基于密度树的应用层多播算法

彭金祥
(广东纺织技术学院　佛山 528041)

摘　要　在IP多播的路由器上转发数据，存在一些弊端并不适应当前网络的需要，在终端主机上选择路由进行多播已经成为目前的趋势。为此，本文在IHC算法上进行优化改进，提出一种新的DHCM(density-based hierarchical clustering multicast)应用层多播模型，DHCM对Cluster的密度进行层次划分，使Cluster满足单调性和同构性，组建一个密度树，实现最短路由，并把Peer-to-Peer技术用在数据传输上，最终把视频服务器的内容分发到密度树上的各个主机，实现应用层多播。实验证明DHCM可以在视频流传输上具有高效性和健壮性。

关键词　应用层多播，密度树，P2P，网络

The Application Level Multicast Based on Density Tree

PENG Jin-Xiang
(Guangdong Textile Polytechnic, Foshan University, Foshan 528041)

Abstract　Currently IP multicast copies and transmits data in router so that it cannot meet the demand for its disadvantages, and it is a trend that multicast is realized through route seletion at end hosts. This paper has proposed a new model of application level multicast named DHCM (density-based hierarchical clustering multicast) which has improved IHC arithmetic. DHCM divides the hosts into many hierarchies according to their density, and constructs a density tree to realize the shortest routing. The tree delivers the content of video server to each host in density tree and uses a P2P scheme in data transmission. By this way the application-level multicast has been realized. This density tree has the homogeneity and monotonic properties. The experiment result has proved that DHCM can transmit the video stream efficiently and robustly.

Keywords　Application level multicast, Density-based hierarchical tree, Electric network, Peer-to peer

1　引言

在当今视频会议、网络游戏、远程教育中，越来越要求对一定范围内特定的主机群发送数据进行通信，并发挥其多播独特的作用。多播技术中又分IP多播和应用层多播。在应用层多播中有集中式算法如ALML[2]和分布式算法如网络优先算法Narada[3]；树优先算法如YOID[4]、NICE[5]。还有一种形成集群的方法，是先随机估计一个集群的中心C_C，然后估计一个集群的阈值半径R_C，一旦C_C和R_C确定下来后，集群就定义为$Cluster=\{g_i \in G \mid \| g_i - C_C \| < R_C\}$，所有阈值半径内的节点都归为一组。但是该方法并不能导致树的深度随着节点数的增加而增大，最终增加了数据传输的延迟。为此，本文提出了一种基于密度层次集群的应用层多播算法(DHCM)，以密度树来保证数据传输在树上的深度，同时，又把Peer-to-Peer应用在层与层之间交换上，这即节省了带宽，又提高了网络效率。

2　密度树的单调性和同构性

为了保证每个集群的单调性和均匀性，根据文[6]中提到的IHC算法，把所有加入多播的终端主机构建为一个密度树DT(density Tree)，且DT能满足单调性和同构性。同构性是指一些相似密度的终端主机node集合成一个集群，各个node之间的距离满足一定要求。单调性是指从根node到叶node，集群的密度逐渐增大。

性质1　假如$D_A=\langle NDP,\mu,\sigma\rangle$表示以节点$A$为中心的集群密度，$NDP=\{d_i \mid d_i \in \mathcal{R}\}$，$d_i$为$A_i$的第$i$个子node和其他兄弟node之间的最近距离，则通过最小生成树算法(MST)得到d_i，μ和σ为NDP的均值和标准差。

性质2　假如$D_A=\langle NDP,\mu,\sigma\rangle$表示$A$的集群密度，取下限函数$L_L(\mu,\sigma)=\mu-\sigma$，上限函数$U_L(\mu,\sigma)=\mu+\sigma$，若$\forall d_i \in NDP$，且满足$L_L \leqslant d_i \leqslant U_L$时，则称该集群满足均匀性。

性质3　假如一个新节点A加入以C为中心的集群时，C子节点中最接近A节点为B，d为A和B的距离，若$d>L_L$，则认为A形成更高的密度区间；若$d>L_L$，则认为A形成更低的密度区间。

