arXiv:2603.07750v1 [cs.NI] 8 Mar 2026

# 结构化 Gossip：面向互联网规模动态网络的分区容错 DNS

Priyanka Sinha
p
riyanka.sinha.iitg
@gmail.com Docyt India

Dilys Thomas
dilys@cs.stanford.edu Tata
Consultancy Services Limited 印度

## 摘要

网络分区给移动自组织网络（MANET）和边缘计算中的分布式名称解析带来了根本性挑战。现有解决方案要么依赖无法扩展的主动协调，要么采用开销过大的非结构化 Gossip。我们提出结构化 *Gossip DNS*，利用 DHT 手指表，通过**被动稳定化**实现分区容错。我们的方法将消息复杂度从 $O(n)$ 降至 $O(n/\log n)$，同时保持 $O(\log^2 n)$ 的收敛速度。不同于需要同步一致的主动协议，我们的被动方法通过可交换操作保证最终一致性，无论消息顺序如何都能收敛。该系统通过版本向量处理任意并发分区，无需全局协调，并支持十亿节点部署。

## CCS 概念

• **信息系统** → **分布式数据库系统**； •**网络** → **网络协议设计**；移动与无线网络。

## 关键词

分布式哈希表、Gossip 协议、网络分区、DNS、移动自组网、最终一致性

**ACM 参考文献格式：**

Priyanka Sinha 和 Dilys Thomas。2026。结构化 gossip：面向互联网规模动态网络的抗分区 DNS。见数据管理国际会议论文集（*SIGMOD '26*）。ACM，纽约，NY，美国，4 页。https://doi.org/XXXXXXX.XXXXXXX

## 1 引言

网络分区是分布式系统中的一种严重故障模式，对于 DNS 这类分层名称解析服务而言尤其具有破坏性。在移动自组网（MANET）、边缘计算部署和灾难场景中，由于节点移动性、链路故障和环境干扰，网络分区经常发生 [2]。传统 DNS 架构依赖稳定的根服务器层级结构，因此从根本上不适用于容易发生分区的环境。

特此授予许可：在不收取费用的情况下，为个人或课堂使用而制作本作品的全部或部分数字或纸质副本，前提是不得为营利或商业利益制作或分发副本，且副本须附有本声明以及首页上的完整引文。对于本作品中由非 ACM 所有的组件，必须尊重其版权。允许在注明出处的情况下摘编。若要以其他方式复制、再版、发布到服务器或重新分发到列表，须事先获得具体许可和/或支付费用。请通过 permissions@acm.org 申请许可。*SIGMOD '26*，美国宾夕法尼亚州费城 © 2026 ACM。ACM ISBN 978-1-4503-XXXX-X/26/06
https://doi.org/XXXXXXX.XXXXXXX

以往关于多跳网络自动配置的研究 [2]指出了核心问题：当 MANET 跨组织边界发生分区时，即使节点在物理上仍可达，也会失去对权威 DNS 服务器的访问。简单地将 CHORD 等基于 DHT 的协议 [3] 应用于 MANET，在网络合并期间会遭遇灾难性失败，因为并发的环修复会创建无法恢复的拓扑结构（详见第 2 节）。

近期的方法可分为两类：(1) 主动协调协议，由于同步开销 [4]„ 在节点数量超过数千后扩展性较差；(2) 非结构化流言协议，能够实现最终一致性，但在扇出为 $k$ [5]时，每轮会产生 $O(kn)$ 条消息。这两类方法都无法同时满足互联网规模部署和分区容错性的要求。

我们作出以下贡献：

- 一种使用结构化 gossip 的**被动稳定化**协议，通过利用 DHT 手指表作为 gossip 网络，将消息复杂度从 $O(n)$ 降低至 $O(n/\log n)$
- 通过满足交换律和幂等律的状态合并操作，对**最终一致性**保证进行形式化证明
- 一种基于版本向量的合并协议，可在无需全局协调或主动同步的情况下处理任意数量的并发分区
- 理论分析证明收敛时间为 $O(\log^2 n)$，总消息复杂度为 $O(n \log^2 n)$
- 一个展示大规模分区容错性的交互式演示

