一种基于衍生树的交互式 P2P 流媒体系统*

陈建忠[1,2]，徐天音[1,2]，李文中[1,2+]，陆桑璐[1,2]，陈道蓄[1,2]，Edward Chan[3]

[1](计算机软件新技术国家重点实验室（南京大学），江苏 南京 210093)

[2](南京大学 计算机科学与技术系，江苏 南京 210093)

[3](香港城市大学 电脑科学系，香港 九龙)

A Derivative Tree-Based P2P Interactive Streaming System

CHEN Jian-Zhong[1, 2], XU Tian-Yin[1, 2], LI Wen-Zhong[1, 2+], LU Sang-Lu[1, 2], CHEN Dao-Xu[1, 2], Edward CHAN[3]

[1](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

[3](Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong)

+ Corresponding author: Phn: +86-25-83597284, E-mail: lwz@dislab.nju.edu.cn, http://www.nju.edu.cn

Abstract: With the wide spread of peer-to-peer (P2P) applications, P2P technology has been shown to be a promising solution to deploy large-scale media streaming systems. Supporting asynchronous user requests and VCR-like interactions in media streaming systems nevertheless remains a challenging task. In this paper, we introduce a derivative tree-based P2P framework for supporting interactive streaming applications. The proposed scheme includes Distributed Discovery Service for resource location and Derivative Tree-based caching structure for dissemination topology maintenance. By introducing the derivative tree into the system, the impact of dynamic node join/departure could be dramatically reduced. With the assistance of Distributed Hash Table (DHT), the overhead of the resource searching, service reconstruction and overlay maintenance could be alleviated to an acceptable level. Extensive simulations show that the derived tree-based strategy performs well. The overhead of interactive operations in our scheme can be reduced by more than 50% compared to existing P2P media streaming systems.

Key words: Derivative Tree; P2P Media Streaming; User Interactivity; Split Cache; Distributed Discovery Service;

摘 要: 随着各种 P2P 应用的普及和发展，P2P 技术在部署大规模流媒体系统上表现出良好的应用前景。然而，在实时流媒体系统中支持异步的用户请求和交互式操作仍然是一个极具挑战的问题。本文提出了一种基于衍生树的分布式 P2P 系统框架，以支持交互式流媒体应用。该系统利用分布式发现服务来进行资源定位，并通过基于衍生树的缓存结构来维护数据传输拓扑。基于衍生树的缓存管理策略可以显著地降低节点动态加入和退出等交

* Supported by the National Natural Science Foundation of China under Grant No. 60803111, No. 90718031, No. 60721002 (国家自然科学基金); the National High-Tech Research and Development Program of China under Grant No. 2006AA01Z199 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No. 2009CB320705 (国家重点基础研究发展规划(973)), and the grant from City University of Hong Kong (No.7002115).

互操作的开销。另外，通过使用分布式散列表（DHT）来维护会话，可以用较低的代价来实现资源查找，服务重构和拓扑维护等任务。仿真实验表明，与现有的 P2P 流媒体系统相比，我们提出的基于衍生树的系统具有良好的性能，其用户交互操作的开销可以降低超过 50%。

关键词: 衍生树; P2P 流媒体; 交互性; 分割缓存; 分布式发现服务;

中图法分类号: **** **文献标识码:** **

1 引言

长期以来，在 Internet 上提供流媒体服务一直是一项极具挑战性的任务，缘于流媒体服务对传输带宽、传输时延、以及传输质量等近乎苛刻的高要求。延时、抖动等因素都会影响流媒体的观看质量，导致用户满意度的降低。过去，基于 IP 组播和基于内容复制（如 C/S 模式和 Server/Proxy 模式）的解决方案都曾被开发并且使用，试图满足网络流媒体服务的特殊要求[1][2][3]。然而这些方法都存在其固有的劣势：IP 组播受限于现有的网络基础设施无法有效支持，并且其高昂的部署成本导致低可扩展性；内容复制则受制于服务器的性能瓶颈，无法在大规模的网络环境中有效应用。

对等网络（Peer-to-Peer, P2P）技术的兴起，给互联网应用模式带来了巨大的影响，已成为大规模分布式数据共享的基础设施，并被广泛应用于流媒体数据分发[4][5]。基于 P2P 的流媒体技术有两方面的主要优势：首先，该技术既不用特殊网络基础设施（如互联网组播路由器）的支持，又无需改变现有的流媒体传输协议和媒体服务器系统的架构，只需增加新的功能模块，因此性价比较高并且容易部署；其次，通过使用该技术，用户在下载的同时，还把媒体流上传给其他用户，易于扩大系统规模：更多的需求会带来更多的资源[6]。目前，对 P2P 流媒体的研究工作主要集中在对直播方式的 P2P 流媒体的研究[7][8][9]。

近期，使用 P2P 方式来支持交互式流媒体服务成为新的研究热点[10][11]。与直播方式的 P2P 流媒体系统不同，交互式流媒体允许用户的交互请求，如执行快进、暂停和跳转等 VCR（Video Cassette Recorder）操作。在 P2P 流媒体系统中提供高效的交互功能是一项极具挑战的任务。首先，流媒体数据的实时传输的问题（较长的等待时延和大量消耗的带宽）在提供交互功能的系统中仍然存在；其次，虽然对所有用户的交互行为的统计呈一定的分布，但单个用户的交互操作具有很强的随意性；再次，仅考虑某一特定的交互操作，触发操作的节点要快速定位能提供交互请求的源节点，对于之前向其请求数据的节点，也要因为它的退出快速重定向数据流，以不影响播放的连续性。在直播系统的解决方案中，由于系统存在很强的同步性，节点很容易从邻居节点处获取后续的数据。然而，在允许异步请求的交互式系统中，这种关系并不成立。另外，系统的节点响应时延和故障恢复也是设计系统时必须考虑的问题。

针对 P2P 的网络环境和交互操作的特性，本文提出了一种基于衍生树的交互式 P2P 流媒体服务系统。针对用户的交互操作导致的系统的动态性，我们设计了一种称为衍生树（Derivative Tree）的缓存结构组织方案来支持用户的交互请求和 VCR 操作的快速响应。衍生树利用交叠缓存来扩展组播树，可以对其进行可控扩展。为了对用户请求快速定位，信息查找借助 DHT（Distributed Hash Table，分布式散列表）机制，将部分节点组织成一个结构化的网络，可以在 $O(\log\ n)$的时间内定位流媒体资源。通过控制衍生树的高度则可以降低数据请求和传输时延，从而降低交互操作的开销。另外，衍生树结构和常用的组播树不同，它的显著的好处是：一个节点的交互操作不会对其它节点（尤其是其下游节点）产生大的影响。仿真实验结果表明，使用基于衍生树的交互式 P2P 流媒体服务可以极大地减轻服务器的负载，提高系统的可扩展性，具有较低的控制开销，有效地支持了用户的交互式请求和 VCR 操作。

2 相关工作

作为流媒体的一个很自然的应用方式，交互式功能的需求很早就被提出了，流媒体点播（VoD，Video-on-Demand）就是交互式流媒体的一个典型应用。早前，在集中式的框架下，从综合业务数字网（ISDN）[12]到 ATM 网[13]，针对 VoD 应用，研究者们提出了一系列分析[14]和解决方案[15]。

