一种基于 P2P 协作的代理缓存流媒体调度算法

唐瑞春 魏青磊 刘 斌
(中国海洋大学计算机科学与技术系 青岛 266100)

摘 要：该文根据流媒体系统中缓存空间不足及服务延迟的问题，提出一种基于 P2P 协作的代理缓存流媒体调度算法 PCSPC(Proxy-Caching Scheduler based on P2P Cooperation)。首先按照流行度高的数据占用较大存储空间的原则，利用媒体文件的存储效率为每个前缀分配相应的存储空间。然后按传输成本将前缀降序排列，代理服务器升序排列，将前缀依次分配到代理服务器上，并且通过理论证明该方法能够有效地减少传输成本。PCSPC 算法能够兼顾存储效率与传输成本。仿真实例说明了算法的有效性。

关键词：流媒体；P2P 协作；代理缓存；存储效率；传输成本
中图分类号：TP393 **文献标识码**：A **文章编号**：1009-5896(2009)11-2757-05

A Proxy-Caching Scheduler Based on P2P Cooperation

Tang Rui-chun Wei Qing-lei Liu Bin
(*Department of Computer Science and Technology*, *Ocean University of China*, *Qingdao* 266100, *China*)

Abstract: Lack of cache and service time-delay in the media streaming system are considered. PCSPC (Proxy-Caching Scheduler based on P2P Cooperation) is proposed. First according to the principle that more popular data are assigned more cache, corresponding cache is allocated to every media file's prefix. And then prefix sequence and proxy sequence are sorted in ascending and descending order respectively by transmission cost, and this method can reduce transmission cost effectively, which is proved by mathematical method. Both cache efficiency and transmission cost are considered in PCSPC. Simulation results show the effectiveness of the strategy.

Key words: Streaming media; P2P cooperation; Proxy-caching; Caching efficiency; Transmission cost

1 引言

流媒体有着重要的应用背景，但是流媒体传输需要大量的带宽和服务器资源，如果单纯地提高硬件来提升流媒体系统的处理能力成本太大。更为科学的调度和传输流媒体才是解决问题的关键。

一种常用的调度方法是采用代理缓存技术。为了减小服务器和网络负载将经常用到的数据缓存到离客户端相对较近的代理服务器上。但是由于媒体文件巨大，且播放时间长，要耗费巨大的缓存空间。通常代理服务器只存储媒体文件中的一部分，用来减少需要缓存的数据量，从而提高代理服务器缓存媒体文件的数量。文献[1]提出只缓存媒体文件中每个块的前缀部分从而减少用户时延增强交互性；文献[2]提出将比较流行的媒体文件的前几块作为前缀进行缓存，用来减少用户时延；杨戈等[3]研究了当在代理服务器上缓存的数据与其流行的程度成正比时，可节约缓存空间，减少补丁通道的数据量；廖建新等[4]提出了基于流行度与网络代价的缓存策略。以上几个调度策略都是基于单个代理服务器，由于单个代理服务器缓存空间有限，缓存技术始终存在着存储空间瓶颈。

另一种方法是客户端进行 P2P 协作，每个节点都贡献自己的一部分存储空间。当媒体文件由中心媒体服务器提供之后，节点之间就可以相互请求和应答，从而减轻媒体服务器的压力并增加了存储空间。文献[5]建立了一种盖然缓存模型，该模型克服了 P2P 网络异构性和服务客户节点稀少等弊端；文献[6]提出了 P2VoD 的多播树技术，用来解决点播服务中 P2P 网络异构性和服务节点失败等问题；文献[7]提出一种混合 P2P 流媒体点播模型 TTVOD，提高了数据冗余度和分发特性。由于媒体文件播放传输时间长，在播放或传输时间内很难保证有足够的资源支持媒体文件正常播放或传输。因为系统内的节点加入和离开都是随机的同时没有任何通知，会引起较大的网络抖动。而流媒体传输对网络的抖动又是极为敏感的，这也是 P2P 发展过程中的一个主要障碍。

为了解决上述出现的问题，一种有效的方法就是综合考虑以上两种技术。本文提出一种基于 P2P

2008-09-10 收到，2009-07-29 改回
国家科技支撑计划项目(2007BAH07B00)资助课题

协作的代理缓存调度算法，同时权衡了存储空间和传输成本。为了节省存储空间必须提高媒体文件的存储效率，即在代理服务器上缓存最为常用的数据，同时将代理服务器进行 P2P 协作以增大代理缓存的空间，使得存储空间能够最大限度地发挥作用。由于代理服务器之间是进行 P2P 协作的，所以不同的代理服务器上会存储不同的数据，需要不断地进行相互的数据传输，从而必须考虑数据的传输成本问题。本文对传输成本建立数学模型并讨论。仿真结果证明本文提出的方法，能够较好地减少用户时延提高系统鲁棒性。

