OXVoD:一个基于分层结构的 P2P 视频点播系统

钱碧伟[1],谢冬青[1,2],周再红[3],熊 伟[3]
QIAN Bi-wei[1],XIE Dong-qing[1,2],ZHOU Zai-hong[3],XIONG Wei[3]

1.湖南大学 软件学院,长沙 410082
2.广州大学 计算机科学与教育软件学院,广州 510006
3.湖南大学 计算机与通信学院,长沙 410082
1.School of Software,Hunan University,Changsha 410082,China
2.School of Computer Science and Educational Software,Guangzhou University,Guangzhou 510006,China
3.College of Computer and Communication,Hunan University,Changsha 410082,China
E-mail:qianbw@qq.com

QIAN Bi-wei,XIE Dong-qing,ZHOU Zai-hong,et al.OXVoD:A video-on-demand system based on P2P delamination structure.Computer Engineering and Applications,2010,46(7):203-207.

Abstract: Expansibility and high degree of playing continuity are key points to large-scale application of video-on-demand system.This dissertation presents a P2P video-on-demand system based on a delamination structure.It combines the accuracy and efficiency of the DHT (Distributed Hash Table) with the simple practicability of the Gossip protocol.The superstratum nodes of this system offer downloading services to the substrate nodes,through which it leads to loading equilibrium.This paper also presents a data downloading strategy,it can help nodes find the optimal download resource and finally enhance the playing continuity of the system by adding a small amount of local information to the information exchange process which based on Gossip protocol.The emulation experiments show that OXVoD can ensure more than 99% of nodes joining the system normally and obtain a playing continuity more than 96% on the condition that server load is stable.

Key words: P2P;video on demand;Gossip protocol;Distributed Hash Table(DHT)

摘 要:可扩展性和高播放连续度是视频点播系统大规模应用的关键。提出了一个分层结构的 P2P 点播系统,融合了 Distributed Hash Table(DHT)的精确高效和 Gossip 协议的简单实用。该系统上层结点为下层结点提供下载服务,有效均衡负载。提出了一种数据调度策略,通过在基于 Gossip 协议数据可用信息交互过程中添加少量本地信息,帮助下载者选择最优下载源,提高系统播放连续度。仿真实验表明,OXVoD 可以在服务器负载稳定的情况下,保证 99%以上的结点正常加入系统,并获得 96%以上的播放连续度。

关键词:点对点技术;视频点播;Gossip 协议;分布式哈希表
DOI:10.3778/j.issn.1002-8331.2010.07.063 **文章编号:**1002-8331(2010)07-0203-05 **文献标识码:**A **中图分类号:**TP393

1 引言

近年来,P2P 视频点播服务作为 P2P 网络下的一种典型应用,越来越得到业界的关注。传统 C/S 模式的 VoD 系统将视频源存储在服务器上,随着用户规模的增加,已经很难满足用户的需求。基于 P2P 结构的 VoD 系统较好地解决了这个问题,其策略是每个结点保存一段接收到的数据,在其他结点请求该数据时可以提供下载服务,进而缓解服务器的负载。

Chaining[1]是最早把 P2P 思想引入视频点播的服务模型,结点按照到达的时间顺序串成一条链,前面的结点缓存部分观看过的视频片断为后面的结点提供服务。ACVoD[2]利用空闲结点,将原本属于不同链的结点或者不能够连接到链的结点最终加入到同一条链,并增强链的稳定性。Narada[3]系统中引入了应用层多播概念,给出协议设计原则并设计了全分布式的多播协议。Hefeeda[4]等人提出的混合 P2P 点播结构根据物理接近性原则组织多播组,数据传播方式为多对多的方式,同时将结点按照性能分为超级结点和普通结点,利用超级结点来协助改进系统性能。P2Cast[5]采用了树状组播树结构,是 Patching[6]的 P2P 应用,结点按照到达的时间加入应用层组播树,新结点会选取与自己到达时间较为接近的结点为父结点,由于父结点可能已经把最开始的一部分内容抛弃,子结点还需要从其他拥有开头部分数据的结点处获得该部分内容,即需要寻找一个 Patching 流的父结点。Zigzag[7]是一种分层的多播流媒体系统,用户结点度被限制在一定范围内,以避免网络瓶颈和减小端到端延迟。P2VoD[8]采用组播树结构,缓存中最小的数据分片相同的结点