当计算模式从 C/S 模式步入 P2P 时代后，P2P 流媒体服务的研究得到了广泛关注。然而，大部分研究集中于直播方式的流媒体 [7][8][9]，只有部分研究工作着眼于交互式流媒体服务[16][17][18]。P2VoD[17]中提出了“代”（Generation）的概念，所有请求相同流媒体文件的节点组成一个会话。同代中节点的到达时间间隔可以用它们实际的缓存内容差来表示。P2VoD 主要解决了节点的异步请求问题；然而，它不能支持节点的交互请求。当交互请求频繁时，系统的性能会下降得很快；此外，在同一代中只有最先到达的节点才能充分利用缓存，其余节点都不能充分利用固定长度缓存，只好使用可变缓存策略。RINDY[18]是一个支持用户交互式操作的流媒体点播服务系统，它使用基于 Gossip 方式的无结构覆盖拓扑来支持由用户交互式请求导致的数据流重定向。基于 Gossip 的无结构覆盖拓扑有其固有的优点，如数据重定向的高效率等，但同时也承受着维护和控制所必须的大量的通信开销。

然而，上述的大部分交互式解决方案中，客户请求的提交是集中式的，需要维护有全局会话（或节点）信息的单元存在；还有一部分基于代理（Proxy）的方式虽能实现“轻量级”的交互操作，但缓存策略大多要依赖于统计的用户请求模式，不能很好支持随机跳转。并且，用户从代理缓存接收数据还是集中式的方式，无法适应大规模交互式应用的需求。不同于上述工作，本文提出了一种完全分布式的交互式 P2P 流媒体系统，实现了流媒体播放点的快速定位和随机跳转，可以支持大规模用户的交互请求，具有良好的性能和可扩展性。

3 系统设计

3.1 系统模型

流媒体服务系统由媒体服务器（Media Server）、会话环（Session Ring）和会话（Session）组成（如图 1 所示）。媒体源文件初始保存在媒体服务器上。所有的媒体流都从媒体服务器上初始化，经过在整个分布式 P2P 覆盖网络中的分发，最后到达终端用户。媒体服务器是普通的内容服务器，仅提供基本的内容分发服务。在我们的系统中，媒体服务器本身不提供任何交互功能，所有的交互操作都由 P2P 覆盖网络中各个节点之间的协同来实现。具有相近播放位置的节点形成一个会话。每个会话是由一组节点形成的数据分发树，媒体流数据从根节点开始，以树状层次的形式向下层节点传递。会话环连接所有会话的根节点，形成一个环状结构。为了实现快速的信息查找，会话环使用 DHT 来组织资源信息，对于用户请求可以实现分布式的快速资源定位。当有新节点加入或是节点发生交互操作进行数据流重定向时，可以通过会话环提供的查询服务，实现快速的资源重定位和高效的服务重构。

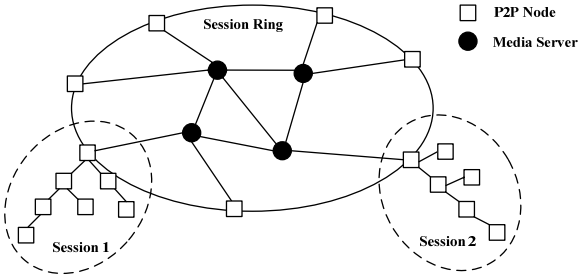


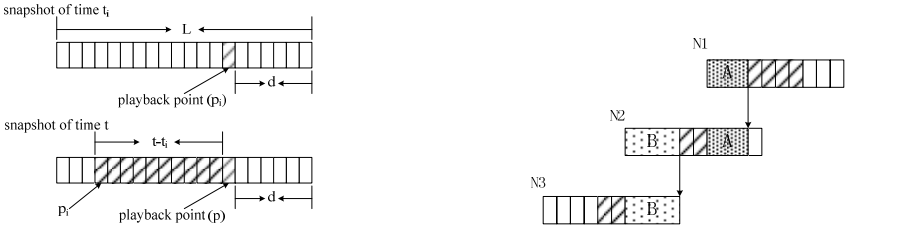
Fig 1. Overview of the system framework

图 1 系统体系结构图

3.2 缓存设计

系统中每个节点都维护一个滑动窗口（又叫缓存队列），用来保存最近播放的数据，以提供后续节点的数据请求，提升系统的服务能力。假设节点能缓存 L 单位时间长度的流媒体数据，节点当前的播放位置是 P（相对于流媒体数据的开始，也以时间单位表示），播放位置距缓存队列的出口的距离为 d，则节点缓存的数据量为 L-d 个数据单元，缓存的数据范围（Data Range）为：$[\max\{0,P-(L-d)\}, P]$。随着时间的推移，节点会持续缓存最近接收的数据，相应的较早的数据会因缓存容量有限而滑出缓存队列。同时，节点缓存的数据可以给任何请求点位于数据范围内的其他节点提供服务。从直观上理解，当节点缓存容量增加时，系统的可扩展性

也会相应增加。



(a) The time variation of the buffer contents (b) The relationship of the buffer contents

(a) 缓存随时间的变化情况 (b) 缓存内容之间的关系

Fig 2. The buffer contents of peers (clients)

图 2 客户端缓存

当某个节点要请求某时刻的数据或是由于发生交互操作要进行数据流重定向时，它会将请求提交到能提供此服务的节点，并与其建立数据连接。为了迅速定位流媒体数据，我们提供了分布式发现机制（Distributed Discovery Service, DDS）。DDS 要求每个节点 i 在本地维护一个五元组信息$(a_i, L_i, P_i, t_i, d_i)$。其中，$a_i$ 是节点对应的 IP 地址，L_i 是节点的本地缓存容量，P_i 是节点刚加入系统时或是由于发生交互操作而恢复播放后的播放点，t_i 是节点开始接受数据时的时间，d_i 是缓存中的播放点距缓存前端的距离。假设影片的总长度为 T，在节点加入系统后的任一时间点 t，节点中缓存的数据范围为（如图 2(a)所示）：

$$[\min\{(P_i+\max\{0,t-t_i-(L_i-d_i)\}),T-(L_i-d_i)\},\min\{P_i+t-t_i,T\}]$$

此时若有其它节点的数据请求属于该数据范围，并且节点有充足的带宽，则此节点可以提供流媒体服务。

在交互式的流媒体系统中，有直接数据联系的节点之间的缓存有交叠，而无直接数据联系的节点之间的缓存内容没有任何关系，这与直播方式的流媒体系统不同。如图 2(b)所示，节点 N1 与 N2 的缓存交叠区域为 A，N2 与 N3 有交叠区域 B，但 N1 与 N3 的缓存内容没有交叠部分。

4 分布式发现服务

分布式发现服务由会话环提供，实现了快速的资源定位和高效的服务重构。为了支持分布式发现服务，会话环中的节点使用 DHT（Distributed Hash Table，分布式散列表）技术自组织成一个结构化的 P2P 网络，并在其上共同构建和维护分布式索引（Distributed Index）以支持高效的资源查询。

4.1 会话环

会话环使用 DHT 技术将节点组织成一个结构化 P2P 网络。本文使用 Chord[19]做为典型的 DHT 技术，其余不同的 DHT 技术可以类似地使用。会话环中的每一个节点都被分配以唯一的散列值，称为节点的 ID。为了尽可能避免由于系统的动态性导致的 DHT 维护开销的增加，我们选择仅仅将系统中的一部分节点（各个会话的根节点）组织进会话环。假设整个会话环的 ID 空间是 2^m，那么任何一个加入会话环的节点都具有一个$[0,2^{m-1}]$之间的 ID，并维护一个有 m 项的路由表。在节点 n 的路由表中，第 i 项指向节点 s，s 是 ID 在$[n+2^{i-1},n+2^i)\ (1\le i\le m)$区间内（模 2^m）的第一个节点，即在顺时针方向到 n 距离至少为 2^{i-1} 的第一个节点。

