TCP 友好速率控制协议的分析及应用

曾晶萍,杨文俊,彭 力,秦肖臻,汪秉文
(华中科技大学 控制科学与工程系,湖北 武汉 430074)

摘 要:传统的 TCP 由于采用速率减半的拥塞退避机制而使其在数据传输时易产生过大的速率波动,而 UDP 不具备拥塞退避机制,在拥塞的网络环境中,UDP 流将大量抢占 TCP 流的网络带宽,同时自身的丢包也迅速增加,并可能带来系统拥塞崩溃的潜在危险,因此 TCP 和 UDP 都不能很好地满足实时流媒体业务的需要。文中研究了一个具有拥塞退避机制、网络吞吐量波动小,且能够与 TCP 协议公平分享带宽的传输协议——TCP 友好速率控制协议(TFRC),并将其应用于实时多媒体流传输应用程序。研究测试结果表明采用 TFRC 后多媒体的实时播放较 TCP 平滑了许多。

关键词:TCP 友好性;拥塞控制;吞吐量
中图分类号:TN915.04 **文献标识码**:A **文章编号**:1673-629X(2007)01-0210-03

Analysis and Application of TCP-Friendly Rate Control Protocol

ZENG Jing-ping, YANG Wen-jun, PENG Li, QIN Xiao-zhen, WANG Bing-wen
(Dept. of Control Science and Engineering, Huazhong Univ. of Science and Tech., Wuhan 430074, China)

Abstract: TCP has a congestion control mechanism of halving its rate, which makes data transmission rate fluctuate badly, while UDP does not have any congestion control mechanism, which means UDP flows will occupy too much bandwidth over TCP flows. At the same time increase its own package loss when the network is congested and make the network environment even worse. So both TCP and UDP are not suitable for real-time streaming application. Analyses TCP-friendly rate control (TFRC), which is a congestion control mechanism and is reasonably fair when competing for bandwidth with TCP flows. It has a much lower variation of throughout over time than TCP, thus it is more suitable for multimedia streaming application. A real-time streaming application using TFRC is given in the paper. The testing results reveal that the smoothness has been improved.

Key words: TCP-friendly; congestion control; throughout

0 引 言

实时媒体流业务需要一个稳定的网络传输速率,以使其在播放端可以平稳流畅地播放,达到用户所期望的播放质量。当前 Internet 网的数据传输业务基本上都是基于 TCP 的。TCP 采用速率减半的拥塞退避机制[1],这很容易引起数据流过大的速率波动,对多媒体的传输是非常不利的。文献[2]的研究表明在传输过程采用 TCP/IP 协议在用户较多时回放将发生延迟和不连续现象。而 UDP 不具备拥塞退避机制[1],在拥塞的网络环境中,UDP 流将大量抢占 TCP 流的网络带宽,同时自身的丢包也迅速增加,并可能带来系统拥塞崩溃的潜在危险。因此,TCP 与 UDP 协议都不能很好地满足实时媒体流业务的需要。随着 Internet 中多媒体实时业务的迅速增长,研究一个适合于多媒体传输,并具有拥塞退避机制,能够与 TCP 协议公平分享带宽的传输协议,成为了 Internet 传输的一个重要课题。

文中研究的 TCP 友好速率控制[3](TCP-Friendly Rate Control, TFRC)正是这样一种协议。它基于数学模型,由发送方根据网络环境调整数据流的发送速率,进而达到拥塞控制的目的。在同等条件下,TFRC 流具有与 TCP 流近似相同的吞吐量,因此,可以"公平地"与 TCP 共享网络带宽。另一方面,TFRC 吞吐量变化稳定、抖动较小,因此更加适合电话、流媒体等对传输速率的平滑性要求较高的应用。

1 TCP 友好速率控制算法

1.1 TFRC 的拥塞控制机制

TFRC 适用于固定数据包大小的应用程序,它根据网络环境的好坏,通过调整每秒钟发送的数据包数来调整数据传输速率。TFRC 是基于接收方的机制,它在接收方计算拥塞控制信息,例如丢包事件率等。

TFRC 的拥塞控制机制[3,4]如下:

* 数据接收方测量丢包事件率 p,然后将其与时间

收稿日期:2006-03-30

作者简介:曾晶萍(1982-),女,河南郑州人,硕士研究生,研究方向为计算机网络、数据共享与应用;秦肖臻,副教授,研究方向为生产过程综合自动化系统集成优化与决策、计算机网络和数据库系统;汪秉文,教授,研究方向为生产过程综合自动化系统集成优化与计算、计算机网络和数据库系统。

戳一起反馈给发送方;

* 发送方利用反馈信息中的时间戳测量回环时间 RTT;

* 将丢包事件率 p 和 RTT 带入 TFRC 的吞吐量方程,经计算得到一个传输速率;

* 发送方然后根据这个计算得到的速率来调整其数据发送速率。

1.2 TFRC 吞吐量方程

基于"公平性",TFRC 与 TCP 共享带宽,因此 TFRC 采取 TCP 的吞吐量计算公式[3],如下:

$$X = \frac{s}{R\sqrt{\frac{2bp}{3}} + 3t\_\mathrm{RTO}\sqrt{\frac{3bp}{8}}\,p(1+32p^2)} \quad (1)$$

其中:

* X 为数据传输速率,单位:B/s;

* s 为数据包的大小,单位:B;

* R 为回环事件 RTT,单位:s;

* p 为丢包事件率,即一个发送窗口内所丢的数据包数与发送的总包数的比值,$0 \leqslant p \leqslant 1.0$;

* $t\_$ RTO 为 TCP 超时重传时间,单位:s,取 $t\_$ RTO $= 4 * R$;

* b 为接收端一次确认的包数,一般取 $b = 1$。

1.3 数据发送协议

1.3.1 数据发送方发送的数据包

```
typedef struct_ sendPacket {
  unsigned long seq; //序列号
  time_t RTT; //回环时间,单位为 ms
  * byte pdata; //数据
} Sends;
```

1.3.2 数据发送协议

数据发送方首先初始化,然后开始数据发送。数据发送分为 2 个阶段:慢速启动阶段和拥塞控制阶段。

慢速启动阶段为发送的初始阶段,此时发送速率比较慢,此时丢包事件率 p 为 0。在此阶段发送方每 RTT 将发送速率翻倍,直到有丢包事件发生,就进入拥塞控制阶段。在慢速启动阶段,发送方保证最少每秒钟发送一包。

拥塞控制阶段,发送方以一定的速率将数据包流发送给接收方。当接收到数据接收方的反馈数据包后,发送方根据反馈信息中的时间戳估算出新的回环时间,并由式(1)计算出的速率相应地改变发送速率。如果在两倍的回环时间内没有接收到反馈包,则将其发送速率减半。拥塞控制阶段保证每 64 秒最少发送一包数据[3]。

1.4 数据接收协议

1.4.1 数据接收方发送的反馈数据包

```
typedef struct feedbackPacket {
  time_t t_recvdata; //接收到数据包的时间
  time_t t_delay; //处理延时
  unsigned long x_recv; //数据接收速率
  double p; //丢包事件率
} Feedbacks;
```

1.4.2 数据接收协议

正常情况下,接收方每 RTT 发送一次反馈数据包。当发送方发送速率小于 1 包/RTT 时,每收到一个数据包时就发送一个反馈包。一旦检测到有丢包事件的发生,就立刻发送反馈数据包而不必等到一个回环时间结束。当接收到一个无序的数据包时,就将其从丢包事件的历史记录中将其删去。

数据接收方的主要功能模块如下:

* 接收方初始化。接收方在收到第一个数据包后进行初始化。

* 数据接收方接收到数据包。

处理步骤流程如图 1 所示。

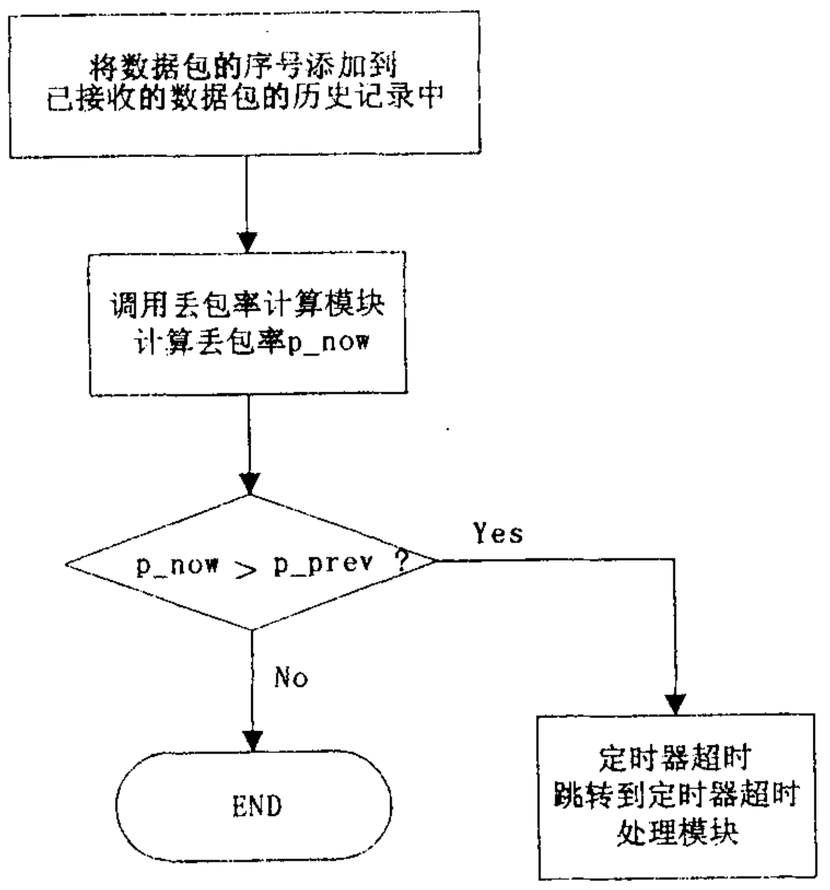

图 1　接收方接收到数据包时的处理过程

* 数据接收定时器超时处理。

数据接收方对定时器超时的处理根据上次发送反馈信息后是否收到发数据包将分别进行处理。若收到数据,则其处理步骤[3]如下:

1) 计算平均丢包事件率 p;

2) 计算数据包的接收速率:

$$x\_\mathrm{recv} = \frac{\text{接收到的数据包数}}{R\_m};$$

3) 发送反馈数据包;

4) 重置定时器在 R_m 秒后超时。

若没有收到,则接收方不发送反馈数据包,只重置定时器超时时间。

1.4.3 丢包事件率的测量

在 TFRC 中,称一个 RTT 内丢失一个或多个包为一次丢包事件(loss event)。丢包事件率测量在接收端被执行。丢包事件率的测量按以下步骤[3]进行:

1) 发现丢包;

2) 判断不同的丢包是否属于同一丢包事件;

3) 用加权平均法计算平均丢失事件间隔;

4) 丢包事件率为平均丢失事件间隔的倒数。

2 采用 TFRC 进行实时多媒体传输的应用程序

应用程序采用传统的客户端/服务器模型。服务器端是多媒体视频源,负责向各客户端发送多媒体视频,客户端是多媒体视频的接收端和播放端。服务器端在固定的端口监听客户端的连接请求,当接收到客户端的连接请求之后,就与客户端建立连接,然后开始向客户端发送多媒体数据。客户端在发送连接请求并得到响应,建立连接后,就开始接受服务端发送过来的多媒体数据。在数据传输时,采用 UDP+TFRC 方式,即传输层采用 UDP 协议,应用层采用 TFRC 拥塞控制策略进行传输速率的控制。

