IP 网络中实时数据流的流量控制

Flow Control for Real-Time Data Based on IP Network

何全胜　姚国祥

(暨南大学信息网络工程研究中心　广州510632)

Abstract　We present a Rate Adaptation Based Flow Control called RBC, which employs an additive-increase, multiplicative-decrease(AIMD) algorithm with congestion avoidance. RBC is used for Real-time Transport Protocol, provides end-to-end flow control for real time data stream.

Keywords　RBC, Real time, Rate, Bandwidth, TCP-friendly

1. 引言

近年来，全球 Internet 以指数的增长速度迅猛发展，成为仅次于电话网的第二大通信基础设施，通信量直逼电话网。音频、视频压缩、实时数据传输技术和网络技术的成熟使在 IP 网络(Internet、Intranet、LAN)上开展多媒体通信业务成为可能。VoIP 技术可以在 IP 网络上以共享网络带宽的方式提供语音、传真业务，从而拉开了与传统语音业务竞争的序幕。据 IDC 预测，在往后的2—3年里世界上10%的传真流量将通过互连网；到2002，美国国内以及国际长途电话有11%通过互联网。据 TeleGeography Inc. 的年度报告统计，2000年 VoIP 业务占整个发送流(outgoing traffic)的3.2%，预计2001年将达到5.5%[2]。然而，目前的 Internet 不保证端到端之间的延迟上限和带宽下限，因此在尽力发送的网络上发送实时业务是无法控制和预测的。

建议 H.323是目前已在工业界广为采用的 IP 网络电话系统最重要的技术基础，它是 ITU-T 制订的基于包交换网络的多媒体互通协议[3~5]。建议 H.323中，话音、图像是用 RTP (Real-time Transport Protocol)协议传输的[6,7]。RTP 协议提供端到端的传输功能，适合于在组播或者单播中传输实时的数据，例如话音、图像、仿真数据。RTP 没有提供资源预留，也不保证实时事务的 QoS。美国南加州大学信息科学院的 Reze Rejaie 等设计了一个 RAP(Rate Adaptation Protocol)协议，这是一个位于传输层的传输协议。RAP 采用 AIMD 算法调整发送速率，为实时数据流提供控制[8]。我们改进 RAP 协议，加上拥塞避免和快速恢复功能，当发现包丢失时就降低发送速率，而不等到超时倍减发送速率，使之用于 RTP 协议上，为 RTP 协议提供数据流控制。

2. RBC 控制方法原理

RBC(Rate Adaptation Based Control)利用反馈包和超时来检测包丢失，并对往还时间(Round-Trip Time RTT)进行抽样。速率调节与包丢失、RTT 两个因素有关。RBC 中包含三个功能模块：决策函数，提高/降低发送率算法和决策频率。具体算法如下：

2.1　决策函数

决策函数的功能可以概括为：①假若没有检测到拥塞，传输速率周期性地(periodically)线性递增；②假若检测到拥塞，立即降低传输速率。

象 TCP 协议一样，RBC 也维持一个 RTT 估计值，简称为 SRTT，而且计算超时和 SRTT 的算法也是 Jacobson/Karel 提出的算法[9]：

$$SRTT_{i+1}=\frac{7}{8}SRTT_i+\frac{1}{8}SampleRTT \quad (1)$$

$$Timeout=\mu\times SRTT+\delta\times VarRTT \quad (2)$$

SRTT 是平均 RTT 估计值，SampleRTT 是根据最近一次应答包计算出来的往还时间，VarRTT 是往还时间的平均偏差值(Mean Deviation)，$\mu=1$，$\delta=4$。在文[9]中有一个 C 源程序实现算法，Timeout 是重传超时。

