# 基于节点异构度的覆盖网络副本一致性维护方法

杨磊，李仁发，胡益明，李肯立
（湖南大学 计算机与通信学院，湖南 长沙 410082）

**摘 要**：分析比较了目前无结构覆盖网络中的副本一致性维护算法，引入了节点能力度量参数——节点异构度，提出了一种基于节点异构度的无结构覆盖网络副本一致性维护方法——NHDCOM。NHDCOM 利用 Chord 组管理协议管理副本节点，采用基于指取表的分割方法动态获取副本节点的异构信息；为优化副本的更新时间，利用动态规划方法提出了一种基于时延的节点度约束生成树算法。给出了 NHDCOM 的维护策略。理论分析和模拟实验结果表明，NHDCOM 能高效地维护副本的强一致性。

**关键词**：无结构覆盖网络；副本一致性维护；Chord；异构度收集树；最小延迟更新内容树
**中图分类号**：TP393.4 **文献标识码**：A **文章编号**：1000-436X(2010)10-0180-10

# Node heterogeneous degree-based consistency maintenance method for unstructured overlay networks

YANG Lei, LI Ren-fa, HU Yi-ming, LI Ken-li
(College of Computer and Communication, Hunan University, Changsha 410082, China)

**Abstract:** Replica strong consistency maintenance for unstructured overlay networks was studied. Current replica consistency maintenance algorithms was summarized, and a novel replica consistency maintenance algorithm: node heterogeneous degree-based of consistency maintenance algorithm (NHDCOM) was proposed. NHDCOM denoted the replica node capability by node heterogeneous degree (NHD). It managed replica nodes using Chord protocol and collected those nodes NHD along with a replica heterogeneous degree collection tree (HDCT) built through the finger table-based ring partition method. Moreover, a replica update problem model was abstracted in NHDCOM, and then a minimum delay update-content tree algorithm based on dynamic programming method was proposed to improve the update speed. Theoretical analysis and simulation results demonstrate that this method is more efficient in maintaining strong replica consistency and updating replica nodes for unstructured overlay network systems.

**Key words:** unstructured overlay networks; replica consistency maintenance; Chord; heterogeneous degree collection tree; minimum delay update-content tree

## 1 引言

副本技术被广泛应用到各种基于覆盖网络的系统中以提高可扩展性、容错性、可用性和减少查询响应时间。P2P 系统就是一种典型的覆盖网络技术。传统 P2P 应用系统中文件主要以只读的形式存在，因此副本技术的研究主要侧重在副本的分发和管理策略上。近年来，P2P 技术被应用到多种新型业务，如分布式存储、游戏、联机拍卖、远程协作等，在这些新型业务中，存储文件由以前的只读形

**收稿日期**：2010-07-02；**修回日期**：2010-09-28
**基金项目**：国家自然科学基金资助项目（90715029）
**Foundation Item:** The National Natural Science Foundation of China(90715029)

式转为可读写形式，数据被多处复制，用户可对系统中的任一副本进行修改操作，副本的高效强一致性维护和如何避免更新冲突成为覆盖网络中副本研究的热点。相比较结构化覆盖网络技术，非结构化覆盖网络技术因其系统拓扑维护简单而且顽健性和容错性能较好而被广泛应用于各种新型应用中。这些新应用的突出特征在于：端特性显著；副本数据具有强一致性；更新数据需要被快速传播。端特性体现在参与新应用加入覆盖网络的节点呈现不同的异构性，保存副本的节点能力参差不齐，在要求副本数据具有强一致性的环境下，端特性将对更新数据的快速传播产生较大影响。

目前，无结构覆盖网络中的副本一致性维护存在的主要需要解决的关键问题是副本节点信息的获取和更新消息的快速传播。集中式方法是针对副本信息获取最直观的解决方法，但不可避免地存在单点故障问题，破坏了覆盖网络的分布式特性；与之相对应的洪泛方法虽然可以避免维护开销，但其在更新时带来的冗余通信开销是不可忽视的。如何以分布的方法动态地组织副本节点信息以避免消息冗余是当前的研究热点。本文利用 Chord 组管理协议进行副本节点的管理[1~3]，在此基础上提出一种基于指取表的环分割方法动态解决副本节点信息的获取问题，这种方法在进行环分割时不会产生额外的环分割冗余通信开销。更新消息的快速传播是避免更新冲突的主要手段，解决这个问题的关键是针对具体应用环境，构建一棵高效的包含所有副本节点的副本生成树，使所有副本节点尽快获得更新消息。这个问题类似应用层的多播问题，有研究将其总结为一个 NP-hard 问题[4]，解决这个问题的关键在于：1）根据应用特性将副本节点组成的覆盖网络映射成图，抽象出问题模型；2）求解映射图的最小延迟生成树。目前关于这个方面的研究较少，本文结合节点的异构度提出一种基于无结构覆盖网络的副本一致性维护问题模型，给出了在这一问题模型下的最小延迟生成树启发式求解算法。

本文第 2 节介绍了无结构覆盖网络中副本一致性维护的相关研究；第 3 节详细介绍了 NHDCOM 算法并对算法性能进行理论分析；第 4 节通过模拟实验评估算法的有效性，第 5 节是结束语。

## 2　相关工作

无结构覆盖网络中的副本一致性维护和更新问题得到了广泛研究，依据思想不同，大致可以分为基于洪泛、基于“推”(push)与“拉”(pull)相结合、基于索引和基于副本链和基于分割树的一致性维护更新算法。

Ripeanu[5]提出了基于泛洪的一致性维护算法。在算法中，更新初始化节点将更新消息通过广播的方式转发到邻居节点，邻居节点如果存在相应文件就更新，然后把更新消息转发到下一轮的邻居节点，依次类推以进行副本一致性的维护。该算法实现简单，但一个副本节点会收到大量的冗余更新信息，占用大量网络带宽，在系统中即使使用 Time-to-Live(TTL)值，副本一致性维护的直径也不可控。Xie[6]在洪泛消息的报文头部增加一个已经收到更新的节点轨迹标签，对已经传输的节点进行记录，收到消息的节点通过检查报文中的节点轨迹标签，完成对冗余消息的提前检测，减少消息在节点间的冗余传输。