性质4　假如节点A与节点B之间距离，以A与B之间的延迟和RP到$Server$之间的延迟比为衡量单位，RP是一个与$Server$保持连接的特殊点，$Server$是发送数据的源出发点，$La(RP,Server)$为RP到$Server$之间的延迟(因为RP到$Server$之间的延迟是一定的，所以可以用$La(A,B)$表示A与B之间的延迟)，则A与B之间的距离，为：

$$dis(A,B)=\frac{La(A,B)}{La(RP,Server)}.$$

3　应用层多播模型算法

假定有个特殊的节点RP(Rendezvous Point)，DT上所有的节点都知道RP的位置，试图加入多播树的主机，都通过"bootstrap"机制连接RP来初始化，并规定RP是多播树DT的最高一层节点$Server$的父亲，这里，RP不属于DT中的节

*)本课题得到全国教育科学十五规划重点课题(No:AYA010034)基金资助。彭金祥　硕士，高级讲师，研究方向：计算机应用。

点，只是和 *Server* 维护一个连接，它只是在多播树的构建和维护上起作用，数据传输的时候会区分 *RP* 与 *DT* 上的节点，不传输数据给 *RP*，集群的多播树 *DT* 的数据流图，如图 1 所示。

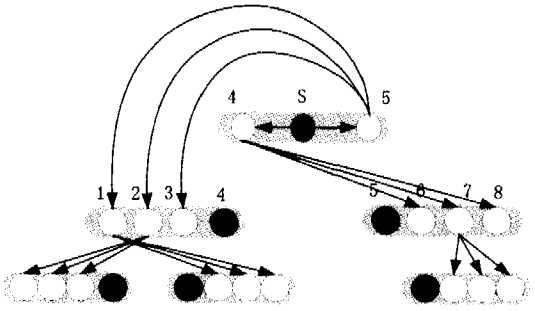


图 1　多播树 DT 的数据流图

由图 1 可知多播树的特点：在第 $j(j>0)$ 层集群中，它的集群中心 node 即是其他节点的父亲节点，在第 $j+1$ 层上，最高层为 *Server*；如果一个 node 父亲节点为孤独节点，并且只有它一个孩子，那么它只能把数据发往它的所有孩子，只有最高层的 Server 可以把数据发往所有它孩子节点，除此之外，node 只能把数据发往自己的外孩子，如：节点 5 只能发往节点 1,2,3；集群中非 head 节点不能直接从父节点接受数据，只能从非上一层的非 head 外父亲接受数据，除非它没有外父亲，如：1,2,3 从 5 获取数据，4,5 只能从 *Server* 获取数据。

3.1　构建数据的密度树

系统构建之初只有一个 *RP* 和一个 *Root*，然后其他节点逐一加入。一个新节点 N_J 的加入可以分为三个阶段：其一，由 *RP* 找到根结点后，再从根节点开始查找，在每一层里找到最接近自己的节点，直到找到最接近自己的叶子节点；其二，从叶子节点 N 开始将新节点 N_J 加入 *DT* 的合适位置，如下述算法 1；其三，从 N_J 开始对 *DT* 进行重构，然后建立相应的数据链路，N_J 的加入会使 *DT* 的单调性和均匀性改变，如下述算法 2、3。其中 $dis(A,B)$，表示节点 A 与节点 B 之间的距离；$d(A,B)$ 表示节点 A 与节点 B 的孩子节点中最近的孩子节点之间的距离。

算法 1　JoinDT(N,N_J)

if N 是 *Root*，就把 N_J 直接作为 *Root* 的孩子；过程结束；

N←N. *Parent*；

While (d(N_J,N)>N. U_L 并且 N 非 Root)；N←N. Parent；

if N 是 Root，就把 N_J 直接作为 Root 的孩子；

算法说明：继续搜索父节点到最接近自己的较小密度，然后根据当前的节点 N 与要加入的节点 N_J 的距离分情况执行：插入新节点 N_J；对一个孩子节点进行层次插入操作；对最近距离的孩子继续搜索。

