应用层组播的最小延迟生成树算法*

曹 佳[1,2+], 鲁士文[1]

[1](中国科学院 计算技术研究所,北京 100080)

[2](中国科学院 研究生院,北京 100049)

A Minimum Delay Spanning Tree Algorithm for the Application-Layer Multicast

CAO Jia[1,2+], LU Shi-Wen[1]

[1](Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-62565533 ext 8837, E-mail: jiacao@ict.ac.cn, http://www.ict.ac.cn

Received 2004-07-16; Accepted 2005-03-11

Cao J, Lu SW. A minimum delay spanning tree algorithm for the application-layer multicast. *Journal of Software*, 2005,16(10):1766−1773. DOI: 10.1360/jos161766

Abstract: Real time transmission, which is delay sensitive, is an important aspect of application-layer multicast. It is crucial to build an efficient multicast tree to guarantee the lower delay. This research is focused on the algorithms of the minimum-delay spanning tree for the application-layer multicast. Firstly, it is stated that the total delay is affected by communication delay, processing delay and the degree of nodes. Then the network is modeled into the node-and-edge-weighted directed graph with the limited degree of nodes. In this model the problem is shown to be NP-hard. Therefore, two kinds of heuristic algorithms are proposed, which are based on the maximum degree and the maximal length path respectively. Finally, the simulation demonstrates that the proposed algorithms are valid.

Key words: application-layer multicast; minimum delay spanning tree; NP-hard; real time transmission

摘 要: 实时传输是应用层组播技术的一个主要应用领域,对网络延迟有严格的限制.保证低延迟组播成功的关键在于构建高效的应用层组播树,研究构建最小延迟应用层组播树的算法.首先分析影响延迟的 3 个因素:链路的传输时间、结点的发送/转发时间和结点度,然后把求解应用层组播树的问题抽象成对边和点都带权的有向图求解“度约束最小延迟生成树”的问题,同时证明这个问题属于 NP-hard,并且提出了两类启发式近似算法:基于度的算法和基于最大延迟路径的算法.最后通过模拟实验说明了所提出算法的有效性.

关键词: 应用层组播;最小延迟生成树;NP-hard;实时传输

中图法分类号: TP393 文献标识码: A

组播是一种一对多的通信方式,常用于视频会议、内容发布、远程教学等网络应用.IP 组播是一种较早的组播实现机制,虽然具有较高的传输效率,但是对底层的网络设备有支持组播协议的要求,所以在现阶段不能在

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA742052 (国家高技术研究发展计划(863))

作者简介: 曹佳(1978−),女,新疆库尔勒人,博士生,主要研究领域为组播路由;鲁士文(1944−),男,研究员,博士生导师,主要研究领域为计算机网络协议.

广大范围内普及.为此,人们提出了应用层组播,希望它既能继承组播的传输特性,例如节约带宽、传输快捷,也可以脱离对底层基础网络提升的依赖,在近期就可以实现较大范围的组播通信.因此,应用层组播成为倍受瞩目的组播实现机制.

人们已经研究了多种应用层组播协议,如 Narada,NICE,Yoid,ALMI 和 Scribe 等[4].无论何种协议,“保证组播参与者尽快收到组播包”一直是应用层组播的主要追求目标,它对实时传输有重要的意义.解决这个问题的关键在于构建高效的应用层组播树,主要涉及两个方面:① 如何将应用层组播的覆盖网络映射成图,即如何抽象问题模型;② 如何求解映射图的最小延迟生成树.

目前求解应用层组播树算法的主要差异是人们根据不同的应用需求提出了不同的问题模型[1−6],见表 1,然后设计出不同的求解算法.这些问题大多属于 NP-complete,不存在多项式时间解.文献[2−6]将覆盖网络抽象成一个边带权、结点度受限的图,未考虑结点的权值,也就是说,忽略了中间结点的处理时间.然而,现实中应用层组播的中间结点大多是普通主机,它们不具备线速转发的能力,因此在抽象问题模型时不能忽略转发处理时间.MDM 模型[1]相对要完善一些,它将覆盖网络抽象成一个结点和边都带权的图,但是没有约束结点的度,这是因为文献[1]的作者认为通过他们设计的算法(H-MDM 算法)求得的生成树本身就具有较小的结点度,同时也具有较低的树延迟.但是,在本文的第 2.2 节中,我们将通过一个反例证明这个假设是不合理的.