## 2 背景与动机

### 2.1 移动自组织网络中的分区问题

考虑一个部署了 DNS 服务器并将其组织成 CHORD 环的移动自组网。当底层网络发生分区时，逻辑上的分布式哈希表环会分裂成彼此断开的片段。Sinha [2] 记录了具体的故障模式：

**单分区场景：** 一个包含 16 个节点的环在节点 5 处分裂，形成分区 $P_1 = \{0, 1, 2, 3, 4, 5\}$ 和 $P_2 = \{6, 7, \ldots\ldots, 15\}$。每个分区都可以通过 CHORD 的被动稳定机制稳定其本地环。然而，当网络合并时，节点可能会尝试并发加入，从而形成环和不可达片段。

**多分区场景：** 如果网络同时分裂成分区 $P_1$、$P_2$ 和 $P_3$，之后 $P_1$ 与 $P_2$ 合并，而 $P_3$ 仍处于隔离状态，现有协议就会失效。主动加入协议会造成协调瓶颈，而被动稳定化则会产生病态情况，如 [2] 图 4.7 和图 4.8 所示：环形成了无效拓扑，并包含不可达片段。

## 2.2 现有方法为何失效

**主动协调：** RANCH 等协议 [4] 采用两阶段提交，每次加入都需要 $O(n)$ 条消息。对于包含 $n$ 个节点的 $k$ 个分区，这会产生 $O(n^2)$ 条消息。关键在于，主动协议要求同步一致——如果任何参与者不可达，协议就会阻塞。动态网络中有效性的代价 [14] 表明，这种阻塞行为是主动协议实现一致性保证所固有的。在频繁发生分区的移动自组织网络中，这会使所有进展停滞。

**被动稳定化（CHORD）：** CHORD 的被动方法 [3] 具有良好的扩展性，但假设网络为单环。发生分区时，各片段会独立稳定。分区合并时，并发稳定化会引发竞争条件，导致病态拓扑 [2]，其中包含环和不可达片段。

**非结构化 Gossip：** 具有扇出的流行病协议 [5] $k$每轮会生成 $kn$ 条消息，收敛总共需要 $O(kn \log n)$ 条消息——对于拥有 $k = 3$ 个节点的十亿级网络，这意味着 900 亿条消息。

# 3 结构化 gossip 协议

## 3.1 核心洞见

DHT 手指表已在标识符空间中提供间隔呈指数增长的链接。在一个含有 $n$ 个节点的 CHORD 环中，节点 $i$ 维护着指向距离为 $2_0$，$2_1$，……，$2^{\lceil \log n \rceil}$ 的节点的指针。这些指针通过在环上建立捷径，实现 $O(\log n)$ 查找。

我们的关键洞见：将手指链接用作流言网络。每个节点不再向随机伙伴传播流言，而是沿着其 DHT 结构进行传播。这能实现指数级的信息扩散，效果类似于 Symphony 的小世界 DHT [11]，但针对分区容错性进行了优化。这种方法利用了前瞻性 [12]：掌握邻居的邻居信息能够加速收敛。

## 3.2 被动稳定化与主动协调

我们的协议采用 **被动稳定化** [1]：节点在不等待确认的情况下做出本地决策。这遵循迪克斯特拉原理：尽管存在分布式控制，系统仍可通过局部操作收敛到正确状态。主动协调需要同步一致、阻塞操作，以及 $O(n)$ 协调开销。被动稳定化则采用异步更新、非阻塞操作，并通过可交换操作实现最终一致性（已在第 4.4 节中证明）。