算法 2　Hierarchy. Restructuring (N)

```
While (N! = Root){
  N_P←N. Parent ;
  Homogeneity. Maintenance (N);
  Recover. Hierarchy (N);
  N←N_P;
}
```

算法说明：从 N 开始进行重构，此处 N 就是刚加进来的新节点或是删除节点的父节点。跟文[6]中提到的 IHC 算法不同，IHC 算法是用来构建数据匹配时建立数据的密度树，所以插入节点和许多 merge、split 操作，可以把两个节点合并和拆分，所以其算法只能通过调节节点的层次关系来重新构建密度树。

算法 3　Homogeneity. Maintenance (N)

```
Repeat {
    Find N_I 和 N_J 是 N 的孩子节点中距离最短的节点；Flag = true;
    If dis(N_I,N_J)<N. L_L
    Then {While (d(N_J,N_I)<N. L_L 并且 N_I 非叶子节点)
            N_I←N_I 的离 N_J 最近的孩子节点；
          JoinDT(N_I,N_J);Flag = false ;}
} Until (Flag)
Repeat {
      Find N_I 和 N_J 是 N 的孩子节点中距离最长的节点；Flag = true;
      If dis(N_I,N_J)>N. U_L Then { JoinDT(N_I,N_J);Flag = false;}
} Until (Flag)
```

算法说明：对违反定义 2 的节点进行层次调节。如果形成更高的密度区间，则把节点往下移动层次，调用 JoinDT 方法加入到合适的位置。如果是形成更低的密度区间，则直接调用 JoinDT 方法加入到合适的位置。

3.2　数据的重构

当多播树进行重构时，一个节点必须进行合适的数据链路的建立，为数据连接的其他节点提供最好的性能服务。一个节点选择外父亲来进行数据连接时，也应该选择最不忙的外父亲建立链路。节点 X 是否忙用 d_X/B_X 来衡量，d_X 表示节点 X 的度，B_X 表示节点 X 的带宽。其一，节点 N 经过层次调节，加入第 j 层，如果 N 的父亲节点为孤独节点 X，而且 X 现在只有 N 这个孩子节点，则寻找 X 的一个兄弟节点 Y。跟其他兄弟节点比较，d_Y/B_Y 最小，N 和 Y 建立数据链路。否则假设 N 的所有兄弟节点的数据链路连接的外父亲为 Y。若 Y 的原来度数为 S_{old}，加入后度数为 $S_{old}+1$，调用 DataLink. Change(Y,S_{old},$S_{old}+1$)。其二，节点 N 的离开，它的孩子节点归并了到新的父亲 N' 下。如果 N' 无兄弟节点，则直接和 N' 建立数据连接通路；否则 N' 原来的孩子节点与外父亲 X（若 N' 不为孤独节点，则 $N'\neq X$）建立数据连接，其中，X 的原来度数为 S_{old}，n 个节点归并后和 X 建立数据连接，度数变为 S_{new}，满足关系 $S_{old}+n=S_{new}$，重新查找正确的数据链路连接的外父亲，则有算法 4：

算法 4　DataLink. Change(X,S_{old},S_{new})

选择 X 的一个兄弟节点 Y，并且 $Y\neq N'$：

$$\left(\frac{d_X}{B_X}-\frac{d_Y}{B_Y}\right)^2-\left(\frac{d_X-S_{new}}{B_X}-\frac{d_Y+S_{new}}{B_Y}\right)^2>0$$

$$\left(\frac{d_X}{B_X}-\frac{d_Y}{B_Y}\right)^2-\left(\frac{d_X-S_{new}}{B_X}-\frac{d_Y+S_{new}}{B_Y}\right)^2 \text{ 的值最大}$$

如果这样的 Y 存在，则把 N' 的所有孩子节点都和 Y 建立数据连接，删除和 X 的数据连接。更新 X 和 Y 的度数。其二，节点 X 的离开，将导致它的外孩子节点要重新查找正确的外父亲建立数据连接通路。如果 X 连接的外集群为 C_1，C_2，…，C_m，每个 C_i 有 S_i 个非集群中心的节点，C_i 的中心节点为 H_i，则有算法 5：