本文的结构如下：第 2 节是相关工作；本文的主要方法基于 P2P 协作的代理缓存流媒体调度算法在第 3 节介绍；第 4 节是实验结果与算法分析；第 5 节是结论。

2 相关工作

2.1 P2P 协作的代理缓存策略中的主要问题

根据 Cuietal 调查，用户最有可能观看媒体文件的开始部分而不是从头看到结束。基于这个发现，代理服务器只需要存储媒体文件开始的一部分，即前缀部分。

在 P2P 协作的代理缓存策略中，决定前缀大小及其分配时，需要同时考虑两个问题：流行度与传输成本。根据 Zipf 分布规律，几乎 80%的用户在点击 20%的流行度较高的媒体文件，其他 80%的媒体文件则很少被请求。为了提高代理缓存空间的利用率，流行度高的媒体文件应该占有比较多的存储空间，流行度低的媒体文件占用较少的存储空间，甚至不占用存储空间。这样既能保证用户点击的命中率也能提高代理缓存的利用率。

为增大流媒体的缓存空间，通常的方法是将多个代理服务器组成簇，簇中的代理服务器上分别缓存不同的媒体文件，并相互共享，进而增大媒体文件的总缓存数量。由此，代理服务器之间必须经常地进行大量的数据交换以满足用户需求，所以必须考虑代理服务器之间，代理服务器与客户之间的传输成本。

在以往的算法中作者往往只考虑存储效率与传输成本中的一个问题。例如文献[8]提出 PDA (Popularity based on Data Amount，PDA)算法，将多个代理服务器组成一个共享的 P2P 网络，利用媒体文件的流行度决定其存储空间以提高缓存效率，并通过缓存预取等技术降低服务器和主干网的压力。文献[9]提出 COPACC (COoperative Proxy And-Client Caching, COPACC)方法，该算法提出在主干网络上传输相同数据要比在边缘网络上需要的传输成本要大，并尽量让数据在边缘网络传输，从而减少传输成本。

本文首先利用流行度决定前缀的大小，然后按一定原则将前缀分配到不同代理服务器上，并使得传输成本尽量小，从而进一步节省服务器资源与网络资源以满足更多用户的需求。

2.2 研究环境

如图 1 所示代理服务器通过直接的或间接的逻辑连接形成一个代理服务器 overlay。并且每一个代理服务器都为一组客户端服务，称为这一组客户端的主代理服务器。假定代理服务器与客户端的通信比中心流媒体服务器与客户端的通信成本要小，中心流媒体服务器存储所有的媒体文件，代理服务器上只存储媒体文件的一部分即前缀部分。

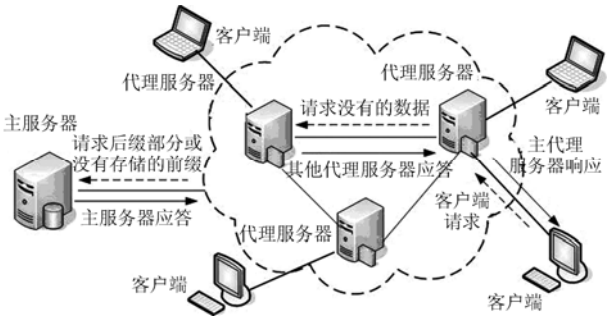

图 1 基于代理服务器 P2P 协作流媒体系统环境

当一个客户端想要播放一个媒体文件时，它首先向自己的主代理服务器发送请求。主代理服务器负责将自己缓存的媒体文件前缀部分发给客户端，同时向其他代理服务器请求其他的前缀部分。主代理服务器还负责将获取的数据组成完整连贯的数据流，保证客户端的正常播放。当媒体文件的前缀部分请求完毕之后，主代理服务器向中心流媒体服务器请求媒体的后缀部分，并传给客户端。

