

# 基于代理缓存的移动流媒体动态调度算法

廖建新<sup>1)</sup> 杨 戈<sup>1),2)</sup> 朱晓民<sup>1)</sup> 黄 海<sup>1)</sup>

<sup>1)</sup>(北京邮电大学网络与交换技术国家重点实验室 北京 100876)

<sup>2)</sup>(辽宁大学信息科学与技术学院 沈阳 110036)

**摘 要** 提出了一种基于代理缓存的移动流媒体动态调度算法 DS<sup>2</sup>AM<sup>2</sup>PC(Dynamic Scheduling Algorithm for Mobile Streaming Media based on Proxy Caching),采用代理缓存窗口自适应伸缩和分段缓存补丁块方案,在代理缓存中根据具体情况每次缓存相同或者不同大小的段补丁块,同时隔一段时间,根据移动媒体流行度更新一次缓存窗口大小,动态决定其最大缓存大小,实现了移动流媒体对象在代理服务器中缓存的数据量和其流行度成正比的原则.仿真结果表明,对于客户请求到达速率的变化,DS<sup>2</sup>AM<sup>2</sup>PC算法比 P<sup>3</sup>S<sup>2</sup>A(Proxy-assisted Patch Pre-fetching and Service Scheduling Algorithm)算法和 OBP(Optimized Batch Patching)+prefix&patch caching 算法具有更好的适应性,在最大缓存空间相同的情况下,能显著减少通过补丁通道传输的补丁数据,从而降低了服务器和骨干网络带宽的使用,能快速缓存媒体对象到缓存窗口,同时减少了代理服务器的缓存平均占有量.

**关键词** 3G;移动流媒体;调度算法;代理缓存;段补丁预取

**中图法分类号** TP393

## A Dynamic Scheduling Algorithm for Mobile Streaming Media Based on Proxy Caching

LIAO Jian-Xin<sup>1)</sup> YANG Ge<sup>1),2)</sup> ZHU Xiao-Min<sup>1)</sup> HUANG Hai<sup>1)</sup>

<sup>1)</sup>(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876)

<sup>2)</sup>(College of Information Science and Technology, Liaoning University, Shenyang 110036)

**Abstract** A novel dynamic scheduling algorithm DS<sup>2</sup>AM<sup>2</sup>PC(Dynamic Scheduling Algorithm for Mobile Streaming Media based on Proxy Caching)for mobile streaming media based on proxy caching is proposed. It employs the scheme that the cache window size can be increased or decreased adaptively according to the popularity of the requested object and the patch bytes that are segmented and cached. Every time the segment patch with different or same number of bytes is cached at the proxy cache according to the different circumstances. The cache window size is updated periodically according to the popularity of mobile streaming media object. The maximum cache size is decided dynamically. The principle is obeyed that the data cached for each mobile streaming media object are in proportion to their popularity at the proxy server. Simulation results show that this strategy is more adaptive than P<sup>3</sup>S<sup>2</sup>A(Proxy-assisted Patch Pre-fetching and Service Scheduling Algorithm) algorithm and OBP(Optimized Batch Patching)+prefix&patch caching algorithm for the variety of the request arrival rates. It can significantly reduce patching data through patching channel under the circumstance of the same maximum cache space. This can effectively reduce the

收稿日期:2006-04-28;最终修改稿收到日期:2008-02-23. 本课题得到国家杰出青年科学基金(60525110)、国家“九七三”重点基础研究发展规划项目基金(2007CB307100,2007CB307103)、新世纪优秀人才支持计划(NCET-04-0111)、国家自然科学基金(90604012)、电子信息产业发展基金项目“基于3G的移动业务应用系统”以及2007年度辽宁大学青年科研基金(2007LDQN08)资助. 廖建新,男,1965年生,博士,教授,博士生导师,主要研究领域为下一代网络. 杨 戈(通信作者),男,1974年生,博士研究生,讲师,主要研究方向为流媒体、下一代网络. E-mail:g20044444@163.com. 朱晓民,男,1974年生,博士,副研究员,主要研究方向为下一代网络. 黄 海,男,1979年生,博士研究生,主要研究方向为下一代网络.

server load and network bandwidth usage on backbone link. It can faster cache the media object than  $P^3S^2A$  algorithm. It reduces the average occupied cache space at the proxy.

**Keywords** 3G; mobile streaming media; scheduling algorithm; proxy caching; segment patch pre-fetching

## 1 引言

在移动通信网上,流媒体正受到越来越多的重视,更有人预言,流媒体将是第三代移动通信(3G)中的杀手业务.在大规模的流媒体系统中,用户的点播往往集中于少数热门节目,这就使得合并用户服务、共享服务器和网络带宽等资源成为可能,于是流调度技术应运而生.目前,对于流媒体的研究主要集中在固定网络如局域网或因特网上,已经取得了一些成果,然而,对流媒体在移动网如 3G 中应用的研究却并不多见.鉴于流媒体技术在 3G 应用中的重要地位,对其在移动网络中应用的研究是非常紧迫的,也是非常有必要的.