Table 1 Delay-Constrained spanning tree problems in application layer multicast

表 1 与延迟相关的带约束生成树问题

Target	Problem
Average delay minimum	Delay-Constrained minimum spanning tree problem (DCM)[4] Minimum average-latency degree-bounded directed spanning tree problem[3]
Minimum delay	Minimum-Diameter degree-limited spanning tree problem (MDDL)[2] Minimum delay multicast problem (MDM)[1] Delay-Constrained shortest path tree problem[6]
Constrained delay	Delay-Constrained minimum spanning tree problem (DCMST)[5]

针对这些问题,我们抽象出一个新的更全面的求解应用层组播树的问题模型——DCMD(degree-constrained,minimum delay spanning tree for node-and-edge-weighted directed graph).DCMD 模型同时考虑了边权、点权和结点度约束 3 方面的问题.另外,本文针对 DCMD 模型提出了一个最小延迟应用层组播树的生成算法,通过这个生成树来发布信息,可以保证所有的组播参与者尽快地收到组播包.

由于 DCMD 是一个新的问题模型,到目前为止,关于同时涉及度约束和结点边均带权 3 个因素的最小延迟生成树的研究很少,尤其是针对结点的权值随着结点度值的变化而变化的情况的相关研究更是少见.因此,我们分别调研了两个应用层组播树的问题模型 MDM 问题(结点和边均带权值的最小延迟生成树问题)和 MDDL 问题(度约束边带权最小延迟生成树问题)的求解算法.对于 MDM 问题,文献[1]提出了一种基于最大延迟路径的贪婪算法的 H-MDM 算法.该算法首先考虑接收延迟最大的结点,然后再逐步处理较“近”的结点.其中所有结点对之间的最短路径的距离采用 Floyd-Warshall 来计算.我们将在第 2.2 节对 H-MDM 算法作详细分析.对于 MDDL 问题,文献[2]提出了 CT 算法,类似于 Prim 方法,先处理接收延迟最小的结点,对相关结点的度没有限定值;文献[10]又分别提出分割式构造算法和极网格构造算法.然而,这些算法解决的 MDM 问题和 MDDL 问题都没有同时考虑度约束、结点和边均带权 3 个因素,而这 3 个因素都直接影响着应用层组播树的树延迟,所以都不可忽略.关于最小延迟生成树,本文所做的主要工作是对这些算法进行修改,从而解决 DCMD 问题.

本文第 1 节定义 DCMD 问题,证明 DCMD 属于 NP-hard.第 2 节给出两类启发式近似算法:基于度的算法和基于最大延迟路径的算法.第 3 节通过模拟实验评估算法的有效性.

1 问题模型

应用组播通过覆盖机制在应用层上实现组播功能,参与者首先在应用层上构建逻辑组播树,然后通过底层的单播 IP 网络进行点到点的实际互连.在应用层组播传输数据包的过程中,组播源准备数据,查询组播路由表,获得孩子的单播 IP 地址,再分别打包、转发.数据包通过相应的单播路径传输给孩子,孩子接收数据包后查询路由表,再分别打包、转发.这一过程重复进行,直到所有结点都收到数据包为止.显然,中间结点需要完成数据的复

制转发工作.如前所述,由于这些结点通常是普通的主机,不能进行线速转发, 所以复制转发工作消耗的时间较大.鉴于此,计算应用层组播延迟(简称“树延迟”,记做 D)不能忽略结点的转发处理时间(下文将中间结点的转发处理时间和源结点的发送时间统称为“发送延迟”,记做 Se).考虑到这个因素,我们需要把树延迟表示成:

$$D=\max\{\sum_{i\in P}(Se_i+Tr_i)\} \tag{1}$$

其中,P 表示从源到叶结点的路径,传输延迟 Tr_i 表示从源结点到结点 i 之间路径的传输延迟.除了发送延迟和传输延迟以外,树延迟还受拓扑结构的影响.拓扑结构与结点度有直接关系,结点度越大,树的深度越小,树延迟也越小.在底层是采用单播传输机制的条件下,某个结点的度大,意味着这个结点需要多次转发同一个数据包,对应的发送延迟越长,导致树延迟增大.由此可见,通过调整结点度来均衡传输延迟和发送延迟这两个参量,可以减小树延迟.因此,影响树延迟有 3 个因素:传输延迟、发送延迟和结点度.

需要说明的是,之所以要适当约束结点的度,除了考虑到可以降低树延迟以外,还有两个原因.一是如果某个主机结点的下行用户数目太多,那么它的工作负载会相当重,所以约束结点的出度可以降低这个结点的工作量;二是从网络传输环境来看,主机的上行带宽受限也要求约束结点的度.

应用层组播的覆盖网络可以被映射成图论中的图,可以将求解最小延迟应用层组播树问题定义如下:

定义 **1**(**DCMD** 问题).

