分布式流媒体分发系统的设计与实现

金 杉,徐 佳,闪 烁
(中科华核电技术研究院有限公司信息技术中心,广东 深圳 518031)

摘 要:研究并实现了一个基于代理的应用层多播原型系统——分布式流媒体分发系统(DDSSM)。该系统按照树优先的方式构建多播路由,并采用分布式和主动式策略利用节点自身的邻居状态信息对多播树的拓扑结构进行动态维护。系统在实现上体现了面向对象的设计思想,达到了模块化的设计目标。在PlanetLab上的实验表明,DDSSM能够根据用户不同的约束条件构造满足QoS要求的应用层多播树,实现对实时音频流数据的分发,并且能够支持较大规模的多播应用,基本实现了为用户提供可扩展、高效率和高可靠的多播服务的目标。

关键词:应用层多播;路由协议;拓扑维护协议;流媒体
中图分类号:TN919.85 **文献标识码**:A **文章编号**:1673-629X(2010)11-0184-05

Design and Implementation of a Distributed Dissemination System for Streaming Media

JIN Shan, XU Jia, SHAN Shuo
(Information Technology Center, China Nuclear Power Technology Research Institute, Shenzhen 518031, China)

Abstract: A novel application layer multicast prototype system based on proxy called Distributed Dissemination System for Streaming Media (DDSSM) was studied and implemented. The multicast node constructs the route with the tree-first strategy, and maintains the topology using the state information of its neighbors with distributed and active strategy meanwhile. The implementation of the system materializes the thought of object-oriented, and achieves the object of modularization. The experiments on PlanetLab proved that, DDSSM can not only construct application layer multicast tree which met different QoS requirements, but also distribute real-time audio streaming data and support large-scale multicast applications. Basically speaking, DDSSM can provide users with scalable, efficient and reliable multicast services.

Key words: application layer multicast; routing protocol; topology maintenance protocol; stream-media

0 引 言

经过40余年的飞速发展,互联网已成为人们日常生产和生活中不可或缺的重要元素[1]。与此同时,伴随着多媒体技术的进步,以远程教育、网络电视、视频点播、在线医疗、网络多媒体广告等为代表的实时流媒体应用越来越受到人们的青睐。这类应用的一个共同特点是要求网络能够对多播通信服务提供支持。

传统的IP多播[2]由于一系列技术和经济方面的原因至今尚未在全球范围内获得大规模部署,于是研究人员提出了应用层多播的概念。按照组成多播网络节点类型的不同,应用层多播系统可以分为基于端系统(Host-based)的模型和基于代理服务器(Proxy-based)的模型两类。

前者直接将多播应用置于端系统中,端系统既是接受多播服务的客户端,同时也是构成多播网络的中间转发节点;后者将一些称为代理服务器的特殊节点按照一定的策略部署于网络当中,由它们首先在应用层构造一个实现多播转发功能的拓扑结构,其它的普通节点作为终端客户,通过接入这些代理服务器获得多播通信服务。这是一种典型的两层结构,其系统架构如图1所示。

文中以实时流媒体应用为背景,以提供高效、可靠的网络服务为目标,研究并实现了一个基于代理的应用层多播原型系统——分布式流媒体分发系统(Distributed Dissemination System for Streaming Media,简称DDSSM)。该系统采用分布式策略[3],按照树优先的方式构建多播路由,不仅能够满足带宽和延时上限等QoS条件,使网络延时和抖动性能达到近似最优化,还能避免集中式算法单点失效的问题[4];同时,系统采用主动式策略,利用节点自身维护的局部邻居状态信息

收稿日期:2010-03-12;**修回日期**:2010-06-07
作者简介:金 杉(1981-),男,硕士,工程师,CCF会员,研究方向为高性能网络,分布式系统。

