

因特网上视频多点传输算法研究

林亚平, 程友清

(湖南大学计算机科学系, 湖南长沙 410082)

摘 要: 本文研究因特网上进行视频多点传输的问题. 在分析了资源预约协议和智体反馈控制机制的基础上, 基于分层编码技术提出一种新的视频传输算法. 文中利用 ns-2 网络模拟器进行了性能评价, 结果表明该算法在保证视频基本服务质量的条件下, 具有较好的公平性和可扩放性.

关键词: 视频多点发布; 资源预约; 分层编码; 反馈智体

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2001)11-1503-04

Research on Multicast Video Distribution Algorithms in the Internet

LIN Ya ping, CHEN Your qing

(Dept. of Computer Science, Hunan Univ., Changsha, Hunan 410082, China)

Abstract: This paper researches the problems of multicast video distribution in Internet. On discussing the characteristics of resource reservation protocol(RSVP) and the mechanism of the feedback agent, a new algorithm for multicast video distribution is proposed based on the layered video encoding. The simulation is performed using the network simulator ns-2. The results show that the algorithm not only assures the basic quality of service of the video applications, but also has better performance in fairness and scalability.

Key words: multicast video distribution; RSVP; layered video encoding; feedback agent

1 引言

近年来, 因特网上的多点实时流媒体应用一直呈迅速增长趋势. 从端到端的角度考虑, 这些应用中的通信延时, 传输带宽和数据丢失等性能指标需要良好的 QoS 支持^[1]. 在缺少 QoS 支持的因特网上, 这些应用不仅不能够满足不同接入速率用户的需求, 而且因采用的协议不具备 TCP 友好的特点, 导致协议间的不公平性, 对目前因特网占主要地位的 TCP 业务 (E-mail, WWW, FTP) 带来严重影响, 导致网络拥塞, 甚至发生崩溃^[2]. 因此研究因特网上的多点实时流媒体的传输控制引起普遍关注, 依据传输控制中涉及的源(发送方)、传输网络和接收方三类对象, 相关的研究工作可归纳为基于源^[3], 基于网络^[4]和基于接收端^[5]的控制. 本文主要讨论基于源和接收方的端到端控制策略.

视频多点传输与视频编码算法密切相关, 研究表明采用分层编码比较适合多点实时流媒体的传输^[5]. 分层编码是要进行编码的视频信息分成若干层(或若干个流), 其中包含最重要视频信息的层称为基本层, 其它附加层称为增强层. 在基本层基础上, 通过增强层可以获取更好的视频质量. 若视频信息采用分层编码, 共有 K 层, 依次记为 n_1, n_2, \dots, n_K 层, 其中 n_1 为基本层, 接收者可以根据当前带宽情况选择前 m ($m \leq K$) 层接收, 由此适应网络的异构性. 但由于网络带宽动态

变化的特性, 采用单纯的分层编码并不能保证理想的视频质量^[5,6]. 针对这一问题, 本文提出一种将资源预约协议和反馈智体机制结合的多点视频发布算法.

2 资源预约协议和反馈智体机制

为了讨论方便, 下面对资源预约协议和反馈智体机制作简要介绍.

2.1 资源预约协议(RSVP)^[7]

在 RSVP 协议中, 当接收者加入多点通信群时, 先估计要达到自己要求的接收质量所需的带宽, 然后接收者向源发资源预约的消息, 预约自己需要的资源. 在预约消息传送的过程中, 经过的每个路由器都会预留接收者所需要的带宽等资源. RSVP 的一个重要特性是具有较好的扩展性, 无需为多点播送的每一个接收方都预约资源, 利用预约合并, 只满足最高 QoS 的资源请求即可. 在多点播送情况下, 各接收方根据自己的 QoS 要求提出资源预约, 发出 RESV 消息, RSVP 的 RESV 消息无需盲目地寻找源, 而只是沿 PATH 消息的路径反向发送, 在多点播送树的分支节点处, RSVP 将来自不同接收方的预约请求进行合并, 只有 QoS 要求最高的预约请求才继续传送. RSVP 针对应用分配资源, 能够有效减少 Internet 多媒体实时传输的迟滞和时延抖动, 保证音频、视频等信息的实时传输.