移动流媒体服务器和移动流媒体代理服务器是提供移动流服务的关键平台,是移动流媒体系统的核心设备.移动流媒体服务器一般处于 IP 核心网中,用于存放流媒体文件,响应用户请求并向移动终端发送流媒体数据.移动流媒体代理服务器,位于移动网络的边缘,靠近用户,在无线应用环境中,无线信道并不稳定,3G 客户端资源有限,移动流媒体代理服务器的作用显得尤其重要,其中移动流媒体调度算法的研究是其中热点.

本文第 2 节对相关流媒体算法进行分析,尤其对它们的不足进行论述;第 3 节详细介绍一种新的移动流媒体调度算法  $DS^2AM^2PC$  (Dynamic Scheduling Algorithm for Mobile Streaming Media based on Proxy Caching);第 4 节对算法进行仿真分析,比较  $DS^2AM^2PC$  算法与  $P^3S^2A$  (Proxy-assisted Patch Pre-fetching and Service Scheduling Algorithm)算法和 OBP (Optimized Batch Patching) + prefix&patch caching 算法的适应性;第 5 节总结全文,指出今后的研究方向.

## 2 相关工作

典型的流调度算法可分为两类:静态调度算法和动态调度算法.静态调度算法采用服务器推模式,

服务器不考虑用户动态行为而调度流,预留信道,周期性广播视频文件,适用于分送最流行的媒体对象;动态调度算法采用客户端拉模式,媒体流的调度由用户请求驱动,服务器根据用户请求到达的情况动态选择相应的调度方案做出响应,使不同的用户尽可能共享同一个数据流,从而降低服务器带宽资源消耗.

静态调度算法主要包括:金字塔算法<sup>[1]</sup>、摩天大楼算法<sup>[2]</sup>等.金字塔算法将节目划分为长度逐渐递增的若干片断,然后利用组播通道播放不同片断.为支持用户连续收看媒体流,金字塔算法要求用户在任意时刻必须从两个组播通道中接收数据.摩天大楼算法不采用倍数增加的方式切分节目片断,而设计了特定数列,再按照数列中的比例切分流媒体对象.以上算法不能根据用户请求到达情况做出动态调整,占有较多缓存.

动态调度算法主要包括:batching 算法<sup>[3]</sup>、patching 算法<sup>[4-5]</sup>、Piggybacking 算法<sup>[6]</sup>、OBP<sup>[7]</sup>、OBP + Prefix caching 算法、OBP + prefix&patch caching<sup>[8-9]</sup>等. Batching 算法将不同用户的请求绑定于一个组播流中,具有很好的带宽效率,但用户具有较大的平均响应延迟;patching 算法中,客户端接收正在组播的节目,同时用一个单播接收已播放的节目前缀,其有效地降低了服务器需要传输的完整流的个数,进一步节省了传输带宽,但随着请求到达率的增大所需的补丁通道个数迅速增大,导致较大的服务器负担;OBP 算法结合了 Batching 算法和 Patching 算法的优点,虽然降低了服务器网络带宽消耗,但在实践中依然会遇到很大困难,因为它依赖一个完全具有网络层组播能力的网络,而且要求客户端具备同时接收多个流的能力,客户端的启动延迟也较大,当网络规模较大时,缺乏足够的灵活性和可扩展性. Piggybacking 算法改变相邻视频流的播放速率,让后面的流赶上前面的流,然后进行合并,其提高了系统资源的利用率,但调节用户的播放速率影响了用户对媒体节目的收看效果. OBP + Prefix caching 算法通过在代理服务器中提前缓存流媒体对象的前缀部分<sup>[10]</sup>,来降低或消除客户端

的启动延迟。以上算法在骨干网络带宽消耗、服务器负载方面都获得了较好性能,但没有考虑占流媒体对象大部分的后缀的缓存策略,对后缀的有效缓存不仅可以进一步降低骨干网络带宽消耗和服务器负载,而且能有效提供用户的 QOS。OBP+prefix&patch caching 算法将补丁数据也进行分段缓存,让不同批处理区间到达的客户可以共享一部分补丁块,从而获得更好的性能,但当客户对“热门”对象的访问请求强度很高时候,这些算法仍然需要消耗较高的骨干网络带宽。文献[11]提出的 P<sup>3</sup>S<sup>2</sup>A(Proxy-assisted Patch Prefetching and Service Scheduling Algorithm)调度算法,根据当前客户请求到达的分布状况,代理服务器为后续到达的客户请求进行补丁预取和缓存,特别是对于那些较流行的媒体对象,客户请求到达率很高时,效果更加明显,而在代理服务器缓存空间的消耗方面和 OBP+prefix&patch caching 算法基本相同,但它没有区分不同流行度的流媒体对象<sup>[12]</sup>,使得代理服务器为每个流媒体对象分配的最大缓存空间都是静态变量,而不能根据流媒体流行度的变化动态调整。

