

PeerCast 广播机制与维护算法的研究与改进

王延伟, 刘 璐

(山东大学信息科学与工程学院, 济南 250100)

摘 要: 在分析 PeerCast 广播机制和节点维护算法的基础上, 增加上游节点向下游节点广播状态信息的函数, 改进节点的加入和维护策略, 并在局域网内进行实验验证。实验结果表明, 改进的广播机制和节点维护算法能够减少节点断线重连的时间, 提高断线重连的成功率, 降低断线后对 YP 的访问次数, 减轻 YP 的压力。

关键词: 对等网络; 流媒体; PeerCast 广播机制

Research and Improvement of PeerCast's Broadcasting Mechanism and Maintaining Algorithm

WANG Yan-wei, LIU Ju

(School of Information Science and Engineering, Shandong University, Jinan 250100, China)

【Abstract】 This paper proposes an improved broadcasting mechanism and a node-maintaining algorithm based on the analysis of PeerCast broadcasting mechanism and node-maintaining algorithm. Experimental results show that the proposed scheme reduces the time of reconnection and improves the success rate of reconnection when disconnected in PeerCast. And it decreases the times of visiting YP (Yellow Page) after disconnection which reduces the stress of YP.

【Key words】 P2P; streaming media; PeerCast broadcasting mechanism

DOI: 10.3969/j.issn.1000-3428.2011.02.032

1 概述

随着 Internet 技术的飞速发展, 流媒体的应用越来越广泛。由于流媒体具有数据量大、持续时间长、占用带宽高等特点, 而传统 C/S 模型可扩展性差、网络资源利用率低, 因此, 基于传统 C/S 架构的流媒体服务器已经成为流媒体发展的瓶颈。针对该问题, 研究人员做了不同方面的探索, 得到了多种解决方案, 概略有以下 3 种: (1) 使用 CDN 分发网络^[1]。(2) 使用 IP 组播技术^[2]。(3) 使用 P2P 技术。CDN 分发网络的成本过高, IP 组播在可靠传输、流量控制、拥塞控制等多方面的固有限制使其难以广泛部署于流媒体系统^[3]。所以, 基于 P2P 技术的流媒体研究逐渐成为研究的热点^[4]。其中, 应用层组播的研究最广泛。应用层组播技术无须改变现有的网络拓扑结构, 只要在底层物理网络的基础上构建一个应用层覆盖网络。根据覆盖网类型的不同, 现有的 P2P 流媒体模型可分为 2 种: 树形和图形, 其中, 树形结构又分为单树和多树。P2P 网络是一个动态构建的自主网, 节点会随时加入或离开网络, 节点间的处理能力、存储能力和带宽存在异构性。且流媒体本身具有独特性质, 如数据存储量大、传输持续时间长、带宽占用高以及 QoS 要求高^[5]。

PeerCast 是一款被广泛研究的典型的 P2P 流媒体软件, 单树结构, “推模式” 数据分发, 但由于系统采用单源模式, 节点的断线或离开造成其下游节点失去源、重新寻找源、加入网络, 影响了收视(收听)效果。文献[6]采用混合结构(tree-mesh-hybrid)对 PeerCast 进行改进, 实现多源下载, 降低了父节点离开时对下游节点的影响, 但同时也增加了节点的维护复杂度。本文着重分析了 PeerCast 的广播机制和节点维护算法, 针对其存在的问题进行了改进。实验结果表明, 改进算法减少了节点在失去源节点后重新加入网络的时间,

改善了用户体验效果。

2 PeerCast 的广播机制

2.1 PeerCast 的系统结构

PeerCast 采用树形结构(单树), 以媒体源为根节点组成应用层组播树, 分发媒体数据。整个系统的节点分为 3 类: YP (Yellow Page), 广播者和普通节点。YP 管理并发布所有频道信息。系统的每一个普通节点都可以发布节目, 从而成为广播者。普通节点从广播者或其他上游节点下载音视频流, 同时也可以向下游节点上传已获得的音视频流。

2.2 PeerCast 的广播机制

PeerCast 节点间的控制信息和数据信息是通过 PCP (PeerCast Protocol) 传递的。在 PCP 协议中, 广播包是传递节点信息和频道信息的重要工具, 主要用于下游节点向上游节点传递自己的节点状态和广播者向 YP 传递自身状态和频道信息 2 种情况。广播包包括包头(与广播相关的内容)和负载(实际的具体内容)两部分。广播包采用泛洪机制, 通过 TTL (Time-To-Live) 来控制。PCP 协议中 PCP_BCST 是与广播相关的内容, 包括生存时间 (PCP_BCST_TTL)、跳数 (PCP_BCST_HOPS)、源 (PCP_BSCT_FROM)、广播范围 (PCP_BCST_GROUP) 等。实际的内容则主要包括主机信息 (PCP_HOST) 和频道信息 (PCP_CHAN)。