应用程序数据传输部分的模块框架如图 2 所示。

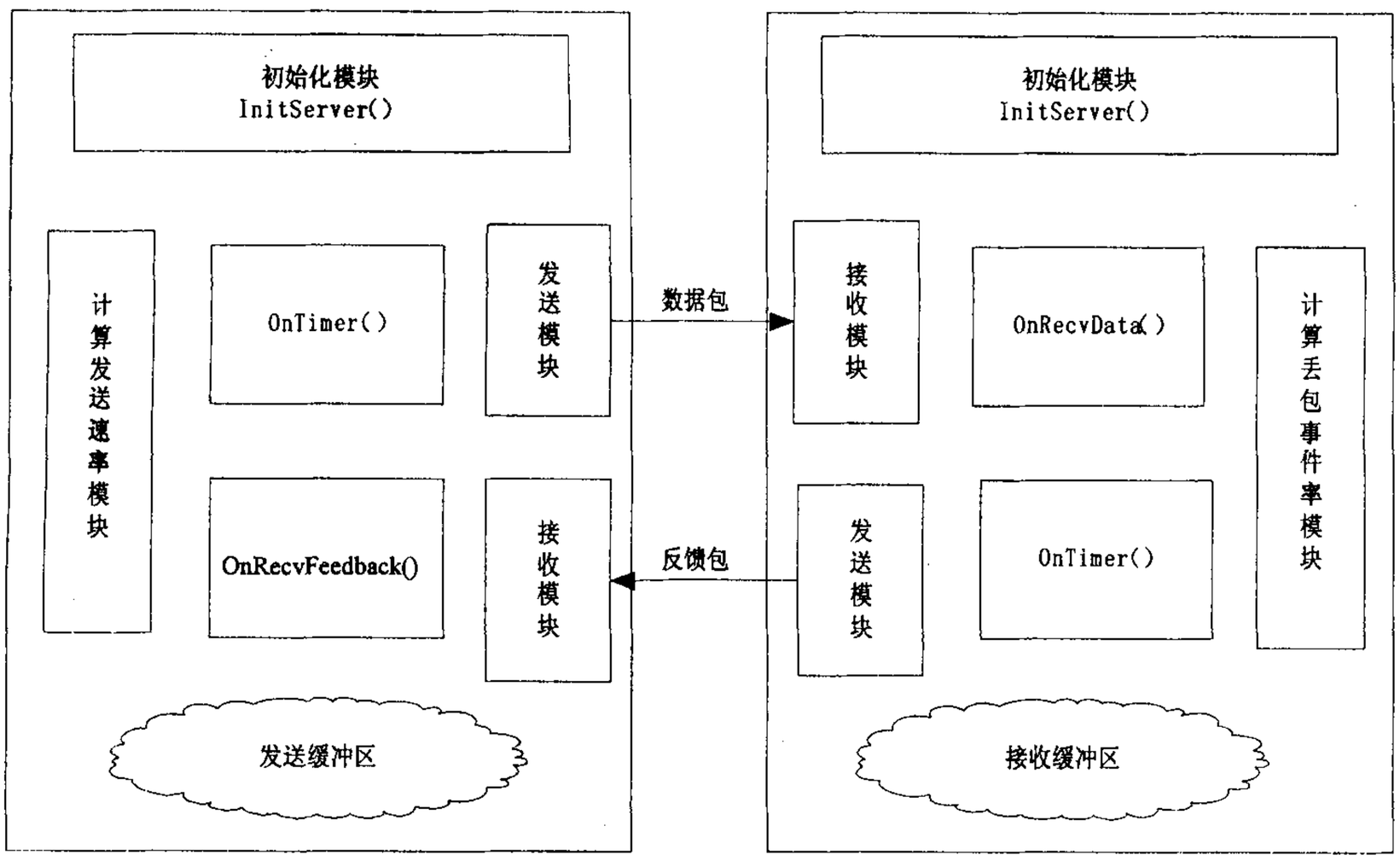

图 2 数据传输部分的模块框架图

服务器端为每一个连接上的客户端设置一个定时器。当定时器超时前收到客户端方发来的反馈包,就调用 OnRecvFeedback()模块进行处理;如果当定时器超时时,服务器没有接收到反馈,将调用定时器超时处理模块 OnTimer()。在多媒体数据传输方面,服务器端的计算量主要集中在发送速率计算方面,因此,将发送速率计算单独作为一个模块供 OnRecvFeedback()和 OnTimer()调用。