对多播树的拓扑结构进行动态维护,具有较强的可扩展性和动态适应性,因而较适合部署在节点和链路状况动态变化的广域网环境中。

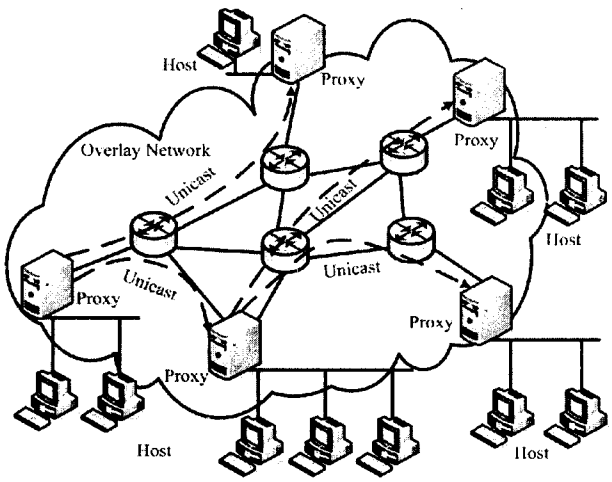

图 1　Proxy－based 系统结构示意图

1　DDSSM 协议机制

DDSSM 的设计目标是在广域网条件下为大规模实时流媒体多播应用提供稳定的服务,因此系统必须具有可缩放性、高效性和可靠性[5]三个方面的特征,从而能够有效处理网络状况的动态变化。由于系统实行分布式策略,所以文中仅以一个多播组为例介绍协议交互过程,其他各组内节点之间的交互均可在组与组之间互不干扰的情况下并发执行。

1.1　组管理协议

组管理协议分为组创建和组取消两个部分,因二者操作过程基本类似,故此处仅以组创建操作为例作详细介绍。为保持客户界面的友好性,可由拥有多播数据源的终端用户为多播组指定一个具有特殊含义的名称来标识该组,以方便用户阅读和使用。系统从用户界面上获得组名后,对该名称进行哈希运算(如 SHA－1[6]),生成一个 128 位长的全局唯一的组标识符,然后将组创建请求消息路由到节点标识符和组标识符最接近的后继节点,该节点记录下组标识符、源节点标识符以及该会话所要求的 QoS 参数,之后向数据源点返回组创建应答消息。

这里的节点标识符和组标识符最接近的节点称为该多播组的汇聚节点,不同于集中式系统中汇聚节点承担了组管理、路由计算、拓扑维护等复杂繁重的工作,DDSSM 中的汇聚节点仅仅作为组信息的维护者,而且对于不同组名的会话,汇聚节点将按照哈希函数的运算结果分布在不同的网络节点上,这无疑大大减轻了汇聚节点的负担,增强了系统的可扩展性和鲁棒性。

1.2　节点加入协议

当多播组创建成功后,其他网络节点可通过多种方式(例如离线通知方式)获得多播组名称,由系统根据这一名称产生组标识符后即开始加入操作。新节点先将组标识符放在组查询请求消息中路由到汇聚节点,后者在自身维护的组列表中找到对应的根节点地址和 QoS 参数,生成组查询应答消息返回给发起查询的节点,查询节点便获得了运行节点加入算法所需要的相关信息。该节点首先将根节点作为当前试探父节点向其发送查询请求消息,收到消息的节点将儿子节点列表放在应答消息中返回给发起查询的节点,新节点根据此消息和 QoS 约束条件运行 AOMRP[7]协议的节点加入算法,直至成功加入多播组,或者因操作失败而退出程序。

1.3　节点离开协议

如果一个端系统欲离开多播组,那么它根据自身在多播树上的位置作两种不同处理:若该用户为叶节点,则直接向其当前父节点发送离开请求消息,该父节点收到消息后,删除此儿子节点的路由选项,并返回应答消息;若该用户为非叶节点,则首先向所有的儿子节点发送离开请求消息,儿子节点收到此消息后,立即与其备用父节点交互执行重新加入的过程,待所有儿子节点均重新加入多播组后,此非叶节点最终执行离线操作。

1.4　拓扑维护协议

应用层的主机相对于路由器来说,除离线频率较高以外,其不稳定性也带来了网络拓扑结构的脆弱性,DDSSM 采用心跳模式进行拓扑维护,其心跳周期和超时阈值分别为 5 秒和 30 秒。邻居间通过周期性地交互保活请求消息和保活应答消息来刷新节点的存活状态。如果某节点的父节点应答超时,则该节点即刻与备用父节点交互完成重新加入过程,如果应答超时的是其儿子节点,则该节点仅仅将此儿子节点从路由表中删除即可。

