 路径确认模块选择下一组可用路径. 交易执行模块再次执行该交易请求, 直至交易成功执行或无可用路径返回交易执行失败.

(4) 通道应用层

通道应用层直接与链下通道网络节点交互, 通过调用下层服务, 提供通道状态管理、节点间通信等功能, 并依据节点操作生成链下交易请求, 交由下层服务完成路径搜索与交易执行等过程. 依据应用场景不同, 通道应用可分为支付类应用与状态类应用两类. 支付类应用面向链上原生代币、智能合约代币等单一可量化资产的链下支付过

程. 状态类应用面向任意状态 (五子棋、围棋、条件合约等) 的链下交互过程.

2.2 链下通道路由算法基础模型

链下通道路由算法的路径搜索与交易执行过程紧耦合. 在设计时需要综合考虑网络状态以及交易请求特点. 在网络架构固定的情况下, 通道路由算法执行主要涉及通道路由层, 即状态获取 (state acquisition)、路径搜索 (path finding)、路径确认 (path confirmation) 与状态更新 (state updation) 4 个阶段. 链下通道路由算法基础模型 (如图 6) 的输入为链下交易请求以及网络状态; 若交易请求执行成功, 模型输出为更新后的网络状态; 若请求执行失败, 模型输出为先前网络状态. 从链下通道网络全局来看, 路由算法 4 个阶段并行执行, 每完成一次, 所涉及路由通道就迁移至下一状态.

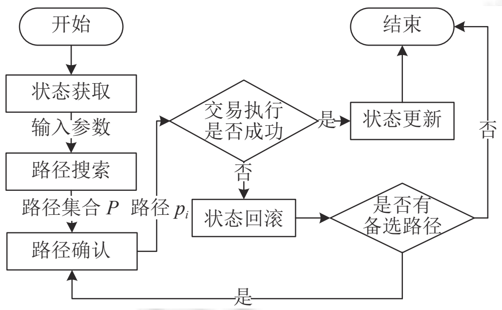


图 6 链下通道路由算法基础模型

(1) 状态获取

状态获取是路径搜索的基础. 节点调用通道网络层状态维护模块实时维护所参与通道状态, 实时或定期获取链下通道网络局部或全局状态, 更新网络拓扑、通道容量、节点参数、通道拥塞情况等信息. 收到应用层发来的链下交易请求 tx 后返回“最新”(节点视角) 网络状态 S_0, 供路径搜索与节点维护等模块使用.

(2) 路径搜索

路径搜索是路由算法的核心. 源节点根据链下交易请求, 基于本地网络状态信息或与网络中其他节点协同得到的网络初始状态 S_0, 在有限成本内, 搜索源节点与目的节点之间的可用路径集合 P, 满足交易请求在通道容量、处理延迟、搜索通信、交易成本等方面的要求. 路径集合 P 中每个元素为一条路径 (单路路由) 或一组路径 (多路路由).

(3) 路径确认

路径确认是链下通道网络平稳运行的关键. 节点依据路径搜索阶段得到的可用路径集合 P, 结合交易成本低优先、处理时间短优先、通道负载低优先等策略, 确定链下交易请求的执行路径 p_i. 适宜的路径确认策略, 可以有效避免通道阻塞、失衡、路由选择中心化等问题, 减少请求处理时间, 降低交易成本, 提升并发性, 从而有助于网络整体通量水平的提升.

(4) 交易执行

状态更新阶段将更新路径 p_i 所涉及到的通道的状态, 包含状态锁定、交易执行、状态更新 3 个步骤. 其中, 状态锁定保证通道拥有足够容量完成交易处理; 交易执行采用 HTLC、AMP 等协议保证原子性. 若交易请求执行成功, 则完成状态更新操作; 否则, 回滚至初始状态 S_0, 返回阶段 3 获取下一执行路径 p_{i+1}, 若无满足条件可用路径, 则交易请求执行失败, 网络状态回滚.

以上是链下通道路由算法基础模型, 在进行实际路由算法设计过程中, 各阶段具体步骤可能有一定交叉或顺序调整.

3 算法分类

本文依据路径搜索与交易处理过程中是否存在拆分交易, 将现有通道路由算法划分为单路路由与多路路由两

类. 单路路由适用于不允许进行交易拆分的场景, 其依据链下通道网络全局或局部信息, 寻找一条路由路径 (即一笔链下交易) 完成链下资产或状态转移, 在网络规模较小、节点连通度高、交易平均转移数量少的情况下, 具有较高通量水平 (与链上通量定义不同, 链下通道网络通量水平指在保证原子性的前提下, 单位时间内转移资产或状态数量) 与路由效率. 但在大额交易处理过程中, 单路路由算法可能长时间阻塞通道, 造成局部网络资产或状态流动性 (或称“通道流动性”) 降低, 甚至引起通道大规模死锁, 使得网络整体通量水平大幅下降.

多路路由算法则是寻找至少一条路由路径 (即至少一笔链下交易) 完成链下交易处理过程. 通过交易拆分, 降低每条路由路径处理压力, 提高交易处理成功率与效率. 在网络规模较大、节点连通度分布随机、交易平均转移数量大的情况下, 多路路由算法往往能够达到更高通量水平. 在通道流动性方面, 多路路由通过交易拆分处理, 能够有效减少通道阻塞问题, 提升网络整体流动性. 但在路径搜索过程中, 多路路由对于网络状态时效性要求更高, 节点间通信开销大, 同时引入了多路交易处理原子性等问题, 使得路由效率较低且隐私保护设计更加困难.

在具体实现过程中, 链下通道网络节点可以本地运行单路路由或多路路由算法. 计算、通信受限的节点, 可以通过请求第三方寻路服务[14]的方式, 获取从第三方服务角度的较优可用路径集合, 完成链下交易处理. 图 7 为代表性链下通道路由算法及其分类.

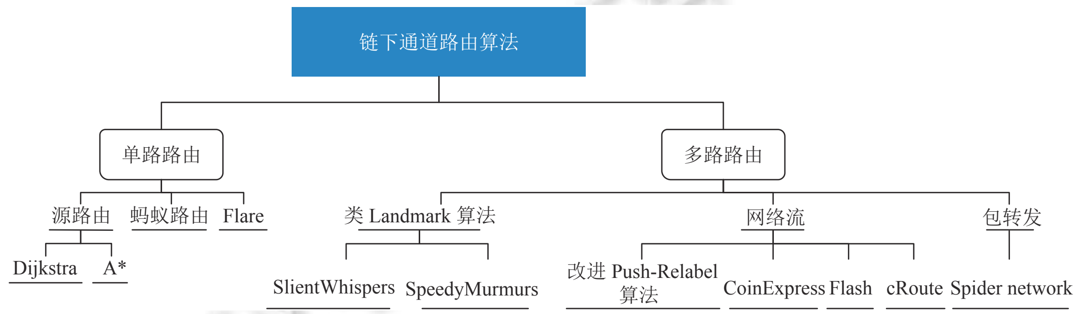


图 7 代表性链下通道路由算法

3.1 单路路由算法

在链下通道网络发展初期, 链下支付为其主要应用场景, 网络规模、交易转移数量较小, 节点通过直接通信或依据链上状态即可投入较低成本完成全网状态维护, 故早期链下通道路由算法采纳并改进传统网络中源路由等路由算法. 但是, 随着网络规模扩大, 节点保存全网信息成本过高且网络变化动态性增强, 国内外研究团队陆续提出了蚂蚁路由[26]、Flare[27]等算法以缓解该问题.

3.1.1 源路由

链下通道的建立与结算, 以及通道容量等信息都需要在上链共识后才可生效, 节点可以通过监测链上信息来维护网络全局状态, 这与源路由算法寻找路由路径的过程相符合. 源节点在发起交易时, 基于网络全局视图, 使用 Dijkstra、A*等经典寻路算法便可找到从源节点到目的节点满足通道容量、处理时间、交易成本等需求的可用路径集合. 在路径确认与交易执行阶段, 采用成本低优先或处理时间短优先等原则, 逐一试错, 完成链下交易处理. 闪电网络[13]与雷电网络[14]早期方案便采用源路由算法进行通道路由搜索.