### 3 移动流媒体动态调度算法 (DS<sup>2</sup>AM<sup>2</sup>PC)

#### 3.1 算法描述

为了便于分析,假定网络处于理想状态,没有抖动和传输延时,不失一般性,服务器到代理之间的网络只提供单播服务,而代理到客户之间支持 IP 组播服务,这样的域内组播更容易部署和管理,流媒体对象采用恒定位数率(CBR)编码,而且用户总是希望从头开始播放<sup>[11]</sup>。

移动流媒体调度算法 DS<sup>2</sup>AM<sup>2</sup>PC 是基于补丁预取的动态调度算法,它的基本思想是利用代理服务器在客户请求转交常规流数据时进行补丁预取并缓存,在缓存窗口  $W$  内,是否进行补丁预取取决于当前批处理区间内是否有客户请求到达以及代理是否已经缓存到  $W$ ,当客户播放完缓存在代理中的部分媒体对象而且代理没有缓存到  $W$ ,这时要从源服务器中提取相应补丁块,每次代理服务器预取的补丁块的数量根据预先定义的数据段大小来定,如图 1。当客户请求到达速率较高时,补丁预存能够很快达到缓存窗口,最大化地利用缓存,可以明显降低通过补丁通道从源服务器提取的补丁数量,从而降低骨干网络的负担及源服务器并发流的个数。

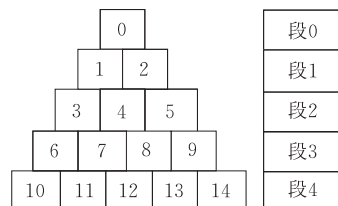


图 1 代理服务器中补丁预取的数据段

缓存在代理中的每个流媒体对象都要建立和保存一个叫媒体对象访问日志的数据结构,如下表示:

$B$ : 每个数据块表示传输的最小数据单位(时间长度表示);

$b$ : 批处理间隔(时间长度表示) $b = mB (m = 1, 2, \dots)$ ;

$P$ : 媒体对象前缀部分(时间长度表示);

$T$ : 媒体对象总长度(时间长度表示);

$L_s$ : 第  $s$  段段长(时间长度表示)如图 1。

为简单,设  $b = B$ ,  $T = kb$ , 缓存窗口大小为  $W$ ,  $W = mb (m = 1, 2, \dots, k)$ ,  $m$  的初始值为  $\left\lfloor \frac{k}{2} \right\rfloor$ 。

调度算法具体过程:

1. 假设第一个客户请求在  $t_0$  时刻到达,这时代理中只有缓存客户请求对象的前缀部分,对于第一个客户和到达时刻  $t \in [t_0, t_1]$  的客户请求(图 2 中 A 段),代理服务器立即通过单播向每个客户传送媒体对象前缀部分  $b$ ,代理服务器将在  $t_1$  时刻请求源服务器通过单播信道开始传输常规流  $T - b$ ,同时代理开始预先分配一个长度为  $L_0$  的缓存空间,如图 1,用于缓存即将到达的常规流的第一个数据段  $[b, 2b]$  作为区间  $[t_1, t_2]$  到达客户请求的补丁块,实现补丁预取,常规流数据到达代理后,代理通过组播通道向客户转交。

2. 对到达时刻  $t \in [t_1, t_2]$  的客户请求(如图 2 中 B),代理服务同样立即通过单播向每个客户传送媒体对象前缀部分  $b$ ,代理服务器在  $t_2$  时刻开始预先分配一个长度为  $L_1$  的缓存空间,用于缓存  $[2b, 4b]$ ,作为下一个区间  $[t_2, t_4]$  内到达的客户请求的补丁,同时代理服务器在  $t_2$  时刻通过补丁通道向客户传输补丁数据,通过组播向客户转交常规流,系统运行到  $t_3$  时刻,代理服务器已经缓存了 2 个长度为  $L_0 + L_1$  的补丁段。

3. 若在某个时间间隔内没有客户请求到达,则该区间为空区间,这时代理暂停分配缓存空间,同时缓存窗口减小一个  $b$ ,如果后面有客户到达,则那时要补上这部分。

4. 同理,若在整个缓存窗口中没有出现空区间,当时刻  $t \in [t_s, t_{s+1}]$  的客户请求到达,代理服务同样立即通过单播向每个客户传送媒体对象前缀部分  $b$ ,代理服务在  $t_{s+1}$  时刻通过补丁通道向客户传输补丁数据,代理服务器在  $t_{s+1}$  时刻应该预先分配一个长度为  $L_s$  的缓存空间,并缓存  $[0, 2 \times (s+1) \times b]$ ,但前面的客户已经预缓存  $[0, 2 \times s \times b]$ ,所以此时代理只要预先分配  $2 \times b$  的空间来预缓存即可,在  $t_{s+1}$  时刻加入常规通道,用于缓存  $[2 \times s \times b, 2 \times (s+1) \times b]$ ,作为以后区间到达的客