基于 DHT 的 P2P 网络可以实现数据对象的快速精确定位，但是在交互式应用中需要一种有效的范围查询机制：只要某个节点的缓存中含有请求的数据位置，该节点就应该属于请求节点的*候选集*。为实现这样的分布式发现机制，我们在会话环上引入了分布式索引，为查询请求建立相应的拓扑。而系统的数据传输拓扑是通过查询请求后才建立的，独立的数据传输拓扑结构可以应功能需求而进行单独的调整。

在具有 n 个节点的会话环中，每个会话根节点查询数据对象（即其它会话中所能提供的流媒体数据）需要的路由跳数为 $O(\log n)$。DHT 的查询开销 $T_{dht\_search}\in O(\log n)$。

4.2 分布式索引

4.2.1 扩展基于 DHT 的 P2P 网络

为了组织分布式索引，我们需要对基于 DHT 的 P2P 网络进行相应的扩展，提供一些由上层调用的 API，如下所述。nodeID =DHTInit()：让节点加入现有的 DHT 网络，并初始化节点的一些状态信息，返回节点的 ID；route(msg, key)：将消息 msg 路由到由 key 指定的节点上，这是对 DHT 路由方法的一个封装；send(msg, nodeAddr)：将消息 msg 发送到 nodeAddr 所指定的节点上，这是点对点方式的消息传递；forward 是在节点接收到由其他节点传来的 route 消息时才触发的；deliver 是在消息被发送到指定节点时触发的。分布式索引就是通过它们来实现索引创建、元数据信息注册以及查询消息的发布。这样建立的索引是完全分布式的：每个节点所作的任何决策都只基于本地的一些信息，所有的节点都具有同样的功能。这些良好的分布式特性都继承自 P2P 模型。另外，在 DHT 网络上，节点间传送的消息类型包括 CREATE、JOIN、LEAVE 和 MULTICAST。CREATE 消息触发索引结构的建立，JOIN 消息在节点加入索引结构时送出，LEAVE 消息是节点在删除索引时发送的，MULTICAST 是节点向分布式索引结构发布查询消息时调用的。另外，msg.group 是消息的组标识，msg.source 是生成此消息的源节点标识，msg.type 是指上述的几种消息类型。本文中，我们仅给出 forward 和 deliver 函数的伪代码（见图 3）。

```
1. forward(msg, key, nextIP)
2.   switch (msg.type)
3.   JOIN: if !(msg.group ∈ groups){
4.           groups ∪= msg.group;
5.           route(msg,msg.group);
6.         }
7.         groups[msg.group].children ∪= msg.source;
8.         nextIP = NULL;

1. deliver(msg, key)
2.   switch(msg.type)
3.   CREATE:      groups ∪= msg.group;
4.   JOIN:        groups[msg.group].children ∪= msg.source;
5.   MULTICAST:  ∀ node in groups[msg.group].children;
6.                send(msg, node);
7.                if ( ismemberOf(msg.group) )
8.                   MessageHandler(msg.group, msg);
9.   LEAVE:       groups[msg.group].children =
                     groups[msg.group].children - msg.source;
10.               if (|groups[msg.group].children| = 0)
11.                  send(msg, groups[msg.group].parent);
```

Fig 3. The pseudo-code of the forward operation and deliver operation

图 3 forward 操作和 deliver 操作的伪代码

4.2.2 分布式索引的建立

当有一个节点要求加入某个特定的索引结构时，它向索引入口节点发送一个加入消息 route(JOIN, groupID)，以创建该节点到入口节点的路径。当某个节点收到其它节点的路由消息时，如果它不是入口节点，需要对这个消息进行转发处理：首先，检查自己是否已经是这个组的一个索引节点了。若是，表明已经存在从入口节点到当前索引节点的路径了，将消息中包含的索引项信息加入到当前节点的索引中，并终止此消息的转发过程；若此节点不是该组的一个索引节点，则需要将当前节点标识为 groupID 对应的索引结构的一个索引节点，将收到的消息中的索引信息加入到当前节点，并将 JOIN 消息路由到下一个节点。

当节点要退出某个索引时，节点会发送 LEAVE 消息给它的直接父节点，此消息会沿着路由路径递归地向索引的入口节点方向发送，一旦某个节点中还有活跃的索引条目，则过程终止。

即使在成员规模很大时，索引结构的维护仍然是高效的。组成员分布在整个 ID 空间中，底层 DHT 良好的随机特性使得整个索引结构具有良好的平衡性，不会造成消息的转发负载集中于少部分节点，索引结构也不会集中于很小一部分节点上。和集中式的索引策略相比，分布式索引可以避免单点失效，也可以比仅用服务器维护更多的索引，具有很强的可扩展性。所有的索引项的建立都是分布式完成的。特别地，入口节点不需要维护所有的索引，这更像是一个多级索引机制。

4.3 基于分布式索引的发现服务

4.3.1 查询过程

分布式索引可以让用户快速地定位可以提供服务的数据源。例如，当有新节点加入系统或是节点发生交互操作时，分布式索引服务可以帮助节点查找那些可以提供服务的节点。基于结构化 DHT 网络的分布式索引结构可以在不需要媒体服务器的辅助下动态地发现和定位组内的其他用户。属于同一个组的成员具有相同的观看兴趣（这里可以理解为观看同一部影片）。用户可以通过加入索引结构把本节点对应的五元组信息(a_i, L_i, P_i, t_i, d_i)更新到对应的索引节点上去，并在退出的时候删除对应节点上的索引项。如果节点上此时索引为

空，则对应此节点应更新为非索引节点。

查询操作是分布式索引的一个主要功能，它允许用户在整个组中以某种查询标准定位相关的节点，例如通过查询可以找到所有能提供流媒体接入的会话集。假设用户要基于某个索引进行查询，首先获取相应的groupID（媒体文件的散列值），然后该用户节点向索引的入口节点发送查询消息，我们把节点在路由消息到根节点时，路径上经过的第一个属于该索引结构的索引节点称为查询的*起始节点*，由起始节点开始发起广度优先搜索。更确切地说，当一个节点接受到查询消息时，首先检测它自己是不是查询起始节点（这可以通过在查询消息中设置一个状态位来实现），然后发起搜索。起始节点首先查看维护的本地索引中有没有满足查询要求的节点，如果存在合适的会话，就把会话的根节点相关信息返回给请求节点，然后将查询消息转发给其邻居索引节点，并继续类似查询过程。

4.3.2　查询开销分析

假设会话环的 ID 空间为 n，由 DHT 的性质可知整个分布式目录树的高度 $h \le \log n$。假设会话环中的节点数为 n，节点个数的概率密度函数是 $p(n)$，则在会话环上分布式索引查找到所有候选结果的平均时间开销为：

$$T_{ring\_search} = \sum_{i=1}^{n}(p(i)\log n) = \log\prod_{i=1}^{n}(n^{p(i)}) \in O(\log n) \quad (1)$$

5　会话的组织

分布式索引利用结构化 P2P 的特性，能为节点快速定位数据请求，但若所有的节点都以基于 DHT 的方式建立索引，当系统动态性增强时，结构化网络的更新将变得异常复杂。所以，本文只将一部分节点（直接从媒体服务器获取流媒体数据的节点）组织成索引结构。当节点通过分布式发现机制得到一个候选的会话后，节点要对这个会话请求加入。如果采用传统的构建组播树方案，不易对树的结构进行控制。在本文中，我们引入了一个基于衍生树的缓存设计方案（如图 4），以指导整个数据拓扑的建立。衍生树是一种扩展的组播树，节点间的数据请求受结构化的约束，其中交叠缓存的设计也使得整个树的高度是可控的。