但实际运行过程中, 链下通道实时容量与节点状态并未存储在链上, 早期方案中节点无法获取实时路由通道容量分布. 若交易执行过程中路由通道容量小于转移容量或节点离线时, 就会造成交易回滚、通道阻塞等问题. 与此同时, 由于节点本地维护网络全局状态, 随着网络规模扩大, 节点运行成本大幅上升. 为缓解以上问题, 降低链下交易失败几率, 在改进方案中[28], 源节点通过发送洋葱数据包对候选路径进行探索, 获取通道即时状态信息. 既保证通道状态时效性, 又尽可能减小节点敏感信息公开范围, 起到一定隐私保护的作用.

3.1.2 蚂蚁路由

蚂蚁路由 (ant routing)[26]受到蚂蚁行为的启发, 利用蚂蚁寻找食物时使用“信息素”标记路径的特点, 实现了一种去中心化的路由算法. 网络中节点只保存局部路由信息, 维护与邻居节点的动态通道状态, 通过三阶段转发消息

来寻找到从发送方到接收方的可用路径. 三阶段信息传递中用到了 3 种信息, 分别称为“信息素种子”“匹配种子”和“确认种子”, 信息传递过程如图 8 所示. 三阶段的具体流程如下.

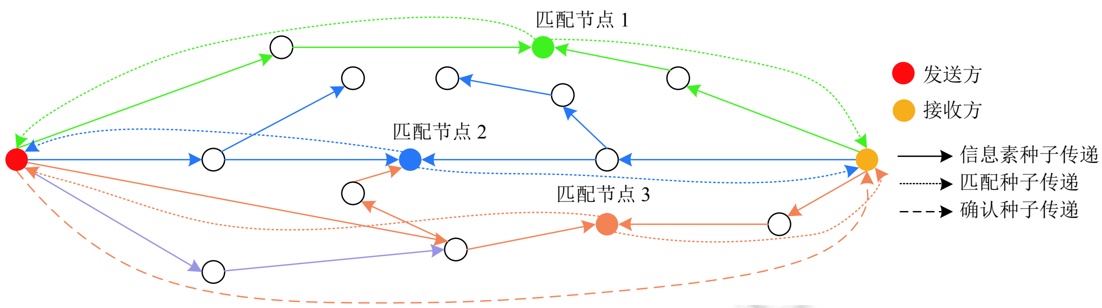


图 8　蚂蚁路由算法示意图

(1) 信息素种子传递

发送方与接收方分别构造信息素种子, 并向网络中相邻节点广播. 每个节点存储种子以及上一跳节点. 当双方的种子在网络中某个节点 (称为匹配节点) 上相遇时, 则说明存在可达性路径. 信息素种子格式如下:

$$S(\varepsilon) = (\varepsilon|R, c, f, F, t),$$

其中, $\varepsilon = 0$ 代表发送方, $\varepsilon = 1$ 代表接收方; R 表示发送方与接收方线下协商一个大随机数; c 表示到达当前节点时经过的跳数; f 为到达当前节点时, 预计累计收取的交易费用; F 表示发送方为转发交易而愿意支付的交易费用阈值, 发送方会将此值发送给接收方; t 表示信息在节点内存中存储的时间阈值.

(2) 匹配种子传递

匹配节点对信息素种子进行检查, 包括两个方向上传来的信息素种子中的累计收取交易费用和是否小于发送方设置的总费用阈值、种子是否超时等. 对于满足条件的信息素种子构造匹配种子, 按照信息素种子的转发路径, 分别向发送方和接收方转发. 匹配节点本地记录从发送方至接收方的下一跳, 从而将两段路径连接起来. 匹配种子格式如下:

$$M(\varepsilon) = (0\varepsilon|R, id, c, f, t),$$

其中, id 为匹配节点随机生成, 用于标识此次路由搜索, 从而避免因发送方与接收方之间可能存在多条路径, 即有多个匹配节点, 仅使用随机数 R 无法区分的问题.

(3) 确认种子传递

发送方根据返回的一个或多个匹配种子做选择, 向接收方发送确认种子, 从而完成路径搜索的工作. 确认种子格式如下:

$$C = (000|R, id, t).$$

基于确认种子, 路由路径上节点即可通过通道间协作完成链下交易处理.

此算法的优点是借鉴蚂蚁寻找食物的思路, 实现了去中心化路由, 网络中所有节点均执行相同操作, 而不依赖于一些特殊权限的节点; 提出了消息格式、存储结构与节点交易费设计方案. 但是, 其关于链下通道交易处理过程中资金锁定、交易处理原子性、通道平衡、安全性与隐私保护的内容仍需进一步论证.

3.1.3　Flare

源路由等传统通道路由算法需要依靠全局的网络拓扑信息来进行路径发现与选择. 但是随着通道网络规模不断扩大, 维护全局的网络状态需要消耗大量的存储空间与网络带宽, 给普通节点带来了巨大压力. 基于 Landmark[29] 等全局信标的路由算法能够通过随机选择几个节点作为全局信标, 其余节点基于到信标节点的路径来恢复出到目的节点的路径, 最终完成交易执行. 但也因此全局信标节点需要承担大部分的路由转发操作, 需要节点拥有较高的计算存储能力与较高的网络带宽, 造成全局信标的候选库相对有限. 此外, 全局信标节点的获利更多, 信息更加公

开, 更容易遭受到 Sybil 攻击与 DDoS 攻击. 因此, Flare 算法[27]通过结合本地信标与全局信标算法, 普通节点本地维护局部网络拓扑信息, 大大降低了对于链下通道网络节点的性能需求. 图 9为 Flare 算法示意图.

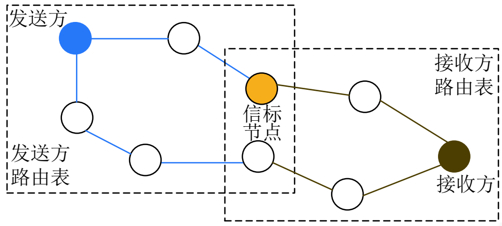


图 9 Flare 算法示意图

在 Flare 算法中, 每个节点以路由表的形式维护着自身在通道网络中的局部视野. 其中路由表包含一定跳数内的邻近通道以及到信标节点集合的有效路径. Flare 算法通过选取在地址空间上距离自身较近的节点作为信标节点, 一方面由于节点地址的随机性大大提高了节点对于整个网络的可见性; 另一方面在路径搜索过程中, 通过不断缩小与目的节点在地址空间上的距离, 保证了算法的收敛性, 并且提高了路径随机搜索成功率. Flare 算法中节点的网络视野就如同战争迷雾一般, 对于较近的区域具有很强的可见性, 对于较远的区域通过随机搜索, 以较小的代价维护高效的路由状态信息, 以很高的概率确保其他节点的可达性.

Flare 算法通过结合源节点与目的节点的路由表信息进行路径探索, 当交易双方路由表中存在共有节点时, 便获取到了一组可达的候选路径. 如果不存在共有节点, 那么源节点寻找目前已知路径中地址空间上距离目的地址较近的节点, 并请求该节点的路由表加入到已有的通道集合中继续路径搜索, 直至有足够多的候选路径或是请求超时. 当候选路径足够多时, 通过发送洋葱路由包, 探测通道的实时可用余额、交易费以及网络情况等动态信息, 对于各个候选路径进行综合评分排序, 选取一条容量充足、时间、成本条件满足的路径进行路径确认与交易执行, 最终完成交易.

相比于传统算法, Flare 以总网络大小为对数的空间复杂度来维持路由状态, 以较小的带宽消耗保证了路由算法的高可靠性. 不过也正是由于 Flare 算法随机路标节点的选择, 最终路径探索的可靠性是概率性的. 虽然 Flare 保持了一个较高的可靠性水平, 但是仍然存在失败的可能, 不能保证路由算法一定在发送者与接收者之间找到可用路径. 如果在路由探索或是路径确认中失败, 节点需重新进行路由搜索或只能通过链上汇款来完成交易. Flare 为单路路由算法, 受到路径上通道容量的限制较大, 对于大额交易, Flare 可能因为高容量通道的缺少而失败; 或倾向于选择高容量通道, 可能造成路由选择中心化、通道严重阻塞等问题, 降低网络安全性与通量水平.