户请求的补丁。客户需要在  $t_{i+1}$  时刻加入补丁通道, 在  $t_{2 \times (s+1)}$  时刻加入常规通道。

5. 若客户到达的区间前面有多个连续空区间, 则代理有可能缺失一定的补丁数据, 在本区间结束时需要代理通过一个单播通道从源服务器中重新获取缺失的补丁数据, 称为补丁服务, 如在  $[t_6, t_7]$  区间到达的客户前面连续有 3 个空区间, 则代理在  $t_7$  时刻需要分配 4 个数据块缓存  $[6b, 8b]$ ,  $[8b, 10b]$ ,  $[10b, 12b]$ ,  $[12b, 14b]$ , 其中  $[6b, 7b]$  的数据块需要代理通过一个单播通道从源服务器中重新获取, 称为补丁服务,  $[t_6, t_7]$  到达的客户请求将在  $t_{14}$  时间加入常规流中。

6. 重复上述步 3~5, 直到缓存窗口边界。

7. 经过上面步 1~6 后, 如果在代理中已经缓存了媒体对象的一部分, 大小等于前面一个服务周期结束时缓存窗口的最终长度, 后来的客户请求将开始一个新的服务周期, 首先由代理向客户提供服务, 在服务到缓存窗口, 需要代理通过一个单播通道从源服务器中获取媒体对象剩余部分  $(T-W)$ 。

8. 经过一段时间, 当客户对媒体对象的访问平稳时, 可设代理缓存窗口  $W = \left( \left\lfloor \frac{Lavg}{b} \right\rfloor + 1 \right) b$  (其中  $Lavg$  为平均访问长度)。

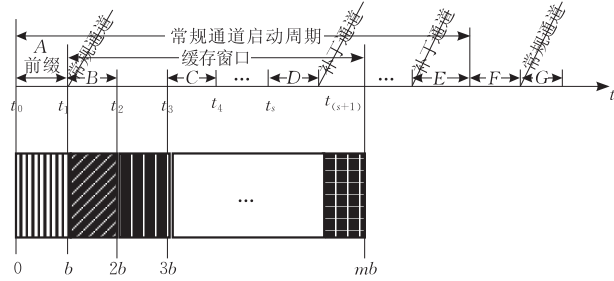


图 2 流媒体调度策略示意图

当媒体对象的缓存窗口  $W=0$  时, 代理中没有缓存该对象,  $DS^2AM^2PC$  算法退化为带前缀缓存的批处理补丁算法; 当  $0 < W < T$  时候, 代理针对不同流媒体对象的流行度差异, 自适应地改变其缓存窗口大小, 有效地利用了代理缓存空间. 在缓存补丁数据块的过程中, 只有在窗口内出现较多连续空区间的时候, 并且空区间后面有客户请求到达, 此时代理才向源服务器提出补丁服务, 当客户请求到达速率很高的时候, 需要补丁服务会非常少. 当  $W=T$  时, 经过一次高密度客户请求的常规通道启动周期, 代理就可以独立为客户请求服务, 不需要源服务器了. 同时  $DS^2AM^2PC$  算法包含多种流媒体调度算法特征: (1) 批处理 ( $b > 0, W=0$ ); (2) 补丁 ( $b=0, W > 0$ ); (3) 批处理补丁 ( $b > 0, W > 0$ ); (4)  $P^3S^2A$  算法 ( $L_s=1$ ).

### 3.2 算法分析

首先将上述算法作如下数学假设和抽象<sup>[11]</sup>.

(1) 假设用户对流媒体对象的请求到达服从泊松过程, 平均请求到达速率为  $\lambda$ , 则在一个批处理时

间间隔内没有请求到达的概率为  $pr = e^{-\lambda b}$ , 有请求到达的概率为  $1 - pr$ .

(2)  $y_i$  是第  $i$  个批处理区间  $[t_{i-1}, t_i]$  中到达的客户请求数,  $y_1, y_2, \dots, y_N$  是相互独立、相同分布的随机变量。

(3) 在第  $i$  个批处理区间是否需要补丁服务, 由  $y_{i-2}, y_{i-1}, y_i$  的值共同决定, 即  $y_{i-2}=0, y_{i-1}=0$  且  $y_i \neq 0$  时, 需要启动一次补丁通道传输补丁块  $[(i-1) \cdot b, i \cdot b]$ ;  $y_i=0$  时, 无论  $y_{i-2}, y_{i-1}$  取任何值, 都不需要启动补丁通道, 特别地, 当  $y_1=y_2=\dots=y_N=0$  或者当  $y_1=y_2=\dots=y_N \neq 0$  时都不需要启动补丁通道。

(4) 假设在补丁窗口  $W$  内, 产生补丁服务的大小为  $u$ , 显然  $u$  取值为  $0, b, \dots, (N-1) \cdot b$  中的各个可能值, 记它取  $i \cdot b$  的概率为  $p_i$ , 即  $p(u=i \cdot b) = p_i, i=0, 1, \dots, N-1$ .