Jiang[7]提出了一种主节点广播与基于动态时间间隔“拉”相混合的副本一致性维护算法，任何节点的更改都必须得到主节点确认才能实现最终的一致性维护，这种方式对于覆盖网络节点的更新存在局限性，当主节点的 IP 改变或者离线时，一致性维护就会失效。Datta[8]提出了一种基于谣言(rumor)的“推”(push)与“拉”(pull)相结合的无结构 P2P 系统一致性维护算法。更新消息的传播类似于谣言传播：某个组成员收到更新消息后，以一定的概率把该消息传播给组内其他若干个成员，这就是“推”。当某个节点加入组后，主动地从组内其他成员获取最新的数据，这就是“拉”。这种算法带来的冗余更新消息还是比较多。文献[9]提出一种分层的混合推/拉技术的一致性维护算法，并且引入了有效责任点机制，这种算法能大量减少冗余信息及更新流量，但不能保证系统副本的强一致性。

文献[10]提出一种基于副本索引的无结构 P2P 副本一致性维护策略，这种集中式的策略由种子节点根据副本索引进行副本的一致性维护。该算法弥补了洪泛更新算法的不足，消除了网络中的大量冗余消息，但存在如种子节点选取的开销及失效等问题。

文献[11,12]提出了一种基于副本链的副本一致性维护算法：更新消息通过副本链传递，每次更新消息传递给链中邻近的 $K$ 个节点。这种方法可以有效地减少冗余消息的产生，但构造和维护一个副本链带来了额外开销。同时，由于采用时间或 TTL 机

制，该方法并不能保证副本间的完全一致性。

文献[2,3]提出了一种结构化覆盖网络中基于分割树的一致性维护算法。文献[1]在其基础上提出一种无结构覆盖网络系统中的副本一致性维护算法——PATCOM。PATCOM 动态、按需建立一颗更新消息分割树——UMPT 来传播更新消息。在 UMPT 中每个节点只出现一次，节点过载的几率大大减少。虽然 PATCOM 在副本节点信息的获取和冗余更新消息方面有所改进，但在构建 UMPT 时采用 Chord 的查询机制带来较大的系统开销，尤其是算法并没有考虑副本节点的异构性对更新传播速度的影响。

无结构覆盖网络应用中的副本更新速度的研究较少，与之相关的是应用层多播技术[13~16]。文献[13]提出了一种求解最小延迟生成树的 DCMD-H 算法，不仅考虑延迟路径长度，还考虑相关节点的度是否超出限制，如果超出就选择次长路径，直到找到合适的节点为止。虽然 DCMD-H 考虑了约束节点度的问题，但只是为每个节点静态定义一个最大度，处理过于简单，并没有考虑到节点随更新时间而变化的异构性对节点度和更新时间的影响。

为了有效提高无结构覆盖网络副本的一致性维护收敛速度，降低维护开销，本文提出一种基于节点异构度的副本一致性维护算法——NHDCOM。考虑到更新内容较大时，节点的异构性对更新速度有重要影响，NHDCOM 采用 Chord 协议管理副本节点，利用指取表生成副本分割树收集副本异构度信息可以将系统通信开销 $O(N\times\log d)$（其中 $d$ 为叉数）[1] 降为 $O(N)$。结合副本异构度抽取一个新的副本多播树的问题模型，在此基础上，提出了一个基于动态规划的最小延迟更新树的生成算法，通过这个生成树来传播更新内容，可以保证所有的副本节点以较优的时间收到更新消息。

## 3 NHDCOM

### 3.1 前提假设及相关定义

无结构覆盖网络中的副本一致性维护问题是指在无结构覆盖网络中，一个或者多个副本节点由于动态读写原因，当副本出现改动时，通过节点间消息的传递，确保相关节点保存的文件副本一致性。在讨论一致性维护算法之前，给出以下假设和定义。

**假设 1** 各副本节点的更新传输稳定同步进行。

**假设 2** 副本节点能随意地加入和离开系统，任一副本节点知道组内至少一个副本节点的信息。

**假设 3** 副本节点与系统中的副本保持一致，副本节点只保存一份相同的副本。

**假设 4** 每个节点都有一个唯一的覆盖网 ID 和 IP 地址，节点只需知道某一节点的 IP 地址，就能建立连接进行更新传播。

**假设 5** 指取表中的每一个表项具有的后继节点是不同的，即在指取表中的每一个 interval 具有唯一且不重复的后继节点。

**定义 1** 副本节点，对于某一文件，拥有该文件的节点称为该文件的副本节点。

**定义 2** 副本更新节点，修改副本节点上的副本，并由该节点发起更新操作。

**定义 3** 副本节点异构度(nhd)，节点异构度取决于机器的物理性能(phyPerformance，取决于 CPU 计算速度、总线带宽及内存大小等)及网络物理带宽(bandwidth，含节点的上行带宽和下行带宽)。

$$nhd=\frac{phyPerformance+bandwidth}{phyPerformance\times bandwidth} \tag{1}$$

**定义 4** 异构度收集树，在纯分布式覆盖网络中组织副本节点，当发生副本更新时根据分割方法组织成副本节点树以收集副本异构度。

**定义 5** 副本内容更新树，根据动态获取的所有节点异构度信息所计算的最小延迟生成树。

### 3.2 节点异构度

#### 3.2.1 节点异构度的确定

定义 3 提到副本节点异构度主要取决于节点的物理性能及网络带宽。CPU 性能越好、内存越大，运算能力水平越强，执行指令速度也就越快。NHDCOM 算法中，每个副本节点在收集节点异构度、计算生成树和传送更新数据时需要频繁运行分布式算法和进行数据读写，这些操作都与节点的 CPU 和内存的性能有密切的关系。当副本节点确定了自己的孩子节点并向它们传送更新内容时，传送数据的速度与网络带宽相关。网络物理带宽越大，单位时间内传输的数据越大，传送更新内容的速度也就越快。从式(1)中可知，理论上节点异构度应该取决于两者的综合值，即节点的异构度由物理性能和带宽中的瓶颈因素决定。在实际覆盖网络环境中，考虑系统的端特性，主机的物理性能相差不大，不同的接入带宽成为影响主机异构能力的主要因素，因此，本文将节点的异构度表示如下：

node_nhd{

1. int ID_node;