3 基于 P2P 协作的代理缓存流媒体调度算法

3.1 流行度与存储效率

流行度是衡量一段时间内媒体文件受欢迎的程度，也是决定媒体文件前缀部分大小的关键因素。通常流行度被定义成为在一段时间间隔内，媒体文件被点播的次数，但是考虑到实际中用户点播的影片并不一定是他喜欢的影片，打开之后可能会立即关闭，为了能够真正衡量媒体文件的流行热度，文中将流行度定义为一段时间间隔内媒体文件播放的数据总量。这样就可以更准确的衡量其流行热度。

如图 2 所示，$s_1 s_2,\cdots,s_n$ 表示在时间间隔 $\Delta_{r1,r2}$ 内传送的媒体文件。其中实线表示在这个时间间隔内正在传送的数据。

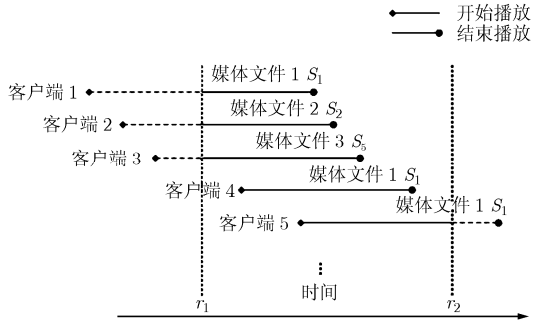

图 2 在一个时间间隔内流播放示意图

在这个时间间隔内，媒体文件 s_i 的数据量 D_i 与这段时间间隔内传输的总数据量 D 的比就是该媒体文件的流行度。

$$Po^i = \frac{\sum_{k=1}^{M} D_{ik}}{D},\quad 0 \le i \le N \tag{1}$$

媒体文件 s_i 的数据量 D_i 与其所占的存储空间的比就是存储效率 CE_i。

$$CE_i = \frac{D_i}{P^i} = \frac{\sum_{k=1}^{M} D_{ik}}{P^i},\ 0 \le i \le N \tag{2}$$

其中 $D_i = \sum_{j=1}^{M} D_{ij}$，$D_i$ 表示在某个特定时间间隔内数据流 s_i 的总数据量。D_{ik} 表示客户端 k 播放媒体文件 s_i 的数据量。N 表示媒体文件的数量，M 表示客户端的数量。P^i 表示分配给 s_i 的磁盘空间。

3.2 前缀的大小

首先在系统运行之初，即还没有统计到各个媒体文件的流行度的时候，我们将存储空间平均分配给每一个媒体文件。这意味着所有媒体文件的前缀部分大小是一样的，而且受媒体文件数量和总的存储空间大小的影响。当用户点击之后，系统开始统计每一个媒体文件 s_i 的数据总量 D_i，并计算其对应的存储效率，然后按存储效率的高低对文件进行排序。

由式(2)可以得出，为了提高某个文件的存储效率，只能提高该文件 s_i 的流行度，或是降低它的存储空间。而该文件的流行度在一段时间内基本是固定的，因此只有改变其存储空间。在代理缓存末端，是存储效率相对较低的媒体文件，当该文件的存储效率低于候补队列中的媒体文件的储存效率，该文件将被候补队列中的文件替代。因此，为了提高存储效率，只能减少流行度相对较低的媒体文件对应的存储空间。经过如此一段时间的调整，媒体文件前缀大小才能在某一段时间内确定下来。至于前缀在若干个代理服务器组成的存储空间上如何分配，将是下文关注的问题。

3.3 前缀的分配

为了更有效地将已经确定了大小的前缀部分分配到不同的代理服务器上去，我们定义 $C(i,j,p_j^i)$ 为媒体文件 i 将前缀部分中的 p_j^i 大小的数据分配到代理服务器 j 上时总的传输成本。表示如下：

$$\sum_{i=1}^{N}\sum_{j=1}^{H} C(i,j,p_j^i) \tag{3}$$

其中 $\sum_{j=1}^{H} p_j^i = P^i$；$\sum_{i=1}^{N} p_j^i \le m_j^p$；式(3)中 $C(i,j,p_j^i)$ 又可以表示为 $\sum_{J=1}^{H} p_j^i \left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right] \lambda_J Po_J^i$，$N$ 表示媒体文件数，H 表示代理服务器数，p_j^i 表示在代理服务器 j 上存储的媒体文件 i 的前缀部分的大小，$c_{j,J}^{p\to p}$ 表示在代理服务器 j 向代理服务器 J 传送一个单位的数据所需的成本，$c_J^{c\leftrightarrow p}$ 表示代理服务器 J 与客户端之间传送一个单位的数据所需的成本，λ_J 表示访问代理服务器 J 的访问频率，Po_J^i 表示在代理服务器 J 上媒体文件 i 的访问概率即代理服务器 J 上媒体文件 i 的流行度。m_j^p 表示代理服务器 j 上的缓存空间。

