第33卷 第23期 计 算 机 工 程 2007年12月
Vol.33 No.23 Computer Engineering December 2007
·网络与通信· 文章编号：1000—3428(2007)23—0132—03 文献标识码：A 中图分类号：TP393

一种带宽前瞻式的应用层组播路由算法

胡迎松，张 旭
（华中科技大学计算机学院，武汉 430074）

摘 要：流媒体直播是应用层组播技术的一个主要应用领域，对网络性能非常敏感，节点失效时快速恢复路由是一个核心问题。该文在几种常见的处理方法基础上，提出了一种带宽前瞻式的快速重建路由的方法。在节点离开或者发生故障之前就为其孩子节点计算备用路由，一旦节点离开，其孩子节点可以迅速找到并平滑地切换新的父节点，尽量选择服务能力较强的节点作为备用路由，从而增加树的稳定性。
关键词：应用层组播；生成树；备用路由

Bandwidth Considered Proactive Route Maintenance Method in Application Layer Multicast

HU Ying-song, ZHANG Xu
(College of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074)

【Abstract】 Live media streaming, which is interruption sensitive, is an important aspect of application layer multicast. It is important to restore the spanning tree when a node leaves. This paper analyses several familiar schemes to solve this problem, and proposes a bandwidth considered proactive route maintenance method. In this method, every non-leaf node computes backup route for its children nodes before it leaves. So children nodes can quickly switch to backup route when their current parent node leaves. In order to improve the stability of the spanning tree, it selects the node having the max service capability as backup route.
【Key words】 application layer multicast; spanning tree; backup route

IP组播对网络层设备有较大的依赖，需要对组播路径上所有的网络层设备进行调整和升级，为大范围部署IP组播带来了阻碍。因此，IP组播很长一段时间以来都没有得到广泛的应用。应用层组播(ALM)在终端主机上实现组播的功能；只需要安装应用软件，并且不需改变现有的网络架构。另外，ALM在路由上更灵活（比如在多路径数据包的转发和负载均衡方面）。目前，应用层组播主要用于实时的多媒体传输领域，包括视频会议、媒体流的分发等[1~2]。

1 问题及已有的解决方法

在ALM会话期间，每个终端主机都是数据传输树的一员，可以随意地离开，有时也可能发生故障。在IP组播中不存在这个问题，因为传输树中的非叶子节点是路由器，它不会不发出通知就离开传输树。在ALM中，必须考虑的问题之一就是当节点突然离去时重新构造组播树[3]。对于组播应用（比如流媒体直播），有节点离开时重新恢复组播树所需要的时间是很重要的，因为所有的孩子节点都会受到影响。因此，在重叠网络的组播中，由节点故障所引起的问题越来越受到人们的关注。目前常用的解决方法有被动的响应式方 法[4~5]和主动的前瞻式方法[6~7]。

1.1 响应式的方法

许多 ALM 系统采用响应式的方法，在节点离开后触发树的恢复过程。在这种方式中，节点正常离开前要发送消息通知其父节点和孩子节点。孩子节点直到加入新的父节点才能继续接收数据流。如果节点发生故障突然离去，它就不能向受影响的节点发出通知，这些节点要经过一段时间才能发现父节点的离开。心跳机制可以帮助这些受影响的节点发现父节点的突然离开。父节点和孩子节点周期性地互相发送 heartbeat 包表明链路通畅。当孩子节点在一定周期之内没有收到父节点发来的 heartbeat 包，它们就认为父节点发生故障离开了。然而，孩子节点需要一段超时时间才能发现故障，且在此期间不能收到数据流。

发现父节点的故障之后，所采取的方法主要有下列几种：

(1)当节点离开或者失效时，它的所有孩子节点都试图去连接根节点，而各个孩子节点的子树保持不变[8]。只要根节点的带宽没有耗尽就接受这些节点的加入，否则根节点就把它们重定向到后代节点。

(2)当节点离开或者失效时，它所有的后代节点都试图连接根节点。

(3)当节点离开时通知所有的孩子节点，让它们加入其祖父节点。当节点发生故障而失效时，因为孩子节点不知道祖父节点的地址，所以直接去连接根节点。

1.2 前瞻式的方法

在这种方法中，每个节点都保留一个备用路由以便父节点离开或故障时使用。一旦父节点离开，受影响的节点直接连接备用路由，这样它们只需很短的间断时间就能继续接收数据。

在PRM[9]系统中，因某个节点离开而重构树时，采用随机发送的方法。每个节点随机选择一定数量的其他节点并以

作者简介：胡迎松(1966－)，男，副教授，主研方向：计算机网络与应用；张 旭，硕士研究生
收稿日期：2006-12-20 **E-mail：**zx_mail2000@yahoo.com.cn