5.1 缓存数据请求策略

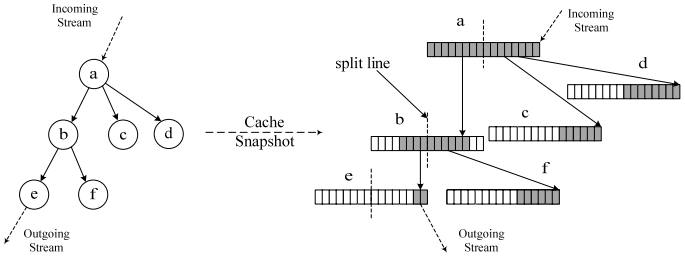


Fig 4.　A derivative tree based session and its corresponding buffer view

图 4　基于衍生树的局部会话和对应的缓存视图

首先，我们先给出衍生树中一些定义和建树规则：

分割线（*Split line*）：缓存的分割线是对整个滑动窗口的划分，节点的缓存被分割线分成两部分（假设被分成相等的两部分）。当另一个节点发起数据请求时，如果请求的数据在分割线的右半部分缓存范围内，请求节点可以成为原节点的右子女，如果请求的数据在分割线的左半部分缓存范围内，则是原节点的左子女。

我们定义如下建树规则：（1）一个节点可以有多个右子女，但是只能有一个左子女。（2）只有左子女可以衍生出左子树，右子女只能作为叶节点，不能为其他节点提供流服务。

内部节点（*Internal node*）：即树的非叶节点。它一定是某一个节点的左子女（根节点除外），能给其它节点提供流服务。会话的根节点是第一个内部节点。数据请求点位于一个内部节点的分割线左侧的子节点是内部节点，若有多个节点请求位于这个区域，将有节点替换发生。

外部节点（*External node*）：即树的叶节点。外部节点的数据请求点位于上层内部节点的分割线右侧，外

部节点不能继续给其他节点提供数据服务，一个内部节点可以有多个外部子节点。

外部节点和内部节点是会话范畴的，它标识了节点的角色和特性。由于系统的动态性，节点在同一个会话中的角色可以发生转变。每个会话的根节点同时属于上层的会话环。为方便问题的讨论，我们不对节点的播放位置作限定，即节点的播放位置不一定要处于缓存的前端。但正常播放时，播放位置在缓存中的相对位置是固定的。这里，我们假设播放点距缓存前端的距离为 d，缓存长度都固定为 L。为支持会话创建，会话中每个节点都要维护一些信息：分割线在缓存中的相对位置 $\alpha = 1/2$，分割线右侧的外部节点列表信息 EX_set，每一项包含了这样的信息(_ex_node, P)，以及它左侧的内部节点信息(_in_node, P)，P 为对应节点的当前播放位置，只要节点之间数据的传输关系存在，则发送节点的 P 和接收节点的 P 的相对位置是固定的。这个设计方案的形式化描述如图 5 所示，假设内部节点 A 当前缓存的流媒体内容为$[C_A, P_A]$，C_A 和 P_A 的值是对应的数据块在整个流媒体文件中的时间编号。若 A 有一个内部节点 B，则需满足 $C_A \le P_B < P_A - (L/2 - d_A)$。若 B 作为内部节点刚加入 A，为了向其它节点提供流媒体服务，B 需要含有$[C_A, P_B]$间的数据，故 $d_B = P_A - (L/2 - d_A) - P_B$。若 E 是 A 的一个外部节点，则：$P_E \in [\max(P_A - (L/2 - d_A), C_A), P_A]$，$d_E = d_A + (P_A - P_E)$。

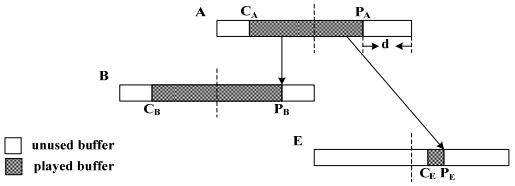
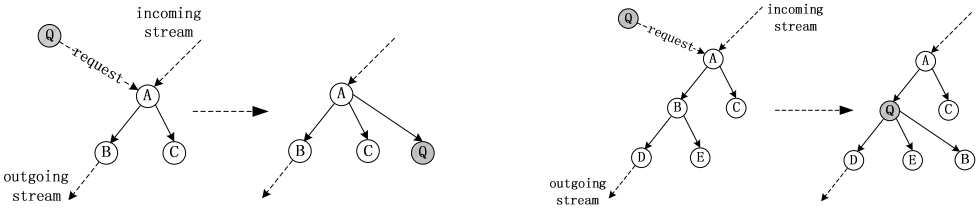


Fig 5. Demonstration of the buffer scheme

图 5 缓存策略示例

5.2 节点加入

5.2.1 节点加入算法



(a) Add as an internal node

(a) 加入成为内部节点

(b) Add as an external node

(b) 加入成为外部节点

Fig 6. A “push-down” join process in a derivative tree

图 6 下推式的衍生树节点加入过程

当有新节点加入时，原来的会话会发生相应的变动。节点加入是一个“下推式”的过程：从根节点开始，将新加入的节点的请求位置与衍生树的内部节点逐个向下匹配，直到找到一个合适的位置加入该节点。节点加入不能违背衍生树的建树规则。如图 6 所示，假设一个新节点 Q 请求加入会话，请求的数据块为 P_Q，Q 从会话的根节点 A 开始向下查找。如果节点 A 的缓存中含有数据块 P_Q，且 P_Q 在节点 A 的缓存中位于分割线的右侧，则 Q 作为 A 的外部节点加入会话，直接从 A 接收数据，如图 6(a)所示。如果请求位置 P_Q 在节点 A 的分割线左侧，需要比较 P_Q 与 A 的内部节点 B 的请求位置 P_B 的关系，有两种情况：1）如果 $P_Q > P_B$，为了不违背衍生树规则，需要用请求节点 Q 替换内部节点 B，并通知节点 B 的子节点从新节点 Q 处接受数据，同时将原内部节点 B 变成新节点 Q 的外部节点，如图 6(b)所示；2）如果 $P_Q \le P_B$，则请求节点 Q 继续从节点 B 查找，递归完成上述的操作。易见，会话内的递归查找的时间开销 $T_{session\_search} \in O(H)$，$H$ 为会话的衍生树的树高，即内部节点的数量。当加入请求失败时，可以重新启动会话环上的分布式索引查找，继续加入过程。若几次尝试未果，节点新建一个会话并加入会话环。如果超过了加入的容忍时延，则向返回失败消息。

5.2.2　节点的替换操作

当发生图 6(b)的情况时，会引发节点的替换过程。如图 7 所示，当新的请求节点 *Q* 加入衍生树并成为节点 *A* 的子节点时，由于 *Q* 的请求点先于原内部节点 *B*，为了不违背衍生树的构造规则，要将 *B* 替换成 *Q*，使 *Q* 成为 *A* 的新的内部节点。为了能让下游节点顺利播放，新加入的节点 *Q* 要预取从 C_B 到 P_Q 之间的一段数据（直接从媒体服务器或者专用的服务器预取）。当 *Q* 预取完相关的数据后，再将 *B* 的子节点的数据连接重定向到 *Q*，同时让原内部节点 *B* 也成为 *Q* 的一个外部节点。