在 3.1 节中已给出本文关于流行度的定义，而采用按点播次数衡量的流行度即 f_J^i (请求媒体文件 i 的数量/请求的总数量)，由于在一次请求中用户可能会在请求之后并立即结束请求，因此这种情况下代理服务器并不需要将自己缓存的 p_j^i 大小的数据全部的发送出去，会产生误差。因此采用流行度 Po 可提高 $C(i,j,p_j^i)$ 的精确度。

式 (3) 中 $C(i,j,p_j^i)$ 可以分解成 $\sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J Po_J^i$ 和 p_j^i，同时用 $C(i,j)$ 表示 $\sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J Po_J^i$。这样 $C(i,j)$ 与 p_j^i 相互独立，由于目标函数 $\sum_{i=1}^{N}\sum_{j=1}^{H} C(i,j,p_j^i)$ 为线性函数，并要在 $\sum_{j=1}^{H} p_j^i = P^i$ 和 $\sum_{i=1}^{N} p_j^i \le m_j^p$ 的限定下达到最小值，所以本问题可以归结为一个线性规划问题。理论上能够最终求得 p_j^i 的大小，从而得到每一个代理服务器 j 上应该分配媒体文件 i 的大小。由于限定条件和参与运算的参数多，使这一算法让目标函数取得最小值，算法的复杂度会很高。所以在实际应用中采用另一种比较易实现的算法，来替代以上的算法。并能够证明替代的算法也能够使得传输成本尽量的小。

分配 p_j^i 大小的算法如下：

(1)按照 $C(1,j)$ 将代理服务器升序排列，将排好的序列存储至 proxy-list 中；

(2)按照 $C(i,1)$ 将媒体文件降序排列，将排好的序列存储至 content-list 中；

(3)While(全部的媒体文件分配完成 || 代理服务器缓存分配完毕)

{

将 content-list 中的第 1 个媒体文件尽可能多地存储到 proxy-list 中的第 1 个代理服务器上

If(content-list 中的第 1 个文件能够全部存储到 proxy-list 中第 1 个代理服务器上)

在 content-list 中删除第 1 个文件，并将下一个设为第 1 个

Else

在 proxy-list 中删除第 1 个代理服务器，并将下一个设为第 1 个

}

(4)if(content-list 中还有数据)

{

将 content-list 中的数据放入候选列表中

}

其中 $C(1,j)$ 表示用同一个媒体文件来评价所有的代理服务器上的传输成本，$C(i,1)$ 表示用同样一个代理服务器评价所有媒体文件的传输成本。

这个算法可以尽可能地接近最优值。证明如下。

3.4 前缀分配算法有效性证明

在一段时间内，人们都会对流行度高的媒体文件表现出相同的兴趣。所以在一定的时间段内，属于不同代理服务器的客户端发送的点播请求分布大体是一致的。可以近似地假设 $C(i,j,p_j^i)$ 中，有 $Po_1^i = Po_2^i = \cdots = Po_H^i = P_o{}^i$ 得

$$C(i,j) = \sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J Po_J^i = Po^i \bullet \sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J \tag{4}$$

其中 Po^i 为当前各个代理服务器上收到请求媒体文件 i 的平均概率。

引理 1 对于任意的媒体文件 $i,i' \in [1,\cdots,N]$，都有 $C(i,j)/C(i',j)$ 是一个常数，其中 $j \in [1,\cdots,H]$。

证明

$$C(i,j)/C(i',j) = Po^i \cdot \sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J / Po^{i'} \cdot \sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J = Po^i / Po^{i'} = C(i,j')/C(i',j')$$

由此可得出引理 1。 证毕

引理 2 对于任意代理服务器 $j,j' \in [1,\cdots,H]$，都有 $C(i,j)/C(i',j)$ 是一个常数，其中 $i \in [1,\cdots,N]$。

证明

$$C(i,j)/C(i',j) = Po^i \cdot \sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J / Po^{i'} \cdot \sum_{J=1}^{H}\left[c_{j,J}^{p\to p} + c_J^{c\leftrightarrow p}\right]\lambda_J = Po^i / Po^{i'} = C(i,j')/C(i',j')$$