较小的概率向它们发送数据。这种方法在某些情况下看起来是有效的，但是会产生大量的冗余信息。文献[10]采用了增加冗余虚拟链路的方法来减少不能接收到多播分组的节点数量；从而增强了生成树的鲁棒性。

在杨氏[5]方法中，每个非叶子节点为其孩子节点计算备用路由。备用路由用重叠树中的剩余度来提供。首先，父节点计算其孩子节点剩余度的和。如果和大于等于孩子节点的数目，那么它们自己可以构成备用路由。否则，父节点计算其孩子节点包括孙子节点的剩余度。其次，选择到祖父节点的延迟最小的孩子节点。这个孩子节点将使用祖父节点作为备用路由。这个孩子节点的子树如果可能就向其他孩子节点提供备用路由。然后这个孩子节点的后代和这个孩子节点测量到其他孩子节点的延迟，延迟最小的被选用。这个过程一直重复，直到所有的孩子节点找到备用路由。这种方法产生的控制信息量较大。

2 考虑带宽的前瞻式路由算法

2.1 基本思路

为方便讨论，用出度来表示节点能够提供的链接的数目。比如媒体流要求的带宽为B，节点i能够提供的带宽为B_i；那么节点i最多可以提供B_i/B个链接。记$d_m(i)=B_i/B$。$d_u(i)$表示节点i已经使用的出度，即已经连接到节点i的节点数。$d_r(i)$表示节点i的剩余出度，即节点i还能提供的链接数。则$d_m(i)= d_u(i) + d_r(i)$。记$C(P)$为节点P的孩子节点集合。

(1)采用前瞻式的方法。流媒体应用对实时性要求比较高，在节点故障离开时需要迅速为其孩子节点找到新的父节点才能保证它们收到连续的数据流。因此，采用前瞻式的方法，即在节点发生故障之前就为其孩子节点计算好备用路由。

(2)每个节点保留至少 1 个出度用于提供备用路由。这样可以保证只需少量的控制信息就能在其祖父节点和孙子节点之间用剩余度来构建它们的备用路由。

(3)在选择备用路由时尽量将服务能力较强的节点放到树的上层。这样生成的树的高度比较低，并且比较稳定。定义服务能力(Service Capability, SC)$SC(x)=(d_r(x) -1)* t(x)$表示节点x提供服务的能力。这里使用$d_r(x)–1$ 是为了保留一个度以方便建立备用路由，$t(x)$表示节点x持续连接的时间。

2.2 节点加入时的处理

节点加入时的算法：

(1)新来的节点n向候选父节点P(初始时为根节点R)发送加入请求；

(2)如果$d_r(P)>1$，那么根节点接受n的加入请求；

(3)如果$d_r(P)<=1$，对所有$C_i \in C(P)$计算$SC(C_i)$，然后选择子节点中SC值最大的节点U，发送重定向消息给节点n，令$P=U$，转(1)。

假设图 1 中节点 R 的最大出度是 4，则节点 R 的活动出度为 3；保留 1 个出度以提供备用路由。新来的节点 H 向节点 R 发出加入请求。节点 R 收到请求之后检查其剩余出度

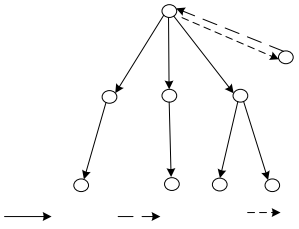

图 1 节点加入

$d_r(R)$，此时$d_r(R)=1$。所以，节点R拒绝节点H的加入请求。然后节点R计算其所有孩子节点的SC值，选择SC值最大的节点(这里假设为C)生成重定向消息发给节点H。节点H根据收到的重定向消息向节点C发出加入请求，结果节点H成为了节点C的孩子节点。

2.3 备用路由的构造

节点加入之后要计算备用路由。由于节点刚加入，因此不考虑其连接时间，只考虑带宽。图 1 中节点H加入并且成为节点C的孩子节点后，节点C更新其孩子节点列表，并且将其孩子节点按照d_r降序排列。d_r最大的节点用节点C的父节点作为其备用路由，d_r次大的节点用d_r最大的节点作为其备用路由，依此类推。假设$d_r(H)> d_r(G)> d_r(F)$，那么节点H用节点R作为其备用路由，节点G用节点H作为其备用路由，节点F用节点G作为其备用路由，如图 2 所示。