已知:一个图 $G(V,E)$.结点 $v\in V$,具有权值 $c[v]\in R^+$,出度 $\deg[v]$ 与 $c[v]$ 满足 $c[v]=f(\deg[v])$,$f(x)$表示权值函数;结点 v 具有最大度约束 $d^{\max}[v]$,v 在 G 中的出度 $\deg_G[v]$ 满足 $\deg_G[v]\le d^{\max}[v]$;边 $e\in E$ 带权 $p[e]\in R^+$.

求解:最小延迟生成树 $T(V,E')$,并且在树 T 中结点的度 $deg_T[v]$满足 $\deg_T[v]\le d^{\max}[v]$.

DCMD 问题可归约到具有 NP-complete 难度的电话广播 TB 问题[8].TB 问题描述这样一类问题:给定一个图 $G(V,E)$,一个处理机 $s\in V$ 和一个处理机集合 $T\subseteq V$;规定每个时间段允许每个处理机结点选择一个邻居结点,并发送一个消息.电话组播 TM 问题要求计算出一个最快的转发方案,使得从 s 向 T 的结点发送消息的时间最短.当 $T=V$ 时,这个问题就成为电话广播 TB 问题.

定理 **1**. DCMD 问题属于 NP-hard.

证明:已知 $TB\propto NP-complete$,为了证明 $DCMD\propto NP-hard$,只需证明 $TB\propto DCMD$.对于任意实例 $I\in TB$ 通过下面这个多项式时间变换可以得到实例 $I'\in DCMD$:(1) $c[v]=f(\deg[v])=1$;(2) $p[v]=0$.从而得到实例 I',然后求解 I'得到最小延迟生成树 T'.显然,这个 T'就是 TB 的最小延迟生成树 T,因此解决 DCMD 的算法可以也可以解决 TB 问题.故得证, $DCMD\in NP-hard$. □

2 算法描述

为了下面叙述方便,我们先描述几个术语.设 $G(V,E)$是连通图,待求树 $T(V,E')$.初始状态时,$T=\varnothing$,即 $V_T=\varnothing$,$\overline{T}$ 表示 T 相对于图 G 的补集,$V_{\overline{T}}=V$;生成树的过程是将 $\overline{T}$ 中的每个结点以及相关的一条边逐一纳入到 T 的过程.这样 T 与 $\overline{T}$ 之间存在一些边,边的一个端结点属于 T,另一个端结点属于 $\overline{T}$,我们称这些边组成的集合为“反圈”.设 $e(i,j)$ 表示反圈中的边,两个端结点分别是 $i\in T$, $j\in\overline{T}$.我们称结点 i 的集合为“反圈在 T 中的点集合”,记作 I;结点 j 的集合为“反圈在 $\overline{T}$ 中的点集合”,记作 J.生成树的过程就是从 J 中选择一个结点 u,从反圈中选择一条相关边 $e(i,u)$,直到所有的结点都纳入到 T 的过程.

求解 NP-hard 类问题通常有两种算法:确切算法和近似算法.确切算法基于枚举,时间复杂度呈指数特性,不适合于具体的应用.近似算法为了获得多项式时间的求解速度,从一定程度上降低了解的准确度,但是因为比较快,得到的结果不是很差,所以较为实用.我们为 DCMD 问题设计了两种启发式近似算法:基于度的算法和基于最大延迟路径的算法.

2.1 基于度的算法

2.1.1 基于最大度的算法

该算法的核心思想是尽量选择度最大的结点.因为只考虑结点的度,所以求得的生成树比较粗壮.

算法 **1**. 基于最大度的算法.

输入:图 $G(V,E)$,图中每个结点的 $d^{\max}$,边权值,源结点 r,权值函数 $f(x)$.

输出:树 T.

初始化待求树 $T=\varnothing$,将源结点 r 纳入到树 T 中.

(1) 从 J 中选择一个 $d^{\max}$ 值最大的结点 u;

(2) 从 I 中选择结点 v,$deg_T[v]$满足 $deg_T[v]<d^{\max}[v]$,且边 $e(v,u)$存在,边权值 $p[v,u]$最小;

(3) 将结点 u 和边(v,u)纳入到树 T 中;

(4) 修改集合 I 和 J,同步反圈.

重复(1)~(4),直到所有的结点都纳入到 T 中.

算法 1 逐一处理每个结点,每次最多进行$|V|$次比较,时间复杂度为 $O(|V|^2)$.设任意两结点的距离 $dis(u,v)$满足 $0<\gamma\le dis(u,v)\le\eta$,结点度 $d[v]$满足 $1\le d^{\min}\le d[v]\le d^{\max}\le|V|-1$.关于延迟的最优解 λ_{opt} 满足$\lambda_{opt}>[\gamma+f(d^{\min})]depth_{opt}$,其中 $depth_{opt}$ 表示度约束树的最小深度;算法 1 得到的生成树 T 的最大延迟 λ_l 满足$\lambda_l<[\eta+f(d^{\max})]depth_{opt}$,由此推出 $\lambda_l<\dfrac{\eta+f(d^{\max})}{\gamma+f(d^{\min})}\lambda_{opt}\le\dfrac{\eta+f(|V|-1)}{\gamma+f(1)}\lambda_{opt}$,所以 λ_l 的近似比满足 $O(f(|V|))$.通常,权值函数 $f(x)$是线性函数,表示结点的度越大,需要的转发延迟越长,因此 λ_l 的近似比满足 $O(|V|)$.