发送方为每一个已经发送的包保留一个记录，包括序列号、发送时间、状态标记。这些记录的集合称之为传输历史记录。在发送一个包之前，发送方采用下面算法遍历传输历史记录，检查已经超时而未收到响应的包，我们认为这个包已经丢失。在这种超时情况下，发送方的发送速率应该立即减半，见公式(7)、(8)。

```
while(DepartTime_i + Timeout ≤ CurrTime){
    if(Flag_i ≠ Acked)then {
        Seq_i is lost.
        A timeout happened.
    }
}
```

为了提高协议和算法的稳定性，不至于受单个响应包丢失太大的影响，RBC 像 RAP 一样，在响应包中添加了冗余信息：1. 序列号码 A_{curr}，被应答的语音包序列号。2. 序列号 N，在 A_{curr} 之前，最近一个没有收到的语音包序列号，0表示没有包丢失。3. 序列号 A_{last}，在序列号 N 之前，已经收到的最近一个语音包序列号，0表示 A_{curr} 是第一个包。每收到一个来自接收方的响应包后，发送方将执行下面算法：

```
For each Seq_i in Trans. History
do {
    if [((A_curr ≥ Seq_i) and (Seq_i > N)) or (Seq_i = A_last)] then {
        Seq_i was received.
    }else if ((A_curr - Seq_i ≥ 3) and (Seq_i not Acknowledeged)) then {
        Seq_i is lost.
    }
}while(Seq_i ≤ A_curr)
```

发送方检测到包丢失，就应该降低速率，但不是像 RAP 协议那样把发送速率减半，而是缓慢降低，见公式(6)。同时改变它的标志为接收，这样就可以避免超时检测。既然发送方接收到响应包，就可以继续发送数据。可以把这种包丢失当作即将发生拥塞的一种预测，此时发送方进入拥塞避免阶段。在数据包超时检测时，发送速率减半导致过大地降低发送速率，降低了协议的吞吐量。当然，为了避免发送速率过渡摆动，不需每检测到一个丢失包都要降低速率。作为改进，对于同一个 SRTT 时间里的一组包丢时的情况，只需采取一次速率调整。

假设第一个检测到丢失的包为 $Seq_{firstlost}$，调整速率控制后发送的第一个包为 Seq_{adjust}。对于 $Seq_{firstlost}<Seq<<Seq_{adjust}$，发现丢失可以不用调整发送速率，如果丢失包 $Seq>Seq_{adjust}$，那么就必须降低发送速率。

在决策函数中，经过包超时检测或者包丢失检测后，对于收到了响应包的相关历史记录中的标志位要改变其状态为已经接收；对于未收到响应包，但函数检测认为已经超时或者已经丢失的包的相关历史记录中的标志位也要改变状态位，表示已经处理。

2.2 提高/降低速率算法

RBC 采用累加倍减(AIMD)来调整发送速率。在没有丢包的情况下，每调整一次，发送速率将增加一个值 α。在实现上，是通过调整 inter-packet-gap(IPG)来控制的，IPG 通过下面公式更新[8]。

$$IPG_{i+1}=IPG_i\times C/(IPG_i+C) \quad (3)$$

$$S_i=PacketSize/IPG_i \quad (4)$$

$$\alpha=S_{i+1}-S_i=PacketSize/C \quad (5)$$

S 是发送速率，C 是一个与时间有关的量，它决定了 α 的值。当检测到包丢失时，立即降低发送速率。为了不使发送速率产生大的摆动，对于丢包有两种不同处理措施。1. 接收方在收到响应包后检查历史记录，发现有丢包，那么立即降低发送速率，这个降低是通过延长 IPG 来实现的，见公式(6)，把公式(3)中的 C 取为负值，相当于减少发包。如果 IPG_{i+1} 小于0，则取其绝对值。同时取消这个包的超时设定。2. 假若发送方在发送包之前检查历史记录，发送有超时而未收到响应包则应该快速降低发送速率，β=0.5，倍减速率，见公式(7)、(8)：