$$Eu = b \sum_{i=0}^{N-1} i \cdot p_i = b \sum_{i=1}^{N-1} i \cdot p_i, \text{ 只需要确定 } p_i \text{ 即可.}$$

$$\text{当 } N=2 \text{ 时, } p_1 = (1-pr) \cdot pr;$$

$$\text{当 } N=3 \text{ 时, } p_1 = (1-pr) \cdot pr^2 + (1-pr)^2 \cdot pr;$$

$$\text{当 } N=4 \text{ 时, } p_1 = (1-pr) \cdot pr^3 + 3 \cdot (1-pr)^2 \cdot pr^2 + (1-pr)^3 \cdot pr;$$

$$\text{当 } N=5 \text{ 时, } p_1 = (1-pr) \cdot pr^4 + 3 \cdot (1-pr)^2 \cdot pr^3 + 5 \cdot (1-pr)^3 \cdot pr^2 + (1-pr)^4 \cdot pr;$$

$$\text{当 } N=6 \text{ 时, } p_1 = (1-pr) \cdot pr^5 + 3 \cdot (1-pr)^2 \cdot pr^4 + 8 \cdot (1-pr)^3 \cdot pr^3 + 7 \cdot (1-pr)^4 \cdot pr^2 + (1-pr)^5 \cdot pr;$$

$$\text{当 } N=7 \text{ 时, } p_1 = (1-pr) \cdot pr^6 + 3 \cdot (1-pr)^2 \cdot pr^5 + 8 \cdot (1-pr)^3 \cdot pr^4 + 16 \cdot (1-pr)^4 \cdot pr^3 + 9 \cdot (1-pr)^5 \cdot pr^2 + (1-pr)^6 \cdot pr;$$

$$\text{当 } N=8, 9 \text{ 时, } \dots$$

可以归纳, 当  $N=10$  时, 构造如下  $N-1$  行  $N$  列矩阵: 矩阵第 2 行元素按照自然数排列, 其它元素全部为 1.

设矩阵

$$matrix1(m, n) = matrix2(m, n) =$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

取

$$matrix1(m, n) =$$

$$matrix1(m,n-1)+matrix1(m-1,n-1),$$

$$3 \leq m \leq N-1, 2 \leq n \leq N;$$

$$matrix2(m,n)=$$

$$matrix2(m,n-1)+matrix2(m-1,n-1),$$

$$3 \leq m \leq N-1, 2 \leq n \leq N;$$

$$matrix(m,n)=$$

$$matrix1(m,n)+(n-1) \cdot matrix2(m-1,n-1),$$

$$2 \leq m \leq N-1, 2 \leq n \leq N;$$

$$p_i = \sum_{m=1}^{N-i} pr^{m+i-1} \cdot q^{N-m-i+1} \cdot matrix(m, N-m-i+1),$$

$$i=1,2,\cdots,N-1.$$

同理,可以求其它  $N$  时的  $Eu^{[11]}$ .

4 DS<sup>2</sup>AM<sup>2</sup>PC 算法性能分析

在 DS<sup>2</sup>AM<sup>2</sup>PC 中,代理服务器利用前缀缓存并通过单播通道为每个客户请求传输媒体对象的前缀部分,从而消除了客户的启动延迟,其余补丁数据和常规流数据都通过代理以组播的方式转发给客户.每次客户请求都使得已经缓存的时刻到请求区间结束时刻的距离逐渐增大,缓存越能较快地达到代理缓存窗口,越能充分利用代理缓存,因此提高了效率,如图 3. 如果在每个区间都有客户到达时,在请求区间结束时刻,客户可以利用前面客户已经缓存的预取补丁,只要再预缓存 2 个  $b$  的补丁.

如果代理缓存窗口大小相同,每个区间都有客

户到达时,DS<sup>2</sup>AM<sup>2</sup>PC 算法比 P<sup>3</sup>S<sup>2</sup>A 算法更快的使代理缓存到整个缓存窗口,使后续的客户更好地利用缓存窗口里的数据,如图 3.

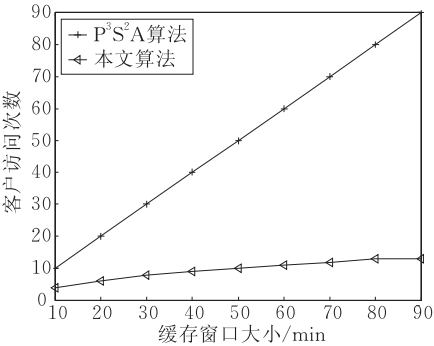
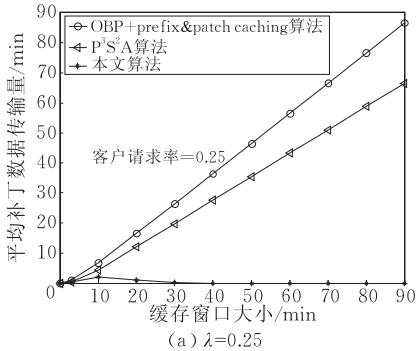


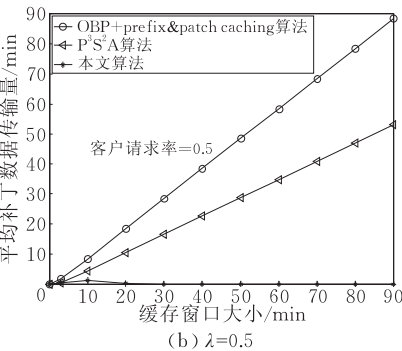
图 3 DS<sup>2</sup>AM<sup>2</sup>PC 算法与 P<sup>3</sup>S<sup>2</sup>A 算法的缓存速度对比