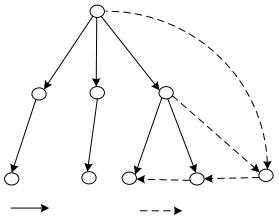

图 2 寻找备用路由

2.4 树的调整

在某些情况下，由于节点切换到备用路由，因此用尽了某个节点的最大出度。这时就要调整组播树。比如，由于节点 C 的输出链路发生故障，其孩子节点切换到备用路由，从而耗尽了节点 R 的出度，如图 3 所示。

节点 R 发现自己的出度耗尽，立即对树进行调整。节点 R 向其所有的孩子节点 A, B, C(新加入的节点 H 除外)发出询问信息。节点 A, B, C 收到此信息之后计算自己的 SC 值，并发给节点 H。假设图 3 中，节点 H 收到的信息为 $SC(A)=8$, $SC(B)=12$, $SC(C)=0$，那么选择 SC 值最大的节点 B 作为其父节点。于是节点 H 成为节点 B 的孩子节点，如图 4 所示。

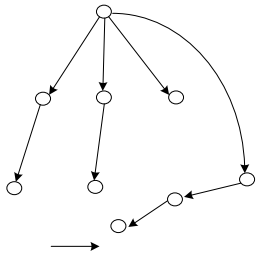

图 3 组播树调整前

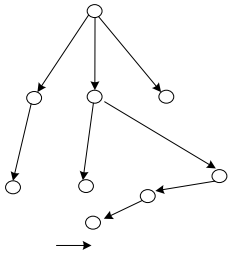

图 4 组播树调整后

2.5 特殊情况处理

某些特殊情况下一个节点不能使用其备用路由：一种情况是其父节点和备用父节点同时离开或发生故障。在这种情况下，采用文献[5]中提供的方法来处理，即使用祖先列表信息。首先试图连接其祖父节点，不管祖父节点是否能接受都遵循重定向算法来执行。当祖父节点因为离开或者故障而不存在时，节点就试图连接祖先列表中更高一层的祖先节点，直到找到新的父节点为止。

有些节点的出度为 0 或者 1 时，提出的方法就会遇到问题。而在 ALM 系统中存在出度为 0 的节点是很普遍的。它们只能作为叶子节点接收数据。如果出度为 1 的节点不能有孩子节点，那么极端情况下如果所有节点的出度都为 1 时，

按照提出的方法就不能构造树了。为了避免这种情况的发生，允许出度为 1 的节点连接一个孩子节点。另外一个问题是，它们用尽了出度或出度本来就是 0，因此它们不能提供备用路由。所以，把这些节点尽量放在末端，这样它们就不需要向其他节点提供备用路由了。

3 性能评估

本文使用软件仿真的方法对考虑带宽的前瞻式方法进行了性能测试。分析了节点发生故障时的恢复时间和维护路由需要的控制信息量。跟响应式的方法和杨氏方法进行了对比。

图 5 显示了在不同节点数量的情况下 3 种方法恢复故障所需要的时间。可以看出，响应式的方法恢复时间比较长，而且随着节点数目的增多恢复时间越来越长。杨氏方法和本文提出的方法恢复时间都比较短，且不会随节点数目的增加而变长。由于响应式的方法不需维护备用路由，只是在检测出节点故障时才开始寻找新的父节点，因此这种方法所需要的控制信息量很小。杨氏方法需要的控制信息量比较多。而本文提出的方法所需要的控制信息量比响应式的方法稍微多一点，如图 6 所示。

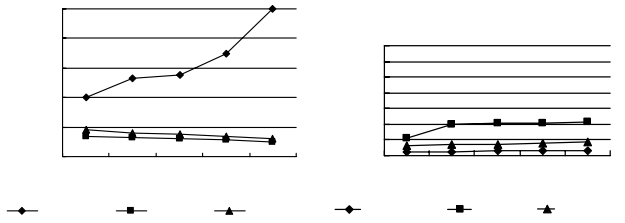

图 5　恢复时间与组播规模的关系　图 6　控制信息量与组播规模的关系

4 结束语

分析了节点故障时应用层组播树快速重建的重要性，介绍了几种常见的解决方法，提出了一种考虑带宽的前瞻式方法。该方法在节点离开或者发生故障之前就为其孩子节点计算备用路由，所以，节点离开或者发生故障时，其孩子节点可以迅速找到并平滑地切换新的父节点，并且尽量选择服务能力较强的节点作为备用路由，从而增加树的稳定性。仿真实验结果表明，节点离开时路由恢复速度比较快，维护备用路由需要的控制信息量比较少。

参考文献

1 陈庆吉. 支持实时多媒体传输的应用层组播系统[J]. 计算机工程, 2005, 31(4).