$$IPG_{i+1}=-IPG_i\times C/(IPG_i-C) \quad (6)$$

$$S_{i+1}=\beta S_i \quad (7)$$

$$IPG_{i+1}=IPG_i/\beta \quad (8)$$

假若丢包非常严峻，IPG 将会成指数增长。当 IPG 增长到某一个值后，将会断开连接，这个值应该是一个连接超时值。

2.3 决策频率

发送速率不能过于频繁调整，否则会引起摆动；也不能反应太迟钝，否则不能及时反映网络状况，无法采取负责的行动。RBC 采用 SRTT 作为两个调整点之间的时间差，即步长。在一个步骤开始前，把计时器的值设定为最近一个 SRTT 的值，IPG 的值在计时器期满时才真正改变。在 RBC 中，C 取 SRTT 的值。在一个 SRTT 时间段里，除了发生丢包或者超时，真正的发送速率是不会改变的，每一次速率改变都在下一个步骤继续之前。在速度调整后，下一个 SRTT 期间将采用调整后的速率发送语音包。

3. 仿真结果分析

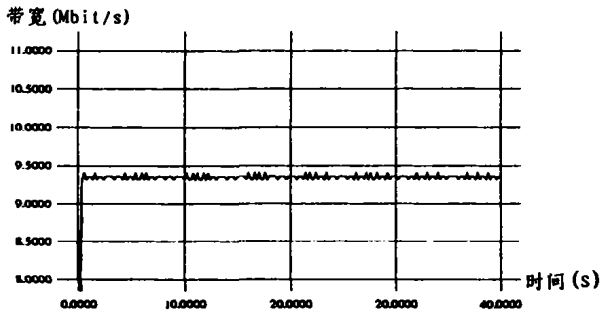

图1 NewReno TCP 带宽

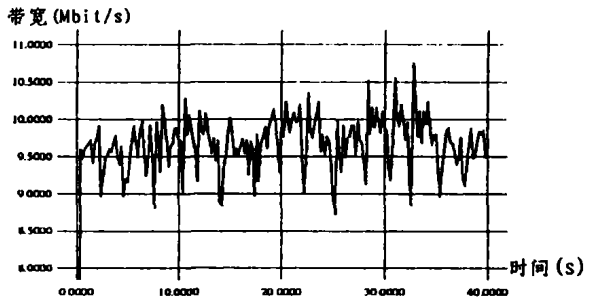

图2 RBC 带宽

本节介绍分析我们仿真的结果。仿真工具是 NS 2.0[10]，操作系统为 RedHat linux 7.0。首先，我们单独测试 NewReno TCP[11]和 RBC，比较 NewReno TCP 和 RBC 的带宽使用情况。NewReno TCP 是对 Reno TCP 的快速恢复和快速重传做了修改的一个协议。仿真环境不存在带宽瓶颈、缓冲、资源竞争等负效应，而只是测试算法的调度，端点之间的连接带宽为 10MB/S，传送延迟10ms。仿真结果如图1和图2。从图中可以看出，RBC 的带宽利用较 NewReno TCP 高，较适合实时的语音和图像传输。同时，RBC 也存在不如 TCP 的缺陷。启动过程慢，而且发送速率不如 TCP 稳定。

为了测试 RBC 与 TCP 共存的友好性，我们分别仿真测试了 RBC 与两组 TCP 共存时带宽的利用情况。测试的网络拓扑结构如图3所示。分别有 n 个 NewReno TCP 源与 Router1连接，连接带宽都是10MBit，网络延迟为6ms。同时有 n 个 RBC 源与 Router1连接，连接带宽也是10Mbit，网络延迟为6ms。接收方分别对应有 n 个 NewReno TCP 接收点与 Router2连接，连接带宽都是10MBit，网络延迟为6ms。有 n 个 RBC 接收点与 Router2连接，连接带宽也是10Mbit，网络延迟为6ms。Router1和 Router2作为路由，通过10Mbit 的带宽连接，网络延迟为5ms。路由队列算法是 RED(Random Early Drop)。