设媒体对象播放的持续时间长度  $T=90\text{min}$ ,前缀长度  $P=1\text{min}$ ,批处理间隔  $b=1\text{min}$ ,常规通道启动周期为  $W+P$ ,媒体播放速率为  $r=1.5\text{Mbps}$  (MPEG-1),  $u$  是需要启动补丁通道重传的部分,  $u=Eu$ ,  $Eu$  可以通过 3.2 节的方法求出,如图 4;骨干链路的归一化带宽(服务器平均并发流个数)为  $R/r=[u+(T-P)]/(W+P+1/\lambda)$ ,其中  $\lambda$  是请求到达速度,  $R$  表示源服务器输出链路的平均传输带宽(即骨干链路的平均传输速率),如图 5;代理服务器的缓存平均占用量为  $S=P \times r+(u+u_1) \times r$ ,其中  $u_1$  表示从常规通道中预取的补丁,  $u_1=b+(W-b) \times (1-pr)$ ,如图 6.

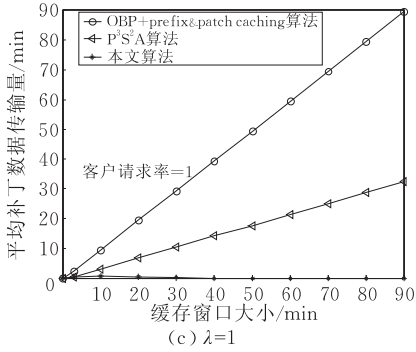
图 4 中(a)~(d)分别显示 OBP+prefix&patch



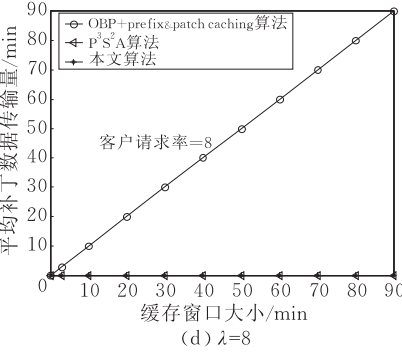
(a)  $\lambda=0.25$



(b)  $\lambda=0.5$



(c)  $\lambda=1$



(d)  $\lambda=8$

图 4 补丁传输量和窗口  $W$  的关系



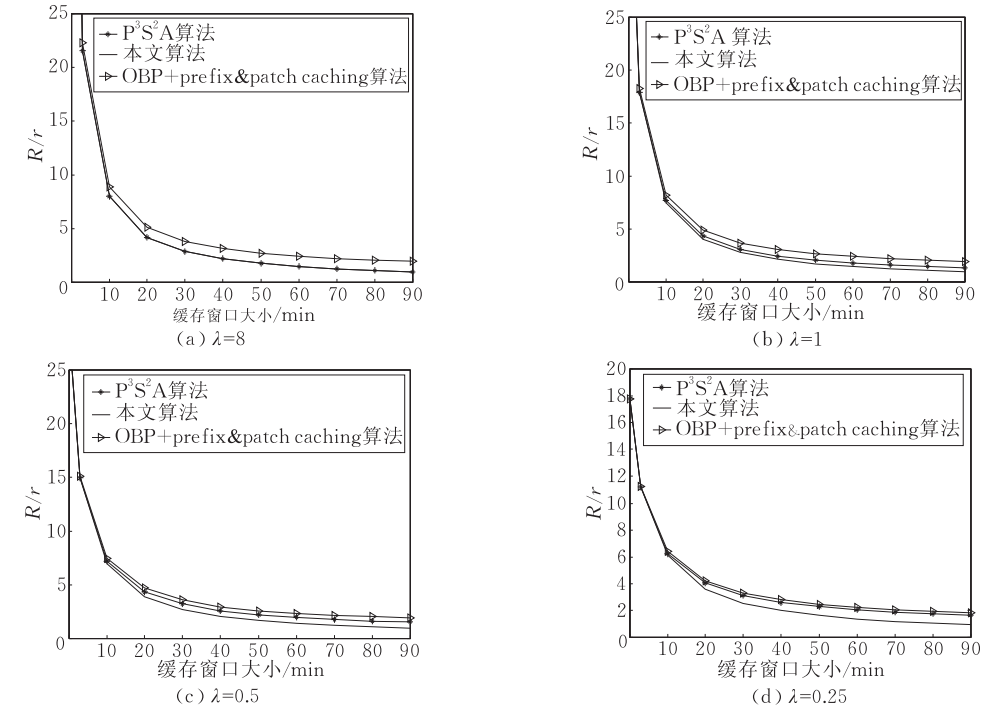


图 5 骨干链路归一化带宽和窗口  $W$  的关系

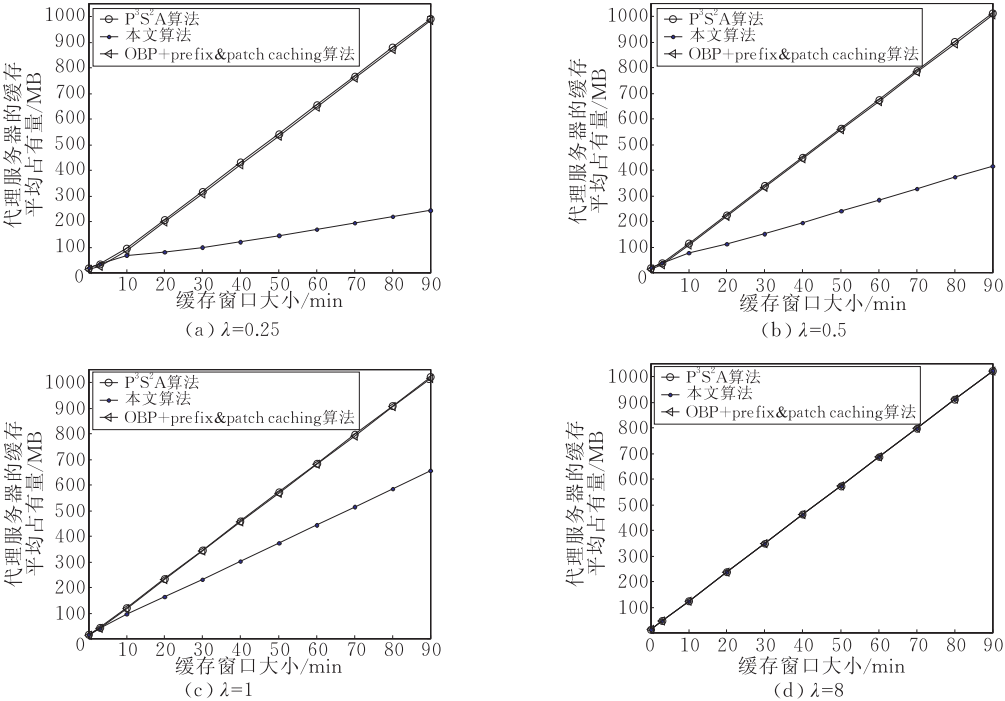


图 6 代理服务器的缓存平均占有量的对比

caching 算法、 $P^3S^2A$  算法和  $DS^2AM^2PC$  算法在不同客户请求强度下通过补丁通道传输补丁数量的比较,前两种算法都随  $W$  变大而增加,OBP+prefix&patch caching 算法增加更快,变化最慢的是  $DS^2AM^2PC$  算法。

图 5 中(a)~(d)分别显示 OBP+prefix&patch caching 算法、 $P^3S^2A$  算法和  $DS^2AM^2PC$  算法在