2. Char IP[8] ;

3. long bandWidth_down;

4. long bandWidth_up;

5. int phycapacity;}

node_nhd 中包含节点的覆盖网 ID 和 IP 地址信息（1～2 行）；同时，根据节点实际情况和算法实现机制，本文将节点带宽分成上行带宽和下行带宽两部分（3～4 行），节点的物理性能（第 5 行）设置为一定值。当节点加入到系统中时，将获取其异构度信息以便系统使用。

### 3.2.2 异构度收集树的建立

异构度可以刻画节点的端特性，而高效地维护和获取所有副本节点异构度信息是一致性维护的前提。这里既要避免集中管理存在的单点失效问题，又要避免纯分布管理带来的低效和消息冗余问题。Chord 协议是一种简单高效的环结构组管理协议，本文提出了一种基于 Chord 协议的异构度收集树（HDCT）构建方法，利用这种方法，通过在无结构覆盖网络中引入较小的通信开销，既可以保证副本节点的分布式管理和维护，又可以在一致性维护阶段使得发起更新的副本节点动态地快速获取所有其他副本节点的信息。NHDCOM 采用 Chord 协议管理所有关键字为 $k$ 的副本节点，所有在线副本节点的加入和离开均遵守 Chord 协议，当更新发生时，由副本更新节点采用环分割的方法将副本节点组织成树。与其他方法[1,2]不一样的是，NHDCOM 利用每个节点所维护的指取表信息进行环分割，理论分析表明算法能显著降低分割开销。

Chord 协议中的每个副本节点维护了一个 $m$ 项指取表。当其中某一副本节点的文件更新时，以更新节点为根节点，根据其 $m$ 项指取表项中的 interval 区间，选取每个表项中的后继节点为该区域的代表节点，对 Chord 环进行逐步的划分，直到每个区域只有一个节点，建立一棵收集副本节点异构度的树。建立 HDCT 过程如算法 1 所示，其中 replica_node.func() 表示函数 func()在节点 replica_node 上运行。

**算法 1**　HDCT 算法

replica_node.scope_partition(Node parent, int begin, int end){

1. replica_node.setParent(parent);

2. for(int i = 1; i <= m; ++i){

3. Node child_node =replica_node. finger[i]. start;

4. int child_begin, child_end;

5. child_begin = replica_node.finger[i]. start. id;

6. child_end = replica_node.finger[i+1]. start. id;

7. if(child_begin∉ [begin,end) )

8. break;

9. child_node = replica_node.finger[i]. successor;

10. temp_begin= replica_node.finger[i]. successor.id;

11. if(temp_begin∈[child_begin,child_ end) )

12. child_begin = temp_begin;

13. else

14. continue;

15. if(child_end∉ [begin,end) )

child_end = end;

16. replica_node.addChildren(child_node);

17. replica_node.sendMessageToChild (child_node, replica_node, child_begin, child_end);}}

HDCT 算法依照副本节点的指取表进行 Chord 环的划分（第 2 行）。先得到指取表中第 $i$ 项的区间，如果起始值越界，则不需对 $i$～$m$ 项进行划分(5～8 行)，因为 $i$+1 项同样也会越界。判断第 $i$ 项的后继是否在区间内，如果不在，则不需进行处理，否则将第 $i$ 项的起始值设为后继节点的 ID(9～14 行)。再判断末尾值是否越界，如果越界则将区间末尾值作为其末尾值（第 15 行）。最后是将父节点及其负责的区间发送给对应表项的后继节点(第 17 行)。如此循环下去，最后每个节点都负责不同的区域，且每个节点在 HDCT 中只出现一次，图 1 给出了在一个由 12 个副本节点组成的 Chord 环上（如图 2 所示）进行分割得到的 HDCT。