2.1.2 基于度和最近距离的算法

在算法 1 中选定结点 $u\in J$ 后,才局部考虑边权,不能得到最优解.为此做如下改进:首先选择 k 个到源的路径最短的点集合 K,然后从中选 $d^{\max}$ 值最大的结点加入到 T 中.参数 k 在这里作为权衡因子.当 k 较大时,侧重于选择结点度较大的结点;当 k 较小时,侧重选择边点权值较小的结点.算法 2 既考虑权值,也考虑结点的度约束,其解比算法 1 更优.详细描述如下:

算法 **2**. 基于度和最近距离的算法.

输入:图 $G(V,E)$,图中每个结点的 $d^{\max}$,边权值,源结点 r,权值函数 $f(x)$.

输出:树 T.

初始化待求树 $T=\varnothing$,将源结点 r 纳入到树 T 中.

(1) 从 J 中选择 k 个距离源最近的结点 u_i,放入集合 K;

(2) 从 K 中选择一个 $d^{\max}$ 值最大的结点 u;

(3) 从 I 中选择结点 v,$deg_T[v]$满足 $deg_T[v]<d^{\max}[v]$,且边 $e(v,u)$存在,边权值 $p[v,u]$最小;

(4) 将结点 u 和边(v,u)纳入到树 T 中;

(5) 修改集合 I 和 J,同步反圈;根据 $c[x]=f(deg[x])$修改结点 u 和 v 的权值.

重复(1)~(5),直到所有的结点都纳入到 T 中.

算法 2 逐一处理每个结点,每次最多进行 $k\cdot|V|+k$ 次比较,时间复杂度为 $O(|V|^2)$.

2.2 基于最大延迟路径的算法

由于基于最大延迟路径的 DCMD-H(DCMD-heuristic)算法起源于 H-MDM (heuristic-MDM)算法,因此,本节首先介绍用来解决 MDM 问题的 H-MDM 算法[1];然后列举一个反例说明在某些情况下算法 H-MDM 不适合用来构建应用层组播树;最后我们改进 H-MDM 算法,设计了一种用于解决 DCMD 问题的 DCMD-H 算法.

在 H-MDM 算法中,结点的“接收延迟”表示从源向这个结点发送一个数据包的延迟,结点的“延迟路径”表示从源到这个结点的最短路径,该路径的长度(通常是“延迟”)叫做延迟路径长度.初始状态下,将源 r 纳入到空树 T 中,然后再按照下列步骤依次进行.

H-MDM 算法.

(1) 计算 $\overline{T}$ 中每个结点的最小接收延迟;

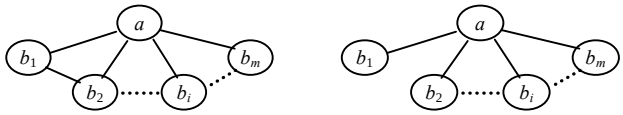
(2) 选择其中具有最大接收延迟的结点;

(3) 将这一结点及其延迟路径加到 T 中.

重复(1)~(3),直到所有的结点都纳入到 T 中.

按照 H-MDM 算法,为图 1(a)所示的覆盖图构建生成树,可能导致一些结点的度很大,如图 1(b)所示.其中结

点 a 到结点 b_i 的距离为 $L-i\cdot p(a)$,i=1,2,3,…,m,$p(a)$表示结点 a 转发一个包的发送处理时间,结点 b_i 到结点 b_j 的距离为ε,ε是一个足够小的正数.结点 a 在生成树中的度 $m_a\approx L/p(a)$.当 L 足够大时,m_a 将很大,此时 a 的转发工作量很大,可能不能及时接收数据,造成数据丢失.所以,H-MDM 算法不能直接用来解决 DCMD 问题.



(a) The input graph (b) The H-MDM tree
(a) 输入图 (b) HMDM 树

Fig.1 Example that node degree is very large in the H-MDM tree
图 1 H-MDM 算法导致结点度很大的例子

为此,我们对算法 H-MDM 做了改进,得到算法 DCMD-H.其基本思路是,对于算法 H-MDM 的第 2 步,不仅考虑延迟路径长度,还考虑相关结点的度是否超出限制,如果超出就选择次长路径,直到找到合适的结点为止.显然, DCMD-H 算法考虑了约束结点度的问题.