但 RSVP 对实时传输数据流缺乏监控, 无法获得网络拥塞、包丢失等传输信息, 因此不能及时调整实时数据的传送。

2.2 反馈智体机制^[3]

基于源端的传输控制算法存在反馈爆炸问题, 为了避免反馈爆炸, 文献[3]提出在网络中设立智体机制聚合接收者的反馈信息, 其基本思想简述如下: 把网络划分为若干域(domain), 每个域中设立一个中间智体(Intemedia Agent, 简称 IA)。域又可划分若干个子网, 每个子网有一个子网智体(Subnet Agent, 简称 SA)。SA 从子网收集各种反馈信息反馈给 IA, IA 根据将这些反馈信息做出聚合处理后形成一条反馈信息发送给源端。由此减轻源端的反馈负载和处理瓶颈。

3 基本层资源预约速率适应算法

本文引入前述的层次反馈智体机制, 提出一种基于资源预约和智体反馈控制的多点视频发布算法, 称之为基本层资源预约速率适应算法, 简称为 BL-RSVP。BL-RSVP 算法的基本思想是: 源对视频分层编码, 其中基本层的速率固定, 增强层的速率则根据接收者的反馈信息予以调节。接收者要加入多点通信群时, 在网络中预约与基本层速率相适应的带宽, 由此保证接收者能接收到最低质量的视频效果; 源在服务过程中根据接收者的反馈信息来改变相应增强层的传输速率; 此外为了减少反馈爆炸问题, 采用智体机制进行反馈消息的聚合处理。算法分为资源预约、反馈智体和速率适应三个方面, 下面分别进行描述。

3.1 基本层资源预约

算法采用 RSVP 协议保证视频编码的基本层被正确接收, 基本过程如下:

- (1) 接收者向本地路由器发送加入多点通信群的请求, 在这一步中并不请求带宽, 路由器只需要将请求信息路由到源。
- (2) 源端通过发送 PATH 消息给接收者确定返回路径和流速率的信息, 接收者接收 PATH 消息后, 通过 RESV 消息进行预约资源, 经过的路由器都预约与基本层速率相适应的带宽。
- (3) 当接收者要离开多点通信群时, 它向本地路由器发出离开的请求, 沿途的路由器将释放所预约的带宽(若有同一多点通信群的其他接收者在使用预约资源则不释放), 当离开的请求到达源后, 源就从多点通信群中删除该接收者并停止向其发送视频信息。

3.2 反馈智体和状态确定规则

BL-RSVP 属于基于源的控制算法, 算法采用反馈智体机制避免源控制算法中的反馈爆炸问题。当接收者收到了源发给它的第一个视频信息帧后, 该接收者就向子网中发询问帧, 若有反馈智体存在, 则该反馈智体发应答消息给接收者, 否则该接收者就将自己或将最近的路由器设置为反馈智体, 并把询问消息发送到上层域中, 域中反馈智体的确定与子网中反馈智体的确定类似, 如图 1 所示。接收者发送的反馈信息中包括源的地址、接收者所接收的视频信息的层数、接收者所处状态等。其中接收者状态是根据一段时间内该接收者所接收数据的丢失率确定, 分为满载(loaded)、未满载(unloaded)和

拥塞(congested)三种状态中的一种。接收者的状态确定规则如下: 设 $r_1 > r_2$ 分别表示两种分组丢失门限值, 若丢失率大于 r_1 , 则状态设置为拥塞; 若丢失率小于 r_2 , 则状态设置为未满载; 否则状态设置为满载。