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60673156)。
作者简介:钱碧伟(1985-),男,硕士研究生,主要研究领域为对等网络;谢冬青(1965-),男,教授,博士生导师,主要研究领域为算法分析与设计,信息安全;周再红(1971-),女,博士研究生,主要研究领域为网络安全,分布式计算;熊伟(1977-),男,博士,主要研究领域为对等网络。
收稿日期:2009-02-18 **修回日期:**2009-04-13

构成一个层,由于第一层的结点只能从服务器下载数据,导致系统的可扩展性不够,其他层的结点都只有一个父结点,但一个父结点可能有多个子结点,由于子结点只能从其父结点处下载数据,可能导致某些子结点获得数据的时效性差。

文章提出一个基于分层结构的 P2P 点播系统 OXVoD,该系统融合 DHT 的精确高效和 Gossip 协议的简单实用,使用 DHT 快速发现父结点,通过 Gossip 协议交换数据可用信息。OXVoD 中每个结点拥有多个父结点,采用多对多的数据下载方式,并提出一种数据调度策略,通过在基于 Gossip 协议的数据可用信息交互过程中添加少量本地信息,以帮助下载者选择最优下载源,从而提高系统播放连续度,提升用户观看体验。

2 系统构建策略

假设结点可用带宽稳定。结点与结点之间是一种多对多的关系,即每个结点拥有多个父结点和多个子节点。使用 Gossip 协议,父结点将部分本地数据分片的可用信息周期性的发送给子结点,子结点根据收到的可用数据分片信息,按一定策略选择最优父结点下载数据。每个结点都预留 1 GB 磁盘空间存储本地播放过的数据供其子结点下载。

2.1 系统结构

结点按"发送周期 T_{send}"将本地数据分片可用信息发送给子结点,并由此构成子结点的待下载数据队列,结点待下载数据队列中的最小数据分片用 $Need_i$ 表示。父结点需记录最近两个"下载周期 T_{down}"收到的下载请求个数的平均值 N_{requst} 和本地未处理完毕的下载请求 $N_{unfinish}$ 作为子结点选择最优下载源的评估条件。子结点按"下载周期 T_{down}"向父结点下载数据。

每个结点维护一张父结点表(表 1)和一张子结点表(表 2)。表 1 记录了父结点的编号、IP 地址、可用上传带宽、$N_{unfinish}$、N_{requst}、上次提供可用下载信息的时间;表 2 记录了子结点编号、IP 地址、$Need_i$。结点向父结点请求下载数据,并为子结点提供下载服务。父结点表项数不超过某一常数,子结点表项数最大值则与其可用上传带宽成正比。

表 1 父结点表

PeerID	IP address	Upload bandwidth	Unfinished request	Average request	Last server time
37	210.43.109.1	103 k/s	47	8	32
…	…	…	…	…	…
158	210.43.109.12	116 k/s	32	7	32

表 2 子结点表

PeerID	IP address	Minimum Date Piece
18	210.43.109.42	1 244
…	…	…
225	18.54.76.158	1 346

设视频总的数据分片数量为 L,影片分成 N 个区间段 $[0,\frac{L}{N}),[\frac{L}{N},\frac{2L}{N}),\cdots,[\frac{(L-1)L}{N},L]$。$Need_i$ 属于同一区间段的结点构成一个层,层的编号由 $Need_i$ 决定。结点所属层的编号为 $G_i\in[1,N]$。若 $Need_i\in[\frac{kL}{N},\frac{(k+1)L}{N})$,则结点 i 属于第 $(N-k)$ 层,即 $G_i=N-k$。若某层中没有一个结点,则称该层为空层,否则为非空层;若 $G_i<G_j$,则 G_i 层为 G_j 层的上层;若 $G_i=G_j-1$,则 G_i 层为 G_j 层的直接上层;若 $G_i>G_j$,则 G_i 层为 G_j 层的下层;若 $G_i=G_j+1$,则 G_i 层为 G_j 层的直接下层。

属于同一层的结点构成一个 Chord[9]环,不介绍其维护及查找机制。会合服务器 S 保存每层的若干个结点信息作为该层的入口地址,结点进入某层时,都会先向 S 请求该层的入口地址。