约束结点度会引起度约束生成树问题,这是一个 NP-complete 问题[7].我们不直接解决这个问题,而是利用应用层组播中主机结点的任意互连性,采用了回避策略.当算法失效时,需要由没有被纳入到树中的结点提供额外的连接信息来保证算法的两次可运行.我们对 320 个中小规模的图采用这种策略后发现,只有 3 例失效,需要两次信息补充,其余的失效情况只需一次信息补充,因此这个策略可以用于实际应用.

基于最大延迟路径的算法关键是求解延迟路径,为此我们引出如下定理.

定理 **2**. 任意延迟路径 l 上的任意结点 i 的延迟路径 l_i 也必定属于 l.

证明:设 l 表示从源结点 r 到结点 g 的延迟路径 l_{rg},设路径 $l_{rx}\subseteq l$,$x\in l$,$|l_{rx}|$ 表示路径 l_{rx} 的长度.设结点 x 的延迟路径 $l'_{rx}\neq l_{rx}$,$|l'_{rx}|$ 表示结点 x 的延迟路径的长度.由延迟路径的定义可知 $|l'_{rx}|<|l_{rx}|$,也就是说,$|l'_{rx}|+|l_{xg}|<|l_{rx}|+|l_{xg}|=|l|$,这个与"$l_{rg}$ 是 r 到 g 的延迟路径"的已知条件矛盾.因此 $l'_{rx}=l_{rx}$,定理得证. □

根据定理 2 容易推出每个结点的延迟路径必定先经过已经纳入树的结点和边,然后再经过还未纳入到树的结点和边,这就为求解结点的延迟路径 $l(r,g)$ 的长度提供了一个简化途径.可以先记录所有的 $i\in I$ 的延迟路径长度 l_{ri},然后计算结点的延迟路径 $l(r,g)=\min\{l_{ri}+l_{ig}\}$,其中 $i\in I$.下面给出 DCMD-H 算法的描述.

算法 **3**. 基于最大延迟路径的算法.

输入:图 $G(V,E)$,每个结点的 $d^{\max}$,边权值,源结点 r,权值函数 $f(x)=w_0+x\cdot w$.

输出:树 T.

初始化待求树 $T=\varnothing$,将源结点 r 纳入到树 T 中.

(1) 修改每条边的权值 $p(u,v)=p(u,v)+w_0$;

(2) 计算每对结点的延迟路径 $Path(u,v)$,$u\in V$,$v\in V$ 和延迟路径长度 $l(u,v)$;

(3) while ($|V|\neq|T|$) do

I. 从 $\overline{T}$ 中选取一个结点 u,满足源 r 到 u 的 $l(r,u)$最大的条件;v 为这条路径在 T 中的最后经过的结点,显然 $v\in I$;同时,选取的 u 必须保证 $deg[v]<d^{\max}[v]$,否则重新选取 $l(r,u)$次小的结点 u;

II. 将延迟路径(r,u)中不在 T 中的边和点添加到 T 中;

III. 修改 v 在 T 中的结点的度 $deg_T[v]+1$ 和结点权值 $c_T[v]$.

采用单源最短距离算法 Dijkstra 来求解结点对之间的延迟路径及长度,时间复杂度为 $O(|V|^3)$;While 循环最多执行$|V|$次,需要重新计算结点延迟路径,时间复杂度为 $O(|V|^4)$.故算法 3 的时间复杂度为 $O(|V|^4)$.

总体来说,DCMD-H 算法的基本思想起源于 H-MDM 算法,二者都基于最大延迟路径的贪婪算法,首先保证距离最"远"的结点的延迟,然后再逐步处理距离较"近"的结点.它们的不同之处在于 DCMD-H 算法用来解决 DCMD 问题模型,而 H-MDM 算法用来解决 MDM 问题模型.第 1 节的分析指出,由于性能方面的原因,必须要约束应用层组播的主机结点的出度.虽然算法 H-MDM 的设计者认为,通过算法 H-MDM 求得的生成树本身不会出现度很大的结点,同时还保证了较低的树延迟,但是,本节的反例证明了在某些特殊情况下 H-MDM 求得的生成树的某些结点度可能很大,也就是说,这个假设不合理.因此,当对应用层组播进行数学抽象时,需要明确地约

束结点的度,所以 DCMD-H 算法更适合用来构建应用层组播树.

3 模拟实验

本节模拟分析对于中小规模网络,我们提出的 3 种算法的平均优化情况.