下游节点向上游节点广播状态信息是最常见的广播现

基金项目: 国家自然科学基金资助项目(60872024); 高等学校科技创新工程重大项目培育资金资助项目(708059); 山东省自然科学基金资助项目(Y2007G04)

作者简介: 王延伟(1983 -), 男, 硕士研究生, 主研方向: P2P 流媒体; 刘 璐, 教授、博士生导师

收稿日期: 2010-06-21 **E-mail:** wangyanweisdu@mail.sdu.edu.cn

象。下面结合此情况简要分析 PeerCast 中的广播机制。

2.2.1 广播包的形成

(1) 获取节点状态

```
numListeners=newLocalListeners;
numRelays =newLocalRelays;
isPlaying=ch->isPlaying();
fwState=servMgr->getFirewall();
lastUpdate=ctime;
```

(2) 用节点状态初始化 ChanHit 类对象 hit

```
hit.initLocal(numListeners,numRelays,ch->info.numSkips,ch->
info.getUptime(),isPlaying,oldp,newp);
hit.tracker=ch->isBroadcasting();
```

(3) 构建 AtomStream 类对象 atom, 并写入广播相关的内容和实际的内容

```
GnuID noID;
noID.clear();
atom.writeParent(PCP_BCST,7);//与广播相关的内容
atom.writeChar(PCP_BCST_GROUP,PCP_BCST_GROUP_
TRACKERS);
atom.writeChar(PCP_BCST_HOPS,0);
atom.writeChar(PCP_BCST_TTL,11);
atom.writeBytes(PCP_BCST_FROM,servMgr->sessionID.id,16);
atom.writeInt(PCP_BCST_VERSION,PCP_CLIENT_VERSION);
atom.writeBytes(PCP_BCST_CHANID,ch->info.id.id,16);
hit.writeAtoms(atom,noID);//具体的内容, 主机信息
pack.len=pmem.pos;
pack.type=ChanPacket::T_PCP;
```

2.2.2 广播包的发送

广播包的发送主要通过 ChanMgr::broadcastPacketUp() 和 ServMgr::broadcastPacket() 2 个函数实现。函数内部的主要调用关系如图 1 所示。

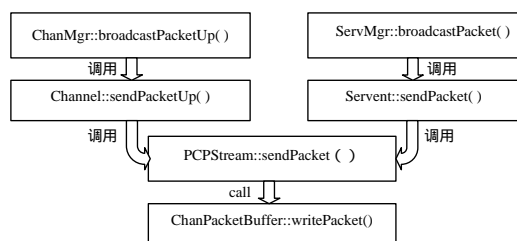


图 1 PeerCast 广播中的函数调用

图中, 上面的函数主要用于下游节点向上游节点更新本地信息和转发下游节点的广播包。下面的函数主要用于向指定类型的连接发送广播包。PCP 中的连接类型主要有以下 4 种:

- (1) Servent::T_RELAY, 普通的中继连接;
- (2) Servent::T_DIRECT, 与播放器的连接;
- (3) Servent::T_COUT, 向上与 Tracker/YP 的连接;
- (4) Servent::T_CIN, 向下与以此节点作为 Tracker/YP 的连接。

2.2.3 广播包的读取与转发

数据包的读取与转发是通过调用 PCPStream::readBroadcastAtoms() 函数实现的。节点根据 PCP 包中是否含有 PCP_BCST 字段可以很容易判断数据包是否是广播包。

(1) 读取。函数通过调用 AutoStream 类的 read() 函数、readChar() 函数和 readBytes() 函数循环读取 PCP 包与广播相关的内容和实际的内容, 做相应处理(TTL-1、HOPS+1 等)后存入 ptom, 以备转发时使用。

(2) 转发。如果包的目的地不是自己并且 TTL>0, 则转发此包。包的转发仍然根据 PCP_BCST_GROUP 限制的范围, 通过 2.2.2 节中提到的 2 个函数实现。

3 PeerCast 的节点维护算法及存在的问题

3.1 PeerCast 的节点维护算法

在 PeerCast 中, ChanHit 表示与客户端播放同一频道的节点, ChanHit 中保存了节点的 IP、端口等主机信息以及 numListeners、numRelays、numHops、time 等多种节点状态信息。ChanHitList 类则维护了一份 ChanHit 链表。收听同一频道的所有节点的信息均保存在 ChanHitList 中。