DDSSM 采用 LTRP[8]协议计算备用父节点,即每一个非根非叶节点周期性地对当前父节点发送父兄集合查询消息并等待其应答,当前父节点收到消息后将自己和除发送请求的节点之外的其他儿子节点的资源使用情况放在父兄集合应答消息中返回给请求者,请求者据此信息计算其儿子节点的可能备用父节点集,并将此集合放在备用父节点探测请求消息中发送给儿子节点。相应的儿子节点运行备用父节点选择算法选取备用父节点,运行结束后返回备用父节点探测应答消息告知请求者对其他的儿子节点继续执行同样的触发动作。

2 DDSSM 系统模块及其实现

按照模块化和扩展性的要求，根据面向对象的设计思想，使用 Java 语言以及其他相关技术和工具实现了 DDSSM 原型系统。在系统所采用的 Proxy－based 体系结构下，参与应用的网络节点分为代理终端(Proxy)和用户终端(Host)两类。前者主要参与构建多播路由、处理低层协议交互；后者主要负责与终端用户交互、处理高层应用。

2.1 代理终端

代理终端是实现系统协议和算法的核心部分，它由组管理、拓扑维护、数据分发和结构化 P2P 路由(如 Pastry[9])等四个模块组成：

1)组管理模块(Group Manage)：实现对多播组的创建、节点加入和节点离开的控制，其中创建组(Group Create)、加入组(Group Join)和离开组(Group Leave)子模块分别对应 DDSSM 的组管理协议、节点加入协议和节点离开协议。

2)拓扑维护模块(Topology Maintain)：实现对多播树结构的维护操作，其中带宽/延时测量子模块(Bandwidth/Latency Measure)和多播树修复子模块(Tree Reconstruct)分别实现对网络拓扑进行性能测量及拓扑修复的功能。该模块与组管理模块一起对应用层多播路由表(Routing Table)进行维护和更新。

3)数据分发子模块(Data Distribute)：实现对流媒体数据一对多的转发。根据节点上的路由表信息，将多播数据在本地进行复制后转发给相应组中的下游节点。

4)P2P 路由子模块(Pastry Routing)：采用美国 Rice 大学开发的 Free Pastry 开源系统组织多播网络节点，首先由各节点运行 Pastry 路由协议构成环状结构，然后利用 Pastry 的应用程序接口传递和接收各种控制信息。

代理终端上运行的主要数据结构包括本地节点状态类 NodeStatus、路由表类 RoutingTable 和组维护类 GroupMaintain 等。

(1)本地节点状态类 NodeStatus。

```
public class NodeStatus {
  Float delayFromParent;
  Float delayFromRoot;
  Vector<Float> delayToChildren;
  Vector<int> resBandwidthOfPareAndBros;
  int maxBandwidth;
  int usedBandwidth;
}
```

其中，delayFromParent、delayFromRoot、delayToChildren 分别为该节点到父节点、根节点以及儿子节点的延时信息，resBandwidthOfPareAndBros 为该节点的父亲节点和兄弟节点的剩余带宽，maxBandwidth 和 usedBandwidth 分别为节点本身的最大带宽和已经使用的带宽，由此二者可以获得节点的剩余带宽。

(2)路由表类 RoutingTable。

```
public class RoutingTable {
  BasicAttribute parent;
  BasicAttribute backupParent;
  BasicAttribute root;
  Vector<BasicAttribute> children;
}
```

其中，parent 为当前父节点，backupParent 为备用父节点，root 为根节点，children 为儿子节点列表。BasicAttribute 是一个 Java 容器类，它以＜String，Id＞属性对的形式存储类对象，借此用每个节点的 IP 地址和 Pastry 节点标识符作为一组相关联属性来共同表示一个节点，极大地方便了编程实现。

(3)组维护类 GroupMaintain。

```
public class GroupMain {
  HashMap<Id,BasicAttribute> groupList;
}
```

其中，groupList 表示当本地节点作为某些多播组的汇聚节点时，保存在该节点上的组列表。这里采用 HashMap 结构，将组标识符和根节点信息作为表项来存储，方便了组的创建、查找和删除等操作。