2 王艳丽, 鲜继清, 白　洁. 基于 P2P 的流媒体技术[J]. 计算机应用, 2005, 27(6).

3 李　伟, 沈长宁. 应用层组播协议的研究[J]. 计算机工程与应用, 2005, 41(24).

4 Noguchi T, Yamamoto M. Construction of a Robust Multicast Tree for Application-level Multicast[J]. IEICE Transactions on Communications, 2005, E88-B(12): 4427-4434.

5 Tan Guang, Jarvis S A, Chen Xinuo, et al. Performance Analysis and Improvement of Overlay Construction for Peer-to-Peer Live Media Streaming[C]//Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems. 2005.

6 Tran D, Hua K, Do T. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming[C]//Proceedings of IEEE INFOCOM′03. 2003.

7 Yang M, Fei Z. A Proactive Approach to Reconstructing Overlay Multicast Trees[C]//Proceedings of INFOCOM′04. 2004-04.

8 Kusumoto T, Kunichika Y. Proactive Route Maintenance and Overhead Reduction for Application Layer Multicast[C]// Proceedings of P2PMMS′05, Singapore. 2005-11.

9 Chu Y, Rao S G, Ses S, et al. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture[C]//Proceedings of ACM SIGCOMM′01. 2001-08.

10 吴国福, 刘志峰, 窦文华. 应用层多播中生成树的鲁棒性技术研究[J]. 计算机工程与科学, 2006, 28(6).

（上接第 101 页）

(2)二进制翻译的循环可扩展性(scalability)更好，随着循环次数的增加，其执行时间的增长比软件解释更为平缓。

二进制翻译利用程序执行的局部性原则，即大部分的程序执行时间花在了小部分的代码块上，这小部分频繁执行的代码块一般对应于源代码中的循环体，可以将其称为程序中“热点”(hot spot)。热点代码经过一次翻译后可以重复执行，这样翻译所带来的额外开销平摊到多次重复执行上以后，其平均开销就减小很多。与其相反的是，软件解释技术只是简单地循环解释每一条即将运行的指令，花费了很多不必要的时间解释相同的代码。因此，二进制翻译在大部分的程序运行中都能获得更好的性能。

4 结束语

本文介绍了二进制翻译技术，描述了动态二进制翻译基础平台 CrossBit 的设计与实现。CrossBit 以可重定向、模块化与可扩展为目标，其中，可重定向性主要通过机器无关的中间指令来实现。实验数据说明，二进制翻译技术和传统的软件解释技术相比，具有更高的执行效率与可扩展性。

参考文献

1 Altman E R, Kaeli D R, Sheffer Y. Welcome to the Opportunities of Binary Translation[J]. IEEE Computer, 2000, 33(3): 40-45.

2 Baraz L, Devor T, Etzion O, et al. IA-32 Execution Layer: A Two-phase Dynamic Translator Designed to Dupport IA-32 Applications on Itanium-based Systems[C]//Proceedings of the 36th International Conference on Microarchitecture, San Diego, CA. 2003.

3 Meloan S. The Java HotSpot Performance Engine: An In-depth Look[Z]. Sun Microsystems, 1999.

4 Sites R L, Chernoff A, Kirk M D, et al. Binary Translation[J]. Digital Technical Journal, 1992, 4(4).

5 Andrews K, Sand D. Migrating a CISC Computer Family onto RISC via Object Code Translation[C]//Proc. of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems. New York, USA. 1992: 213-222.

6 Bala V, Duesterwald E, Dynamo S B. A Transparent Dynamic Optimization System[C]//Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation, New York, USA. 2000: 1-12.

7 Burger D, Austin T M. The SimpleScalar Tool Set Version 2.0[R]. Technical Report 1342, Computer Sciences Department, University of Wisconsin, 1997.