3.2 多路路由算法

针对单路路由存在的大额交易成功率低、路由选择中心化、通道阻塞严重等问题, 国内外研究团队借鉴 Landmark、网络流、包转发等算法思想, 重新建立链下交易处理模型, 利用部分链下交易转移状态可拆分的性质, 将一笔链下交易请求拆分为多笔实际链下交易, 降低每笔交易路由负载. 在网络状态与负载动态变化的情况下, 能够保证较高交易处理并发性与成功率, 达到较高链下通道网络可扩展性与通量水平.

3.2.1 类 Landmark 算法

Landmark 的概念由 Tsuchiya 于 1988 年提出[29]. 类 Landmark 算法以 Landmark 节点为根节点构造整个网络的生成树, 生成树的质量直接决定路由质量. 算法中可以有一个或多个 Landmark: 当只选取一个时, 只有一棵生成树来提供路径搜索服务, 类似于单路路由; 当有多个 Landmark 时, 不同生成树中的多条路径共同提供路由服务.

类 Landmark 算法的大致流程包括: (1) Landmark 节点选取: 依据节点连通度大小、节点可信度高低等因素或采用随机抽取的方式选定 Landmark 节点, 作为生成树的根节点; (2) 生成树构造: 采用传统的 BFS 算法或者其变种构造全网生成树; (3) 可达路径搜索: 在一棵或者多棵生成树中搜索从发送方到接收方的可达路径, 根据算法设计可以经过/不经过 Landmark 节点; (4) 可用路径计算: 在可达路径的基础上, 增加链下交易对于通道容量、处理

时间等方面的需求, 计算出可用路径集合; (5) 交易多路执行: 发送方基于可用路径集合, 确定交易拆分策略与路由路径, 完成交易执行; (6) 网络状态更新: 由于交易执行会引起通道容量变化, 生成树的边权值也随之变化, 生成树需实时/定期更新 (即路由信息更新). 算法示意图如图 10 所示. Landmark 算法中生成树包括两种: (1) 前向生成树: 从根节点到其他节点最短路径构成的树; (2) 反向生成树: 从其他节点到根节点最短路径构成的树. 生成树的构造方法可以采用传统的 BFS 算法或者其变种. 下面主要介绍两种代表性算法.

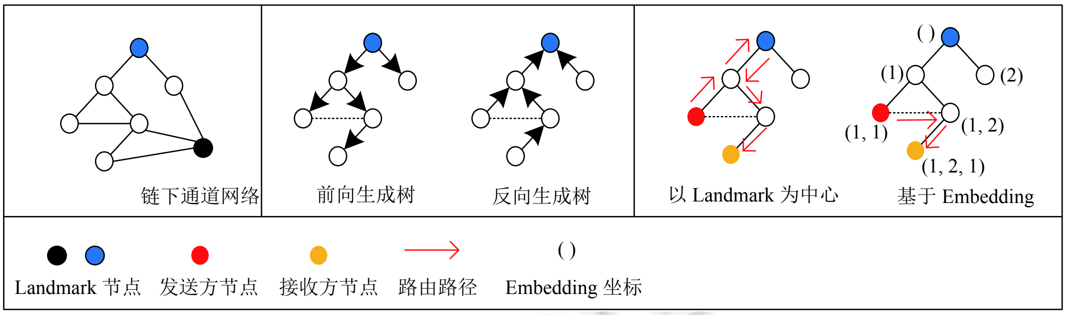


图 10 Landmark 算法示意图

3.2.1.1 SilentWhispers

SilentWhispers[30]是第一个在分布式信贷网络中实现路由转发并支持隐私保护的算法, 其采用传统的基于 Landmark 路由的信贷网络设计理念, 以 Landmark 为中心完成在分布式环境下的路径搜索与交易执行, 并使用安全多方计算等技术保护节点隐私.

SilentWhispers 假定 Landmark 节点固定, 且全网熟知. 为提升网络可扩展性, 其采用分布式 BFS 的方法构建生成树, 每个节点仅维护自身到所有 Landmark 节点的下一跳, 以及从 Landmark 到邻居节点的下一跳. 当收到邻居节点的寻路请求时, 根据本地路由信息判断是否转发寻路.

在可用路径计算过程中, 交易发送方需确认每条路径的可用容量 (即此路径上所有通道中的最小容量). 为了避免通道容量的信息泄露, 本方案提出了基于数字签名链与安全多方计算的隐私保护机制: 可达路径中每个节点对所在路径的前一个与后一个节点的身份信息签名, 将签名信息与其可用容量的部分秘密值一同发送给该路径对应的 Landmark 节点. Landmark 节点根据收到的所有签名信息构造完整数字签名链, 结合可用路径信息验证节点身份; 根据所有部分秘密值, 本地计算本路径可用容量, 并将结果反馈给交易发送方.

SilentWhispers 算法的主要缺点有: (1) Landmark 节点固定, 且可达路径必须经过 Landmark 节点, 未充分利用生成树之外通道, 通道利用率低, 造成一定资源浪费; (2) 引入安全多方计算, 显著增加网络中节点的计算与通信开销; (3) 生成树周期性更新, 可能导致因网络拓扑、通道容量等信息未及时更新而交易执行失败的情况.

3.2.1.2 SpeedyMurmurs

SpeedyMurmurs 算法[24]的基本思想来源于 Friend-to-Friend 网络中的 VOUTE 算法[31], 适用于较大规模网络的链下交易处理, 其网络架构与 SilentWhispers 类似, 主要区别在于 SpeedyMurmurs 方案的生成树中每个节点会设置唯一坐标用于可用路径搜索.

为了缓解 SilentWhispers 中因先寻路、后计算造成的并发交易冲突、通道阻塞等问题, 本方案将可达路径搜索与可用路径计算过程合并, 发送方首先将交易总转移数量随机拆分后, 在生成树路径搜索过程中判断通道可用容量是否满足锁定需求. 若部分生成树寻路失败, 则发送方依据失败信息重新拆分交易, 直至所有路径容量锁定成功. 在所有路径完成容量锁定后, 便可开始交易多路执行过程. 由于路径搜索过程中使用坐标计算最短路, 每个节点向离接收方更近的方向寻路, 存在“捷径”的情况下不必经过 Landmark, 可有效缩短路由长度并充分利用非生成树中的捷径通道, 通道利用率有所提升.

为了提升生成树的质量, 本方案选择高连通度的节点作为 Landmark 节点, 在树的构造过程中, 优先选择容量

充足的双向通道, 确保路由能力强的节点处于更高层次, 并依据节点所处位置 (子节点坐标基于父节点坐标生成) 生成其对应坐标[32], 便于计算节点间距离. 此外, SpeedyMurmurs 采用了实时局部更新通道容量的方法, 仅在两节点间通道容量归零或由零变为正值的情况下, 更新对应节点及其孩子节点的坐标以及在树中的位置, 相对于 SilentWhispers 的周期性重构生成树, 本方案开销更低. 但在某些情况下, 如果靠近根部节点的通道失衡 (双向通道变单向), 依旧需要重新建立生成树, 生成树构建开销会显著增加.

3.2.2 网络流

链下通道网络为链下通道及其节点构成的有向图, 通道每个方向容量为有向边的容量. 链下交易的路径搜索过程可视作一种最大流模型的求解过程: 即寻找源节点与目的节点之间的可用路径组合, 在满足容量、成本、延迟等方面需求的情况下完成资产转移或状态更新. 基于该思路国内外研究团队陆续提出了改进 Push-Relabel 算法[25]、CoinExpress[33]、Flash[17]、cRoute[34]等算法. 网络流类算法示意图如图 11 所示.