2.2 用户终端

用户终端是直接与用户交互，接受终端命令的系统节点。其中，终端用户控制界面位于用户终端的上层，是人机交互的唯一接口。系统从该界面获得创建组、加入组、离开组的动作信息和媒体类型、格式、QoS 约束条件等参数信息，传递给下层的终端用户控制子模块和流媒体处理子模块进行动作响应处理和媒体格式转换等操作。该模块的主要数据结构包括媒体处理类 MediaProcessor 和 QoS 约束类 QosConstraint。

1) 媒体处理类 MediaProcessor。

```
public class MediaProcessor {
  DataSource audioDatasource;
  DataSource videoDatasource;
  Processor audioProcessor;
  Processor videoProcessor;
  TrackControl track[];
}
```

其中，audioDatasource、videoDatasource、audioProcessor、videoProcessor 和 track 分别为进行流媒体处理所需用到的媒体处理器、媒体数据源和媒体控制轨道

类对象。

2)QoS 约束类 QosConstraint。

```
QosConstraint {
  Float delayBound;
  int reqBandwidth;
  String mediaFormat;
}
```

其中,delayBound 和 reqBandwidth 分别为该组会话所规定的延时上限和所需求的带宽大小,mediaFormat 为规定的媒体格式。

2.3　流媒体数据的传输

DDSSM 对流媒体数据的处理和传输采用了 Sun 公司的 Java 媒体框架(Java Media Framework,简称 JMF)技术[10]。JMF 是对应 Java 2 平台标准版(J2SE)的应用程序接口(API),其强大的媒体工具包可以在 1.1.x 及以上版本的 Java 上运行。JMF 技术提供了先进的媒体处理能力,从而扩展了 Java 平台的功能,其体系架构如图 2 所示。

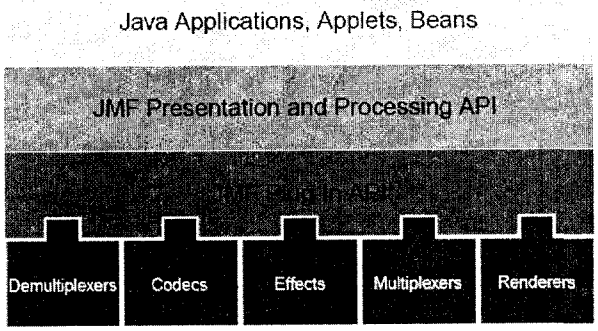

图 2　JMF 体系架构图

JMF 与 VCR 的运作模式极为相似,首先由数据源类(Data Source)将媒体流封装起来,然后交给播放器类或处理器类(Player 或 Processor),由后者对数据进行处理和控制。为了对网络流媒体传输提供支持,JMF 采用模块化的编程手段,在 javax.media.rtp、javax.media.rtp.event 和 javax.media.rtp.rtcp 等三个 package 中实现了 RTP/RTCP 协议,其端到端的传输过程如图 3 所示。在发送端,首先对多媒体数据文件(File)进行采集,形成本地格式(Format A)的数据源(DataSource),然后利用媒体处理器(Processor)对其进行格式转换,获得适合远程传输的网络流媒体格式(Format B)的数据源(DataSource),接下来使用 RTP 管理器(RTPManager)将数据源按照 RTP/RTCP 协议的规范发送至远程主机;在接收端,一方面采取相反的过程获得多媒体数据用于本地播放,同时根据需要对数据进行复制(clone),然后继续传输给下游节点。

