一种拥塞感知的 TFRC 协议慢启动算法

蒋 翊[1],吴春明[1],姜 明[2]
(1. 浙江大学人工智能研究所,浙江杭州 310027;2. 杭州电子科技大学计算机学院,浙江杭州 310018)

摘 要: 本文分析了 TFRC(TCP-Friendly Rate Control)协议在慢启动阶段采用类似 TCP 协议的倍增发送速率机制存在的问题,提出了一种利用回路响应时间(Round Trip Time,RTT)来自适应调节慢启动阶段速率的算法.通过分析实际 RTT 值和 EWMA(Exponentially Weighted Moving Average)处理后的平均 RTT 值来感知网络当前的拥塞状况,以调节发送速率的激进程度.仿真实验表明,该方法对 TFRC 协议具有明显的改进作用,减少了慢启动阶段结束时的报文丢失率,提高了协议的传输平稳度和吞吐量,从而能更有效地适应多媒体流的传输要求.

关键词: TFRC 协议; 慢启动; 拥塞感知
中图分类号: TP393 **文献标识码**: A **文章编号**: 0372-2112 (2009) 05-1025-05

A Congestion Aware Slow-Start Algorithm for TFRC Protocol

JIANG Yi[1],WU Chun-ming[1],JIANG Ming[2]
(1. *AI Institute of Zhejiang University, Hangzhou, Zhejiang* 310027, *China*;
2. *Computer Science College of Hangzhou Dianzi University, Hangzhou, Zhejiang* 310018, *China*)

Abstract: This paper analyzes the problem of TFRC (TCP-Friendly Rate Control) protocol in slow-start phase in which the sending rate increases exponentially like that in the TCP protocol, and proposes a method of using round trip time (RTT) to adjust the sending rate adaptively in this phase. The method can detect the network congestion degree by comparing the sample RTT with the average RTT and then adjust the aggressiveness of sending rate in slow-start phase. The simulation results indicate that with this method, TFRC reduces the packet loss rate before entering the congestion avoid phrase, and improves the throughput and smoothness property, thus meeting the requirements of multimedia stream more effectively.

Key words: TCP-friendly rate control (TFRC); slow-start; congestion aware

1 引言

近年来计算机网络的爆炸式发展,特别是大量多媒体实时业务的广泛应用,对传统的以 TCP 为主的传输层协议提出了新的挑战.大部分多媒体应用对时延很敏感,但却具有一定的容错机制,能够容忍少量的报文丢失.此类应用要求传输协议开销小、响应快,并且不必保证传输完全可靠.此外,抖动也是对多媒体应用有较大影响的问题,而 TCP 拥塞控制使用的 AIMD 策略,会引起发送速率很大的抖动.目前多媒体实时应用多采用 UDP 协议来传输,但由于 UDP 协议缺乏拥塞控制机制,当基于 TCP、UDP 的应用共享网络带宽时,TCP 流因具有拥塞退避机制而导致"饥饿",而 UDP 将不公平地获得大量带宽,同时可能导致网络拥塞,影响 Internet 的鲁棒性,因此随着 UDP 协议的应用在 Internet 所占的比例的增加,其不足之处逐渐暴露.

针对 Internet 上支持实时多媒体流传输所存在的上述问题,研究人员提出了 TCP 友好拥塞控制协议(TCP-Friendly 协议),TCP 友好的定义为"非 TCP 流在长期范围内吞吐量不超过相同情况下的 TCP 流的吞吐量".与 TCP 相比,TCP 友好拥塞控制算法在速率调节和对拥塞进行响应时具有更好的平滑性,更适合于多媒体流的传输.根据 TCP 友好协议的机制特点不同[1],当前对 TCP 友好协议进行分类的主要依据有:

(1) 基于窗口的 TCP 友好协议和基于速率的 TCP 友好协议.

(2) 单播的 TCP 友好协议和组播的 TCP 友好协议.

(3) 单速率的 TCP 友好协议和多速率的 TCP 友好协议.

(4) 端到端的 TCP 友好协议和需要路由器支持的 TCP 友好协议.

TFRC 是其中一种基于公式控制发送速率的端到端单播 TCP 友好协议,IETF 于 2003 年组织编写了关于

收稿日期:2008-06-26;修回日期:2008-10-26
基金项目:国家自然科学基金(No. 60773182);国家"863"重大项目课题(No. 2008AA01A323);浙江省科技计划项目(No. 2007C21031)

TFRC 协议的 RFC3448 文档[2]. 近年来该协议受到国内外研究人员的广泛关注与深入研究,主要集中于 TFRC 性能分析与改进,另外也有很多学者也尝试将其用于 Ad-hoc、音频实时传输、无线网络、Diff-Serv 网络等领域.