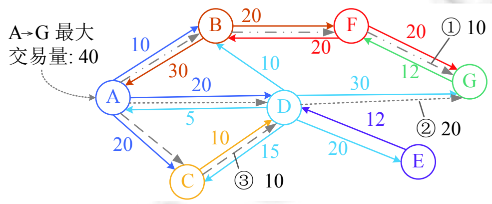


图 11 网络流算法示意图

3.2.2.1 改进 Push-Relabel 算法

链下通道网络路由算法与传统的网络流算法具有一定的相似性, 尽管传统的 Push-Relabel 算法[35]可以高效且分布式的求解类似问题, 但是传统的 Push-Relabel 算法无法应用于实际并行的交易环境中. 因为对于并行路径搜索过程, 一条通道的容量可能被多条流共同使用, 使得实际转移资产或状态数量远大于通道容量, 进而导致交易失败.

为了解决交易处理并行化的问题, 改进 Push-Relabel 算法[25]引入容量锁 (capacity locking) 机制, 即在路径探索的过程中, 各条交易流锁定自身正在使用的通道容量资源, 而其他交易流只可以使用当前未锁定的容量进行交易处理, 保证了通道上实际流量小于通道容量.

相比于单路路由算法, 改进 Push-Relabel 算法可以满足更大交易金额的应用环境, 并且适用于实际并发与分布式场景. 但改进 Push-Relabel 算法在路径探索中的锁定机制可能会导致一笔交易在网络中造成大面积的容量锁定, 降低了通道容量的利用效率, 使得部分交易因为可使用的容量不足而交易失败.

3.2.2.2 CoinExpress

CoinExpress 算法[33]基于 Ford-Fulkerson 最大流算法, 引入改进 Push-Relabel 算法中提出的容量锁机制, 通过分布式动态路由的方式, 寻找一组可完成状态转移的路由路径并解决交易处理并行化问题. 具体而言, 当接收到链下交易请求时, 发起者将依据链下通道网络局部信息, 使用广度或深度优先搜索算法寻找可用路径, 并由交易请求接收者确定交易请求最终的执行路径.

CoinExpress 为满足路由算法与交易处理的及时性限制 (timeliness), 在路径搜索过程中, 为每条 P2P 搜索消息增加通道等待与处理时间、通道有效时间、请求处理截止时间, 保证及时过滤超时请求. 为避免网络规模增大后信息洪范, 增加消息生存时间 (time to live, TTL) 字段, 并且各节点在收到前序节点发来的消息后会等待一段时间, 减少低效信息转发, 提升路由搜索与交易执行效率.

根据实验测试结果[33], 在网络规模适中且考虑及时性限制的情况下, CoinExpress 实际吞吐量相较于改进 Push-Relabel 算法得到了大幅提升, 且路由搜索的通信成本控制在较低水平. 但是, 由于 CoinExpress 算法需全网公开地址, 其隐私保护机制设计需要进一步完善. 此外, 该算法在路径搜索与交易执行过程中并未考虑通道平衡问题, 可能影响其系统性能进一步提升与网络长期平稳运行.

3.2.2.3 Flash

由于交易请求容量分布、节点之间交易频率等因素都能够很大程度地影响路由算法性能. 因此, Flash 团队[17]首先分析了两大代表性区块链系统 (Ripple 与比特币) 一段时间内的交易数据, 发现交易请求容量呈现出长尾分布的特征; 每对发送方与接收方之间的交易具有一定周期性, 且发生在某一节点集合内 (聚类).

基于以上特点, Flash 算法对于“大象流”(转移数量大) 与“老鼠流”(转移数量小) 采取了不同的路由策略.

“老鼠流”单笔转移资产数量小, 单路或少数路径即可满足其容量需求, 但节点在发送请求之后希望尽快获得交易结果反馈. 结合“老鼠流”周期性与聚类的特点, Flash 网络中节点基于全网拓扑, 使用 Top-K 最短路算法在本地建立到其他节点的路由表. 在交易请求处理过程中, 直接依据路由表中路径逐一试错, 从而有效减少了 Flash“老鼠流”路径搜索时间与成本, 降低交易请求处理延迟.

“大象流”单笔转移资产数量大, 单路路由很难满足其容量需求且容易造成通道长时间阻塞, 影响其他交易请求执行. 因此,“大象流”处理需要发送方与接收方之间的多路路由路径协作共同完成资产转移. Flash 算法通过改进 Edmonds-Karp 最大流算法, 使用广度优先搜索找到发送方与接收方之间的 k 条路径, 求解多路路由能够转移的最大容量, 能够有效避免 Edmonds-Karp 算法本身收敛速度慢等问题. 与此同时, 引入交易费估算函数, 对比不同路径交易成本. 尽管增加了路径搜索成本、交易费与处理延迟, 但是能够保证“大象流”以较快速度、较高成功率完成执行, 对于系统整体通量提升与长期、稳定运行有重要意义.

根据 Flash 团队[17]的测试结果可知: 相较于 Spider[7]与 SpeedyMurmurs[24], Flash 在保证交易请求成功率较高的情况下, 大幅提升成功转移资产数量. 通过控制“大象流”比例, Flash 的平均路由搜索延迟有明显降低. 但是 Flash 算法对于系统安全性、隐私保护、通道平衡等方面的分析与设计仍需完善, 且节点本地维护网络全局拓扑结构算法可扩展性较低.

3.2.2.4 cRoute

针对通道失衡造成的链下通道网络拓扑结构频繁变化、交易请求处理严重阻塞、性能下降等问题, Celer Network 提出了一种可证明最优的分布式均衡路由算法 (cRoute).

cRoute 算法[34]受无线网络中 BackPressure 路由算法[36]启发, 节点依据外界交易请求以及所在通道历史资产转移情况, 计算通道实时向各目的节点的拥塞失衡权重 (congestion-plus-imbalance weight, CPI weight), 从而得出网络实时拥塞梯度并确定路由方向, 而不计算交易请求发送方与接收方之间实际路由路径. cRoute 算法运行过程可以看似水从高往低的流动过程, 交易请求的发送方处于“高地势”, 而接收方对于发往自己的请求地势为 0, 即可保证在无需计算具体路由路径的情况下, 完成状态转移过程.

cRoute 算法给出网络稳定、网络平衡、吞吐量可支持、吞吐量范围、吞吐量最优的形式化定义, 通过建立 Lyapunov 方程及其条件偏移, 证明了 cRoute 算法在无需提前获取任何关于链下交易的统计信息的前提下能够达到吞吐量最优: 即只要存在一种路由算法能够使得支付通道网络保证稳定且平衡, 那么 cRoute 算法也能够达到.

在隐私保护方面, cRoute 算法由于无需获取全网路由路径, 因此路由中继节点并不易推测出交易请求初始发起节点信息, 能够保护交易处理路由路径以及发送放与接收方节点隐私. Celer Network 团队提出在保证吞吐量最优的情况下, 可以结合洋葱路由保护交易请求的目的地址隐私.

通过实验模拟[34], 在包含 77 个节点, 254 条双向通道的支付通道网络拓扑中, cRoute 算法与 Flare[27]、SpeedyMurmurs[24]相比具有较高的实时吞吐量与通道利用率. 但是, cRoute 现有测试方案缺乏对于参与节点作恶、网络通信成本、交易请求处理延迟、算法可扩展性、交易费等方面更详细的设计与对比分析.

3.2.3 包转发

链下通道网络交易金额受限于路径容量, 直接应用传统网络路由算法还会导致网络拥塞以及通道容量失衡等问题. 特别是当使用最短路等算法进行路由探索时, 缺乏对于整个通道网络交易负载环境的考虑, 往往会让交易始终沿着一条固定路径传播 (路由选择中心化). 例如, 存在两条 A 到 B 的候选路径 $A \to C \to B$、$A \to D \to E \to B$, 最短路算法将一直推荐 A 节点沿着 $A \to C \to B$ 路径传播交易, 直至该路径上的资金全部被锁定在通道一端, 从而造成该通道的拥塞, 并且另外一条长路径无法得到充分利用. 与此类似, 传统网络流等其他的多路路由算法, 也会

导致某些通道在单个方向上有大量的交易, 最终使得这个方向上的余额耗尽.