在现实中,网络活动地点往往围绕着不同的城市,呈“簇”状分布,城市的分布呈随机态.因此,我们这样来描述应用层组播的用户分布:各个“簇”呈随机分布,每个簇的用户相对自己所在簇的中心点呈正态分布.故在模拟网络中,分别采用随机拓扑和正态拓扑两种结构.我们直接用 GT-ITM 生成随机网络,$scale$=100,n=$|V|$,α=结点对之间的互连概率 p,表示$|V|$个结点随机分布在 100×100 的平面上.正态拓扑表示结点的 x,y 坐标分布呈正态性,x~N(50,50),y~N(50,50),且 0<x,y<100.因为 GT-ITM 不支持正态拓扑,所以我们自己构建了新的拓扑生成器.利用当λ≥20 时泊松分布近似于正态分布的性质,通过泊松数列生成算法 Knuth 来改进 GT-ITM,从而生成仿正态拓扑网络.然后为每个结点随机生成一个结点的最大度约束值 $1\le d^{\max}\le 10$.假设线路延迟与距离成正比.

由于问题 DCMD 属于 NP-hard,不容易精确计算出树延迟最小的生成树.为衡量本文算法的优化度,我们以 $dis^{\max}=\max\{\text{延迟路径长度}_i\}$ 为参考值,其中延迟路径长度 $_i=\sum_{e\in i}\text{线路延迟}_e+f(1)$,i 表示延迟路径.显然,$dis^{\max}$ 小于等于树延迟的最优解 λ_{opt} .

3.1 与参考值$dis^{\max}$的比较

注意,我们采用结点的平均邻居个数 Ne 来替代 p,例如当 Ne=4 时,如果$|V|$=20,那么 p=0.2;如果$|V|$=400,那么 p=0.001.我们测试 320 个随机图/正态图,规格见表 2,每个规格有 20 个图.权值函数 $f(x)=w_0+deg[v]\cdot w$,w_0=4,w=1.算法 2 的权衡因子 k=5.表 2 的数据是相应 20 个图的平均值.从表 2 可以看出,用算法 3 所得的解离 $dis^{\max}$ 最近,近似比最好;算法 1 的所得解较差,平均情况都可能是 $dis^{\max}$ 的 5 倍,并且随着结点个数的增多,解越来越差;从图 2、图 3 可以明显看出,用算法 2 和算法 3 求得的生成树的树延迟在平均情况下比较接近 $dis^{\max}$.

Table 2 Computational results of the three algorithms for DCMD problem

表 **2** 3 种算法求解 DCMD 问题的计算结果

	Scale	Ne=4				Ne=6				Ne=8			
		$dis^{\max}$	Spanning tree delay			$dis^{\max}$	Spanning tree delay			$dis^{\max}$	Spanning tree delay		
			Alg 3	Alg 2	Alg 1		Alg 3	Alg 2	Alg 1		Alg 3	Alg 2	Alg 1
Random graph	20	211	216	229	333	154	160	174	253	122	134	142	197
	50	286	293	309	545	184	204	216	397	154	176	192	339
	100	300	316	345	605	250	269	287	607	183	200	240	566
	150	291	304	348	777	240	262	282	682	181	203	243	638
	200	364	379	412	904	232	275	318	820	207	234	259	692
	300	362	379	383	1 098	262	289	319	942	220	241	285	828
	400	414	432	474	1 242	280	301	331	1 086	218	227	311	1 040
	500	425	425	505	1 355	301	321	367	1 357	–	–	–	–
Normal distribution graph	20	67	74	84	112	47	54	57	87	40	48	53	79
	50	87	97	103	172	58	73	87	133	51	65	75	120
	100	98	110	107	204	75	88	100	196	64	79	89	183
	150	96	109	126	256	76	92	100	224	64	83	90	202
	200	113	128	136	310	82	99	104	272	68	89	102	228
	300	105	122	130	366	81	104	110	314	79	96	107	273
	400	126	142	153	383	95	113	126	377	73	95	106	325
	500	133	150	153	451	92	118	131	431	–	–	–	–

Note: (1) Alg=algorithm;(2) “Normal distribution graph” means the graph where nodes’ coordinate follow Normal Distribution.

为了进一步定量描述算法 3 的优化度,我们对每个图计算百分比相对差值的均值 $\delta=\frac{1}{20}\sum_{i=1}^{20}\frac{|L_i-dis_i^{\max}|}{dis^{\max}}\times 100\%$,这样可以较为准确地衡量树延迟与 $dis^{\max}$ 之间的差距,其中 L_i 表示算法 3 得到的生成树的延迟,δ 的计算结果见表 3.

我们得到如下结论:

- 当网络比较稀疏、结点比较分散时,算法 3 所得解较好,δ≈3%~5%.
- 如果结点位置不变,增加互连概率,则δ增加.δ越大表示算法的解越来越差.