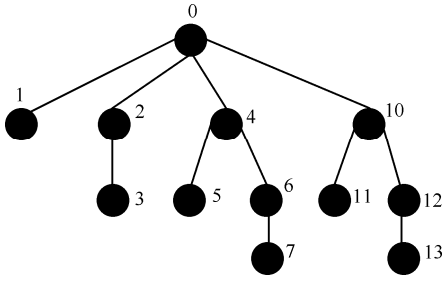


图 1　由 12 个副本节点组成的 Chord 环上进行分割得到的 HDCT

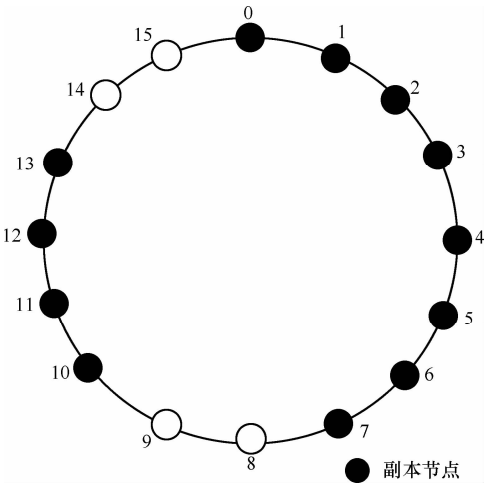


图 2 由 12 个副本节点组成的 Chord 网络，$m$=4

**定理 1** 在一个 ID 空间大小为 $2^m$ 的 Chord 环上建立的 HDCT 高度为 $O(m)$，且每个节点只在 HDCT 中出现一次。

**证明** 假设 Chord 环的 ID 空间大小为 $2^m$，根据假设 5，每个副本节点维护一个小于等于 $m$ 项的指取表。Chord 指取表项中的间隔逐项递增，第 $m$ 项负责的区间最大，是整个 ID 空间的一半。每次对 Chord 环进行划分时，第 $i$ 层 $(0 \leqslant i \leqslant (m-1))$ 节点指取表中的第 $m$-$i$ 项负责的区间最大，为其父节点第 $m-i-1$ 项区间的一半，由于每个副本节点负责的区间不重复，算法最终收敛到指取表的第 1 项，即第 $m$–1 层节点的后继节点，因此，HDCT 的高度为 $O(m)$。在 HDCT 算法中，每个副本节点都只包含在一个唯一的区间内且区间不重复，所以每个节点只出现一次。

### 3.2.3 节点异构度的收集

HDCT 建立副本分割树后，根节点向所有副本节点发送一个短的更新消息以收集所有副本节点的异构信息，每个副本节点收到更新消息后都需要向其父节点返回其异构度。子节点向父节点传送的确认消息是一个在 3.2.1 节中定义的关于节点属性的结构体 node_nhd。分割树中的副本节点收到的所有孩子异构信息 node_nhd 数组后，将其发送给其父节点，依次递归，直到根节点收到确认消息，具体算法过程如算法 2 所示。

**算法 2** 异构信息的收集算法

```
replica_node.HDInfoCollection(){
1. for  i =0 to replica_node children nodes size
2. if(replica_node.receiveFrom(nhd_infos, children_ node[i]).Time > TTLh)
3. RepresentNode=replica_node.select RepresentNode
(children_node[i].begin,children_node[i].end);
4. RepresentNode.region_partition (replica_node, children_node[i].begin,children_node[i].end);
5. RepresentNode.HDInfoCollection();
6. replica_node.setToParent(nhd_infos);
}
```

为了避免节点失效，父节点维护一个计时器，如果收到子节点发送信息超时，父节点则在该区间选择一个新的负责节点，重新收集节点异构度信息。考虑到在 HDCT 不同高度收集到所有子节点的时间不同以及某个节点失效将延迟上层所有的节点收集子节点异构信息，所以层次越小的节点 TTL 也越大。每个副本节点需要收集所有的子节点信息，如果收到任何其中一个子节点的信息超时(第 2 行)，则在其该节点区域内选择一个代表节点(第 3 行)，并重新生成一个 HDCT 收集该区间的节点异构度信息(第 4，5 行)。当副本节点收集到所有子节点的消息，将它们发送给其父节点（第 6 行）。

## 3.3 更新内容树的建立

### 3.3.1 问题模型

副本更新节点收集到所有副本信息后，接下来就是要依据这些信息，将更新内容在最短时间内传播到所有副本节点。根据假设 4，覆盖网中的任意节点之间都可以建立 TCP 连接进行数据传输，因此，算法的目的是要在一个完全图中寻找从副本更新节点到所有副本节点的最短路径并求得这样的一棵最小延迟生成树。由此抽象出一个新的求解最小延迟更新内容树(minimum delay update-content tree)的问题模型：

已知：一个完全图 $G(V,E)$，节点 $v \in V$，具有初始权值 $c[v]=D_H(v)$ 且 $c[v]_t=f(t)$，边 $e \in E$ 且具有权值 $c[v,u]=\min\{c[v]_{up}, c[u]down\}$，节点的初始出度 $\deg[v]=V_N-1$。其中 $D_H(v)$为节点 $v$ 的异构度，$V_N$为总的副本节点数。

求解：最小延迟更新内容树 $T(V,E')$。

问题模型中，考虑在实际环境中副本节点同时传播更新的能力受到异构度的限制，而节点的出度除了受节点的异构度限制外，还受到时间的限制，在不同的时刻节点的传播能力是不一样的，即节点的权值是时间的函数，MDUT 问题与 DCMD[13]问题一样，类似于电话广播 TB 问题[4]，是一个 NP 完

全问题。这类问题不能用精确算法求解，只用寻求其近似求解算法。

### 3.3.2 MDUT-H 算法

本文通过对迪杰斯特拉算法进行改进，提出一种求解最小延迟更新内容树的启发式算法MDUT-H。由于深度越小的节点能力越强，为充分利用已经传播完更新内容的节点能力，使得这些节点在整个更新传播时间内能够不停地为孩子节点传播更新，考虑节点异构度与节点度之间的关系，算法为每个节点引入一个动态变化的时延值 *Tran_ delay*，并为每条以该节点为源的边（出边）引入一个时间戳变量 *timestamp*。每个节点的 *Tran_delay* 初始值为其作为接收节点接收更新数据时所花费的时间，出边的时间戳变量 *timestamp* 与节点的 *Tran_delay* 初始值相等。当节点的出度增加时，只要节点已利用的总下行带宽值小于节点的异构度，节点的 *Tran_delay* 不变，所增加边的 *timestamp* 不变；当节点的出度增加到总下行带宽值大于节点的异构度时，将节点的 *Tran_delay* 加上已有的边权值中的最大值，并将所增加出边的 *timestamp* 设置为节点最新的 *Tran_delay* 值。实际环境中，边时间戳将触发一次相应连接的更新传输操作。

算法的伪代码如算法 3 所示，根节点收到一个 node_nhd 数组异构信息(nhd_infos)后，重新生成一棵最小延迟更新内容树－MDUT。其中 root 为副本更新节点，$delayUV = Tran\_delay_U + Pro\_delay_U + Max$(发送时延 $U$，接收时延 $V$)，$Pro\_delay_U$ 为节点 $U$ 的处理时延，发送时延 $U$ 与节点 $U$ 的上行带宽相关，接收时延 $V$ 与节点 $V$ 的下行带宽相关。