Spider 算法[7]则通过采用一种类似于包转发机制来达到平衡路由的目的, 实现全网通量的提高, 其示意图如图 12 所示. Spider 将交易切分成多个交易单元, 通过多条路径并行转发. 当路由通道容量不足时, 将交易单元转移至等待队列中. 当容量恢复后, 立即将剩余容量用于队列中交易单元的转发, 避免因暂时容量缺乏所导致的交易失败. 与此同时, 通过对于不同交易单元的状态的感知, 在路径探索中, 调整不同路径上的交易单元的发送窗口, 促使网络整体达到平衡. 具体而言, 为了达到通道平衡, Spider 算法将提升通道阻塞方向的路由费用, 降低相反方向的路有费用, 进而吸引更多资金或状态平衡通道, 保证整个网络的负载均衡.

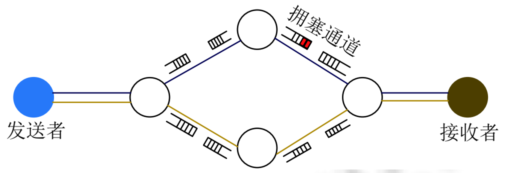


图 12 Spider 算法示意图

相比于其他算法, Spider 更加注重链下通道网络的整体拥塞状态, 以一种启发式的算法尽可能让一条通道双向交易单元数量相同, 使得整个网络交易图接近于一个循环图, 从而保证整个网络的负载均衡以及交易通量的提高, 充分利用了通道网络的全网性能, 并为其提供了理论支持. 但是 Spider 算法中节点本地需要维护全局网络状态可扩展性较差, 并且对于恶意行为的鲁棒性以及网络服务商的激励措施设计仍需进一步研究.

3.3 第三方寻路服务

雷电网络[14]中提供了一种路径发现服务 (pathfinding service, PFS), 向雷电节点提供一条或多条可用路径. PFS 将路径短、费用低并且具备多样性 (寻找到的多条路径中尽量不包含相同的通道) 作为寻路的目标, 但是这些目标之间会有所冲突. 因此, 在通道中使用了称为惩罚项的属性, 通道的基础惩罚项设置为 1 代表如果选择一条通道, 那么通道的权重增加 1. 对于通道内的费用和多样性依照节点的需求设置为不同的值, 比如某条通道的费用的惩罚项为 100, 代表如果选择这条通道, 这条通道每收取 1RDN (雷电网络中使用的代币) 的费用, 通道的权重就增加 100. 对于多样性同理. 通道最终权重为多个惩罚项之和.

PFS 将整个雷电网络视为加权图, 运行双向的 Dijkstra 算法得到一组路径, 从中选择权重最小的路径作为最终路由路径. 为了运行 Dijkstra 算法, PFS 需要实时掌握链下通道网络全局状态, 通过监听区块链上发生的通道打开、关闭、容量增加等事件, 记录通道的增加和减少. 对于活跃通道内的容量变化情况, 需要雷电节点向公共矩阵室 (public matrix room) 发布节点当前的通道容量和以及中继时想要收取的费用, 对外发布节点自身路由服务情况. 不愿意做中继的节点可选择不公开通道信息, 对于此类通道, PFS 将其容量记录为 0. 综上所述, PFS 是面向效率的中心化路由算法, 并且会牺牲一定隐私性.

4 评价体系

链下通道网络路由算法的设计初衷是实现链下通道网络中资产或状态的快速、低成本流通, 但由于通道容量受限且分布不均等原因, 通道阻塞、通道失衡、成本上升、路由选择中心化等问题频发. 为实现链下通道网络长期高效平稳运行, 现有代表性通道路由算法结合特定场景, 对以上问题进行了局部优化. 但由于仍缺乏统一的通道路由算法评价体系, 不同算法评价指标交叉、重合, 不利于算法间对比分析与新型路由算法设计. 因此, 本文综合现有方案[13,17,24,26,27,33,34,37]评价指标, 从基础目标、扩展目标、性能指标 3 个层次建立链下通道路由算法评价体系, 涵盖有效性、并发性、可扩展性、通道平衡、路由选择中心化、成本效益、隐私保护、吞吐量、处理延迟、成功率、搜索效率 11 个方面, 并通过总结归纳现有测试结果中的共性结论，得出算法间在不同指标的定性对比结果.

4.1 评价指标定义

4.1.1 基本目标

基本目标指通道路由算法在网络负载 (规模、交易频率等) 变化的情况下, 能够完成路径搜索与交易执行, 并初步解决通道阻塞、失衡等问题, 是链下通道网络运行的基本需求.

(1) 有效性 (effectiveness)

基于全局或局部网络状态信息, 路由算法必须能够找到一条或一组能够成功执行交易请求的路径. 根据网络状态不同, 有效性可分为静态有效性与动态有效性: 静态有效性是指网络状态固定, 路由算法能够搜索出成功执行交易请求的路径集合; 动态有效性是指在网络状态 (通道容量、网络拓扑等) 动态变化的情况下, 路由算法依旧能够搜索出满足条件的路径集合, 并且能够全网分布式执行.

(2) 并发性 (concurrency)

在某一时刻, 多笔交易请求可能选择一条或多条通道同时进行资产或状态转移操作, 会造成通道阻塞, 甚至出现网络死锁的情况. 通道路由算法并发性依据链下通道容量流动性 (实时利用率) 衡量, 通道容量流动性越高, 路由算法并发性越强.

(3) 可扩展性 (scalability)

可扩展性是指随着链下通道网络规模扩大 (通道数或节点数增多)、交易请求发送频率加快, 路由算法依旧能够保证有效性与并发性.

4.1.2 扩展目标

扩展目标是链下通道网络交易处理过程中对路由服务稳定性、成本、隐私保护方面的更高要求, 涵盖网络架构、路径搜索、交易执行过程中的诸多方面,

(1) 通道平衡 (channel balanced)

通道平衡是指在满足通道内节点直接交易需求的前提下, 通道各链接用于提供路由服务的容量基本相等, 从而使得通道能够长期稳定高效的提供路由服务. 为达到该目标, 路由算法设计时需依据一段时间内通道所接收的交易特点, 制定适宜的通道失衡感知与调整策略. 在通道容量与链下交易分布一定的情况下, 链下通道网络阻塞发生频率越低、通道各链接用于路由服务的资产或状态差越小, 通道平衡的效果越好. 一种设计良好的通道平衡策略, 还能够提高路由服务选择的公平性与交易处理并发性, 缓解路由选择中心化的问题, 使得网络资源得到充分利用.

(2) 路由选择中心化 (routing centralization)

路由选择中心化是指链下通道节点经常选择相同 (或部分相同) 的路由路径完成链下交易处理. 在路由服务通道容量充足、负载较低的情况下, 链下节点经常选择此类通道能够短期提升网络整体通量水平. 但路由选择中心化可能由于多笔链下交易并发造成共用通道阻塞; 由于通道长期单向提供路由服务造成通道失衡; 由于长期与固定通道交互造成节点隐私信息泄露等问题, 进而使得网络无法长期高效、稳定运行. 因此, 路由算法设计中, 需要根据链下交易请求、历史交易规律等信息的特点, 避免路由选择中心化, 在满足链下交易处理需求 (通量、延迟、费用等) 的情况下, 减少通道阻塞、失衡等情况的发生, 提升网络整体路由服务质量.

(3) 成本效益 (cost-effectiveness)

成本效益是指从源节点到目的节点平均转移单位资产或状态, 所需支付的交易费 f_{avg}. f_{avg} 越低, 成本效益越高. 在通道容量、处理时间接近的情况下, 通道路由算法选择成本效益高的通道有助于进一步降低链下交易成本. 现有通道路由算法中, 较少考虑成本效益, 仅有少数项目依据转移资产数量、资产锁定时间与通道拥塞情况收取路由费用.

(4) 隐私保护 (privacy protection)

链下通道网络隐私保护主要包括交易信息与节点状态两方面, 涵盖发送方/接收方信息、交易转移数量、路由路径、通道容量、通道负载 5 个部分. 恶意节点独立或联合少数节点作恶的情况下, 链下通道路由算法应保证以上 5 个方面的隐私保护, 具体如下.

1) 交易信息

发送方/接收方信息: 恶意节点无法依据链下通道网络的公开信息获取某一笔交易请求的发送方/接收方信息.