• 我们所测试的 320 个图的平均情况 $\bar{\delta}$=14.5% ,最差情况δ=150%.这表明,在我们的实例中,算法 3 所得的解平均比 dis^{max} 大 14%,其中最差的解是 dis^{max} 的 5 倍.

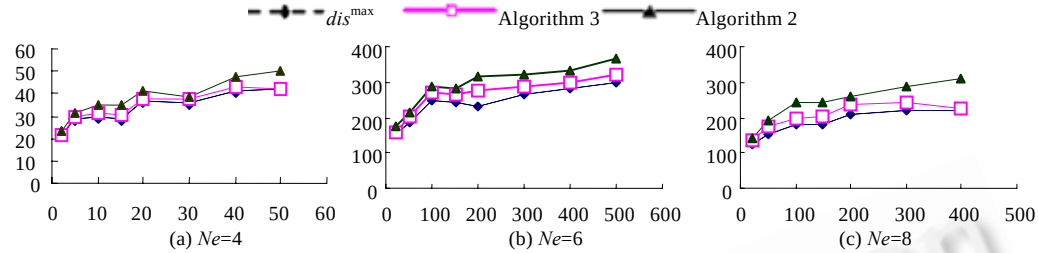


Fig.2 The relation graph of scale (x axis) and the spanning tree delay (y axis) in the random graph

图 2 随机图的规模(x 轴)与树延迟(y 轴)曲线关系图

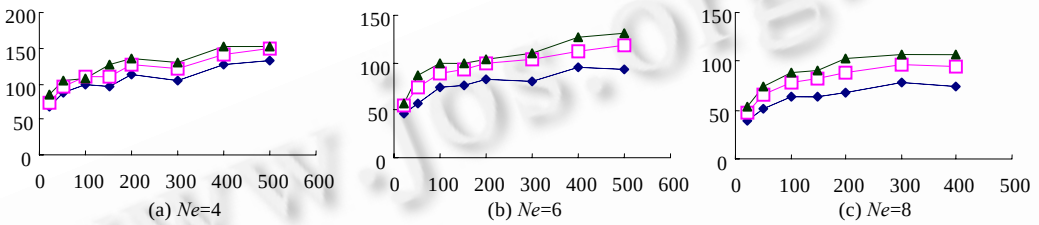


Fig.3 The relation graph of scale (x axis) and the spanning tree delay (y axis) in normal distribution graph

图 3 正态图的规模(x 轴)与树延迟(y 轴)曲线关系图

Table 3 Comparsion between dis^{max} and the computional results of Algorithm3

表 3 算法 3 的解与 dis^{max} 的对比

Scale	δ					
	Random graph			Normal distribution graph		
	Ne=4	Ne=6	Ne=8	Ne=4	Ne=6	Ne=8
20	3	4.2	10.4	11	14.9	21.5
50	3	11.1	15.6	10	29.1	29.2
100	5	7.9	9.4	13	17.3	26
150	4	9.5	12.6	14	22	32.5
200	4	6.6	13.1	14	19.9	32.4
300	5	11	10.1	16	26.0	22.6
400	4	7.8	14.2	13	20.8	32.5
500	5	6.8	-	13	20	-
Average	4.125	8.1125	12.2	13	21.25	28.1

3.2 与网络稠密度的关系

第 3.1 节的模拟显示:δ随着网络稠密度的增加而增大(正态图相对随机图要稠密一些),即算法所得的树有变坏的趋势.那么是否随着网络稠密度的增加,解是否会越来越坏?为此我们测试 20 个规模为 500 个结点的随机图,通过算法 2 得到最大树延迟和互连概率 p 的曲线关系(如图 4 所示).可以看出,当 p≥0.2 时,随着 p 的增加,最大延迟下降较为平缓;p 从 0.2 上升到 0.5,最大延迟仅仅降低了 1/10.当 p=0.2 时,算法 2 的解是 dis^{max} 的 2 倍.所以随着稠密度的增加,我们的算法不会越来越远离参考值 dis^{max}.

这个结果很容易理解,结点位置基本决定了延迟属性.虽然增加互连概率可以减少链路的中转次数,节省转发处理时间,但是延迟属性还是没有发生变化;另外,受结点度的限制,不是所有的结点对都能够直接连接,必须经过一些中转结点的中转处理,这样延迟路径上中转结点的数目必然会随着网络规模的扩大而增加.所以,当互连概率达到一定的值后,优化的余地就越来越小.

总体而言,当结点比较分散、网络比较稀疏时,算法 3 与参考值 dis^{max} 十分接近,所得解比 dis^{max} 大约多 10%左右;算法 2 略差,所得解大约比 dis^{max} 多 20%左右.算法 1 虽然计算简单,但是所得解很差,同等情况下,大约是 dis^{max} 的 3~5 倍左右.当网络比较稠密时,例如 500 个结点随机分布在 100×100 的平面上,当互连概率达到 0.2 时,

算法 2 所得的解大约为 dis^{max} 的 2 倍;此后,随着互连概率的增加,这种情况基本保持不变,解有轻微变好的趋势.