理想情况下,第一层的结点从服务器获得数据后,逐层往下传递。图 1 显示了理想情况下的三层圆柱体结构:A、B 构成第一层;C、D、E 成第二层;F、G、H、I 构成第三层。结点 C 把 A、B 作为自己的父结点,结点 H 把 C、E 作为自己的父结点。

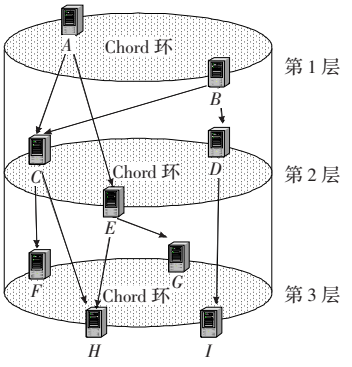

图 1 理想情况下的结构图

2.2 算法说明

请求父结点:理想情况下结点的父结点都在其直接上层。首先结点向 S 请求其直接上层的入口地址进入该层,若直接上层为空层或者没能选中足够的父结点,则往更上层搜索,直至种子结点。若种子结点也没有足够带宽,则选择同层的结点作为父结点。算法见图 2。

```
输入:结点 i 所属层 G_i、常数 FatherCount、常数 Step
输出:结点 i 的候选父节点集合 F_i
算法:
1  for(j=G_i-1;j>0;j--)
2    向会合服务器 S 请求第 j 层入口地址;
3    if(第 j 层为非空层)
4      while(F_i 中结点个数小于 FatherCount){
5          随机选择第 j 层中某个合适的结点加入 F_i
           若连续 Step 次没能选中合适的节点,则 break;
6          }
7        }
8    }
9    while(F_i 中结点个数小于 FatherCount){
10         在本层随机选择某个合适的结点加入 F_i
           若连续 Step 次没能选中合适的节点,则 break;
11   }
```

图 2 请求父结点算法

结点加入:结点 i 根据当前要播放的数据分片的区间计算出所要加入的层 G_i,发送 G_i 给会合服务器 S,并执行请求父结点过程;同时会合服务器选择第 G_i 层的若干个结点,并将他们的信息发送给结点 i,结点 i 选择某个结点(其余结点作为 Candidates)作为入口加入第 G_i 层。算法见图 3。

```
输入:结点 i 当前播放的数据分片编号 ID_play,常数 k
算法:
1 根据 ID_play 所属区间段得到结点 i 等待加入的层 G_i
2 执行请求父节点算法;
3 会合服务器选择第 G_i 层 k 个结点构成结点集 S_i 发送结点 i
4 for each(结点 j in S_i){
5   if(结点 i 未加入第 G_i 层){
6     选择结点 j 作为入口加入第 G_i 层;
7     if(结点 i 加入成功) break;
8   }
9 }
```

图 3 结点加入算法

结点离开:通知子结点将其从父亲列表中删除,并选择新的父结点;通知父结点将其从子结点列表中删除;通知兄弟结点,友好的从本层 Chord 环中退出。通知会合服务器 S,若 S 将其作为该层 Chord 环入口地址,则会选择新的入口地址。算法见图 4。

```
输入:结点 i 所属层 G_i、父结点集合 Father、子结点集合 Son
算法:
1  for each(结点 j in Son)
2    通知节点 j 将结点 i 从其父结点集中删除;
3    结点 j 执行请求父结点算法;
4  }
5  for each(结点 j in Father){
6    通知结点 j 将结点 i 从其子结点集中删除;
7  }
8  结点 i 从 G_i 层 Chord 环中退出;
9  if(结点 i 为会合服务器 G_i 层的入口地址){
10   会合服务器选择新的 G_i 层入口地址;
11 }
```

图 4 结点离开算法

结点崩溃:在连续几个调度周期内未接收到某个子结点的下载请求时,父结点会将其从子结点表中删除。子结点检测到某个父结点崩溃后,会选择新的父结点,并通知会合服务器。同层结点检测到该结点崩溃后,会更新自己的相关信息,并通知会合服务器。

所属层改变:结点所属层由其 $Need_i$ 决定。在正常播放情况下,结点由某个层迁移到上一层时,父结点不会改变。若结点执行各种交互操作(向前或向后跳跃),可由结点离开和结点加入这两个动作来完成。结点所属层改变时,会通知会合服务器 S,若 S 将其作为该层 Chord 环入口地址,则会选择新的入口地址。