3.3 速率适应机制

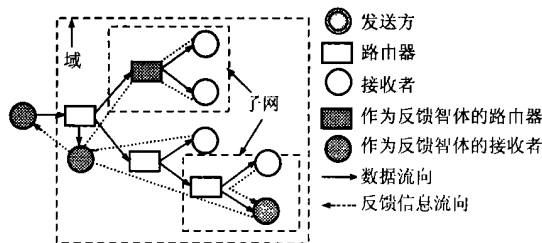


图 1 反馈智体的确定

算法采用以下的速率适应机制调节发送速率。源每隔一段时间 T , 便向各接收者发送状态查询信息, 接收者接到状态查询信息后, 统计上两次状态查询信息时间内所接收信息的丢失率, 利用前述的状态确定规则设置当前状态。然后组合成一条反馈信息, 传送到最接近的反馈智体, 反馈智体收集各接收者的反馈信息并组合成汇总信息传送到上层反馈智体, 其过程如图 1 所示。源接收到反馈信息后, 根据处于不同状态接收者的比例作出提高或降低增强传输速率的决策。设 m_i 表示接收第 i 层且处于拥塞状态的接收者的比例; w_i 为第 i 层当前传输速率; w_{i_min} 和 w_{i_max} 分别表示源第 i 层的最小和最大传输速率。令 a 、 b 分别表示源降低和提高发送速率的比例; f_1 和 f_2 分别表示两种门限值, 则源对第 i 层的速率适应算法描述如下:

rate_adjust() // 速率适应算法

```

{
    if (( $m_i > f_1$ ) and ( $w_i > w_{i\_min}$ ))
         $w_i = \max(w_i * (1 - a), w_{i\_2min});$ 
    else
        if (( $m_i > f_2$ ) and ( $m_i > f_1$ ) and ( $w_i \leq w_{i\_min}$ ))
             $w_i = \min(w_i * (1 + b), w_{i\_max});$ 
}

```

有关接收者加入和退出增强层的适应算法参见文[6]。当本地路由器收到 R 加入某个增强层的请求后, 就将其加入该层的多点通信群中, 并开始向 R 发送

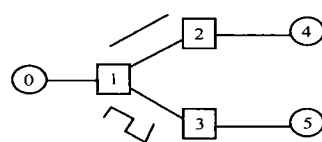


图 2 网络模拟拓扑图

该层视频信息流; 同理, 收到 R 发送的退出某层的请求后, 本地路由器就将 R 从该层的多点通信群中删除并停止向 R 发送该层信息流。

4 性能分析

为了评价 BL-RSVP, 本文采用流行的 ns-2 网络模拟器^[8]进行性能评价。评价时主要考虑算法的以下几个性能指标: (1) 速率适应性; (2) 服务质量; (3) 不同业务之间带使

用网络带宽的公平性.

4.1 模拟拓扑结构及参数设置

参照文献[9], 本文采用图 2 所示的网络模拟拓扑结构, 其中节点 0 为源主机, 可发送 0、1、2 共 3 层信息流 (0 层为基本层, 1、2 层为增强层), 节点 1、2、3 为路由器, 4 和 5 为接收者. 网络中各链路的带宽为 10Mbps, 延迟为 10ms, 路由器队列排队采用 RED^[10] 算法. 为了讨论算法的性能, 在节点 1、2 间的链路中加入速率为 2Mbps 的 CBR 流; 在节点 1、3 间的链路中加入峰值为 2Mbps 和 3Mbps 的方波, 形成背景流量. 本文使用以下模拟参数: 在 60ms 时间内, 接收者数据丢失率小于 5% 为未满载状态; 丢失率介于 5% 和 7% 之间为满载状态; 丢失率 > 7% 为拥塞状态. 发生拥塞时, 信息流的速率每次降低 20%; 未满载时, 信息流的速率每次提高 10%. 接收者连续 3 次处于拥塞状态, 则退出接收数据流当前最高增强层; 当接收者连续 3 次处于未满载状态时开始接收较上一层的增强层. 基本层的速率固定为 2Mbps, 增强层 1 的最大速率为 3Mbps、最小速率为 1Mbps, 增强层 2 的最大速率为 4Mbps、最小速率为 2Mbps.