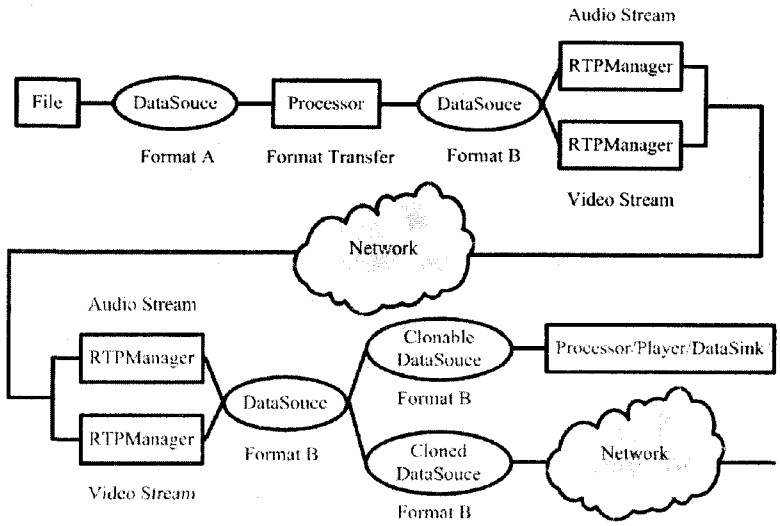

图 3　JMF 流媒体数据传输过程

3　测试与分析

按照实时流媒体多播应用的特点和要求,从实际部署的角度出发,利用开放式全球网络试验平台 PlanetLab[11],将 DDSSM 系统部署在位于亚洲、欧洲和北美洲的 15 个科研机构的节点上进行流媒体音频数据的分发,有关性能评价指标定义如下。

1) 根延时惩罚 D_{pr}(Delay Penalty from Root):

$$D_{pr}(v) = \text{delay}_T(r,v)/w(r,v) \quad (1)$$

式中,v、r 为多播节点,且 r 为根节点;$\text{delay}_T(r,v)$ 表示多播树 T 上从 r 到 v 的应用层多播路径延时;$w(r,v)$ 表示从 r 到 v 的直接单播路径延时。

2) 资源耗费压力 S_{ru}(Stress of Resource Usage):

$$S_{ru} = U_o/U_i \quad (2)$$

U_o 和 U_i 分别表示应用层多播和 IP 单播中最多的节点资源耗费,它们的定义如公式(3) 所示:

$$U = \max \sum_{v \in \text{Children}(u)} (b(u,v) \cdot w(u,v)) \quad (3)$$

u 表示参与应用的任一网络节点;v 表示 u 的儿子节点;$b(u,v)$ 表示 u 向 v 发送流媒体数据的带宽;$w(u,v)$ 表示 u 到 v 之间的单播路径延时。

图 4 显示了 DDSSM 构造的多播树中,所有节点根延时惩罚 D_{pr} 的最小值、最大值以及平均值随组规模的不同而变化的情况。首先,对于 D_{pr} 的平均值,随着组规模的扩大,多播树的层次明显增加,所以总体上来看,平均值逐渐增大。然而在实际网络中,由于节点间延时的绝对数值差异较大,因此延时惩罚的变化量并不与树的层次变化量同步。其次,从最大值变化情况来看,节点规模从 10 到 20 之间有个明显的跃升,20 以后的变化则趋于平缓。造成这种现象的主要原因有以下两个方面:第一,当节点数为 20 时,虽然它们都在同一

个区域内，但由于算法综合考虑了延时和度的约束，致使少数节点处于多播树的第一层以下，从而提高了 D_{pr}；第二，当超过 20 个节点后，其他的节点分别来自欧洲和北美两个地区，它们和根节点之间的延时主要受区域之间的延时绝对数值影响，故而将多播延时与 IP 单播延时相除后所得结果并没有显著的增幅。再次，从最小值变化情况来看，当节点均在同一地区时，多播树上的路径延时最小值总是等于 IP 路径延时值（1 跳的情况下），但是，当节点跨越不同地区时，均出现了多播路径延时最小值小于 IP 路径延时的情况，该结果实际上反映了广域网环境下 IP 路径并不总是优于应用层路径这一现象[12]。