下载数据分片:每个调度周期,父结点会根据子结点的 $Need_i$ 将特定范围内的数据分片可用信息发送给子结点,子结点按照一定策略向父结点请求下载数据,下一节将详细介绍数据下载策略。若连续几个周期某个父结点都无法提供下载服务,则删除该父结点并执行请求父结点算法。

3 数据调度策略

3.1 参数说明

假设 i 为数据分片编号;j 为结点编号;M 为每个数据分片的大小;N_j 为结点 j 上个发送周期未处理完毕的下载请求个数;λ_j 为结点 j 在最近两个下载周期接收到的下载请求个数的平均值;B_j 为结点 j 可供邻居下载使用的带宽;$W(i,j)$ 为邻居向结点 j 请求数据分片 i 的等待时间;S_i 为可供当前结点下载数据分片 i 的邻居结点集合;$size$ 为每个发送周期父结点发送的数据分片可用信息的个数。

3.2 数学模型

父结点按照“发送周期 T_{send}”将本地满足条件 $i\in[Need_m, Need_m+size-1]$ 的数据分片可用信息发送给子结点 m;对于本地没有的数据分片,子结点 m 依照一定策略,按“下载周期 T_{down}”向父结点提出下载请求,同时子结点 m 会通知父结点更新 $Need_m$。

子结点 m 向其父结点 j 请求下载数据分片 i 时,j 可能有大量未处理完毕的下载请求,所以 m 必须等待一段时间 $W(i,j)$ 后才能开始下载 i。等待时间 $W(i,j)$ 与结点 j 上个调度周期未处理完毕的下载请求个数 N_j、本调度周期接收到的下载请求个数 X 有关。以下将对不同周期下载请求个数 X 做出评估。

ContinuStreaming[10] 使用泊松过程作为数据分片的到来模型,P2VoD[8] 使用泊松过程作为结点的到来模型,与此类似,该文也使用泊松过程作为基于 Gossip 协议的流媒体系统中每个调度周期内下载请求的到来模型,且以 λ_j 作为该泊松过程的均值。这样结点 j 在某一调度周期内接收到的请求个数 X 分布满足:

$$P(X=k)=\frac{(\lambda_j)^k e^{-\lambda_j}}{k!},k=0,1,2,\cdots \tag{1}$$

设在本调度周期内,本次请求之前 j 已经接收到 y 个请求,这样本次请求的等待时间为:

$$W(i,j)=(y+N_j)\frac{M}{B_j} \tag{2}$$

近似让 $y=\left[\frac{X}{2}\right]$。由式(2)得结点 j 提供下载数据分片 i 的等待时间 $W(i,j)$ 分布函数为:

$$F(w)=P(W(i,j)\leqslant w)=P(X\leqslant\frac{2wB_j}{M}-2N_j) \tag{3}$$

令 $a=\frac{2wB_j}{M}-2N_j$,因为 $X\geqslant 0$,故 $a\geqslant 0$。由式(1)、(3)两式得等待时间 $W(i,j)$ 等于 w 的概率为:

$$\varphi(w)=P(W(i,j)\leqslant w)-P(W(i,j)<w)=P(X\leqslant a)-P(X<a)$$

当 $a\notin Z$(Z 为整数集合)时,因为 X 为整数才有意义,有 $P(X\leqslant a)=P(X<a)$,则 $\varphi(w)=0$;当 $a\in Z$ 时:

$$\varphi(w)=P(X\leqslant a)-P(X<a)=\sum_{k=1}^{a}\frac{(\lambda_j)^k e^{-\lambda_j}}{k!}-\sum_{k=1}^{a-1}\frac{(\lambda_j)^k e^{-\lambda_j}}{k!}=\frac{(\lambda_j)^a e^{-\lambda_j}}{a!}$$

等待时间 $W(i,j)$ 的期望值为:$E(W(i,j))=\sum(w\varphi(w))$,由式(2)知等待时间 w 为 $\frac{M}{B_j}$ 的整数倍,令 $w=l\frac{M}{B_j}$,$l\in Z$,则有 $l=\frac{a}{2}+N_j$,所以 a 必为偶数且 $l\geqslant N_j$。$W(i,j)$ 概率密度 $\varphi(w)$ 表达式为:

$$\varphi(w)=\begin{cases}\frac{(\lambda_j)^a e^{-\lambda_j}}{a!}, & a\text{ 为偶数}\\ 0, & \text{其他}\end{cases} \tag{4}$$