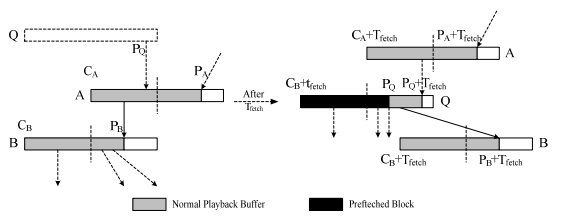


Fig 7.　Node replacement in a derivative tree

图 7　衍生树节点替换过程

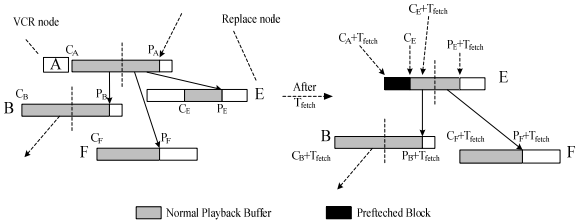


Fig 8.　A “lift-up” node recovery in a derivative tree

图 8　提升式的衍生树节点恢复过程

替换的预取过程需要经过一段时间。假设预取$[C_B, P_Q]$数据段的时延为 $T_{fetch}(C_B, P_Q)$，则经过 $T_{fetch}(C_B, P_Q)$ 时间后，节点 *Q* 中的缓存范围应该是$[C_B + T_{fetch}, P_Q + T_{fetch}]$。假设播放速率为 *S*，预取带宽为 *BW*，则在预取结束后应该满足 $T_{fetch}(C_B, P_Q) \times BW = (P_Q - (C_B + T_{fetch}(C_B, P_Q))) \times S$。解这个方程得，

$$T_{fetch}(C_B, P_Q) = \frac{|P_Q - C_B| \times S}{BW + S} \tag{2}$$

由于 $P_Q - C_B < L$，因此预取时延 T_{fetch} 的值不超过 $L \times S / (BW+S)$，其中，*L* 为节点缓存的大小。

节点替换的时间开销主要由预取时延和重定向时间等构成。其中，预取时延占节点替换时间的主要部分，重定向时间等与之相比可以看作一个常数 C。由此，节点替换的时间开销为：

$$T_{join\_sub} = T_{fetch} + C$$

5.2.3　节点的加入开销

完整的节点加入过程由两个部分组成：首先，请求加入的节点在基于 DHT 的会话环上找到可以提供所需流媒体的会话；然后，该节点通过 5.2.1 节介绍的加入算法成为基于衍生树的缓存结构中的节点。故节点整个加入过程的时间开销主要由会话环上的查找开销、会话内的查找开销和替换开销三部分组成，即：

$$T_{join} = T_{ring\_search} + T_{session\_search} + T_{join\_sub}$$

其中，$T_{ring\_search} \in O(\log n)$，*n* 为会话环中根节点的数量（见 4.3.2 节式(1)），$T_{session\_search} \in O(H)$，*H* 为衍生树的树高（见 5.2.1 节），而 T_{join_sub} 在 5.2.2 节中分析。

5.3　节点的退出

在交互式流媒体中，随时发生的交互操作触发频繁的节点退出。当交互操作发生时，节点可能要退出当前的会话，重定位请求的数据流。这个过程称为*数据重定向*。数据重定向造成的一个严重后果是，在数据传输拓扑中，从这个节点接收数据的所有下游节点都会受到影响。在已有的大多数交互式流媒体系统（[17][18]）中，节点在加入组播树过程中主动创建候选父节点集，并定期地主动更新，在获取数据失败时，及时向候选集中的节点发起数据请求。传统的方法并不能很好地解决由于数据重定向对下游节点的服务造成影响的问题，服务恢复的代价很大。

在基于衍生树的会话组织框架下，发生交互操作的节点会主动为下游节点调度数据流，使得下游节点几乎不受到由交互操作导致的节点退出的影响，仍然能正常提供流媒体服务，极大降低了交互式操作的开销。节点的退出操作主要分为以下 3 种情况考虑：

1)　若发出交互请求的是外部节点，则其不会对其它节点造成任何影响，节点自身启动重定向过程即可；

2) 若发出交互请求的是内部节点，且该内部节点存在外部子节点，则可以从其直接外部子节点中选择一个播放点位置最前的子节点作替换，替换后仍不能违反会话组织的构造规则。与节点加入过程的下推式（push-down）的替换不同，节点发生交互操作时，替换是提升式（lift-up）的。如图 8 所示，当节点 *A* 发生交互操作时，它的内部节点为 *B*，外部节点有 *E* 和 *F*，*E* 的请求点最前，被提升到上一层来替换 *A* 节点。若该节点是当前会话的根节点，则还需将选中的外部子节点加入会话环；

3) 若发出交互请求的是内部节点，并且该节点没有外部子节点，则其下游的内部子节点自动形成一个新的独立会话，并且退出节点的内部子节点成为该会话的根节点。新的根节点加入会话环，并从媒体服务器上接收流媒体数据。

下面我们结合这三种情况，对节点退出的开销进行分析。

假设当前会话中节点的个数为 *N*，树高为 *H*。对于情况（1），由于只有 *H* 个内部节点，故一个节点是外部节点的概率为 (*N*-*H*) / *N*，外部节点的退出对其他节点的服务不会产生任何影响，因此，这部分开销可以看作一个很小的常数。对于情况（3），当要退出的内部节点没有外部节点时，它的左子树会形成一个新的会话加入会话环，由于一个内部节点没有外部节点的概率很小，这部分的开销也可以近似看作常数。因此，节点退出的开销主要取决于情况（2）。当情况（2）发生时，交互节点需进行替换。如图 8 所示，假设 *A* 的缓存为[C_A, P_A]，*E* 的缓存为[C_E, P_E]，如果 $C_E \le P_E$，则可以直接用 *E* 来替换 *A*，其开销为常数；如果 $C_E > C_A$，为了使原先从 *A* 处直接接收数据的节点能连续地从新的父节点 E 处接收数据，*E* 要预取 C_A 到 C_E 之间的数据，预取的时延为 $T_{fetch}(C_A, C_E)$，可由公式(2)来计算。由此可见，情况（2）的开销不超过 $T_{fetch}(C_A, C_E)$，它发生的概率不超过 *H* / *N*。综上所述，节点退出的开销满足：

$$T_{leave} \le \frac{H}{N} \times T_{fetch}(C_A, C_E) + C$$

其中，C 是一个常数，$T_{fetch}(C_A, C_E) = |C_E - C_A| \times \frac{S}{BW + S} < L \times \frac{S}{BW + S}$ 。

5.4 交互操作

当节点发生交互请求时，数据的重定向过程是从会话内扩展到会话间的。交互节点首先查看当前所在的会话是否能够满足其交互请求。若满足，则从父节点往上到根节点查询加入点；若不满足，则需要重新执行加入过程。注意到当节点发生交互操作时，其退出操作和数据重定向是可以并行的。

用户的交互请求可以通过以下三种方式被满足，也导致了三种不同的交互请求开销：

1) 用户的交互请求可以在当前的会话内被满足，此时交互请求的时间开销为：

$$T_{interactive} = \max(T_{leave},\ T_{session\_search}) + T_{join\_sub}$$

2) 用户的交互请求无法在当前的会话内被满足，但可以被当前基于 DHT 的会话环中其它会话满足。这种情况下，交互节点退出当前会话，加入其它会话。此时交互请求的时间开销为：

$$T_{interactive} = \max(T_{leave},\ T_{session\_search} + T_{ring\_search}) + T_{join\_sub}$$

3) 用户的交互请求无法由会话环中的任何一个会话满足，只能通过媒体服务器得到。这种情况下，交互节点在完成会话环的查询后，在会话环上新建一个会话，并作为该会话的根节点。此时开销为：

$$T_{interactive} = \max(T_{leave},\ T_{session\_search} + T_{ring\_search}) + T_{create\_session}$$