**算法 3**　MDUT-H（MDUT－heuristic）算法

```
root.CreateContentTree(vector<node_nhd> nhd_infos){
1. vector<node_nhd> I,J(nhd_infos);
2.  I.push_back(root);
3.  int index = 0;
4.  for(int j_index=0; j_index =J.size(); j_index++)
5.      select u from I,select v from J,make delayUV is the shortest;
6.      edgeUV.Timestamp=u.Tran_delay;
7.      T.addNode(V);
8.      T.addEdge(edgeUV);
9.      I.push_back(v);
10.     J.Remove(v);
11.     u.children.add(v);
12. v.Tran_dealy = u.Tran_delay+filesize/ (min(u.bandWidth_up,v.bandWidth_down));
13.  u.bandWidth_up -= min(u.bandWidth_up, v.bandWidth_down);
14.  if u.bandWidth_up is less all J elements' bandWidth_down
15.  u.Tran_delay＋= max(u children delay);
16.  u.bandWidth_up= u.bandWidth_ down;
}
```

集合 $I$ 表示纳入树 $T$ 的节点，集合 $J$ 表示未纳入树的节点。算法首先初始化待求的更新消息树 $T = I = \Phi$，$J$=*nhd_infos*（1～3 行）。从 $J$ 中选择一个节点 $v$，从 $I$ 中选择一个节点 $u$ 使得 *delayUV* 最小，修改边的时间戳并把边和节点加到树 $T$ 中，然后修改集合 $I$、$J$（5～10 行）以及 $v$ 的时延和 $u$ 的上行带宽（12，13 行）。如果 $u$ 的上行带宽小于集合 $J$ 的所有下行带宽，则相应的修改其时延和下行带宽（14～16 行），以上过程循环执行直到所有节点都放到树 $T$ 中。

算法 3 实际上是一种动态规划算法，利用这种动态改变时延的方法，既约束了节点在不同时刻的出度能够不超过节点的异构度，同时能够充分利用已传播完更新内容的节点能力，使副本节点更快地得到更新内容。

## 3.4 NHDCOM 的维护

### 3.4.1 副本节点的加入与离开

副本节点的加入遵循 Chord 组管理协议。根据假设 2，副本节点可以通过任一组内副本节点加入，当副本节点加入时，首先初始化其前驱和后继，再更新 Chord 环上其他节点的前驱和路由表项以反映节点的加入[17,18]。完成这些操作后，副本节点与其指取表中的节点通信，获得副本的最新内容。

当副本节点删除某一副本或退出覆盖网络应用系统时，副本节点将离开 Chord 环，只需通知其他节点更新路由表，然后再清空前驱和后继。如果离开的节点是更新内容树上的节点，则由父节点代替它将更新内容传送给其孩子节点，使整个更新操作能继续进行，并且不影响其他节点的更新。

### 3.4.2 副本节点的失效

根据副本节点失效的时间点，可以分为以下几种情况。

1) 在建立异构度收集树前失效，这时节点失效不会影响更新操作，Chord 协议将处理这种节点失效。

2) 在异构度收集树建立好并等待孩子节点返回异构信息时失效，这时将会带来冗余消息。因为其中的一个节点失效将会使其父节点收不到异构信息，导致更新失败，算法需要重新生成相应区间的异构度收集树。由于异构度信息收集过程中所传送的内容较小，时间很快，使得在这种情况下失效的概率非常小。

3) 建立更新内容树但没有收到更新内容时失效，这时只会增加更新的时间，但不会产生冗余消息。因为更新树建立后，父节点拥有整个副本节点的全局视图，如果在向子节点传送更新内容时，子节点失效，父节点可以通过调整使失效子节点的孩子代替，再重新构造这部分更新内容树，而不影响其余树枝的更新，减少了失效时冗余消息的产生。

### 3.5 算法性能分析

Stoica 等人[17]详细分析了 Chord 组管理协议对系统带来的性能影响，本文主要是通过更新通信（消息数）开销、算法改进所带来的额外内容开销、算法本身的复杂度来衡量 NHDCOM 算法的性能。

1) 更新通信（消息数）开销

**定理 2** 在一个由 $N$ 个副本节点组成的 Chord 环上建立 HDCT 和收集节点异构度，总共需要产生的通信开销（消息数）为 $2N-2$。

**证明** 由于 HDCT 是根据指取表对 Chord 环进行划分，不会产生节点查找消息。每个节点找到其对应区间的孩子节点时，只需向其发送一条包括负责区间、父节点等内容的消息，并且叶子节点不要发送消息，故建立 HDCT 需要产生的消息数为 $N-1$。收集节点异构度信息的过程与上述过程相反，每个节点（除根节点外）都需向其父节点发送节点异构度信息，也需要 $N-1$ 条消息，因此，整个异构度收集过程所需的消息数为 $2N-2$。

PATCOM 算法在构建 UMTP 时总共所产生的查询消息为 $O(N\times\log d)$[1]（$d$ 最优值等于 8），从定理 2 可以看出，虽然 NDHCOM 算法中收集副本节点信息由环分割和收集异构度两部分过程组成，但由于分割算法采用指取表进行，故总的查询消息数开销还要优于 PATCOM。

2) 传播内容的开销

NHDCOM 算法在收集节点异构度和传播 MDUT 信息时都需要产生额外的内容开销。假设每个节点的 node_nhd 和其指针共占用 40 字节。

**定理 3** 在一个由 $N$ 个副本节点组成的 ID 空间大小为 $2^m$ 的 Chord 环上，关键字 $k$ 的副本节点在 $d(2\leqslant d\leqslant m)$ 叉 HDCT 上收集节点异构度时每个节点传播的冗余消息内容大小最大不超过 $(20\times N)$ byte。