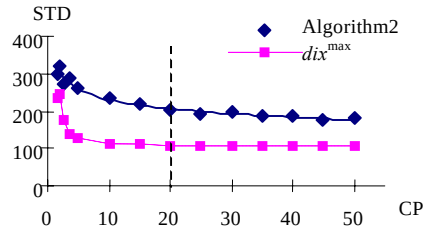


Fig.4 $|V|$=500, the relation graph of the spanning tree delay (SPD) and the connection probability (CP)

图 4 $|V|$=500,树延迟与互联概率 p 的曲线关系

4 结 论

本文主要研究了构建应用层最小延迟组播树的问题.文章的主要贡献包括,专门针对应用层组播定义了一种更加全面的问题模型——DCMD,并且证明这个问题属于 NP-hard.DCMD 与现行的模型不同,结点与边都带权,且结点的最大度受限.我们提出两类近似的启发式算法:基于度的算法和基于最大延迟路径的算法.其中基于最大度的时间复杂度为 $O(|V|^2)$,近似比为 $O(|V|)$;基于最大延迟路径算法的时间复杂度为 $O(|V|^4)$.模拟结果显示,在所提出的 3 种算法中,算法 3 相对最优;当结点比较分散、网络比较稀疏时,算法 3 求得的生成树延迟比参考值 dis^{max} 大约多 10%左右,算法 2 多 20%左右.随着网络稠密度的增加,算法 2、算法 3 求得的最小延迟生成树延迟大约与参考值 dis^{max} 保持常数倍的关系,也就是说,此时算法 2、算法 3 的近似比是常数.

在今后的研究中,我们将继续改进求解 DCMD 的算法,以获得更好的近似比;同时我们将致力于大规模应用层组播树的研究,对于大规模组播树的构建,不能仅仅依靠算法来获得优化解,需要实施某种策略才能获得较好的应用层组播树.

References:

[1] Broash E, Shavitt Y. Approximation and heuristic algorithms for minimum delay application-layer multicast trees. In: INFOCOM 2004, the 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies. Vol 4, 2004. 2697−2707. http://www.ieee-infocom.org/2004/Papers/56_1.PDF

[2] Shi SY, Turner JS. Multicast routing and bandwidth dimensioning in overlay networks. IEEE Journal on Selected Areas in Communications, 2002,20(8):1444−1455.

[3] Banerjee S, Kommareddy C, Kar K, Bhattacharjee B, Khuller S. Construction of an efficient overlay multicast infrastructure for real-time applications. In: INFOCOM 2003, the 22nd Annual Joint Conf. of the IEEE Computer and Communications Societies. Vol 2, 2003. 1521−1531. http://www.informtik.uni-trier.de/~ley/db/conf/infocom/infocom2003.html

[4] Tan SW, Waters G, Crawford J. A survey and performance evaluation of scalable tree-based application layer multicast protocol. Technical Report, No.9-03, Canterbury: University of Kent, 2003.

[5] Salama HF, Reeves DS, Viniotis Y. The delay-constrained minimum spanning tree problem. In: Proc. of the 2nd IEEE Symp. on Computers and Communications. 1997. 699−703. http://portal.acm.org/citation.cfm?id=845348

[6] Mokbel MF, El-Haweet WA, El-Derini MN. A delay-constrained shortest path algorithm for multicast routing in multimedia applications. In: Proc. of the IEEE Middle East Workshop on Networking. 1999. http://www-users.cs.umn.edu/~mokbel/Beriut99.pdf

[7] Jungnickel D. Graphs, Networks and Algorithms. Springer-Verlag, 1999. 120−123.

[8] Elkin M, Kortsarz G. A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. In: Proc. of the Annual ACM Symp. on Theory of Computing. Montreal, 2003. 438−447. htpp://www.crab.rutgers.edu/~guyk/pub/newbr/nabs.ps

[9] Tan SW, Waters G. Building low delay application layer multicast trees. In: Merabti M, Pereira R, eds. Proc. of the 4th Annual PostGraduate Symp.: The Convergence of Telecommunications, Networking & Broadcasting, EPSRC. 2003. 27−32. http://www.cms.livjm.ac.uk/pgnet2003/submissions/Paper-05.pdf

[10] Riabov A, Liu Z, Zhang L. Overlay multicast trees of minimal delay. In: Proc. of the 24th Int'l Conf. on Distributed Computing Systems. 2004. 654−661. http://www.informatik.uni-trier.de/~ley/db/conf/icdcs/icdcs2004.html