且有:

$$E(W(i,j))=\sum(w\varphi(w))=\sum_{l\in N_j}^{\infty}\left(\frac{lM}{B_j}\varphi\left(\frac{lM}{B_j}\right)\right) \quad (5)$$

定理 1 邻居向结点 j 请求下载数据分片 i 的等待时间的期望值 $E(W(i,j))$为有限值。

证明 令 $U_l=\frac{lM}{B_j}\times\varphi\left(\frac{lM}{B_j}\right)$，由式(5)得只需证明正项级数 $\sum_{l\in N_j}^{\infty}U_l$收敛即可。因为$\lim_{l\to\infty}\frac{U_{l+1}}{U_l}=\lim_{l\to\infty}\frac{l+1}{l}\times\frac{\lambda_j^2}{(2l-2N_j+2)(2l-2N_j+1)}=0<1$，由达朗贝尔判别法得到正项级数 $\sum_{l\in N_j}^{\infty}U_l$收敛，命题得证。

编号小的数据分片具有更高的下载优先级，这是因为即使数据分片 i 之后的所有数据都下载完毕，但只要 i 还没有下载，就无法继续播放。对于可供当前结点下载数据分片 i 的邻居结点集合 S_i，根据式(5)求出各个提供者等待时间的期望值，并选择最小期望值的结点下载数据分片 i。数据调度算法见图 5，时间复杂度为 $O(mn)$，其中 m 为待下载数据分片的个数，n 为父结点个数。

输入：(1)按下载优先级由大到小排列的数据分片 $D_1,D_2,\cdots,D_m$；
(2)可供下载数据分片的结点 $Node_1,Node_2,\cdots,Node_n$及其带宽 $B_1,B_2,\cdots,B_n$；
(3)数据分片 D_i的提供者集合 S_i；
(4)结点 $Node_j$在上个周期未处理完毕的下载请求个数 N_j。
输出：每个数据分片 D_i的提供者 $Supplier_i$
算法：

```
for(i=1;i≤m;i++){
    min=∞;
    foreach Node_j in S_i
        计算 Node_j的等待时间期望值 E(W(i,j));
        if(E(W(i,j))<min){
            min=E(W(i,j));
            best=j;
        }
    }
    Supplier_i=Node_best;
    Node_best++;
}
```

图 5 数据调度算法

4 模拟实验及数据分析

OXVoD 构建在 DHT 和 Gossip 协议之上，根据结点当前请求下载的最小数据分片将结点层次化。一般情况下，上层结点为下层结点提供下载服务，有效均衡了负载。同时采用了多对多的数据下载方式，下载数据之前，会评估出最高效的下载源，有利于提高系统播放连续性。

4.1 实验环境

实验包括两种结点：种子结点和普通(非种子)结点。种子结点包含所有的数据分片，每个分片大小为 M=4k。影片时间为 100 min，共 60 000 个数据分片，分为 25 层；非种子结点加入系统后，开始接收第一个数据分片，之后他可以进行各种交互操作。非种子结点会在每个周期 T_{play}=0.5 s 播放 n=5 个数据分片。每个发送周期 T_{send}=0.15 s 父结点向子结点最多发送 $size$=15 个数据分片的可用信息，下载周期 $T_{down}=T_{send}$。每个结点的信息还包括：带宽、加入系统时间。实验中所用到的结点的带宽属于集合 S_1=[80 k/s,250 k/s]。每个结点最多拥有的父结点个数为 $FatherCount$=5，子结点个数和其带宽成正比，并限定其范围为 S_2=[4,10]。根据经验，当结点下载速度达到 40 k/s 时播放就非常流畅了。实验中，结点在每个周期 T_{play}=0.5 s 内需播放 5 个分片，则播放速率刚好达到 40 k/s。

仿真实验在 P4 3.0 GHz 处理器，512 MB 内存，Windows XP 操作系统，VS2005 平台下使用 C# 完成，并测试了多种结点规模下的运行性能，每种情况都只有一个种子结点。