4.2 模拟结果分析

(1) 速率适应性质

实验比较了拥塞时固定速率的分层编码算法与 BL-RSVPa 算法的速率适应性能. 在 8.0s 时, 节点 1 和节点 2 之间链路上因加载了 2Mbps 的 CBR 流造成了拥塞; 10.0s 时, 节点 1 和节点 3 之间链路上因加载了 3Mbps 的方波造成了拥塞. 由于没有速率适应机制, 固定速率分层编码算法下, 源的发送速率没有变化. 而对于同样的背景流量, BL-RSVPa 算法通过反馈和速率适应机制, 源降低了增强层的速率, 对拥塞进行了控制, 使得源的发送速率可以适应网络带宽的动态变化, 图 3 给出了 BL-RSVPa 算法的模拟结果, 图中 X 轴表示模拟时间 (单位为秒), Y 轴表示源的发送速率 (单位为 Mbps), 注意图中得到调节的是增强层的速率.

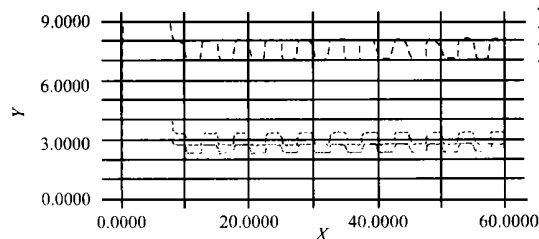


图 3 BL-RSVPa 下的速率适应机制

(2) 服务质量

为了评价接收者的接收质量, 在前述模拟条件下, 对节点 4 (图中用粗线表示) 和节点 5 (图中用细线表示) 进行了数据丢失率统计. 在固定速率和 BL-RSVPa 两种算法的模拟实验中, 节点 4 和节点 5 在 3.50 秒和 5.0 秒开始接收节点 0 发出的数据, 图 4 和图 5 给出了对应的模拟结果. 由于固定速率算法没有速率调节机制, 导致节点 4 和节点 5 丢失数据幅度较大. BL-RSVPa 算法则通过改变增强层的传输速率, 对拥塞进行了控制, 降低了数据丢失率, 从而改善了所接收视频的质

量.

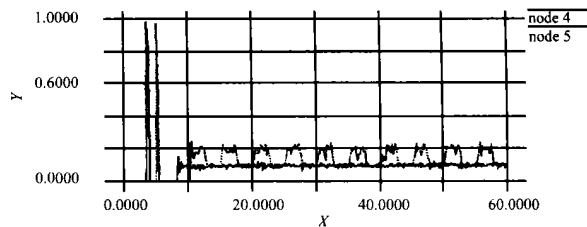


图 4 固定速率算法下接收者的丢失率

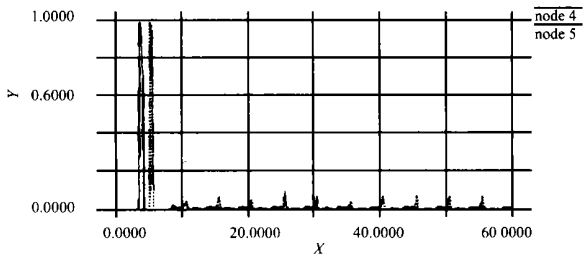


图 5 BL-RSVPa 下接收者的丢失率

(3) 带宽占用的公平性

公平性是指视频发布流与其它种类的应用所占用带宽之间的比例关系, 具体表现在公平性是指视频发布流与其它种类的应用所占用带宽之间的比例关系, 具体表现在视频信息与背景流量 CBR 和方波之间占用带宽的公平性. 实验模拟了 BL-RSVPa 算法下 CBR 流和方波的丢失率, 如图 6 所示. 图中 x 轴表示时间, y 轴表示数据丢失率, y 轴上数据 200.0000 表示丢失率为 $200.0000 \times 10^{-3} = 20\%$. 在 BL-RSVPa 算法下, 由于源端降低了增强层的发送速率, 比较固定速率算法, CBR、方波的数据丢失率明显减少, 说明 BL-RSVPa 算法通过速率调节, 表现了较好的公平性质.