不同客户请求强度下消耗的骨干链路带宽.3 种算法的骨干链路消耗的带宽随缓存窗口增大而减少,显然  $DS^2AM^2PC$  算法减少得更快,OBP+prefix&patch caching 算法减少得最慢。

图 6(a)~(d)分别是 3 种算法在不同客户请求强度下,所需要的代理平均缓存空间的对比.从图中看出,平均缓存空间随  $W$  线性增长, $DS^2AM^2PC$  算

法增长得最慢,OBP+prefix&patch caching 算法和  $P^3S^2A$  算法比较接近,当  $\lambda=8$  时,三者已经基本重合。

图 7 是  $DS^2AM^2PC$  算法在不同客户请求强度下,所需要的代理平均缓存空间的对比,随着缓存窗口的增加,需要的代理平均缓存空间也增加,客户请求强度越大,所需的平均缓存空间也越大。

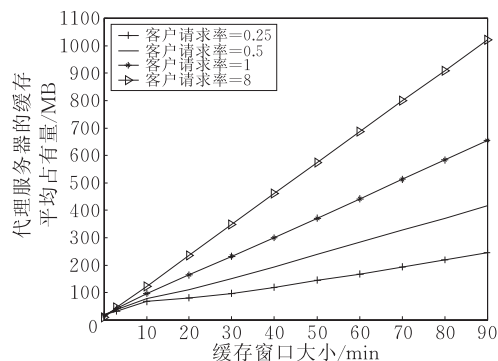


图 7  $DS^2AM^2PC$  算法代理缓存占用与窗口  $W$  的关系

## 5 结 论

在 3G 的流媒体分送系统中,由于 3G 客户终端资源相对于基于 Internet 的流媒体的客户终端非常有限,因此 3G 流媒体系统中,流媒体的传输调度算法更加重要. 为了提高流媒体分送系统的效率,节省网络带宽,本文根据用户对流媒体对象的访问行为,提出了  $DS^2AM^2PC$  调度算法,在代理缓存中根据客户请求情况,每次分配不同大小的缓存段,实现基于常规流的补丁预取和缓存,同时隔一段时间,更新一次缓存窗口大小,根据媒体流行度动态决定其最大缓存大小,尽量拖延了加入常规流的时间,这样不但降低了补丁块传送的数量,从而降低了骨干网络带宽的消耗和流媒体服务器的负担,而且在少量客户请求时,能够节省代理缓存. 仿真分析表明, $DS^2AM^2PC$  算法比  $P^3S^2A$  算法和 OBP+prefix&patch caching 算法具有更好的性能,但是在客户请求相对较少的情况下, $DS^2AM^2PC$  算法有可能在某一时刻占有多一些的代理缓存,但经过一段时间的缓存窗口自动调整,这种情况会有很大改善. 考虑目前缓存成本, $DS^2AM^2PC$  算法还是具有非常大的实际用途. 另外,3G 用户也非常需要