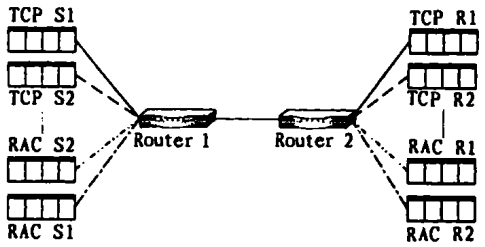

图3 仿真网络拓扑结构图

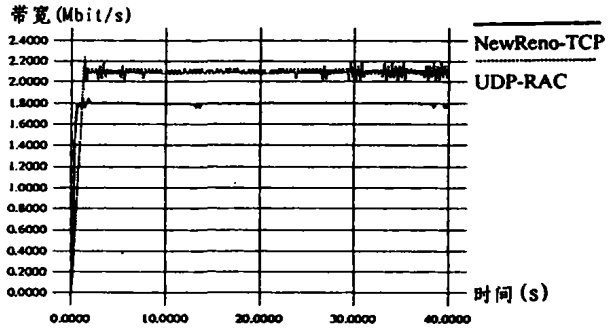

图4 RBC 与 NewReno TCP 共存

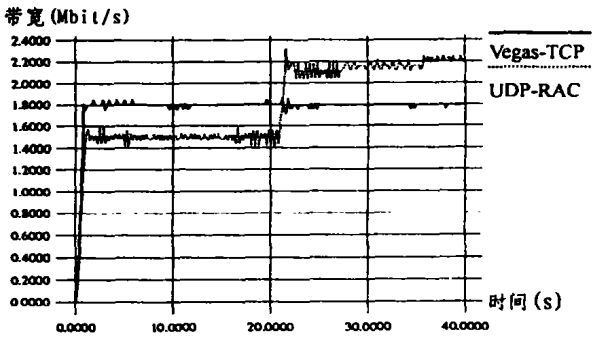

图5 RBC 与 Vegas TCP 共存

图4是测试了 RBC 与 NewReno TCP 共存时，TCP 流量与 RBC 流量带宽的占用情况。纵坐标表示带宽，横坐标表示时间。实线代表 TCP 流量，虚线代表 RBC 流量。从这个图可以看出，RBC 的带宽利用率较 TCP 要高，而且并没有太多抢占 TCP 的带宽。也看出 RBC 带宽波动比较大，不如 TCP 稳定；而且启动过程比 TCP 慢，这一点可以在以后的研究中改进。

图5比较的是另一个 TCP 协议，Vegas TCP[12]协议。Vegas 比 Reno TCP 更及时地决定重传丢失的数据包，并能预测拥塞。在这次仿真中，RBC 的带宽波动更大。开始阶段，RBC 的带宽低于 TCP 带宽；经过20秒钟后线性上升，最后达到稳定状态。这里 RBC 带宽上升是因为 RTT(Round-Trip Time)在连续几秒内降低所致，达到一个最低点后趋于稳定。

从上面的仿真结果看出，RBC 也不是极其 TCP 友好的实现算法，而只是部分 TCP 友好。带宽占用的不公平性主要有下面原因：1. TCP 的内部性能限制；2. 实现时的参数配置；3. RBC 抢占了部分带宽。TCP 存在性能限制。特别在一个窗口里同时多个数据包丢失或者窗口小于4时，TCP 或者等待重传超时或者进入慢启动状态。当然，RBC 在实现时还是能控制带宽抢占，不是不负责任的实现算法。虽然我们没能对 RAP 协议进行仿真，但从文献资料结果来看，我们的改进方法确实起到预期作用，带宽利用率高，而且比之稳定，能快速恢复。