**证明** 在收集节点异构度信息过程中，每个副本节点在 HDCT 上传送其子节点及自身的异构度信息给父节点，因此，第 1 层节点向根节点传输的内容是最大的。由于 DHT 随机地把节点分布在整个 ID 空间上，根据定理 1，第 1 层副本节点中最大的子树最多覆盖了 1/2 的节点，其向根节点传输的内容不会超过 $(20\times N)$ 字节，而以下每层节点传输的内容呈指数递减，定理得证。

NHDCOM 的传播开销还包括每次传播更新内容时要附加传送相应的 MDUT 信息。

**定理 4** 关键字 $k$ 的 $N$ 个副本节点在传播更新内容时每个节点平均附加传送的 MDUT 最大为 $(40\times N)$ byte。

**证明** 与定理 3 相似，传输给第 1 层节点 MDUT 内容最大，等于 $(40\times N)$ byte。以下各层节点接收的 MDUT 内容逐层递减。因此，每个节点需传播的内容平均开销最大等于 $(40\times N)$ byte。

通过定理 3、定理 4 可知，相对更新内容，附加的消息相对较小，这种开销很小，可以接受。

3) MDUT-H 算法时间复杂度

在建立更新内容树的时候，由于构造一棵传播延时最小的树是一个 NP 难问题，所以只能用启发式的算法得到一个较优的结果。MDUT-H（MDUT－heuristic）算法中每个节点加入时动态选定一个能使本节点接收时延最小的节点作为父节点，同时动态更改父节点的接收时延，直到所有节点都加入生成树。从算法 3 可知，算法的时间复杂度是 $O(N^2)$，与最小延迟生成树 DCMD-H 的时间复杂度在同一个级别,考虑到现有覆盖网络的副本节点规模多在 10 000 个节点以内，更新源节点能在较快的时间内构造更新内容树，即便用于更大规模的网络环境中，也可以通过将节点分簇的方法来管理，本文不考虑这种情况。如果更新内容传播的过程中出现节点失效的情况，那么父节点也可在较短时间

内对其负载的树的分枝按照该算法重新构建更新内容树。

## 4　实验与讨论

本文按照 Optorsim[19]中副本更新仿真的基本思想模拟构建了一个中小规模的无结构覆盖网络一致性维护评测环境，分别实现了 NHDCOM 算法、PATCOM[1]算法和 DCMD[13]算法，并对他们进行了性能的比较。考虑实际环境中，覆盖网络中端节点的能力值一般处在若干不同级别（如节点多为笔记本、PC、服务器等类型，连接带宽主要区分为家庭接入、局域网接入、无线接入等方式）上，因此，设计的节点能力值按相应级别呈正态分布：设置不同级别的节点异构度在[0, 100]之间正态分布，通过改变正态分布参数$\sigma$的值来体现节点异构度的区别，$\sigma$越大，分布越分散，节点之间的异构性也就越大；反之越小。在评测环境中，根据文献[1]描述，将 PATCOM 算法中更新消息传播树的叉数设置为最优叉数 $d$=8，父节点在此基础上随机选择孩子节点。在构建更新内容树时为更有效地体现 DCMD 算法的优势，相较于原算法随机的为节点生成最大度约束方法，将每个节点的最大度设置为正比于该节点的能力值。为提高测试数据的稳定性，图中的结果均是 100 次实验所得结果的平均值。

1) 不同的节点异构度分布对一致性维护时间的影响

实验选取副本节点数为 $N$=500,需要更新的副本文件大小分别为 5MB 和 50MB，图 3 和图 4 给出了不同异构度情况下 3 种算法的更新速度。节点异构度的差异由$\sigma$来决定：$\sigma$=10，节点相对比较集中；$\sigma$=100，则节点之间的异构比较明显。

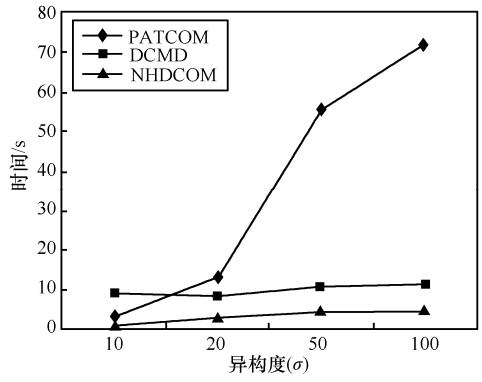


图 3　异构度与更新时间的关系（$N$=500, $Size$=5MB）

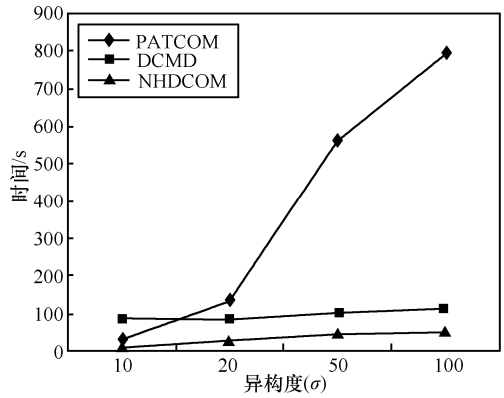


图 4　异构度与更新时间的关系（$N$=500, $Size$=50MB）

从图中可以看出，当异构度较小时，如$\sigma$=10 时，3 种算法的更新时间相差不太大，且 PATCOM 比 DCMD 的时间还短。这是由于节点的能力基本相同，DCMD 与 NHDCOM 考虑节点边权的特点没有其优势。随着异构度的增加，相较 DCMD 和 NHDCOM，PATCOM 的更新时间显著的增加，这主要是因为 PATCOM 仅仅通过对 Chord 环的分割来组成更新消息树，没有考虑到节点异构性对更新时间的影响，当异构性增加时，能力大的节点并没有发挥优势，相反能力小的节点成为了更新时间的瓶颈。从图中还可以看出，异构度的增加对采用 DCMD 与 PATCOM 2 种算法的更新时间影响较小，主要是因为这 2 种算法在构造更新树时充分考虑了副本节点的能力以及 2 个节点之间的传输时延，所以受到节点异构性的影响相对较小。DCMD 在考虑节点度的时候比较简单，仅为每个节点随机生成一个最大度，并没有考虑节点能力差异对出度的影响。NHDCOM 依照节点的异构度来决定节点的出度，同时根据引入的节点时延动态的确定边的时间戳，在整个更新周期中能充分利用每个节点的能力，从而使更新时间最优。