## 3.3 算法描述

每个节点维护三种类型的状态：

**硬状态**（权威数据）：

- DNS 映射：$(name, IP, TTL, version)$ 元组
- 版本向量：$VV[i]$ 跟踪来自节点 $i$ 的最新更新

**软状态**（可通过流言重建）：

- 后继节点和前驱指针
- 手指表：$\{(2^k, \text{节点})\}$，用于 $k \in [0, \log n]$

**分区状态**：

- 本地分区 ID
- 已知分区集合
- 跨分区链接集合

**算法 1：** 结构化闲聊轮次

**输入：** 节点 $i$，分区 ID 为 $p_i$
**输出：** 已发送八卦消息，状态已更新
// 确定同一分区中可访问的流言目标
$targets \leftarrow \emptyset$;
$samePart \leftarrow \{ j \in V : partition[j] = p_i \wedge active[j] = \text{true} \}$;
// 如果后继节点可访问，则始终向其发送流言
**如果** $successor[i] \in samePart$ **那么**
  $targets \leftarrow targets \cup \{successor[i]\}$;
// 选择用于指数传播的最远指针
$fingers \leftarrow \{f \in fingerTable[i] : f \in samePart\}$;
**如果** $fingers \neq \emptyset$ **那么**
  $furthest \leftarrow \arg\max_{f \in fingers} \text{距离}(i, f)$;
  $targets \leftarrow targets \cup \{furthest\}$;
// 使用版本向量发送八卦消息
**对** $j \in targets$ **执行**
  $msg \leftarrow \langle knownNodes[i], VV[i], p_i, partitionVersion[i] \rangle$;
  发送流言$(i, j, msg)$;
// 检测跨分区链接，以便进行合并检测
$structureLinks \leftarrow fingerTable[i] \cup \{successor[i], predecessor[i]\}$;
**对每个** $k \in structureLinks$ **执行**
**如果** $active[k] =$ 为真 $\wedge\ partition[k] \neq p_i$ **则**
  $crossPartitionLinks[i] \leftarrow crossPartitionLinks[i] \cup \{k\}$;
  $knownPartitions[i] \leftarrow knownPartitions[i] \cup \{partition[k]\}$;

**系统不变量：** 遵循迪杰斯特拉的自稳定原则 [1],，我们的系统维护以下关键不变量：(I1) 版本向量单调性：$VV[i][j]$ 永不递减；(I2) 分区 $ID$ 最小化：连通分量中的节点会收敛到最小分区 ID；(I3) 环连通性：每个分区内的后继链路构成一个环；(I4) 手指表正确性：指针指向可达节点，或被标记为无效。这些不变量在流言传播期间得以保持，并使收敛证明成为可能。

## 3.4 分区合并协议

当底层网络恢复、各分区可以通信时：

**阶段 1 - 检测：** 节点检测其 DHT 链接（后继、前驱、手指表）是否指向不同分区中的节点。这些链接将成为跨分区链接。

**阶段 2 - 合并决策：** 采用确定性规则：最小分区 ID 胜出。正在合并的各分区中的所有节点都采用最小分区 ID。此过程无需协调——每个节点在本地做出决策。

**阶段 3 - 收敛：** 通过流言传播新的分区分配。节点从合并分区接收到流言后，更新其分区 ID。随着节点发现新的可达指针，DHT 结构会自行修复。

**版本向量协调：** 合并时，版本向量按元素逐一合并，采用

**算法 2：** 分区合并检测与执行

**输入：** 已检测到跨分区链接的节点 $i$
**输出：** 更新后的分区分配
// 检查此节点是否位于分区边界
**如果** $|crossPartitionLinks[i]| > 0$ **那么**
  // 收集所有已知分区 ID
  $knownPartitions[i] \leftarrow \{partition[i]\}$;
  **对每个** $j \in crossPartitionLinks[i]$ **执行**
    $knownPartitions[i] \leftarrow knownPartitions[i] \cup \{partition[j]\}$;
  // 确定性合并：选择最小分区 ID
  $targetPartition \leftarrow \min(knownPartitions[i])$;
  // 若与当前分区不同，则更新分区
  **if** $partition[i] \neq targetPartition$ **then**
    $oldPartition \leftarrow partition[i]$;
    $partition[i] \leftarrow targetPartition$;
    $partitionVersion[i] \leftarrow partitionVersion[i] + 1$;
    // 维护不变量 I2：分区 ID 最小性
    日志事件("已将 P$oldPartition$ 合并到 P$targetPartition$");