交易转移数量: 恶意节点无法推算发送方设置的初始交易转移数量, 但其可能通过与其他节点联合, 推算出交易转移数量的大致范围.

路由路径: 恶意节点无法获取到某一笔交易请求的完整路由路径.

2) 节点状态

通道容量: 除通道参与双方以及授权访问节点[14], 其余节点无法获取某一通道在特定时刻各方向上的具体容量. 若通道容量信息泄露, 恶意节点独立或联合部分节点便可对链下通道或链下通道网络发起攻击 (DDoS 攻击、Sybil 攻击等). 例如, 利用通道容量的实时信息, 恶意节点能够通过发送“垃圾”交易 (哈希时间锁超时后也不提供解锁信息) 的方式, 长时间阻塞该通道, 使其无法提供路由服务, 干扰正常链下交易处理过程, 严重时造成引起 (局部) 网络“瘫痪”.

通道负载: 恶意节点无法依据通道交易处理时间等信息, 推算某一通道在特定时刻的具体负载信息, 避免恶意攻击行为. 尽管通道负载是通道提供路由服务质量的重要因素, 目前尚无方案提出通道负载隐私保护方案.

4.1.3 性能指标

性能指标用于衡量在网络状态一定的情况下 (输入一定), 通道路由算法在吞吐量、处理延迟、成功率、搜索效率方面的性能高低,

(1) 吞吐量 (goodput)

吞吐量是指基于某种通道路由算法, 在网络状态与待处理交易集合相同的情况下, 路由算法处理完成全部交易请求成功转移的资产或状态总量以及单位时间内完成链下交易请求数量. 从交易转移量与处理速率两个维度衡量通道路由算法吞吐量水平. 对于在截止时间仅完成部分资产或状态转移的交易请求, 并不计算在内.

(2) 处理延迟 (latency)

处理延迟是指路径搜索完成后, 路径确认至交易原子性执行完成所消耗的平均处理时间.

(3) 成功率 (success rate)

成功率是指网络状态与待处理交易请求一定时, 在保证原子性的前提下, 通道路由算法依次执行全部交易请求的成功率.

(4) 搜索效率 (efficiency)

搜索效率是指从路径搜索开始至获取到可用路径集合的过程中, 全网节点所投入的节点通信、数据计算成本. 通信量越少、数据计算量越小、搜索延迟越短的路由算法, 其路由搜索效率越高.

4.2 路由算法综合对比

下面依据第 4.1 节中定义的评价指标, 结合现有方案[13,17,24,26,27,33,34,37]的公开测试数据, 综合对比目前主流链下通道路由算法. 对比结果如表 1 所示, 由于雷电网络寻路服务为基于 Dijkstra 算法的中心化寻路方案, 与其他方案面向场景不同, 故此次未加入对比当中. 针对 表 1 说明如下: (1) SR: 源路由; AR: 蚂蚁路由; Si.: SilentWhispers; Sp.: SpeedyMurmurs; PR: 改进 Push-Relabel 算法; CE: CoinExpress. (2) “√”表示该路由算法能够满足或考虑特定指标; “×”表示该路由算法不能满足或未考虑特定指标; “－”表示路由算法方案中未明确指出. (3) cRoute 测试过程仅构建了 77 个节点、254 条双向通道. (4) 除 SilentWhispers、SpeedyMurmurs 外, 其余方案未明确阐述隐私保护方案, 但引入洋葱路由即可实现发送方/接收方信息隐私保护. (5) 单路路由在交易转移金额较小且通道容量充足的场景中, 交易处理延迟明显低于多路路由算法. 但若在交易处理集合相同的场景下, 由于大额交易的存在, 路由通道可能长时间拥堵, 单路路由交易处理时间较高. (6) Flash 算法对于“大象流”与“老鼠流”采取不同策略,“大象流”基于最大流算法搜索效率较低,“老鼠流”基于最短路径路由能与源路由达到相近效率.

(1) 有效性

现有算法普遍具有很好的静态有效性. Flare 算法[27]路由搜索是一个随机过程, 即便网络状态良好, 依旧存在

无法找到交易请求对应可用路径的情况. SilentWhispers 算法[30]交易处理过程均需要经过 Landmark 节点, 且路由搜索有效性很大程度取决于生成树质量, 路由搜索存在一定随机性. 在动态有效性方面, 通道容量、网络拓扑频繁变化, 将严重影响无法即时获取或更新路由搜索路径上通道容量与链接情况的源路由算法[13,14]与类 Landmark 算法[24,30]. 与此同时, 类 Landmark 算法路由搜索基于生成树, 网络状态频繁变化还将使得其路径搜索质量下降, 甚至无法求解潜在可用路径.

表 1　链下通道路由算法评价指标

评价指标		单路路由			多路路由						
					类Landmark算法			网络流			包转发
		SR	AR[26]	Flare[27]	Si.[30]	Sp.[24]	PR[25]	CE[33]	Flash[17]	cRoute[34]	Spider[7]
有效性	静态	高	高	中	中	高	高	高	高	高	高
	动态	中	—	—	低	低	高	高	高	高	高
并发性	通道利用率	低	低	低	中	中	中	中	高	高	高
可扩展性	网络规模	低	中	中	高	高	低	低	低	—	低
	交易频率	低	—	—	低	低	中	中	高	高	高
通道平衡	是否支持	×	×	×	×	×	×	×	×	√	√
路由选择中心化	能否避免/降低路由选择中心化风险	×	—	×	×	×	×	×	×	√	√
成本效益	是否考虑	√	√	√	×	×	×	√	√	—	√
隐私保护	发送方/接收方信息	√	√	√	√	√	—	—	—	√	—
	交易转移数量	—	—	—	√	√	√	√	√	√	√
	路由路径	—	—	—	—	—	—	—	—	√	—
	通道容量	—	—	—	—	—	—	—	—	—	—
	通道负载	—	—	—	—	—	—	—	—	—	—
吞吐量	交易转移量	低	—	—	中	中	—	—	高	高	高
	单位时间执行交易量	—	—	低	—	中	—	—	—	高	—
处理延迟	交易处理延迟	低/高	低/高	低/高	中	中	高	中	中	—	中
成功率	成功率	低	—	—	中	高	高	高	高	—	高
搜索效率	通信开销	低	高	中	中	中	高	中	低/高	低	高
	计算开销	高	低	低	高	中	高	中	低/高	低	高
	搜索延迟	低	中	中	中	低	高	中	低/高	低	—

(2) 并发性

现有路由算法测试数据缺乏对于通道拥塞的统计分析, 故采用通道利用率对比不同路由算法并发性高低. 多路路由通过交易请求拆分的方式, 将交易处理压力分散至多条路由路径中, 通道利用率高于单路路由算法[34]. 多路路由算法中 Spider[7]、cRoute[34]的路由单位粒度更小; Flash[17]算法对“大象流”“老鼠流”分类处理充分利用链下通道交易周期、聚类的特点, 通道利用率高. 改进 Push-Relabel 算法[25]、CoinExpress[33]、SilentWhispers[30]、SpeedyMurmurs[24]在交易执行过程中的容量锁或资金锁定设计, 通道利用率有所降低.

(3) 可扩展性

随着网络规模扩大, 源路由算法、改进 Push-Relabel 算法[25]、CoinExpress[33]、Flash[17]本地维护全局路由网络状态所需的存储空间与网络带宽显著增加, 造成路由算法可扩展性下降. 蚂蚁路由[26]、Flare[27]、SilentWhispers[30]、SpeedyMurmurs[24]节点本地仅维护网络局部路由状态信息, 且后两者有 Landmark 节点辅助路由搜索与处理过程, 故随着网络规模扩大, 可扩展性依旧能保持在较好水平. 随着交易频率上升, 源路由算法、类 Landmark 算法网络状态更新慢, 更易产生通道阻塞等问题, 可扩展性受限. 改进 Push-Relabel 算法[25]、CoinExpress[33]基于最大流算法与容量锁可扩展性有所上升. Flash[17]对于不同交易差异化处理; Spider[7]、cRoute[34]算法通过增加通道拥塞队列等

设计, 在交易频率上升的情况下依旧能够保持较高可扩展性.