4.2 性能分析

OXVoD 采用的分层思想主要来源于 P2VoD，本节将比较 OXVoD 和 P2VoD 的性能。这里 P2VoD 种子结点的子结点个数最多为 K=6 个，每个结点缓存最大为 MB=0.2，由文献[8]知此时 P2VoD 能获得较好的性能。

结点无法正常加入系统的比例：图 6 显示了不考虑结点的失效和离开行为，在不同的结点规模情况下，结点申请加入系统时被拒绝的概率。从图中可以看到，新结点加入 OXVoD 时，几乎可以正常加入。实际上从 OXVoD 的新结点加入策略中可以看到，新结点申请父结点时，会首先向其上层结点提出请求，在失败的情况下，还会向同层结点发出请求。收到该请求的结点中，只要有一个结点的子结点集合未饱和，新结点就能正常加入系统。而新加入 P2VoD 的结点需要找到一条路径从其他结点获得流，随着结点规模的增加，找到一条拥有足够带宽的路径的概率越来越小，导致新结点加入 P2VoD 时被拒绝的概率越来越大。

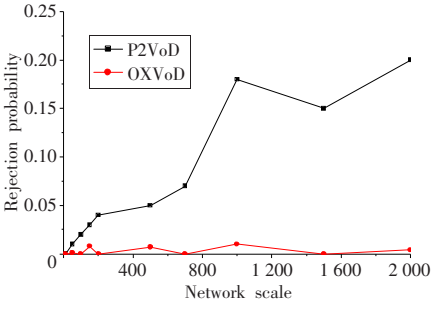

图 6 结点加入失败概率

种子结点的压力：定义稳定状态下，每个调度周期结点提供的平均下载次数为结点负载。种子结点的负载是决定基于 P2P 的 VoD 系统可扩展性重要指标。图 7 显示了 OXVoD 和 P2VoD 的种子结点负载变化。从图中可以看到，随着网络规模的增加，P2VoD 的种子结点负载逐渐的增加，当网络规模达到

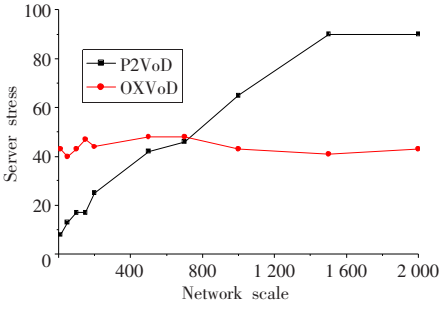

图 7 种子结点负载

一定程度后,其压力处于比较稳定了状态了。而 OXVoD 的种子结点负载则一直保持在比较稳定的状态，这是因为 OXVoD 根据种子结点带宽分配其子结点个数,在不同网络规模中都能根据其带宽能力为子结点提供下载服务。

播放连续度:设结点 i 在 T 时刻实际观看的数据分片个数为 $N_{watch}(i)$,而在正常情况下应该播放的数据分片的个数为 $N_T(i)$,定义 $\frac{N_{watch}(i)}{N_T(i)}$ 为结点 i 当前的播放连续度,$\frac{\sum_{i=1}^{n}\frac{N_{watch}(i)}{N_T(i)}}{n}$ 为系统的当前的播放连续度。

图 8 显示了 500 个结点规模时，稳定状态下 P2VoD 和 OXVoD 的播放连续度。可以看到 OXVoD 的播放连续度达到了 97%以上,相对 P2VoD 有了较大的提升。这主要时因为 OXVoD 中的结点会缓存其播放过的数据，并为后来者提供下载服务，较大程度地提高了系统的容量。按周期 5 min 随机的让 5%的结点执行向前跳跃操作,同时让 5%的结点执行向后跳跃操作,跳跃幅度为 5 min 的影片长度。若当前结点正在观看第 10 min 的内容,执行向前跳跃操作,则会从第 15 min 开始观看;若执行向后跳跃操作,则从第 5 min 开始观看。图 9 显示了 500 个结点规模时,OXVoD 和 P2VoD 在执行交互操作时的播放连续度。从图 9 中可以看到 P2VoD 在中间时刻获得了较差的播放连续度,这是因为该时刻系统中的结点数量比较大,同时执行交互操作的结点比较多,造成很多结点无法及时找到新的父结点造成的。同时由于 P2VoD 采用了一个结点只能拥有一个父结点的策略，导致了结点下载数据的效率比较低。而 OXVoD 中的结点缓存了其播放过的所有数据,即使用户频繁的执行交互操作,仍然能够通过DHT 快速的找到新的父结点;同时由于采用了多对多的数据传输模式,结点在下载数据时,可以有选择的向父结点下载数据,进而提高播放连续度。对比图 8 和图 9,可以得到 OXVoD 在动态环境中的播放连续度相对静态环境只是略有减小，这时因为每个结点都保存了多个父结点信息,即使某个父结点不再有效,子结点依然能从其他父结点获得所需要的数据分片。