2) 更新文件大小对一致性维护时间的影响

更新文件大小对副本一致性维护时间显然有较大的影响。实验分别测定当节点规模为 100 和 500、$\sigma$为 20、取更新文件大小为 1MB、5MB、10MB、50MB、100MB 时 3 种算法的更新时间，结果如图 5 和图 6 所示。

图 5 和图 6 可以看出，在节点规模及$\sigma$相同的情况下，随着文件大小的增加，相较 NHDCOM 算法，PATCOM 和 DCMD 算法的更新时间速率增长较快。这主是要因为在构建更新树时，尤其在传输大文件时，能力强的节点尽量放在树深度较小的地方可以节约大量更新时间；同时，上层节点在整个

更新周期内能够持续参与传送更新数据对减少整个更新周期无疑具有更加重要的影响。PATCOM 算法在构建更新树时，既不考虑节点的异构度影响，又没有考虑节点传送数据的动态规划，其性能较差。DCMD 虽然考虑了节点的异构性但只对节点度进行了静态约束，缺乏动态规划，其性能虽然比 PATCOM 好，但比 NHDCOM 算法有较大的差距。

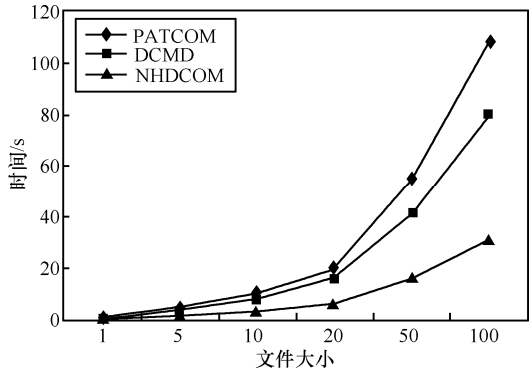


图 5 文件大小与更新时间的关系（$N$=100, $\sigma$=20）

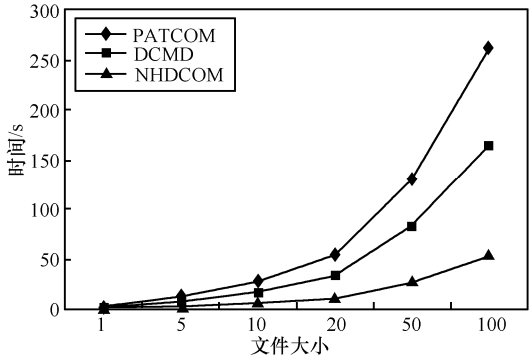


图 6 文件大小与更新时间的关系（$N$=500, $\sigma$=20）

3) 副本节点规模对一致性维护时间的影响

本文最后模拟测试了不同副本节点规模对算法性能的影响。将文件大小固定为 5MB，设定节点规模从 50 到 2000，在节点异构度分布系数分别为 10 和 100 时的测得 3 种算法的更新时间如图 7 和图 8 所示。

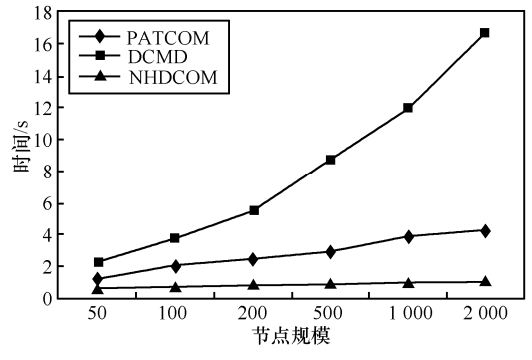


图 7 节点规模与更新时间的关系(*size*=5MB, $\sigma$=10)

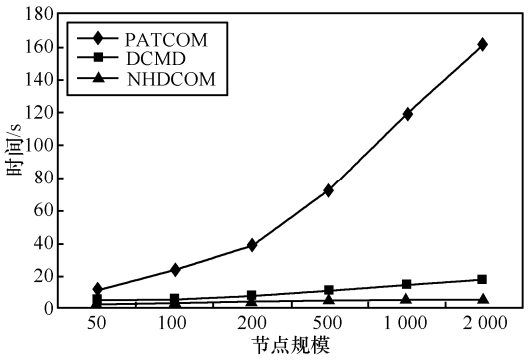


图 8 节点规模与更新时间关系(*size*=5MB, $\sigma$=100）

从图 8 可以看出，当节点规模增大时，节点的度组织对更新时间具有很大的影响。PATCOM 算法不考虑节点的异构度对节点组织的影响，且始终保持固定的出度，当节点的异构度分布较大时，对其性能具有较大的影响。DCMD 算法考虑了节点的异构度，但为节点随机规定了一个最大度约束，当节点异构度不大时，DCMD 算法的性能甚至比 PATCOM 算法还要差，这从图 7 可以看出。NHDCOM 算法利用动态规划的方法根据节点的异构度动态组织节点的出度，在不同异构度分布的情况下具有稳定的性能。

## 5 结束语

本文主要研究了无结构覆盖网络中的一致性维护问题，提出了一种基于节点异构度的无结构覆盖网络副本一致性维护方法——NHDCOM。为刻画无结构覆盖网络中节点的端特性，提出了一种节点异构度的表示方法 node_nhd。本文利用 Chord 协议来组织副本节点，并在更新时提出一种根据节点指取表进行分割的算法，该算法能够以较小的开销帮助更新源节点获得所有其他副本节点的异构度信息。结合节点异构度，本文提出一种求解最小延迟更新内容树(minimum delay update-content tree)的问题模型，利用动态规划的方法提出了一种启发式算法——MDUT-H，理论分析和模拟实验表明，相较类似算法，在不同的节点异构度分布、副本文件大小以及节点规模等环境下，NHDCOM 算法具有出色的效率和稳定性。

## 参考文献：

[1] 李振宇, 谢高岗, 李忠诚. PATCOM：基于分割树的无结构 P2P 系统一致性维护方法[J]. 计算机学报, 2007,30(9):1500-1510.