(4) 通道平衡

现有算法中仅有 Spider[7]与 cRoute[34]支持通道平衡设计 (详见第 3.2 节), 通过引入通道平衡机制, 减少通道阻塞, 降低通道容量锁定时间, 并取得了更高通道利用率与交易转移量 (吞吐量)。

(5) 路由选择中心化

源路由算法与 Flare[27]使用路径最短作为最优路径的选择标准; SilentWhispers[30]、SpeedyMurmurs[24]的路由路径往往需要经过 Landmark 节点; CoinExpress[33]等最大流算法将选择最大容量通道, 因此, 以上算法均无法避免路由选择中心化的问题. Spider[7]与 cRoute[34]由于引入通道平衡机制, 路径确认时, 基于通道平衡、降低交易费、避免通道阻塞等多目标进行最优路径选择, 能够一定程度缓解路由选择中心化的问题.

(6) 成本效益

Spider 算法[7]依据通道不同方向阻塞情况动态收取交易费, 吸引更多资金或状态平衡通道, 保证整个网络的负载均衡. 源路由、Flare[27]等算法在路径搜索过程中基于交易转移资金或状态数量及锁定时间按比例收取交易费用. 目前, 链下通道网络交易费设计原则尚处于早期阶段, 后续算法设计有较大研究空间.

(7) 隐私保护

现有路由算法对于链下通道网络隐私保护方面的设计较少, 大多停留在引入洋葱路由实现发送方/接收方信息隐私保护的阶段. 其中, 类 Landmark 算法引入多方安全计算等过程实现发送方/接收方信息以及交易转移数量的隐私保护. CoinExpress[33]等其他多路路由算法在交易转移数量隐私保护方面具有一定优势, 恶意节点仅基于部分路由路径很难获取初始交易转移数量, 但可通过节点共谋, 基于路由通道容量, 推算出交易转移数量的大致范围. cRoute[34]算法基于拥塞失衡权重完成资产或状态转移过程, 并未直接计算发送方与接收方之间可用路径集合, 且每条通道路由方向随交易请求时刻变化, 能够一定程度实现对路由路径的隐私保护.

(8) 吞吐量

交易转移量方面, 基于现有方案实验结果[7,17,34]可得, Spider[7]、Flash[17]、cRoute[34]算法在网络状态与待处理交易集合相同的情况下, 通道利用率高, 交易中资产或状态转移量能够明显高于其他路由算法. 单位时间执行交易量方面, 现有方案普遍缺乏此类指标测试, 依据 cRoute[34]测试结果, cRoute 算法平均每秒执行交易量远高于 SpeedyMurmurs[24]与 Flare[27].

(9) 处理延迟

单路路由受交易请求转移资金或状态数量影响较大, 在交易平均转移数量较小且通道容量充足的情况下, 单路路由交易处理延迟明显低于多路路由算法. 但在交易平均转移数量较大的情况下, 单路路由交易处理时间将大幅上升. 在多路路由算法中, 结合现有方案实验结果[7,17,33], 由于改进 Push-Relabel 算法[25]因缺乏延迟控制机制设计, 其交易处理时间远高于其他多路路由算法.

(10) 成功率

在网络状态与待处理交易请求一定时, 综合现有方案实验结果[7,17,24,33], 多路路由算法成功率高于单路路由算法. 其中, SilentWhispers 算法[30]由于交易处理过程必须依赖 Landmark 节点, 交易处理成功率低于其他多路路由算法.

(11) 搜索效率

源路由算法通过有限的节点间通信便可本地计算出可用路径集合, 属计算资源集中型算法, 通信开销与搜索延迟较低但计算开销高. 蚂蚁路由[26]算法路径搜索依赖于节点间多次通信获取网络最新状态, 通信开销大时间长, 但可用路径计算与确认过程简单, 故计算开销较低. 与蚂蚁路由类似, Flare[27]路由搜索过程需频繁进行节点通信以获取不同节点路由表信息, 但因引入信标节点, 通信开销适当降低. SilentWhispers[30]可达路径搜索与可用路径计算串行执行且引入数字签名链等机制, 相较 SpeedyMurmurs[24]计算开销较大. SpeedyMurmurs[24]使用节点坐标计算最短路, 每个节点向离接收方更近的方向寻路, 不必经过 Landmark, 允许经过非生成树中的捷径通道, 有效降低搜索延迟. CoinExpress[33]增加 TTL、请求处理时间等设计, 相较改进 Push-Relabel 算法[25], 节点间通信效率提升,

通信开销、计算开销、搜索延迟降低. Flash 算法对于“大象流”与“老鼠流”采取不同策略,“大象流”基于最大流算法搜索效率较低,“老鼠流”基于最短路径路由能与源路由达到相近效率. cRoute[34]算法在路径搜索过程中主要与邻居节点通信, 路由搜索开销低效率高. Spider 算法[7]需通过节点间频繁通信维护通道拥塞队列状态与拥塞控制, 通信与计算开销较大.

5 总结与展望

链下通道网络能够有效提升区块链系统可扩展性, 对于区块链生态发展具有重要意义. 其中, 链下通道路由算法是通道网络高效、平稳运行的关键, 已成为信息领域一个新的研究热点. 本文系统梳理目前主流的 10 种通道路由算法, 从基本目标、扩展目标、性能指标 3 个层次综合对比算法差异, 以期为后续算法研究与技术应用工作提供参考. 基于路由算法发展现状与对比分析结果, 链下通道路由算法未来研究趋势将主要侧重于网络架构、经济激励、算法性能 3 个方面.

(1) 网络架构

现有链下通道网络从节点角色 (普通节点、路由服务节点) 入手, 大多采用点对点架构[7,13]或多中心与层次化架构[24,27], 未依据节点交易数据进行针对性优化. 例如: 利用链下交易周期性聚类的特点[17], 可借鉴区域路由思想, 将具有连通度高、呈现周期性交易行为的节点划分至同一区域, 降低链下通道网络直径, 域内域间采用不同路由策略, 有助于提升搜索效率与交易执行速度.

(2) 经济激励

经济激励机制, 特别是链下交易费计算规则将很大程度影响区块链生态用户参与链下通道网络的积极性. 现有方案主要依据交易转移数量、容量锁定时间、通道拥塞程度等因素实现链下交易费用计算与调整. 未来需针对链下通道网络应用场景与交易特点, 更深入地开展经济激励机制研究工作. 不仅有助于调动全网用户积极性, 还对网络负载均衡、通量水平提升有着重要意义.

(3) 算法性能

算法性能直接影响链下通道网络应用范围, 是衡量通道路由算法优劣的关键. 未来可从路由评价机制、通道平衡策略、容量锁定策略等方面入手开展研究工作. 合理的路由评价机制能够更细粒度控制通道状态, 提升链下交易处理并发度与通道利用率, 雷电网络寻路服务[14]即为该方向初步尝试.

通道平衡是通道能够对外稳定提供路由服务的基础, 基于链下通道网络应用场景, 设计适宜通道平衡策略能够有效缓解通道阻塞、路由选择中心化等问题, 从而达到全网路由服务负载均衡的目的, Spider[7]与 cRoute[34]算法已从包转发与拥塞梯度的角度进行了相应设计优化.

现有容量锁定大都在路径搜索过程中采用“独占”策略, 通道容量易全部锁定. 在交易执行阶段通道利用率不高, 容易引起通道长时间阻塞且交易处理延迟增加. 未来可深入研究容量锁共享机制, 例如: 在路径搜索阶段仅锁定当前路由交易所需最大容量, 并依据 FIFO 原则与路由评价机制执行交易, 有效降低交易处理延迟.

References:

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf

[2] Eyal I, Gencer AE, Sirer EG, Van Renesse R. Bitcoin-NG: A scalable blockchain protocol. In: Proc. of the 13th USENIX Conf. on Networked Systems Design and Implementation. Santa Clara: ACM, 2016. 45–59.

[3] Wang JP, Wang H. Monoxide: Scale out blockchains with asynchronous consensus zones. In: Proc. of the 16th USENIX Symp. on Networked Systems Design and Implementation (NSDI 19). Boston: USENIX Association, 2019. 95–112.