当用户第 1 次加入网络时, 向广播者 Tracker 索取数据, Tracker 此时记录下此用户的信息。如果 Tracker 还可以接收连接, 则吸收此节点作为其子节点; 反之, 则反馈给用户一份正在收看相同节目的节点链表。这份链表是用户获得的第 1 份 ChanHitList, 里面保存的节点均可能成为其上游节点, 用户会依次询问这些节点直到加入网络。在 2.2 节 PeerCast 的广播机制中已经提到, 节点在接收数据的同时, 定期向其上游节点广播自己的状态信息, 这就是 P2P 系统中的心跳(HeartBeating)机制, 也是 PeerCast 系统中检测节点是否有效的关键。上游节点接收到下游的广播包后, 解析广播包并发送包的节点通过“头插法”加入到自己的 ChanHitList 中。久未收到这种心跳信息的节点(PeerCast 中设为 90 s)被视为 DeadHit, 进而被 ChanHitList::clearDeadHits() 函数从 ChanHitList 中删除。

3.2 PeerCast 的节点维护算法中存在的问题

由上面的分析可知, 在 PeerCast 中, 节点加入到网络后就拥有了一份与自己收看(听)同一频道的 ChanHit 链表 ChanHitList。此链表会随时间的推移和用户的变化不断更新, 更新的动力源是组播树上的节点不断送来的广播包。网络中的下游节点频繁地向上游广播自己的状态, 上游节点不断地更新自己的 ChanHitList。由于更新链表采用在表头加入新节点的方法, 这样做虽然保证了链表前面的节点都是最近更新过的活跃节点, 但却使原来保存在链表中的 Tracker 和上游节点被新节点挤到后面, 进一步会被作为不活跃节点被删除掉, 因此链表最终保留了其直接下游节点和间接下游节点, 如图 2 所示。从而, 在出现上游节点断开而重新连接时, 只能向自己链表中的下游节点寻求新的连接, 造成环路问题。

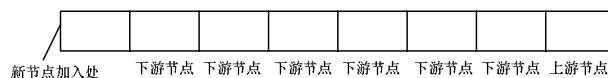


图 2 ChanHitList 示意图

4 PeerCast 的改进

针对上述问题, 本文提出一种改进的广播机制和节点维护算法, 其过程如下:

(1) 保留节点第 1 次加入网络时得到的链表, 调整 Tracker 到链表的尾部。这样就保存了 Tracker 和一定的上游节点。

(2) 改变原有链表的维护策略。将链表分为 3 部分: 上游节点段为第 1 部分; Tracker 始终保存在链表中上游节点段的后面, 为第 2 部分; 下游节点为第 3 部分。在断线重连时, 先试着连接保存在链表中的上游节点。无效后, 再连接 Tracker。这种方法是有效的, 因为它至少保留了 Tracker 在链表中, 在连接所有上游节点无效的情况下直接连接 Tracker, 既避免了徒劳地连接下游节点, 又省去了遍历链表无果后再向 YP 索要 Tracker 的麻烦, 节省了时间, 还减小了 YP 的负

担。改进后的链表见图 3。

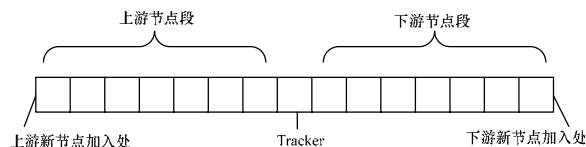


图 3 ChanHitList 改进后的示意图

(3)增加一个全局变量 `bool reconnect`，用于区别是否是断线后的重连，初始化为 `false`，断线后设置为 `true`；在 `ChanHitList` 类中增加一个变量 `bool lowernode`，初始化为 `false`，在收到下游节点的广播信息加入新节点时设置为 `true`，此变量用于区分节点是否是下游节点，避免在重连时连接下游节点。

(4)改变节点加入策略，上游节点加入到链表头部，调用 `addUpHit()` 函数，下游节点追加到链表尾部，调用 `addDownHit()` 函数。修改 `ChanHitList` 类的 `addHit()` 函数，分析包的来源后调用上面 2 个加入节点的函数之一。`addUpHit()` 函数与原 `addHit()` 函数相似，加入到链表头部。`addDownHit()` 函数需要在 `ChanHitList` 类中增加一个 `ChanHit` 类的指针，`*listend` 表示链表的尾部，帮助将下游节点追加到链表尾部。

(5)修改 `deadHit()` 函数，增加节点是否是 Tracker 的判断。Tracker 要始终保留在链表上游节点段与下游节点段之间，不删除。上游节点与下游节点采用不同的时间间隔标准，上游节点的时间间隔稍长。修改其他几个删除节点相关的函数，增加 Tracker 的判断，确保不删除 Tracker。