**算法 3：** 处理收到的 Gossip 消息

**输入：** 节点 $i$ 从节点 $j$ 接收 $msg$
**输出：** 已更新的本地知识和版本向量
$knownNodes_j, VV_j, p_j, pVersion_j \leftarrow msg$;
// 合并同一分区内的知识集
**如果** $p_j = partition[i]$ **则**
  **对每个** $k \in knownNodes_j$ **执行**
    **如果** $k \in V \wedge partition[k] = partition[i]$ **则**
      $knownNodes[i] \leftarrow knownNodes[i] \cup \{k\}$;
  // 逐元素合并版本向量
  **对每个** $node \in (VV_i \cup VV_j)$ **执行**
    $VV[i][node] \leftarrow$ 最大值$(VV[i][node], VV_j[node])$;
    // 保持不变量 I1：单调性
**否则**
  // 跨分区消息：触发合并检测
  $crossPartitionLinks[i] \leftarrow crossPartitionLinks[i] \cup \{j\}$;
  $knownPartitions[i] \leftarrow knownPartitions[i] \cup \{p_j\}$;
$lastUpdate[i] \leftarrow$ 当前时间();

$VV_{merged}[k] = \max(VV_1[k], VV_2[k])$，适用于所有节点 $k$。这样无需协调即可确保因果一致性。

# 4 复杂度分析

## 4.1 消息复杂度

**定理 1**。结构化 gossip 每轮达到 $O(n/\log n)$ 条消息。

证明概要：每个节点向 $\leq 2$ 个目标发送流言。最远的指针指向相距 $\approx n/2$ 的位置。根据鸽巢原理，每个节点都是对 $\approx \log n$ 个其他节点而言最远的指针。每个节点都会从 $\approx n/\log n$ 个发送者处收到流言。□

## 4.2 收敛时间

**定理 2**。结构化 gossip 在 $O(\log^2 n)$ 轮内收敛。

证明概要：环形 gossip 以 $O(n)$ 的速率传播。指针 gossip 呈指数扩散：信息在 $O(\log d)$ 轮内可到达距离 $d$。这与最优 CHORD 路由界限 [13] 相符。环形 gossip 需要 $O(\log n)$ 轮、手指表需要 $O(\log n)$ 轮才能实现完全覆盖，因此总计为 $O(\log^2 n)$。□

## 4.3 分区容错性

**定理 3**。合并协议无需协调即可处理任意并发分区。

证明：合并规则 $target = \min(\cup_i P_i)$具有确定性。对于进行通信的节点 $u \in P_i$，$v \in P_j$，由于 min 满足交换律和结合律，它们都会计算出相同的 $target$。版本向量合并满足幂等律和交换律。跨分区检测是本地的。不变量 I2（分区 ID 最小性）确保所有节点在没有全局协调的情况下收敛到单个分区 ID。□

## 4.4 空间复杂度

每节点状态：手指表需要 $O(\log n)$，DNS 转换需要 $+O(m)$，版本向量需要 $m(n)$。在实践中，可以使用 [9] 中的技术压缩版本向量。

## 4.5 最终一致性保证

现在，我们正式证明我们的被动稳定化协议能够保证最终一致性。

**定义 1（最终一致性）：** 如果在消息传递最终成功且没有新的更新发生的任何执行中，所有节点最终都收敛到相同状态，则该分布式系统具有最终一致性。

**定理 4.** 结构化 gossip 可确保分区成员关系和版本向量的最终一致性。

证明：我们通过证明所有状态合并操作都满足最终一致性的充分条件来证明这一点：交换律、结合性和幂等性。

**(1) 分区 ID 收敛：** 合并操作为：
$merge(p_1, p_2) =$ 分钟$(p_1, p_2)$
即：
- 交换律：$\min(p_1, p_2) = \min(p_2, p_1)$
- 结合律：$\min(\min(p_1, p_2), p_3) = \min(p_1, \min(p_2, p_3))$
- 幂等律：$\min(p, p) = p$

因此，无论消息顺序如何，所有处于相互通信的分区中的节点都会收敛到最小分区 ID。

**(2) 版本向量收敛：** 版本向量的合并操作是逐元素最大值：

$VV_{merge}[k] = \max(VV_1[k], VV_2[k])$ 对所有节点 $k$ 这是：

- 交换律：$\max(v_1, v_2) = \max(v_2, v_1)$
- 结合律：$\max(\max(v_1, v_2), v_3) = \max(v_1, \max(v_2, v_3))$
- 幂等律：$\max(v, v) = v$