流媒体系统提供 VCR-like 功能,下一步是如何在  $DS^2AM^2PC$  算法的基础上实现 VCR-like 功能。

## 参 考 文 献

- [1] Lu Hai-Bin, Lan Jun-Qiang, Zhuang Xin-Hua. Advanced pyramid broadcasting for video-on-demand//Proceedings of the IEEE 6th International Symposium on Multimedia Software Engineering (ISMSE'04). Miami, USA, 2004: 28-34
- [2] Hua Kien A et al. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand system//Proceedings of the ACM SIGCOMM Conference. Cannes, France, 1997: 89-100
- [3] Dan A et al. Scheduling policies for an on-demand video server with batching//Proceedings of the ACM Multimedia. San Francisco, California, 1994: 15-24
- [4] Cai Y, Hua K A, Vu K. Optimizing patching performance//Proceedings of the ACM/SPIE Multimedia Computing and Networking. San Jose, California, 1999: 203-215
- [5] Hua Kien A, Cai Ying, Sheu Simon. Patching: A multicast technique for true video-on-demand services//Proceedings of the ACM Multimedia. Bristol UK, 1998: 191-200
- [6] Golubchik L et al. Adaptive piggybacking: A novel technique for data sharing in video-on-demand storage servers. ACM Multimedia Systems, 1996, 4(3): 140-155
- [7] White P P, Crowcroft J. Optimized batch patching with classes of service. ACM SIGCOMM Computer Communication Review, 2000, 30(4): 21-28.
- [8] Verscheure Olivier, Venkatramani Chitra, Frossard Pascal, Amini Lisa. Joint server scheduling and proxy caching for video delivery. Computer Communications, 2002, 25(4): 413-423
- [9] Frossard P, Verscheure O. Batch patch caching for streaming media. IEEE Communications Letters, 2002, 6(4): 159-161
- [10] Sen S, Rexford J, Towsley D. Proxy prefix caching for multimedia streams//Proceedings of the IEEE Infocom'99. New York, USA, 1999, 3: 1310-1319
- [11] Qin Shao-Hua, Li Zi-Mu, Cai Qing-Song, Hu Jian-Ping. Study on dynamic scheduling algorithms for streaming media based on proxy caching. Chinese Journal of Computers, 2005, 28(2): 185-194(in Chinese)  
(覃少华, 李子木, 蔡青松, 胡建平. 基于代理缓存的流媒体动态调度算法研究. 计算机学报, 2005, 28(2): 185-194)
- [12] Chesire M, Wolman A, Voelker G M et al. Measurement and analysis of a streaming media workload//Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems(USITS-01). San Francisco, California, 2001: 1-12



**LIAO Jian-Xin**, born in 1965, Ph.D., professor, Ph. D. supervisor. His research interests include mobile intelligent network, broadband intelligent network and 3G core networks.

**YANG Ge**, born in 1974, Ph. D. candidate, lecturer.

**Background**

This work is supported by the National Science Fund for Distinguished Young Scholars (No. 60525110); National 973 Program (Nos. 2007CB307100, 2007CB307103); Program for New Century Excellent Talents in University (No. NCET-04-0111); NSFC (No. 90604012); Development Fund Project for Electronic and Information Industry (Mobile Service and Application System Based on 3G); 2007 Year's Science Fund Project for Young Scholars of Liaoning University (2007LDQN08). An important objective of these projects is to design scheduling algorithms, which can improve the transmission performance of a streaming media system.

Delivering efficiently streaming media to asynchronous clients is a challenging task due to the high bandwidth requirements. A lot of existing solutions have focused on developing IP multicast-based scheduling algorithm to reduce serv-

His current research interests include streaming media and next generation network (NGN).

**ZHU Xiao-Min**, born in 1974, Ph. D., associate professor. His research interests span the area of intelligent networks and next-generation networks with a focus on IN/Internet interworking and protocol conversion.

**HUANG Hai**, born in 1979, Ph. D. candidate. His research interests focus on next generation network (NGN).

er load and network bandwidth. However, even today IP multicast deployment in the Internet remains severely limited. This paper solves the problem of streaming media in the Internet-like environment through the dynamic scheduling algorithm based on proxy caching. Simulation results shows that the algorithm effectively reduces the server load and network bandwidth usage on backbone link and the average occupied cache space at the proxy.

Their previous work on this area include a proxy caching algorithm based on segment popularity for mobile streaming media, an active prefetching algorithm for streaming media based on natural number segmentation, peer-to-peer oriented admission control for streaming media and a data assignment algorithm based on peer to peer for streaming media and so on.