其中 $C_{j,J}^{p->p}$, $C_{j',J}^{p->p}$ 分别表示代理服务器 j 与 j' 与其他代理服务器之间的传输成本。

由此可得引理 2。 证毕

定理 1 如果将 content-list 中媒体文件 i 与其位置前面的媒体文件 $i-i'$ 互换，传输成本变大，即

$$C(i,j) - C(i,j-j') \le C(i-i',j) - C(i-i',j-j') \tag{5}$$

其中 $i' \in [1,\cdots,(i-1)]$, $j' \in [1,\cdots,(j-1)]$。

证明 根据引理 1 得出

$$\frac{C(i,j)}{C(i-i',j)} = \frac{C(i,j-j')}{C(i-i',j-j')} = \alpha$$

其中 $0 \le \alpha \le 1$ ($C(i,1)$ 按照降序排列，当服务器相同的情况下， $C(i,j) \le C(i-i',j)$)

$$C(i,j) = \alpha C(i-i',j) \tag{6}$$

$$C(i,j-j') = \alpha C(i-i',j-j') \tag{7}$$

式(6)与式(7)相减，消去 α 得

$$C(i,j) - C(i,j-j') \le C(i-i',j) - C(i-i',j-j') \tag{8}$$

由此可得定理 1。 证毕

定理 1 说明，在 proxy-list 列表中，如果把媒体文件 i 放到 j 以前的位置，那么在 content-list 中比 i 位置更靠前的文件将要放到代理服务器 j 上，$C(i,j)-C(i,j-j')$ 表示媒体文件 i 更换位置后，减少的传输成本。$C(i-i',j)-C(i-i',j-j')$ 表示将媒体文件 $i-i'$ 更换位置后增加的传输成本，减少的传输成本小于增加的传输成本，这样总的传输成本将会变大。所以媒体文件 i 应该放到代理服务器 j 上，而不是更前的位置上。

同理可得定理 2。

定理 2 如果将 content-list 中媒体文件 i 与其位置后面的媒体文件 $i+i'$ 互换，传输成本变大，即

$$C(i+i',j+j') - C(i+i',j) \le C(i,j+j') - C(i,j) \tag{9}$$

其中 $i' \in [1,\cdots,(N-i)]$, $j' \in [1,\cdots,(H-j)]$。

定理 2 说明，在 proxy-list 列表中，如果把媒体文件 i 放到 j 以后的位置，那么在 content-list 中比 i 位置更靠后的媒体文件将要放到代理服务器 j 上，这样传输成本将会变大。所以媒体文件 i 应该放到代理服务器 j 上，而不是更后的位置上。

综合上述定理 1 和定理 2 得出结论：媒体文件 i 放到代理服务器 j 上是最合适的，不管放到 proxy-list 中更前或是更后的位置上传输成本都会变大。所以我们得出媒体文件 i 放入代理服务器 j 中，需要的传输成本最小，即该分配算法能够有效

的控制传输成本。

4　算法分析

为更好地与 PDA 算法比较，本文采用文献[8]的仿真环境。假定有 3 台代理服务器相互协作，每个代理服务器在一分钟内平均收到 50 个请求，同时我们设定一个单元的数据在主服务器与代理服务器之间，代理服务器之间，代理服务器与客户端之间传输所需的传输成本的比是 10:3:1。

图 3 表示 PDA 算法与 PCSPC 算法在同样的环境下所需传输成本的比较，图中纵坐标是测试算法所需的传输成本与没有缓存情况下所需传输成本之比。从图 3 中可以看出随着客户端的增加，中心服务器接到的传输后缀请求增加，同时代理服务器之间的协作频繁，使得传输成本增加。由于算法 PCSPC 中将不同传输成本的媒体文件分配到合适的代理服务器上去，当客户请求较多时，能够有效地减少传输成本，因此，从图中可以看到在客户端数量大于 100 时 PDA 算法需要的传输成本与 PCSPC 算法的差距逐渐增加，证明 PCSPC 算法能够在同等情况下服务更多用户。