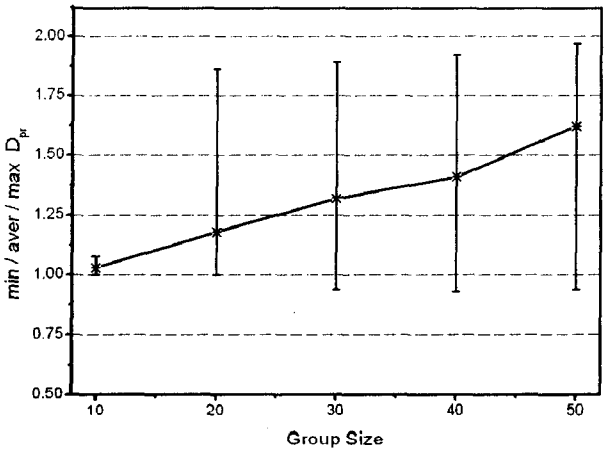

图 4 根延时惩罚 vs. 多播组规模

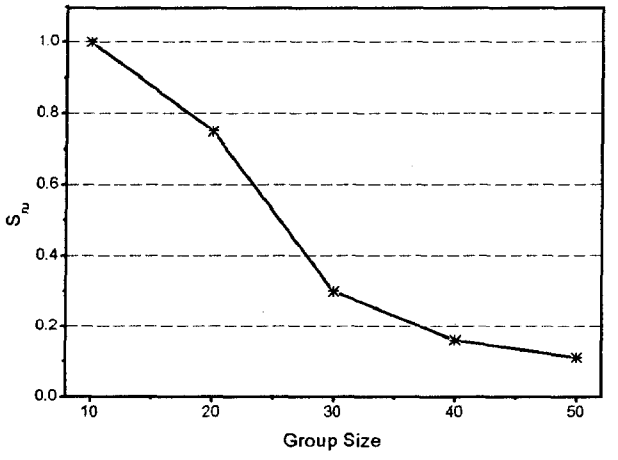

图 5 资源耗费压力 vs. 多播组规模

图 5 反映了资源耗费压力随组规模的变化情况。从图中可以看出，由于在 IP 单播中，耗费最多的节点始终是源点，所以随着多播组规模的增大，源点越来越成为整个系统的瓶颈；而在 DDSSM 中，网络资源的耗费被越来越多的中间节点分担，起到了很好的负载均衡作用，并且这一趋势还将随着节点数的增多而进一步发展，因此不难看出，DDSSM 系统具有较好的可扩展性，能够适合大规模的多播应用。

4 结束语

文中以实时流媒体多播应用为背景，研究并实现了一个基于分布式协议机制的流媒体分发系统——DDSSM。系统中实现了用于完成多播组管理、节点加入、节点退出、拓扑修复等控制功能的核心协议。为保证系统的可扩展性，支持大数据量的流媒体多播应用，系统采用控制信令和媒体数据信道相分离的原则实现报文传输，一方面利用结构化的 P2P 路由协议传递控制消息，另一方面使用 JMF 技术结合 RTP/RTCP 协议直接传输流媒体数据，从而优化了系统结构，提高了传输效率。实验结果表明，该系统能够根据不同的约束条件构造满足应用 QoS 要求的应用层多播树，实现对实时音频流数据的分发，并且能够支持较大规模的多播应用，基本实现了为用户提供可扩展、高效率和高可靠的多播服务的目标。

参考文献：

[1] Tanenbaum A S. 计算机网络[M]. 第 4 版. 潘爱民，译. 北京：清华大学出版社，2004.

[2] 包怀忠. IP 组播关键技术研究[J]. 计算机技术与发展，2009，19(4)：138－142.

[3] Tanenbaum A S，Steen M. 分布式系统原理与范型[M]. 第 2 版. 杨剑峰，常晓波，李 敏译. 北京：清华大学出版社，2008.

[4] 吴家皋. 覆盖多播路由的算法及协议研究综述[J]. 计算机科学，2007，34(6)：7－12.

[5] Almeoth K C，Ammar M H. Collection and modeling of the join/leave behavior of multicast group members in mbone[J]. IEEE Transactions on Communication，1997，45：224－229.

[6] FIPS 180－1. Secure hash standard[R]. Washington DC：National Institute of Standards and Technology，US Department of Commerce，1995.

[7] 金 杉，刘林峰，吴家皋. 一种新的自适应覆盖多播路由协议[J]. 东南大学学报：自然科学版，2007，37(3)：374－379.

[8] 金 杉，熊少学. 基于主动策略的覆盖多播拓扑维护协议[J]. 计算机工程，2009，35(17)：120－122.

[9] Rowstron A，Druschel P. Pastry：Scalable，distributed object location and routing for large－scale peer－to－peer systems[C]//In Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware). Heidelberg，Germany：[s. n.]，2001：329－350.

[10] Sun Corp. Java Media Framework[EB/OL]. 2010. http://java. sun. com/products/java－media/jmf/.

[11] Princeton University. PlanetLab[EB/OL]. 2010. http://www. planet－lab. org/.

[12] Anderson D，Balakrishnan H，Kasshock F，et al. Resilient Overlay Networks[EB/OL]. 2001. http://nms. lcs. mit. edu/projects/ron/.