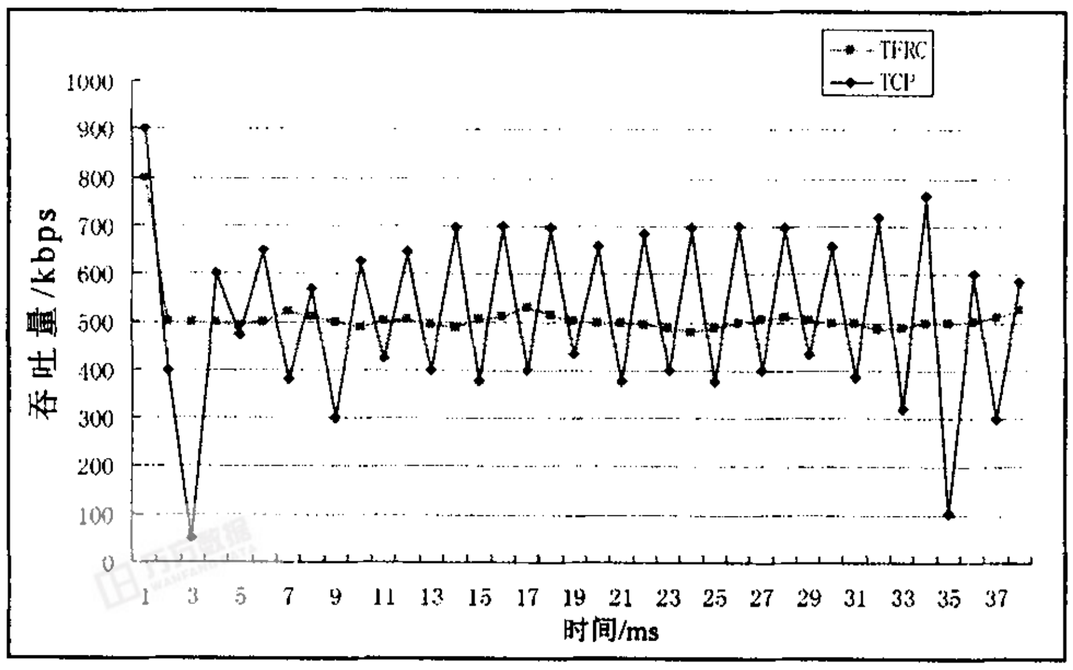

图 3 TFRC 与 TCP 吞吐量对比

每个客户端也都设置一个定时器。在定时器超时前接收到服务器发来的数据包,则调用 OnRecvData()模块进行处理,否则,定时器超时就调用定时器超时处理模块 OnTimer()。对于客户端来说,最主要的就是丢包事件率的计算。因此将计算丢包事件率单独作为一个模块。

对该应用程序进行了测试,如图 3 所示。采用 TFRC,网络吞吐量的波动明显比 TCP 小,应用程序的实时播放也较 TCP 平滑了许多,等待和缓冲的次数明显减少。

3 结 论

由此可见,TFRC 在与 TCP 公平竞争带宽的前提下,平滑网络吞吐量抖动方面具有比 TCP 明显的优势。

TFRC 把在同一个 RTT 中所有的丢包都算作一个丢失事件。在 TFRC 中并没有用丢失的包数除以发送的总包数这样简单的方法来计算丢包率,而是引入了“丢失事件”的概念。“丢失事件”的概念的引入,可以减少丢包的偶然性对发送速率的影响,这是减小吞吐量抖动的一个原因[5]。TFRC 的吞吐量变化缓慢的另一个重要原因是 TFRC 在计算丢失事件率时采用了加权平均丢失间隙法[5]。间隙即为相邻丢失事件的时间间隔。在 TFRC 中,丢失事件率是平均时隙的倒数;平均时隙的计算是通过多个时隙进行加权平均得到的,即在计算时当前的时隙所占的权值大,历史越久的权值越小。这样,丢失事件率的变化就比较平滑,而且与当前的丢失事件关联比较紧密。

由于保证了传输的平滑性,所以当可用带宽发生变化时,TFRC 在反应上就较 TCP 迟钝。

参考文献:

[1] 谢希仁. 计算机网络[M]. 第 4 版. 北京:电子工业出版社, 2003.

[2] 王 鹏,季桂树,张志武. 基于 TCP 协议控制的实时视频流传输系统[J]. 微机发展, 2005,15(10):93-95.

[3] Handley H, Floyd S, Padhye J, et al. RFC 3448 TCP Frien-dly Rate Control (TFRC) Protocol Specification[S]. [s. l.]:[s. n.], 2003.

[4] 李 骐,陈 涤,高振明. TFRC 拥赛控制策略的分析和改进[J]. 计算机工程,2005,31(8):108-110.

[5] 王光阳,徐昌彪,陈前斌,等. TFRC 与 TCP 流数之比对协议间公平性影响的研究[J]. 计算机应用研究,2005(11):16-18.