**(3) 已知节点集合：** 节点知识的合并操作是集合并：

$KnownNodes_{merge} = KnownNodes_1 \cup KnownNodes_2$

这是：

- 交换律： $A \cup B = B \cup A$
- 结合律： $(A \cup B) \cup C = A \cup (B \cup C)$
- 幂等律： $A \cup A = A$

**(4) 单调进展：** 每个谣言传播轮次都会使已知节点集合单调增加。由于其受 $n$ 限制，收敛会在有限时间内发生——这一性质已在流式算法中形式化 [15]。

**(5) 消息传递：**假设在分区内最终能够送达。根据定理 2，信息会在 $O(\log^2 n)$ 轮内传播。

综合 (1)-(5)：所有操作都是无冲突复制数据类型 [10, 17], , 可保证最终一致性。□

**推论 1：**被动稳定化使每个分区都能独立取得进展；主动协调则会阻塞。

**推论 2：**该协议具备分区容错能力（CAP 定理）：在分区期间仍可用，分区恢复后最终达到一致。

## 5 演示

我们提供了一个交互式网页演示，地址为：https://priyankaiitg.github.io/chordgossip

- 使用 CHORD 分布式哈希表创建最多包含 100 个节点的网络
- 触发任意网络分区（2–5 个分区）
- 可视化沿手指链接传播的闲聊消息流
- 观察各分区内的独立收敛
- 模拟渐进式合并和并发合并 • 比较消息数量与非结构化 Gossip 的差异
- 测试分区内及跨分区的 DNS 查询

可视化中，节点按分区着色，分区内链接以绿色线条表示，检测到的跨分区链接则以红色虚线表示。用户可以观察信息如何通过指针流言呈指数级传播，同时环形流言维持连通性。演示还包含版本向量可视化，展示因果排序，其方式类似于分布式流处理 [15] 和隐私保护型分布式系统 [16]中使用的技术。

## 6 相关工作

**分布式哈希表架构：**Symphony [11] 开创了小世界分布式哈希表路由，实现了 $O(\log n)$ 跳。最优 CHORD 路由 [13] 为结构化覆盖网络证明了紧确界。我们的工作在此基础上扩展，实现了分区容错性。**动态网络：**Bawa 等人 [14] 分析了动态点对点系统中的一致性与可用性权衡，指出主动协议代价高昂。我们的被动方法避免了这一问题。**点对点网络中的前瞻：**Manku 等人 [12] 表明，邻居信息可以加速收敛——我们通过手指表利用这一点。**基于分布式哈希表的 DNS：**CoDoNS [7] 使用 CHORD 实现 DNS，但缺乏分区处理能力。**移动自组网：**Khan 等人 [18] 通过复制解决移动自组织网络中的分布式哈希表故障；我们采用被动收敛。原始论文 [2] 指出了这些问题；我们的无冲突复制数据类型 [17] 解决了这些问题。

## 7 结论

结构化 gossip 表明，通过**被动稳定化**而非主动协调，可以实现互联网规模的分区容错。通过利用 DHT 结构进行流言传播，我们将消息复杂度从 $O(n)$ 降至 $O(n/\log n)$，同时保持对数级收敛时间。我们的形式化证明表明，满足交换律和幂等律的状态合并操作，无需同步一致即可保证最终一致性——这对于分区频繁且持续时间长的移动自组网部署而言，是一项关键属性。

关键见解在于，只要将状态操作精心设计为无冲突复制数据类型（CRDTs），被动稳定化就能兼具正确性与高效性。与在分区期间会阻塞的主动协调协议不同，我们的方法可以在各个分区中独立持续推进，并在网络连通性恢复后确定性地合并。

未来工作包括：（1）面向十亿节点部署的版本向量压缩技术；（2）与现有 DNS 基础设施集成，以实现向后兼容；（3）针对分区场景中拜占庭节点的安全机制；（4）根据分区稳定性指标自适应调整反熵传播速率。