结束语 在早期的网络中，由于 TCP 协议中没有拥塞避免和控制机制，过度重发、抢占资源等因素使得网络利用率很低，这促使了网络控制机制的研究。发送多媒体包的 RTP 协议底层通过 UDP 传播，也没有任何控制。为了提高话音的 QoS，我们在 RTP 协议上加了一个流量控制。基于速率控制的 RBC 算法在一定程度上要优于基于窗口控制的算法。加上 RBC 的 RTP 是一个负责的传输协议。对于目前网络流量绝大部分是基于 TCP 的情况，RBC 基本上还是一个 TCP 友好的实现方法。同时 RBC 也不失 UDP 的优点，能够提供实时的多媒体传输。

当然，我们的工作还需要进一步改进，RBC 还无法处理多个响应包同时丢失的情况。而且 RBC 还不够稳定，带宽波动比较大。为了更好地提高接收端的话音和图像质量，可以在接收方加上队列缓冲，降低传输延迟带来的延迟抖动和话音图像间隙。采用好的算法对丢包情况进行恢复，使得话音图像能够更平滑、自然地播放。

参 考 文 献

1 ITU INTERNET REPORTS 2001: IP TELEPHONY. http://www.itu.int/ti/index.htm
2 IP Telephone Design and Implementation. http://www.ti.com/
3 ITU-T RECOMMENDATION H.323 Packet-based Multimedia Communications Systems
4 舒华英，赖平漳，等编著. IP 电话技术及其应用. 人民邮电出版社，1999
5 糜正琨编著. IP 网络电话技术. 人民邮电出版社，2000
6 ITU-T RECOMMENDATION H.225 Media Stream Packetization and Synchronization on Non-Guaranteed Quality of Service LANs. ITU-T. May 1996
7 Schulzrinne H, et al. RTP: a Transport Protocol for Real-Time Applications. IETF RFC1889, Jan. 1996
8 Rejaie R, Handley M, Estrin D. RAP: An End-to-end Rate-based Congestion Control Mechnism for Realtime Stream in the Internet. Proc. IEEE Infocom, March 1999
9 Jacobson V, Karels M J. Congestion avoidance and control. In ACM SIGCOMM '88, 1988
10 The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/
11 Floyd S, Henderson A T. RFC2582 The NewReno Modification to TCP's Fast Recovery Algorithm. U.C. Berkeley. April 1990
12 Brakmo L, Peterson L. TCP Vegas : End to End Congestion Avoidance on a Global Internet. IEEE Journal on Selected Areas in Communication, 1995, 13(8): 1465～1480

(上接第46页)

·电子白板数据的同步反馈，从而更好地支持底层可靠组播协议拥塞控制的实现。

·增加新的媒体类型，如动画、视频、XML 文档等。

我们将在今后的工作中逐步改进现有系统，增加新的功能，从而更好地支持基于组播的协同工作。

参 考 文 献

1 Prakash A, et al. Data Management Issues and Tradeoffs in CSCW Systems. 1987
2 McCanne S. A Distributed Whiteboard for Network Conferencing. May. 1992
3 Tung T-L. MediaBoard: A Shared WhiteBoard Application for the Mbone
4 Raman S, McCanne S. Scalable Data Naming for Application Level Framing in Reliable Multicast, 1998
5 Floyd S, et al. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. In: Proc. of ACM Sigcomm' 95. 342～356
6 Liu Ching-Gung. A Scalable Reliable Multicast Protocol: [Ph. D]. Dissertation Proposal, 1995
7 Tang J C. Findings from observational studies of collaborative work. International Journal of ManMachine Studies, 1991, 34: 143～160
8 Clark D, Tennenhouse D. Architectural Considerations for a New Genenration of Protocols. In: Proc. of ACM SIGCOMM '90, Sep. 1990. 201～208
9 吴文峻，等. 可靠组播研究综述. 计算机科学，2001(2)
10 UCL MICE project http://www-mice.cs.ucl.ac.uk/
11 UCB Mash project http://www-mash.cs.ucb.edu/