6 性能评估

6.1 实验的建立

我们通过仿真实验来评估基于衍生树的交互式 P2P 流媒体服务系统的性能。系统中除了媒体服务器，所有客户节点具有相同的属性（如缓存和带宽等）。实验的网络拓扑使用 GT-ITM[20] 拓扑生成器来构造，网络拓扑符合 Transit-Stub 模型，网络节点数量为 5000 个。假设流媒体的播放速率为 256Kbps，流媒体文件的长度 T=1600 秒。每个客户节点的默认缓存大小 L 为 100 个数据单元，即每个客户节点可以缓存 100 秒的流媒

体片段。每个客户节点默认可以为 K 个节点提供流媒体服务。客户节点的到达满足泊松过程。为获取多种情况下的模拟实验结果，将节点的请求到达率 λ、K 和 L 等都设为可调的参数。

在仿真实验中，我们以可扩展性、节点加入开销、交互操作的开销等作为性能参数，来评估基于衍生数的 P2P 交互式流媒体服务的性能。我们将基于衍生树的流媒体服务系统与另一个交互式流媒体系统 RINDY[18] 进行性能比较，实验结果表明，我们的系统在节点平均控制开销、交互节点的重定向开销、以及交互操作对系统的影响等几方面都表现出更好的性能。

6.2 服务器负载

我们通过检查媒体服务器所提供的会话数目来衡量媒体服务器的负载情况。我们在设定节点请求到达率 $\lambda=5$ 和影片时长 T=1600 确定的情况下，记录媒体服务器需要维护的会话数目与 K（允许接入的子节点数）和 L（缓存大小）的关系。由图 9 中的曲线变化趋势可以看出，当客户端缓存很小时，增加客户端缓存大小可以极大地减小服务器负载。例如，当 L/T 从 1%增加到 6%，服务器负载分别降低了 34%（K=2），24%（K=4）和 39%（K=6）。当缓存/影片大于 6%时，会话的数量趋向于常值，表明再增加节点缓存并不会提高系统的可扩展性。这是因为过小的缓存导致覆盖网络中存在的媒体文件的资源不足，而当覆盖网络中某个媒体文件的缓存足够多时，增加缓存大小不会显著提高性能。此外，客户节点允许接入的子节点数量也对服务器负载产生重要的影响。当缓存大小相同时，每个客户节点允许接入的子节点数目越大，平均会话数目将会越少。在图 9 中可见，如果每个客户节点只为一个外部节点提供流媒体服务时（即 K=2 时），系统会话的数量比较大，使得服务器的负载远大于 K=4 至 K=6 的情况。

6.3 节点加入开销

当客户节点请求加入系统时，节点通过分布式索引获取可以提供服务的会话列表，请求节点同时向这些会话发出加入请求。当节点请求成功时，加入开销为发出请求到接收到第一个请求成功消息的时延；当节点请求失败时，加入开销为接受到最后一个失败消息的时延。这里，我们以查询的次数作为加入开销的单位。

实验结果如图 10 所示：当 K 增大时，外部节点增多，故总的会话数量将会减少，查询开销得以降低。此外，较大的缓存容量使节点的服务能力提高，也使节点所衍生的内部节点链的长度变长，从而导致查询次数的相应增加。但是，这种情况下查询次数的增加很小，从图 10 中可见，查询次数的增量不超过 4 跳。

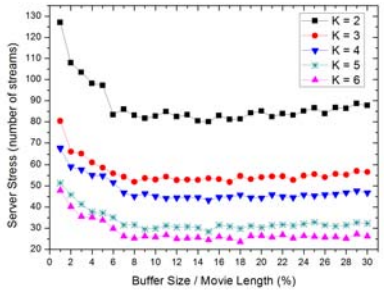


Fig 9. Server stress against varying buffer size

图 9 服务器负载和缓存大小的关系

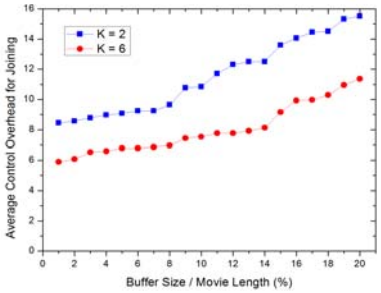


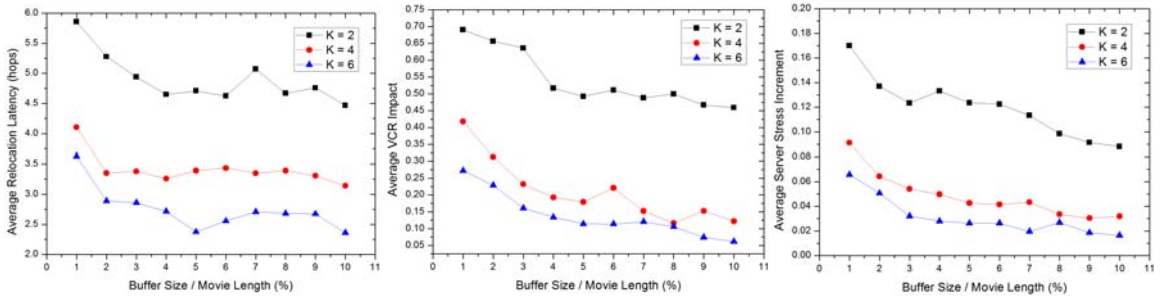
Fig 10. Average joining latency against varying buffer size

图 10 节点平均加入时延和缓存大小时间的关系

6.4 交互操作影响

当节点发生交互操作时，不仅要使交互节点进行快速的数据重定向，而且必须尽快恢复其它受影响节点的正常的流媒体播放。在实验中，我们设定节点的到达率为 $\lambda=5$，当系统的节点个数到达 200 时，10%的节点产生随机跳转。根据 5.3 节所述，如果交互节点是外部节点，或是连有外部节点的内部节点，则系统中的其它节点基本不受影响；如果交互节点没有外部节点连接时，则所有以其为根的子树都会受影响。假设子树会直接向服务器请求新的数据通道，则交互操作的影响可以用建立新的数据通道和子树节点数的积来表示（本文余下部分中的交互操作的影响也均使用这个度量）。

图 11(a)和 11(b)记录了对于不同的 K 值，交互操作的平均重定向时延和平均受影响节点数量。从图中可以看出，重定向开销和交互操作对于不同节点的缓存大小的影响是基本一致的。此外，交互操作的影响在 K 值增大时会有所降低。例如，当 K=6，L/T=6%时，交互操作的平均重定向开销为 2.55，小于 K=4 时的 3.43 和 K=2 时的 4.62；类似的，平均受影响节点数量为 0.114，小于 K=4 时的 0.221 和 K=1 时的 0.551。图 11(c)记录了平均每次交互操作导致的会话数目的平均增量。当 K 值较小时，由于节点提供的数据流数目有限，所以每次交互产生新的会话的概率比较大。而当带宽足够大，使会话数目趋于稳定时，由交互操作导致的会话数目的增量也趋于稳定。并且，从图中可见，会话数目的增量随着缓存变大而逐渐变小。较大的缓存意味着每个客户节点有更大的概率为其它节点提供服务，而无需从媒体服务器上新建会话连接。



(a) Average relocation latency (b) Average impact nodes (c) Average server stress increment

(a) 交互操作的平均重定向时延 (b) 平均受影响节点数量 (c) 平均服务器负载的增量

Fig. 11 VCR impact against varying buffer size

图 11 VCR 操作的影响和缓存大小时间的关系

6.5 与其他P2P流媒体系统的比较

我们将基于衍生树的流媒体服务系统与现有的交互式 P2P 流媒体系统 RINDY[18]进行性能比较。RINDY 使用基于 Gossip 方式的无结构覆盖拓扑，来支持用户的交互式请求。选择 RINDY 作为比较对象，缘于 RINDY 和基于衍生树的交互式 P2P 流媒体服务相似的缓存管理策略：每个节点仅仅关注当前缓存中的媒体内容，以支持对其它节点的内容分发（文献[21][22]等都通过使用额外的硬盘存储，来支持用户的交互式请求）。