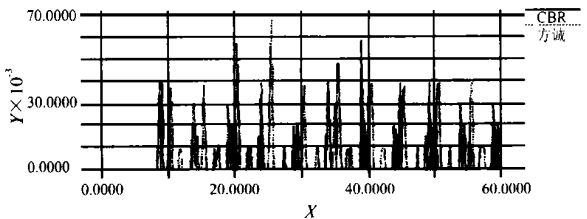


图 6 BL-RSVPa 算法下 CBR 和方波业务丢失率

值得指出, 由于 BL-RSVPa 算法引入了资源预定和分层编码速率调节机制, 源端的控制比较单纯的速率调节机制更为复杂, 此外采用反馈智能体减少源端的反馈爆炸是要求域或子网智能体牺牲一定开销.

5 结语

本文针对视频应用的特点和网络异构性之间存在的矛盾, 基于分层编码概念, 提出了一种资源预约和反馈智能体机制结合的新算法. 模拟结果表明该算法较好地保证了接收者所接收视频的基本质量, 同时较好地解决了速率适应、服务质量和公平性等问题. 由于网络的异构性和多播大规模通信组群的可扩放性问题, 导致了多点视频发布的复杂性, 因此值得进

一步研究复杂网络模型下各种算法的性能.

参考文献:

- [1] D D Clack, et al. Support real time application in an integrated service packet network: Architecture and mechanism [A]. ACM SIGCOMM [C], Baltimore, Maryland, 1992: 14– 26.
- [2] S Floyd, K Fall. Promoting the use of end-to-end congestion control. in Internet [J]. IEEE/ACM Trans. on Networking, 1999, 7(4): 458– 472.
- [3] X Li, S Paul, M H Ammar. Layered video distribution with retransmission (LVMR): evaluation of hierarchical rate control [A]. Proc. of INFOCOM [C], San Francisco, 1998.
- [4] D L Tennenhouse, et al. Towards an active network architecture [J]. Computer Communication Review, 1996, 26(2): 5– 18.
- [5] S McCarne, et al. Receiver-driven layered multicast [A]. Proc. of SIGCOMM [C], Stanford, CA., 1996: 25– 28.
- [6] 朱英, 林亚平. 视频多点播送中的代理控制分层传输算法研究 [J]. 小型微型计算机系统, 2001, 22(1): 109– 111.
- [7] Lixia. Zhang, et al. RSVP: A new resource reservation protocol [J]. IEEE Trans. on Network, 1993, 7(5): 8– 18.
- [8] UCB/LBNL/ VINT. Network simulation ns [CP]. <http://www-mash.cs.berkeley.edu/ns/>.
- [9] Célio Albuquerque, Brett Vickers, Tatsuya Suda. An End-to-End Source Adaptive Multi-Layered Multicast (SMM) Algorithm [A]. In Proc. 9th International Packet Video Workshop [C], New York, 1999.
- [10] S Floyd, V Jacobson. Random early detection gateways for congestion avoidance [J]. IEEE Trans. on Network, 1995, 1(4): 397– 413.

作者简介:

林亚平 男. 1955 年生于湖南省邵阳市. 1982 年获湖南大学学士学位, 1985 年获国防科技大学硕士学位, 2000 年获湖南大学博士学位. 现为湖南大学计算机科学系教授, 博士生导师. 主要从事计算机网络和进化计算领域的研究.

程友清 女. 1973 年生, 湖南大学计算机科学系硕士研究生, 从事 Internet 上流媒体传输技术的开发和研究工作.