## 参考文献

[1] Edsger W. Dijkstra。分布式控制下的自稳定系统。*ACM* 通讯，17(11):643–644，1974年11月。[2] Priyanka Sinha。多跳移动自组网中的自动配置。硕士论文，奥本大学，2007年。[3] Ion Stoica、Robert Morris、David Karger、M. Frans Kaashoek 和 Hari Balakrishnan。Chord：面向互联网应用的可扩展点对点查找服务。载于*ACM SIGCOMM* 会议论文集，第149–160页，2001年。[4] Xiaozhou Li、Jayadev Misra 和 C. Greg Plaxton。环的并发维护。技术报告 TR-04-36，德克萨斯大学奥斯汀分校，2004年。[5] Alan Demers、Dan Greene、Carl Hauser、Wes Irish、John Larson、Scott Shenker、Howard Sturgis、Dan Swinehart 和 Doug Terry。用于复制数据库维护的流行病算法。载于第六届 *ACM* 分布式计算原理研讨会论文集，第1–12页，1987年。[6] Giuseppe DeCandia、Deniz Hastorun、Madan Jampani、Gunavardhan Kakulapati、Avinash Lakshman、Alex Pilchin、Swaminathan Sivasubramanian、Peter Vosshall 和 Werner Vogels。Dynamo：亚马逊的高可用键值存储。载于*ACM SIGOPS* 操作系统原理研讨会论文集，第205–220页，2007年。[7] Venugopalan Ramasubramanian 和 Emin Gün Sirer。Beehive：针对幂律查询分布的点对点覆盖网络 O(1) 查找性能。载于*NSDI* 论文集，第4卷，第8–8页，2004年。[8] Nicholas J. A. Harvey、Michael B. Jones、Stefan Saroiu、Marvin Theimer 和 Alec Wolman。SkipNet：具有实用局部性特征的可扩展覆盖网络。载于*USENIX* 互联网技术与系统研讨会论文集，2003年。[9] David Ratner、Peter Reiher 和 Gerald J. Popek。Roam：面向移动计算的可扩展复制系统。载于移动计算系统与应用研讨会论文集，第96–101页，1994年。[10] Marc Shapiro、Nuno Preguiça、Carlos Baquero 和 Marek Zawirski。无冲突复制数据类型。载于第*13*届分布式系统稳定性、安全性与安全保障国际研讨会论文集，第386–400页，2011年。[11] Gurmeet Singh Manku、Mayank Bawa 和 Prabhakar Raghavan。Symphony：小世界中的分布式哈希。载于第*4*届 *USENIX* 互联网技术与系统研讨会（*USITS*）论文集，2003年。[12] Gurmeet Singh Manku、Moni Naor 和 Udi Wieder。了解邻居的邻居：前瞻在随机点对点网络中的作用。载于第*36*63，2004 54届 *ACM* 计算理论研讨会（*STOC*），第 – 页。[13] Prasanna Ganesan 和 Gurmeet Singh Manku。Chord 中的最优路由。载于第*15*届 *ACM-SIAM* 离散算法研讨会（*SODA*）论文集，第169–178页，2004年。[14] Mayank Bawa、Aristides Gionis、Hector Garcia-Molina 和 Rajeev Motwani。动态网络中有效性的代价。载于*ACM SIGMOD* 数据管理国际会议论文集，第515–526页，2004年。[15] Gurmeet Singh Manku、Mayank Bawa 和 Prabhakar Raghavan。Symphony：小世界中的分布式哈希。载于第*4*届 *USENIX* 互联网技术与系统研讨会（*USITS*）论文集，2003年。[16] Rakesh Agrawal、Ramakrishnan Srikant 和 Dilys Thomas。保护隐私的联机分析处理。载于 *ACM SIGMOD* 数据管理国际会议论文集，第251–262页，2005年。[17] Paulo Sérgio Almeida。无冲突复制数据类型的实现方法。arXiv:2310.18220，2023年10月。[18] Najeeb Khan、Kashif Naseer Qureshi、Gwanggil Jeon 和 Rajesh Kumar。基于分布式哈希表的移动自组网容错路由。《*Sensors*》，22(11):4280，2022年6月。