我们就节点平均控制开销、交互操作的请求个数、请求频度带来的额外开销、以及缓存大小对交互操作的影响四个方面来对基于衍生树的交互式 P2P 流媒体服务和 RINDY 的性能进行比较。实验结果表明，基于衍生树的交互式流媒体服务有其固有的优势：在对用户的交互请求良好支持的同时，仅仅产生较小的额外控制开销，并且节点的交互操作仅对系统整体性能产生非常小的影响。

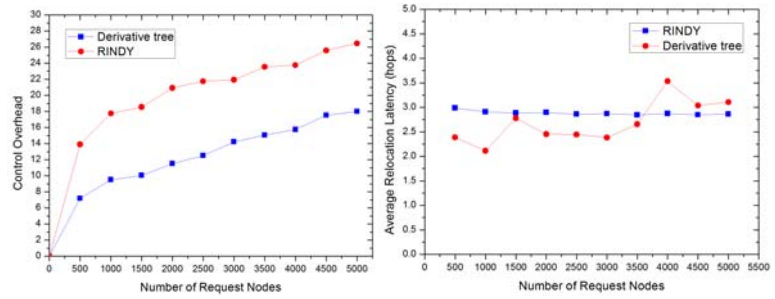
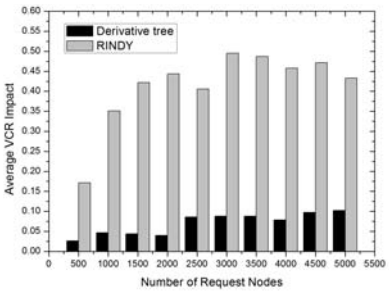


Fig 12. Comparison for control overhead against varying node scale

图 12 对于不同节点规模的平均控制开销的比较

(a) Average redirection latency (a) 交互节点的平均重定向时延



(b) Average impacted nodes (b) 平均受影响节点数量

Fig. 13 Comparison for VCR impact against varying node scale

图 13 对于不同节点规模的 VCR 操作的影响的比较

6.5.1 交互操作的控制开销

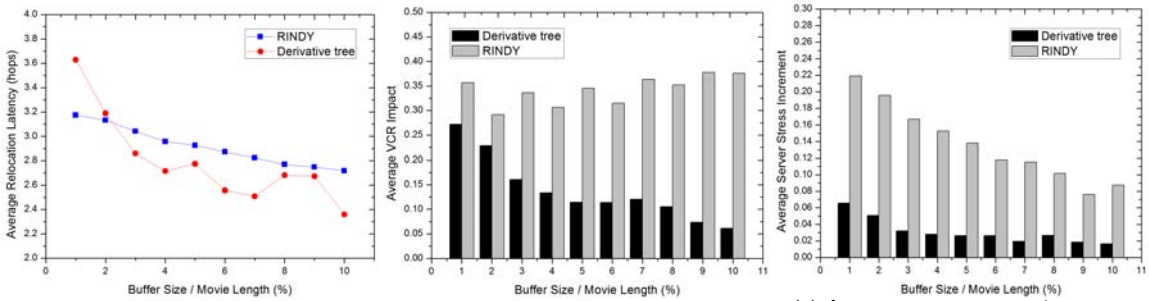
我们首先比较两个系统中，部分节点产生交互操作请求所带来的控制开销。这里，控制开销由所发送的控制消息（control message）的个数来度量。我们记录了在一定的时间段内，基于衍生树的交互式流媒体服务和 RINDY 系统的节点平均发送的控制消息个数。

图 12 显示了当系统中 10%的节点随机产生交互操作请求时，随着请求数量的增加，节点的平均的控制开销的变化情况。由图可见，随着请求数量的增加，两种系统的控制开销都增大。但是，RINDY 系统的控制开销的增长远大于基于衍生树的系统。例如，当请求数量为 1000 时，RINDY 系统的用于控制的平均消息数为 18，衍生树系统为 9，而当请求数量为 5000 时，RINDY 系统的平均控制消息数量增加到 26，而衍生树系统只增加到 17，后者只有前者的 65%。这是因为 RINDY 对节点的组织基于 Gossip 方式，节点间完全通过相互发送消息彼此联系。当某个节点产生交互操作时，节点间必须发送大量的消息，来广播这个交互事件的发生，以便相关节点更新交互节点的当前状态。而在基于衍生树的交互式流媒体服务中，播放位置相近的节点构成一颗衍生树。衍生树结构在创建过程中规定了节点之间的相互关系，节点退出当前位置无需通知大量的其它节点，故导致的开销非常小。在基于衍生树的交互式流媒体服务中，交互操作的主要开销为数据重定向开销。

6.5.2 交互操作的影响

图 13 比较了在不同节点规模下，在基于衍生树的交互式流媒体服务和 RINDY 中，节点产生的交互请求的重定向开销和对系统整体性能的影响。从图 13(a)中可以看出，基于衍生树的交互式流媒体服务中交互节点的重定向开销要大于 RINDY，但当会话的数量稳定后，这个重定向开销也趋于稳定。图 13(b)验证了基于衍生树的交互式流媒体服务的优势，即交互操作对系统的整体性能的影响很小。正如之前的分析，仅当产生交互请求的节点本身是内部节点，并且该节点没有挂载外部节点时，系统的整体性能才会产生实质的影响（这种情况的概率非常小，已经在之前的部分说明）。当节点个数为 5000 时，基于衍生树的系统中，由交互操作受到的影响的节点数量仅为 RINDY 系统的 30%。

图 14 比较了对于不同的缓存大小，基于衍生树的交互式流媒体服务和 RINDY 中，节点产生的交互请求的重定向开销和对系统整体性能的影响。从图 14(a)可见，在 RINDY 系统中，缓存大小的改变对交互操作的平均重定向开销影响很小。这是由于 RINDY 中的节点使用 Gossip 方式来了解其它节点的缓存情况。而在基于衍生树的交互式流媒体服务中，随着缓存尺寸的增大，相应建立的会话数量减少，导致重定向时间变小。并通过实验可以看到，当节点的“缓存/影片比”到达一定阈值(约为 2%)时，交互节点的平均重定向开销将小于 RINDY 中具有相同缓存大小的节点的重定向开销。图 14(b)记录了对于不同的缓存大小，交互操作对于系统整体性能的影响。RINDY 中使用 Gossip 的通信方式，交互操作发生后，受影响的节点将从邻居节点表中选择一个拥有所需数据流的节点，并重定向数据通道。而在基于衍生树的交互式流媒体服务中，通过选择外部节点进行替换操作，可以减少由于节点的退出对下游节点造成的影响。从实验结果可见，基于衍生树的交互式流媒体服务中交互操作对系统的影响要明显小于 RINDY。



(a) Average redirection latency (b) Average impacted nodes (c) Average server stress increment

(a) 交互节点的平均重定向时延 (b) 平均受影响节点数 (c) 平均服务器负载增量

Fig. 14 Comparison for VCR impact against varying buffer size

图 14 对于不同缓存大小的 VCR 操作的影响的比较

缓存越大，节点产生交互操作后，交互节点本身以及受影响的节点到服务器上取数据的可能性就越小。对于基于衍生树的交互式 P2P 流媒体服务，大尺寸的缓存意味着节点所能提供的数据流的范围变大，故服务器的负载将变小。缓存大小与交互操作对服务器负载的影响的关系见图 14(c)。从实验结果中可以看出，相比 RINDY，基于衍生树的交互式流媒体服务具有更小的服务器负载，即更好的系统可扩展性。当“缓存/影片比”取 6%时（此时服务器负载趋于稳定），基于衍生树的系统中，交互操作的影响仅为 RINDY 系统的 30%。