在广播机制方面，通过以下 3 点完善 PeerCast 中的心跳机制。

(1)增加 PCP 协议中的一个广播组 `PCP_BCST_GROUP_DOWN`，以限制向下游广播心跳信息的范围。增加主机字段 `PCP_HOST_FLAG1_UPDOWN`，区别节点是否是上游节点。向下游广播时对应位为真。

(2)根据 `PCP_HOST_FLAG1_UPDOWN` 是否置位，在 `ChannelStream::getStatus()` 函数中分别选择不同的广播组。在 `Servent::sendPCPChannel()` 中待发完一个数据包后，调用 `ChannelStream::updateStatus()`，广播心跳信息。

(3)修改广播包的读取和转发函数，增加对广播组为 `PCP_BCST_GROUP_DOWN` 时包的向下传递。

5 实验结果

对上述改进进行了多次对比实验，让网络中的节点随机地离开，并对节点断线后重新连接所需的平均时间和断线后对 YP 的平均访问次数做了比较。

实验环境如下：

局域网：带宽 100 Mb/s；

YP：CPU 奔腾 4 2.7 GHz，内存 1 GB，硬盘 120 GB；

Tracker 和普通节点：CPU AMD Turion 64 1.9 GHz，内存 512 MB，硬盘 120 GB；

节点数目：25；

文件格式：wmv 文件，251 MB；

编码器：Windows Media 编码器，编码速率为 548 Kb/s。

主要对比参数：断线重连所需的平均时间 t_{rec} ，断线重连时对 YP 的平均访问次数 cnt_{yp} 。

其表达式分别为：

$$t_{rec} = \frac{\sum_{i=1}^n t_{rec_i}}{n}, \quad cnt_{yp} = \frac{\sum_{i=1}^n cnt_{yp_i}}{n}$$

其中， t_{rec_i} 为第 i 个节点断线重连所需的时间； cnt_{yp_i} 为第 i 个节点断线重连时访问 YP 的次数； n 为断线重连的节点数。

图 4、图 5 为实验结果。由图 4 可以看出改进后的算法明显降低了断线重连所需要的时间。图 5 则显示改进后的 PeerCast 断线时减少了对 YP 访问，降低了 YP 的负担。其主要原因是本文的节点维护算法保存了一份结构清晰的链表，避免了节点在其上游节点断开后既浪费时间又只能连自己下游节点的情况。

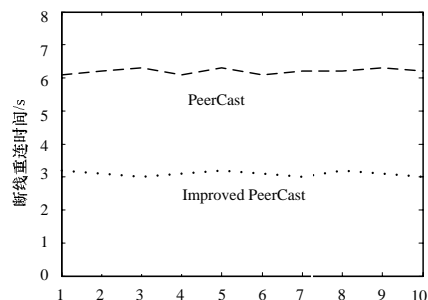


图 4 节点断线重连所需的平均时间

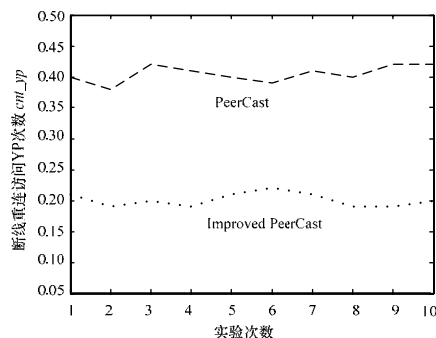


图 5 节点断线重连时访问 YP 的平均次数

6 结束语

本文在研究 PeerCast 广播机制和节点维护算法的基础上，提出了一种增强型广播机制和节点维护算法，并进行了实验对比。实验结果表明，改进后的算法降低了断线后重连所需的时间。

由于 PeerCast 基于单树型拓扑结构的固有限制，整个系统的稳定性和可扩展性有限。随着节点数和广播树高度迅速增加，播放延迟也不断延长，基于多树和基于网的流媒体研究成为热点。今后将在这方面做进一步研究。

参考文献

- [1] Jussi K. Internet Content Distribution[EB/OL]. (2009-10-03). eeweb.poly.edu/faculty/yongliu/docs/p2pvs.pdf.
- [2] Deering S. Host Extension for IP Multicasting[S]. RFC 1112, 1989-08.
- [3] Jannotti J, Gifford D K, Johnson K L. Overcast: Reliable Multicasting with an Overlay Network[C]//Proc. of USENIX Symposium on Operating System Design and Implementation. San Diego, USA: [s. n.], 2000.
- [4] Liu Yong, Yang Guo, Chao Liang. A Survey on Peer-to-Peer Video Streaming Systems[EB/OL]. (2009-10-23). www.springerlink.com/index/C62114G6G4863T32.pdf.
- [5] 胡平, 聂朋朋, 陆建德. 典型 P2P 流媒体模型及其关键技术[J]. 计算机工程, 2009, 35(3): 60-62.

(下转第 99 页)