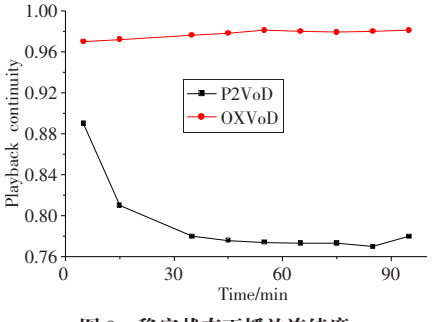

图 8　稳定状态下播放连续度

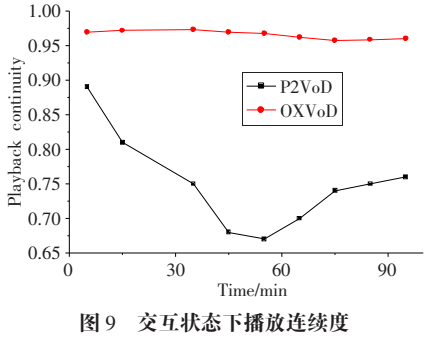

图 9　交互状态下播放连续度

5　结语

文章融合 DHT 的精确高效和 Gossip 协议的简单实用,设计了一个基于分层结构的视频点播系统。该系统充分利用各结点资源,较好平衡了负载,提高了系统的可扩展性。充分考虑资源提供者的异构性,并评估出最佳资源提供者,以此设计较好的数据调度算法。假设了结点可用带宽稳定,这与实际网络环境存在一定差异，同时没有具体研究分层多少对系统的影响,这些工作都将在下一步的研究中完成。

参考文献:

[1] Sheu S,Hua K,Tavanapong W.Chaining:A generalized batching technique for video-on-demand systems[C]//Proc of the Int'l Conf on Multimedia Computing and System.Washington:IEEE Computer Society,1997:110-117.

[2] Lin F,Zheng C,Wang X,et a1.ACVoD:A peer-to-peer based video-on-demand scheme in broadband residential access networks[J]. Int'l Journal of Ad Hoc and Ubiquitous Computing,2007,2(4):225-231.

[3] Chu Y H,Rao S G,Zhang H.A case for end system multicast[C]// Kurose J,Nain P.Proc of the ACM SIGMETRICS 2000.Santa Clara: ACM Press,2000:1-12.

[4] Hefeeda M M,Bharat K.A hybrid architecture for cost effective on demand media streaming[J].Computer Networks,2004,44(3):353-382.

[5] Guo Y,Suh K,Kurose J,et al.P2cast:peer-to-peer patching scheme for VoD service[C]//Proc of the 12th Int'l Conf on World Wide Web. New York:ACM Press,2003:301-309.

[6] Cai Y,Hua K,Vu K.Optimizing patching performance[C]//Dilip D. Proc of the MMCN'99.Washington:SPIE Press,1999:204-216.

[7] Tran D,Hua K,Do T.Zigzag:An efficient peer-to-peer scheme for media streaming[C/OL].(2006).http://www.ist.psu.edu/tran03zigzag.html.

[8] Do T,Hua K,Tantaoui M.P2VoD:Providing fault tolerant video-on-demand streaming in peer-to-peer environment[C]//Proc of the IEEE ICC 2004.Paris:IEEE Communications Society,2004:1467-1472.

[9] Stoica I,Morris R,Karger D,et a1.Chord:A scalable peer-to-peer lookup protocol for internet applications[C]//Proceedings of ACM SIGCOMM,2001.

[10] Li Zhenhua,Cao Jiannong,Chen Guihai.ContinuStreaming:Achieving high playback continuity of gossip-based peer-to-peer streaming[C]// Proceedings of IEEE INFOCOM,April,2008:14-18.