[4] Lombrozo E, Lau J, Wuille P. Segregated witness (consensus layer). Bitcoin Core Develop. Team, Technical Report, BIP, 2015. 141.

[5] Ozisik AP, Andresen G, Levine BN, Tapp D, Bissias G, Katkuri S. Graphene: Efficient interactive set reconciliation applied to blockchain propagation. In: Proc. of 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 303–317. [doi: 10.1145/3341302.3342082]

[6] Zhang XH, Hu HG. Optimization of hash function implementation for bitcoin mining. In: Proc. of the 3rd Int'l Conf. on Mechatronics Engineering and Information Technology (ICMEIT 2019). Atlantis Press, 2019. 448–452. [doi: 10.2991/icmeit-19.2019.73]

[7] Sivaraman V, Venkatakrishnan SB, Ruan K, Negi P, Yang L, Mittal R, Fanti GC, Alizadeh M. High throughput cryptocurrency routing in payment channel networks. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation (NSDI 20). Santa Clara, 2020. 777–796.
[8] Khalil R, Zamyatin A, Felley G, Moreno-Sanchez P, Gervais A. Commit-chains: Secure, scalable off-chain payments. In: IACR Cryptology ePrint Archive. 2018. 642.
[9] Hearn M. Micro-payment channels implementation now in bitcoinj. 2013. https://bitcointalk.org/index.php?topic=244656.0
[10] Coleman J, Horne L, Li XJ. Counterfactual: Generalized state channels. 2018. https://l4.ventures/papers/statechannels.pdf
[11] Wiki B. Hash time locked contracts. 2018. https://en.bitcoin.it/wiki/Hashed_Timelock_Contracts
[12] Osuntokun O. AMP: Atomic multi-path payments over lightning. 2018. https://lists.linuxfoundation.org/pipermail/lightning-dev/2018-February/000993.html
[13] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments. 2016. https://lightning.network/lightning-network-paper.pdf
[14] AG B T. Raiden network 2.0.0 documentation. 2017. https://docs.raiden.network/
[15] Lightning Network statistics. 1ML. 2020. https://1ml.com/
[16] Bitmain. BTC.com. 2020. https://btc.com/
[17] Wang P, Xu H, Jin X, Wang T. Flash: Efficient dynamic routing for offchain networks. In: Proc. of the 15th Int'l Conf. on Emerging Networking Experiments and Technologies (CoNEXT 19). Orlando: ACM, 2019. 370–381. [doi: 10.1145/3359989.3365411]
[18] ACINQ. Lightning network explorer. 2020. https://explorer.acinq.co/
[19] Khalil R, Gervais A. Revive: Rebalancing off-blockchain payment networks. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. Dallas: ACM, 2017. 439–453. [doi: 10.1145/3133956.3134033]
[20] Miller A, Bentov I, Bakshi S, Kumaresan R, McCorry P. Sprites and state channels: Payment networks that go faster than lightning. In: Proc. of the 23rd Int'l Conf. on Financial Cryptography and Data Security. Frigate Bay: Springer, 2019. 508–526. [doi: 10.1007/978-3-030-32101-7_30]
[21] Dziembowski S, Eckey L, Faust S, Malinowski D. PERUN: Virtual payment channels over cryptographic currencies. In: IACR Cryptology ePrint Archive. 2017. 635.
[22] Decker C, Wattenhofer R. A fast and scalable payment network with bitcoin duplex micropayment channels. In: Proc. of the 17th Int'l Symp. on Stabilization, Safety, and Security of Distributed Systems. Edmonton: Springer, 2015. 3–18. [doi: 10.1007/978-3-319-21741-3_1]
[23] Burchert C, Decker C, Wattenhofer R. Scalable funding of Bitcoin micropayment channel networks. Royal Society Open Science, 2018, 5 (8): 180089. [doi: 10.1098/rsos.180089]
[24] Roos S, Moreno-Sanchez P, Kate A, Goldberg I. Settling payments fast and private: efficient decentralized routing for path-based Transaction. In: Proc. of the 25th Annual Network and Distributed System Security Symp. (NDSS 18). San Diego, 2018. 1–15. [doi: 10.14722/NDSS.2018.23252]
[25] Rohrer E, Laß JF, Tschorsch F. Towards a concurrent and distributed route selection for payment channel networks. In: Proc. of the ESORICS 2017 Int'l Workshops on Data Privacy Management, Cryptocurrencies and Blockchain Technology. Norway: Springer, 2017. 411–419. [doi: 10.1007/978-3-319-67816-0_23]
[26] Grunspan C, Pérez-Marco R. Ant routing algorithm for the Lightning Network. arXiv: 1807.00151, 2018.
[27] Prihodko P, Zhigulin S, Sahno M, Ostrovskiy A, Osuntokun O. Flare: An approach to routing in lightning network white paper. 2016. https://pdfs.semanticscholar.org/4392/166a1194010c844ec915694fd5c56da94301.pdf
[28] Lightningnetwork. Onion routed micropayments for the lightning network. 2020. https://github.com/lightningnetwork/lightning-onion
[29] Tsuchiya PF. The landmark hierarchy: A new hierarchy for routing in very large networks. ACM SIGCOMM Computer Communication Review, 1988, 18(4): 35–42. [doi: 10.1145/52325.52329]
[30] Malavolta G, Moreno-Sanchez P, Kate A, Maffei M. SilentWhispers: Enforcing security and privacy in decentralized credit networks. In: Proc. of the 24th Annual Network and Distributed System Security Symp. (NDSS 17). San Diego, 2017.
[31] Roos S, Beck M, Strufe T. Anonymous addresses for efficient and resilient routing in F2F overlays. In: Proc. of the 35th Annual IEEE Int'l Conf. on Computer Communications (INFOCOM 16). San Francisco: IEEE, 2016. 1–9. [doi: 10.1109/INFOCOM.2016.7524553]
[32] Papadimitriou CH, Ratajczak D. On a conjecture related to geometric routing. In: Proc. of the 1st Int'l Workshop on Algorithms Aspects of Wireless Sensor Networks. Turku: Springer, 2004. 9–17. [doi: 10.1007/978-3-540-27820-7_3]
[33] Yu RZ, Xue GL, Kilari VT, Yang DJ, Tang J. CoinExpress: A fast payment routing mechanism in blockchain-based payment channel networks. In: Proc. of the 27th Int'l Conf. on Computer Communication and Networks (ICCCN 18). Hangzhou: IEEE, 2018. 1–9. [doi: 10.

1109/ICCCN.2018.8487351]
[34] ScaleSphere Foundation Ltd. Celer network: Bring internet scale to every blockchain. 2018. https://www.celer.network/assets/doc/CelerNetwork-Whitepaper.pdf
[35] Goldberg AV, Tarjan RE. A new approach to the maximum-flow problem. Journal of the ACM, 1988, 35(4): 921–940. [doi: 10.1145/48014.61051]
[36] Neely MJ, Modiano E, Rohrs CE. Dynamic power allocation and routing for time varying wireless networks. In: Proc. of the 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies. San Francisco: IEEE, 2003. 745–755. [doi: 10.1109/INFCOM.2003.1208724]
[37] Gudgeon L, Moreno-Sanchez P, Roos S, McCorry P, Gervais A. SoK: Off the chain Trans. In: IACR Cryptology ePrint Archive. 2019. 360.

贾林鹏(1995－), 男, 博士生, CCF 学生会员, 主要研究领域为区块链, 链下通道.

于雷 (1981－), 男, 博士, 高级工程师, CCF 专业会员, 主要研究领域为区块链, 大数据.

裴奇 (1995－), 女, 硕士, 主要研究领域为区块链, 链下通道.

张珺 (1975－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为区块链.

王鑫 (1996－), 男, 博士生, CCF 学生会员, 主要研究领域为区块链, 链下通道.

孙毅 (1979－), 男, 博士, 研究员, 博士生导师, CCF 杰出会员, 主要研究领域为区块链.

张瀚文 (1981－), 女, 博士, 副研究员, CCF 高级会员, 主要研究领域为区块链.