7 结论与进一步的工作

在现有的流媒体系统中，媒体服务器承担了很重的数据发送负载，并且对用户交互请求的支持较差。针对这些问题,本文提出了一种完全分布式的交互式流媒体系统框架——基于衍生树的交互式 P2P 流媒体服务。基于衍生树的交互式 P2P 流媒体服务使用基于 DHT 的完全分布式覆盖拓扑，并在此基础上为交互功能提出了一种全新的缓存请求方案，以此来降低节点交互操作的响应延时和因交互操作给系统带来的影响。通过理论分析和模拟实验，我们证明了该系统的确能够有效解决在当前的交互式流媒体服务中用户的交互请求时延过大，以及由交互节点跳转导致其它节点的观看质量受到影响等问题。

我们未来的工作考虑以下三个方面：首先，在适合基于衍生树缓存管理基础上，准备引入缓存预取策略，并设置专有的用于预取的缓存段。预期策略已被证明可以有效地增强节点对突发性数据包丢失的适应能力，平滑流媒体的播放效果。其次，我们考虑通过对用户的交互行为的统计和追踪分析，设计一种适应用户交互习惯的缓存替换策略，这不仅能够极大改善系统的性能，而且将使系统能够更好地支持用户的交互行为。最后，我们考虑在现有的系统中提供 QoS 保障，使得用户的观看质量可以得到保证。

References:

[1] Deering S, Cheriton D. Multicast routing in datagram internetworks and extended LANs. ACM Transaction on Computer Systems (TOCS). ACM Press. 1990. 8(2): 85-110.

[2] Diot C, Levine BN, Lyles B, Kassem H, Balensiefen D. Deployment issues for IP multicast service and architecture. IEEE Network. IEEE Communications Society. 2000. 14(1): 78-88.

[3] Sen S, Rexford J, Towsley D. Proxy prefix caching for multimedia streams. In: Proc. of the 8th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1999). New York: IEEE Computer Society and IEEE Communication Society. 1999. 3: 1320-1319.

[4] PPLive website. 2009. http://www.pplive.com

[5] Joost website. 2009. http://www.joost.com

[6] Liu J, Rao SG, Li B, Zhang H. Opportunities and challenges of peer-to-peer internet video broadcast. In Proc. of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications. IEEE Communications Society. 2008. 96(1): 11-24.

[7] Sripanidkulchai K, Ganjam A, Maggs B, Zhang H. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In: Proc. of the Annual Conf. of the Special Interest Group on Data Communication (SIGCOMM 2004). Portland: ACM Special Interest Group on Data Communication. 2004. 34(4): 107-120.

[8] Zhang X, Liu J, Li B, Yum T. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In Proc. of the 24th Annual Conference Sponsored by IEEE Communications Society (INFOCOM 2005). Miami: IEEE Communications Society. 2005.3: 2102-2111.

[9] Castro M, Druschel P, Kermarrec AM, Nandi A, Rowstron A, Singh A. SplitStream: high-bandwidth multicast in cooperative environments. In: Proc. of the 19th ACM Symposium on Operation Systems Principles (SOSP 2003). Session: Overlay & peer –to-peer networks. New York: ACM Press. 2003. 298-313.

[10] Huang C, Li J, Ross KW. Can internet video-on-demand be profitable? In: Proc. of the Annual Conf. of the Special Interest Group on Data Communication (SIGCOMM 2007). Tokyo: ACM Special Interest Group on Data Communication. 2007. 133-144.

[11] Huang Y, Fu T, Chiu DM, Lui J, Huang C. Challenges, design and analysis of a large-scale p2p-vod system. In: Proc. of the Annual Conf. of the Special Interest Group on Data Communication (SIGCOMM 2008). Seattle: ACM Special Interest Group on Data Communication. 2008. 275-388.

[12] Hoshi T, Mori K, Takahashi Y, Nakayama Y, Ishizaki T. B-ISDN multimedia communication and collaboration platform using advanced video workstations to support cooperative work. IEEE Journal on Selected Areas in Communications (JSAC). IEEE Communications Society. 1992. 10(9): 1403-1412.

[13] McManus JM, Ross, KW. Video-on-demand over ATM: constant-rate transmission and transport. IEEE Journal on Selected Areas in Communications (JSAC). IEEE Communications Society. 1996. 14(6): 1087-1098.

[14] Little T, Venkatesh D. Prospects for interactive video-on-demand. IEEE Multimedia. IEEE Computer Society. 1994. 1(3): 14-24.

[15] Chang Y, Coggins D, Pitt D, Skellern D, Thapar M, Venkatraman C. An open-systems approach to video-on-demand. IEEE Communication Magazine. IEEE Communications Society1994. 32(5): 68-80.

[16] Cui Y, Li B, Nahrstedt K. oStream: asynchronous streaming multicast in application-layer overlay networks. IEEE Journal on Selected Areas in Communications (JSAC). IEEE Communications Society. 2004. 22(1): 91-106.

[17] Do TT, Hua KA, Tantaoui M. P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: Proc. of the IEEE International Conference on Communications (ICC 2004). Paris: IEEE Communications Society. 2004. 3: 1467-1472.

[18] B. Cheng, H. Jin, and X. Liao. Supporting VCR functions in P2P VoD services using ring-assisted overlays. In: Proc. of the IEEE International Conference on Communications (ICC 2007). Glasgow: IEEE Communications Society. 2007. 1698-1703.

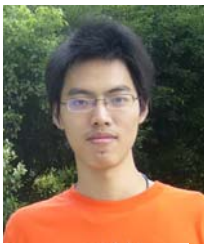
[19] Stoica I, Morris R, Karger D, Kaashoek MF, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proc. of the Annual Conf. of the Special Interest Group on Data Communication (SIGCOMM 2001). San Diego: ACM Special Interest Group on Data Communication. 2001. 149-160.

[20] Zegura E, Calvert K, Bhattacharjee S. How to model an internetwork. In: Proc. of the 5th Annual Joint Conference on Computer and Communications Societies (INFOCOM 1996). San Francisco: IEEE Computer Society and IEEE Communications Society. 1996. 2: 594-602.

[21] Yiu WPK, Jin X, Chan SHG. VMesh: distributed segment storage for peer-to-peer interactive video streaming. IEEE Journal on Selected Areas in Communications (JSAC). IEEE Communications Society. 2006. 25(9): 1717-1731.

[22] Xu C, Muntean GM, Fallon E, Hanley A. A balanced tree-based strategy for unstructured media distribution in p2p networks. In: Proc. of the International Conference on Communications (ICC 2008). Beijing: IEEE Communications Society. 2008. 1797-1801.

陈建忠(1983－),男,硕士生,主要研究领域为应用层组播技术,对等网络中的数据管理和传输技术,普适计算中的交互式流媒体服务.



徐天音(1985－),男,硕士生,主要研究领域为多媒体通信技术,对等网络和覆盖网络,服务计算,无线传感器网络.

李文中(1979－),男,博士,讲师,主要研究领域为 P2P 计算,分布式数据管理和无线网络.

陆桑璐(1970－),女,博士,教授,博士生导师,主要研究领域为分布计算与并行处理,无线传感器网络,普适计算.

陈道蓄(1947－),男,教授,博士生导师,CCF 高级会员,主要研究领域为分布式并行计算.

Edward Chan, 男,博士,副教授,主要研究领域为数据管理和实时系统.