算法 5　DataLink. Change (X)

For (i=1,$i\leqslant m$;i++){

选择 X 的兄弟节点 Y，并且 $Y\neq H_i$：

$$\left(\frac{d_X}{B_X}-\frac{d_Y}{B_Y}\right)^2-\left(\frac{d_X-S_i}{B_X}-\frac{d_Y+S_i}{B_Y}\right)^2>0$$

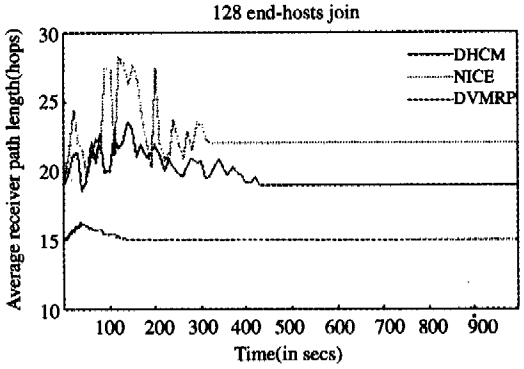
$$\left(\frac{d_X}{B_X}-\frac{d_Y}{B_Y}\right)^2-\left(\frac{d_X-S_i}{B_X}-\frac{d_Y+S_i}{B_Y}\right)^2 \text{ 的值最大}$$

如果这样的 Y 存在，则把 C_i 的所有非中心节点都和 Y 建立数据连接，删除和 X 的数据连接，更新 X 和 Y 的度数。

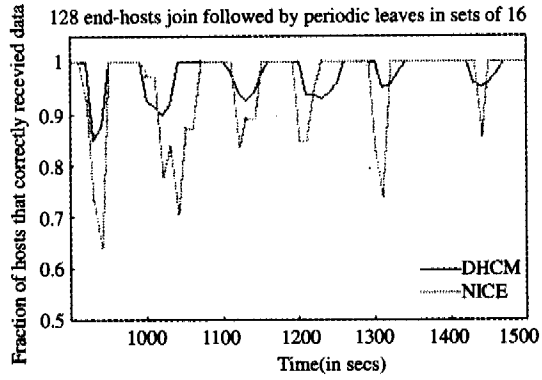
4 模拟试验结果与分析

为了更好地显示应用层多播算法特点，运用模拟分析方法来分析 DHCM 协议的性能。同 IP 层和应用层的多播进行比较，应用层选择 Nice 协议，IP 层选择距离向量多播协议 DVMRP；使用 GT-ITM 拓扑生成器[7]生成网络拓扑。并对节点加入和离开进行三种协议的性能比较，其中，加入阶段是在头 200 秒时加入 128 个节点，离开阶段是在 1000 秒后按照每次 10 秒离开 16 个节点开始进行统计，如图 2 所示。

分析知道，DHCM 在链路强度上和平均路径长度都好于 NICE 协议，但是 IP 多播在平均路径长度还是占有绝对的优势。同时也发现因为 DHCM 在节点加入，需要不断调整层次来达到最优效果，所以最后稳定时刻要比 NICE 协议滞后一段时间。但是稳定下来后的效果要优于 NICE 协议。离开阶段 DHCM 要稍微好于 NICE 协议。



(a) 加入阶段的平均路径长度图



(b) 离开阶段的主机接受数据的比例

图 2

结论 经过实验证明，本文的 DHCM 多播方案，用密度树来划分集群，可以高效的聚集相似密度主机，让数据最短路由到达所有主机；同时用 Peer-to-Peer 进行数据传输，可以节约带宽，不至于出现同一节点向太多节点传输数据而产生瓶颈；其算法保证了可靠的错误恢复，低维护率。