LI Z Y, XIE G G, LI Z C. PATCOM: partition tree-based consistency

maintenance for unstructured P2P systems[J]. Chinese Journal of Computers, 2007, 30(9):1500-1510.

[2] CHEN X, REN S S, WANG H N. SCOPE: scalable consistency maintenance in structured P2P systems[A]. Proc of IEEE Infocom 2005[C]. Washington, 2005. 1502-1513.

[3] WANG Y. Maintaining replica consistency using replica information broadcast tree in P2P storage system[A]. International Conference on Research Challenges in Computer Science[C]. Shanghai, China, 2009.

[4] ELKIN M, KORTSARZ G. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem[A]. Proc of the Annual ACM Symp on Theory of Computing[C]. Montreal, 2003. 438−447.

[5] MATEI R. Peer-to-Peer Architecture Case Study: Gnutella Network[R]. University of Chicago, 2001.

[6] 谢鲲，张大方，谢高岗等. 基于轨迹标签的无结构 P2P 副本一致性维护算法[J]. 软件学报, 2007, 18(1):105-111.
XIE K, ZHANG D F, XIE G G, *et al*. A trace label based consistency maintenance algorithm in unstructured P2P systems[J]. Journal of Software, 2007,18(1):105-111.

[7] LAN J, LIU X T, SHENOY P. Consistency maintenance in peer-to-peer file sharing networks[A]. Proc of the 3rd IEEE Workshop on Interact Applications[C]. Washington, 2002. 90-94.

[8] DATTA A, HAUSWIRTH M, ABERER K. Updates in highly unreliable, replicated peer-to-peer systems[A]. Proc of the 23rd International Conference on Distributed Computing Systems[C]. Washington, 2003.76-85.

[9] 郭晓梅，李仁发，文吉刚等. 基于 P2P 网络环境下的副本一致性维护算法[J]. 计算机科学, 2009,36(1):43-45.
GUO X M, LI R F, WEN J G, *et al*. Replication consistency maintenance algorithm in unstructured P2P systems[J]. Computer Science, 2009, 36(1):43-45.

[10] 蒋试伟, 欧阳松. 基于副本索引的 P2P 副本一致性维护策略[J]. 计算机工程, 2008, 34(19): 123-126.
JIANG S W, OUYANG S. Replication index based consistency maintenance strategy in unstructured P2P systems[J]. Computer Engineering, 2008, 34(19):123-126.

[11] WANG Z J, DAS S K, KUMAR M. Update propagation through replica chain in decentralized and unstructured P2P systems[A]. Proc of the 4th International Conf on Peer-to-Peer Computing[C]. Washington, 2004.64-71.

[12] 苏长根, 欧阳松. P2P 系统中基于副本链的一致性维护算法[J]. 计算机工程,2008, 34(18):145-150.
SU C G, OUYANG S. Replica chain based consistency maintenance algorithm in P2P systems[J]. Computer Engineering, 2008, 34(18): 145-150.

[13] 曹佳，鲁士文. 应用层多播的最小延迟生成树算法[J]. 软件学报, 2005,16(10):1766-1773.
CAO J, LU S W. A minimun delay spanning tree algorithm for the application-layer multicast[J]. Journal of Software, 2005,16(10): 1766-1773.

[14] SALAMA H F, REEVES D S, VINIOTIS Y. The delay-constrained minimum spanning tree problem[A]. Proc of the 2nd IEEE Symp on Computers and Communications[C]. Alexandria, 1997. 699-703.

[15] TAN S W, WATERS G, CRAWFORD J. A Survey and Performance Evaluation of Scalable Tree-Based Application Layer Multicast Protocol[R]. Canterbury: University of Kent, 2003.

[16] BROASH E, SHAVITT Y. Approximation and heuristic algorithms for minimum delay application-layer multicast trees[A]. Proc of 2004 IEEE Conference on Computer and Communications[C]. 2004. 2697-2707.

[17] STOICA, ROBERT M, DAVID K. Chord: a scalable peer-to-peer lookup service for internet applications[A]. Proc of the ACM SIGCOMM'01 Conference[C]. San Diego, California, 2001.

[18] 陈贵海，李振华. 对等网络：结构、应用与设计[M]. 北京: 清华大学出版社, 2007.83-94.
CHEN G H, LI Z H. Peer-to-Peer Network: Structure, Application and Design[M]. Beijing: Tsinghua University Press, 2007.83-94.

[19] BELL W H, CAMERON D G, CAPOZZA L, *et al*. Optorsim: A grid simulator for studying dynamic data replication strategies[J]. International Journal of High Performance Computing Applications, 2003, 17(4):403-416.

**作者简介：**

**杨磊**（1976-），男，湖南临湘人，湖南大学博士生、副教授，主要研究方向为分布式系统与网络。

**李仁发**（1957-），男，湖南郴州人，湖南大学教授、博士生导师，主要研究方向为嵌入式系统与网络。

**胡益明**（1986-），男，湖南株洲人，湖南大学研究生，主要研究方向为对等网络。

**李肯立**（1971-），男，湖南娄底人，湖南大学教授、博士生导师，主要研究方向为并行处理、科学可视化。