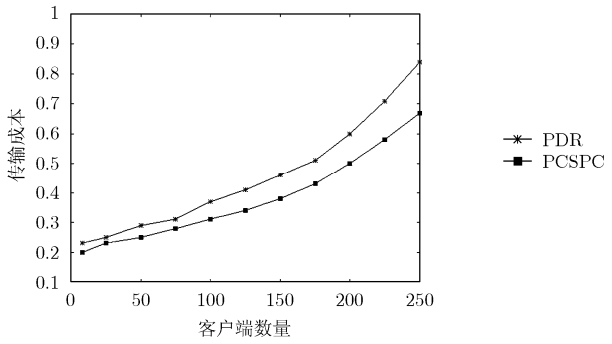

图 3 传输成本与客户端数量关系曲线

图 4 表示算法 COPACC 与算法 PCSPC 在同样的环境下比特命中率的比较，图中纵坐标的比特命中率表示客户端请求命中代理服务器缓存中媒体文件的概率。随着存储空间的增大，代理缓存中存储媒体文件的数量增多，命中率必定增加。因为在算法 PCSPC 中，利用媒体文件的流行度确定媒体文件前缀大小，提高了代理缓存的利用率，使得在同样大小的存储空间下，缓存有效的媒体文件数量更多，因此，图中可看到当缓存空间在 4 GB 到 8 GB 的时候两算法的差距最大，说明 PCSPC 算法在存储空间受限的情况下，更能够有效利用缓存，减少用户时延。

5　结论

本文研究流媒体系统中缓存空间不足与服务延迟问题，提出一种基于 P2P 协作的代理缓存流媒体调度算法 PCSPC。针对代理缓存空间不足的问题，根据常用数据占用多的存储空间的原则，在存储空间相同的情况下，利用本算法能够存储更多的媒体文件，从而最大限度发挥了存储空间的作用。利用传输成本将媒体文件与代理服务器排序，并依次将媒体文件分配到相应的代理服务器上去，尽量减少了传输所用的成本，使得在同能情况下能够服务更多用户。仿真实例说明了算法的有效性。

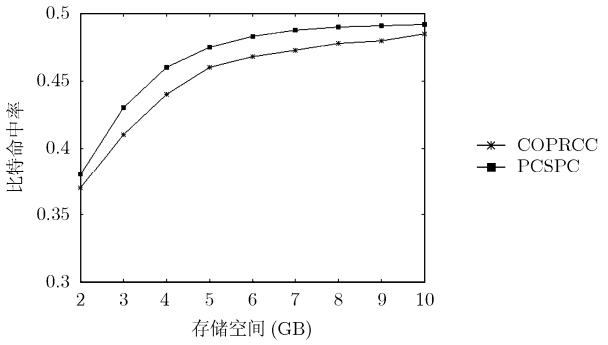

图 4 比特命中率与缓存大小关系曲线

参 考 文 献

[1] Lee S J, Ma W Y, and Shen B. An interactive video delivery and caching system using video summarization [J]. *Computer Communications*, 2002, 25(4): 424–435.

[2] Gruber S, Rexford J, and Basso A. Protocol considerations for a prefix-caching proxy for multimedia streams [J]. *Computer Networks*, 2000, 33(1): 657–668.

[3] 杨戈, 朱晓民, 廖建新, 等. 基于缓存窗口的段补丁预取的流媒体动态调度算法[J]. 电子与信息学报, 2007, 29(5): 1198–1201.

[4] 廖建新, 杨波, 朱晓民, 等. 基于网络代价的流媒体缓存策略研究[J]. 电子与信息学报, 2007, 29(9): 2239–2243.

[5] Tian Y, Wu D, and Ng K W. A novel caching mechanism for peer to peer based media on demand streaming [J]. *Journal of Systems Architecture*, 2007, 54(1–2): 55–69.

[6] Do T T, Hua K A, and Tantaoui M A. Robust video-on-demand streaming in peer to peer environments [J]. *Computer Communications*, 2008, 31(3): 506–519.

[7] 黄晓涛, 郑涛. P2P 流媒体点播的缓存机制研究[J]. 计算机工程与科学, 2008, 30(3): 44–47.

[8] Guo H, Shen G B, and Wang Z G, *et al.*. Optimized streaming media proxy and its applications [J]. *Journal of Network and Computer Applications*, 2007, 30(1): 265–281.

[9] Ip A T S, Liu J C, and Liu J C S. COPACC: An architecture of cooperative proxy-client caching system for on-demand media streaming [J]. *IEEE Transactions on Parallel and Distributed Systems*, 2007, 18(1): 70–83.

唐瑞春：　女，1968 年生，副教授，研究方向为网络流媒体.
魏青磊：　男，1983 年生，硕士生，研究方向为网络流媒体.
刘　斌：　男，1984 年生，硕士生，研究方向为网络流媒体.