参 考 文 献

1 Saltzer J, Reed D, Clark D. End-to-end arguments in system design. ACM Trans. Compute. Sits, 2004, 2: 195～206
2 Pendarakis D, Shi S, Verma D, et al. ALMI: An application level multicast infrastructure. In: Third USNIX Symposium on Internet Technologies and Systems (USITS '01), March 2001
3 Chawathe Y, Rao S G, Zhang H. A case for end system multicast. In: ACM SIGMETRICS, June 2005
4 Paul F. Yoid distribution protocol (yap) specification. Unreformed Report. http://www.yallcast.com, April 2004
5 NICE working group home page. http://www.math.princeton.edu/tsp/
6 Widyantoro D, Yen J. An incremental approach to building a cluster hierarchy[A]. In: Proceedings of the 2002 IEEE International Conference on Data Mining[C]. Maebashi City, Japan, 2002
7 Calvert K, Zegura E, Bhattacharjee S. How to Model an Internet work. In: Proceedings of IEEE Infocom, 2005

(上接第 36 页)

相关部分被封装到一些额外的类中(Session 和 State)，这些类相对比较容易创建和维护。并且这个结构的扩展性，灵活性非常好，另外在性能上也没有产生用户可以察觉的性能下降。

XML、URI 技术的出现(电子邮件通信方面 Email 的 XML 表示[12]，以及 POP URL[13]标准)促进了通信格式、命名和定位的统一。在应用层方面协议的工作也有待展开。本文从软件的通讯体系结构上做了初步的尝试，有很多工作还有待进一步的展开：

(1)在 State 类层次中，对应每个协议的每个状态都存在一个状态类，如何减少 State 类层次中的状态类数目，对一些通用状态类进行抽象，复用，这是一个难题。

(2)这个结构用于我们的通讯平台，经过适当的改良和封装，可以作为一个支持多协议网络通信的类库发布。

(3)协议通常描述为(FSM)，那么可以通过规范的描述来自动生成 State 框架代码。

(4)当前我们只针对 pop3 和 SMTP 协议进行了试验，随着项目的展开和其他协议的加入可能会产生新的问题。

参考文献

1 Alexander C. The Timeless Way of Building. New York: Oxford University Press, 1979
2 Alexander C, Ishikawa S, Silverstein M, et al. A pattern Language - Towns Buildings Construction. New York: Oxford University Press, 1977
3 Gamma E, Helm R, Johnson R, et al. Design Patterns - Elements of Reusable Object-Oriented Software. New York: Addison-Wesley, 1995
4 Buschmann F, Meunier R, Rohnert H, et al. Pattern-Oriented Software Architecture. Vol 1. A System of Patterns. Chichester: John Wiley & Sons Ltd, 1996
5 Cunningham & Cunningham, Inc. Anti Pattern. http://www.c2.com/cgi/wiki? AntiPattern, April 22, 2005
6 Gabriel R P. Patterns of software: tales from the software community. New York: Oxford University Press, 1996. 65～66
7 Schmidt D C. Experience Using Design Patterns to Develop Reusable Object-Oriented Communication Software. Communications of the ACM. Special Issue on Object-Oriented Experiences. Lecture Notes in Computer Science. 1995, 38(10)
8 Pont M J, Banner M P. Designing embedded systems using patterns: A case study. The Journal of Systems and Software, 2004, 71: 201～123
9 Beck K, Johnson R. Patterns Generate Architectures. Springer-Verlag, 1994
10 Menzies T. Object-Oriented Patterns: Lessons from Expert Systems. Software-practice and Experience, 1997, 1(1)
11 Diaz A, Fernandez A. A pattern language for virtual environments. Journal of Network and Computer Applications, 2000, 23
12 王守芳，樊闻斌，金浩，等. 统一消息中的邮件的 XML 标准化方法. 计算机科学，2004
13 RFC1939. Post Office Protocol - Version 3. 1996
14 RFC2060. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. 1996
15 RFC0821. Simple Mail Transfer Protocol. Aug 01, 1982
16 Tanenbaum A S. Modern Operating Systems. New Jersey: Prentice Hall. 1992
17 Coplien J O. Advanced C++ - Programming Styles and Idioms. New York: Addison-Wesley, 1992