2 TFRC 协议机制介绍

Padhye 等人[3]给 TCP Reno 建立了著名的流量模型

$$B(p) \approx \min\left(\frac{W_{\max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + t_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)}\right) \quad (1)$$

其中 $B(p)$ 为平均享有带宽; $W_{\max}$ 为接收端的窗口通告值; RTT 为平均 round-trip time; b 为单个 ACK 回复所确认的包的数量,通常的 TCP 实现设定为 1 或 2; p 为分组丢失率; t_o 为 TCP 传输超时的时间设定.

TFRC 采用以上公式来调整发送端的发送速率,目标是达到与相同的网络竞争环境下的 TCP 传输数据流相比具有基本相同的吞吐量,即是 TCP 友好的,同时避免 TCP 的抖动性等不利因素. 其主要的控制机制如下:

接收端测量丢失事件率、接收报文的时间戳,将其反馈给发送端.

发送端利用接收端反馈过来的信息根据公式

$$R_{sample} = (t_{now} - t_{revdata}) - t_{delay}$$

计算出当前 R_{sample}, R 初始值设为 R_{sample},然后每收到一个报文则按照以下公式计算

$$R = q * R + (1 - q) * R_{sample}$$

式中 R 即用来计算发送速率的参数 RTT,而 q 一般被设为 0.95.

接收端根据 RTT 以及报文的时间戳,决定多个丢包是否归为相同的丢失事件,而丢失事件则决定了怎样调整发送速率.

如果 $p = 0$,则进入慢启动阶段,即每个 RTT 对发送速率倍增,直到出现报文丢失事件. 如果反馈计时器超时则发送速率直接减半,这种情况虽然少见,但它的出现往往标志着网络中的严重拥塞,必须立即减小发送速率.

如果丢失事件率 $p > 0$,则发送端把丢失事件率和 RTT 和 t_{RTO}(一般直接设为 $4 * RTT$)代入由式(1)在通常情况下的简化公式(2):

$$x = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO} * 3\sqrt{\frac{3bp}{8}} p(1+32p^2)} \quad (2)$$

计算出合适的带宽.

TFRC 的以上机制确保了它在大部分网络情况下具有稳定的传输性能,但同时也不可避免的存在一些问题,如 RTO 估计存在缺陷,对报文丢失响应较慢使丢失事件率 p 不同于 TCP,以上偏差导致用于 TFRC 计算发送速率的参数与相同网络中的 TCP 流有差异,影响了 TFRC 的公平性,而且在较高拥塞程度时会导致吞吐量严重下降[4,5].

从以上介绍不难看出,TFRC 用来退出慢启动阶段的标志即为报文丢失,这一点与 TCP 协议相同.

Qi. Li 等人[6]提出一种在 TFRC 接收端利用 RTT 抖动判断由于节点队列引起的拥塞度增加,从而预测丢包的方法,即利用网络节点队列造成的网络传输延迟变化,以一定阈值设置最大的可接受抖动程度,如超出此阈值则认为潜在的丢包即将到来,随即调整发送速率,并预测进入拥塞避免阶段的初速度,使发送速率从慢启动到拥塞避免阶段过渡更趋平稳.

耿福泉等人[7]也提出了一种基于 RTCP 反馈的拥塞控制机制,改进了 TFRC,主要针对实时流媒体要求低延迟和低抖动的特点修改了 TFRC 的慢启动阶段,使得 TFRC 流开始的发送速率不至于太低而使得接收端在开始有较大的延迟和抖动.

3 慢启动性能分析与改进

为了验证慢启动机制对 TFRC 协议的影响,我们使用 NS-2.31 对 TFRC 与 TCP 协议竞争时的情况进行仿真实验[8],实验采用哑铃型网络拓扑,各参数如表 1:

表 1 实验基本场景参数

参数名称	参数值
瓶颈带宽	1Mb
瓶颈延迟	50ms
受监控 TFRC 链路带宽	1Mb
TFRC 使用应用层	CBR
受监控 TCP 链接带宽	1Mb
TCP 使用应用层	FTP
所有边缘链路延迟	2ms～5ms
实验时间	120s

对实验中丢包情况监测结果如下:

图 1 所示为 TFRC、TCP 协议每秒丢包绝对数量的情况. 从图中可以看出,由于慢启动时没有考虑网络中剩余带宽而不断倍增发送速率,如果网络可利用带宽小于应用所需带宽,则在经历第一次报文丢失时由于传输速率相对过高,导致短时间内报文丢失的绝对数量明显大于其他时间段,有可能造成短时间多媒体实

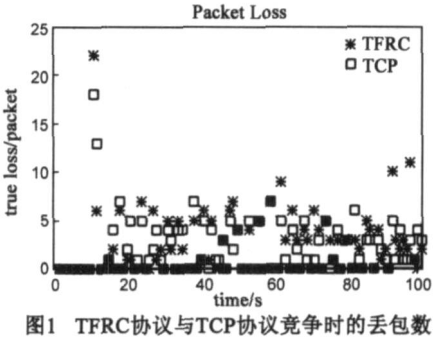

图1 TFRC协议与TCP协议竞争时的丢包数

时流的质量下降,如果 TFRC 接入网络前网络状态已经比较拥塞,则这种现象并不明显;相反,如果 TFRC 接入网络前网络状态比较好,则会加剧这种现象.

假如 TFRC 协议在慢启动阶段能对网络拥塞状况加以考虑,拥塞程度逐渐增大的时候适当减小发送速率,以避免首次的较高丢包率,从而使 TFRC 在进入拥塞控制阶段时根据式(2)计算出的发送速率增大,减小了慢启动结束并进入拥塞控制阶段发送速率的剧烈波动,同时还可以提高传输吞吐量.

为了监测网络中可以用来表征拥塞程度的回路响应时间 RTT,实验在静态路由 5 个 long-term TCP 背景流的网络仿真环境下测得 TFRC 协议 RTT 的 EWMA 均值与实际当前 RTT 的比值(曲线 1), $\sqrt{RTT}$均值与 $\sqrt{RTT}$ 的比值(曲线 2)具有以下特点:

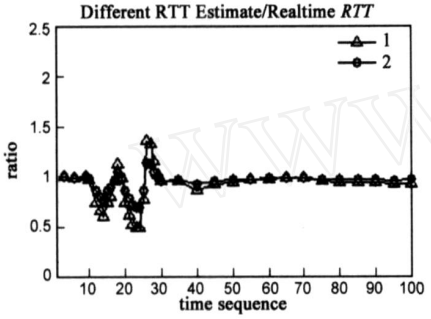

图2　TFRC协议RTT的EWMA、平方根均值及当前值的比值曲线

图 2 反映出随着节点队列不断增加使得 *RTT* 不断增加,各数据流如果有拥塞控制,则会为应对这种情况减小发送速率,随着节点队列缩短,各协议传输所需的 *RTT* 值也会下降. 在 NS 仿真实验中实时采样得到的 *RTT* 值与其 EWMA 均值的比值, $\sqrt{RTT}$及 $\sqrt{RTT}$均值的比值一般在 1 附近波动变化,后者由于加入平方根,变化幅度较前者小. 以上情况在不断变化的实际网络中可能有一定差异. 基于以上考虑,我们对 TFRC 的慢启动阶段做了如下改进.

既然队列长度能反映到 *RTT* 变化中,我们以真实 *RTT*(以下称为 RTT_{now})和 *EWMA* 处理后的 *RTT*(以下称为 RTT_{ewma})作为参数,则慢启动阶段发送速率的公式:

$$R_{inst} = \left(1 + \min\left(\frac{RTT_{ewma}}{RTT_{now}}, k\right)\right) * \text{rcvrate} \qquad (3)$$

其中 R_{inst}是当前的实际发送速率,rcvrate 是根据公式(2)计算出的发送速率, k 为常数, k 值大于 1 时能使 TFRC 在当前网络 *RTT* 减小时更快速的获得带宽,但同时又可能因为网络中节点队列长度的剧烈变化而导致加剧慢启动结束时的突发性丢包; k 值小于 1 则 TFRC 慢启动阶段竞争力相对改进前更加保守. 实验中经过比较我们将其定为 1.5,根据以上公式来调节实际的发送速率时,如果 *RTT* 与 EWMA 处理后的平均 *RTT* 相比增加了,则实际发送速率将动态的根据当前网络中的拥塞情况加以调整,介于 1～2.5 倍的 *rcvrate*,而不是简单的等于 2 倍 *rcvrate*.

4　NS 仿真性能对比

为了对改进效果进行较全面的衡量,我们基于 NS 仿真实验对 TFRC 和 TCP 流竞争时的主要性能指标进行了分析. 网络实验场景与表 1 中所示的相同. 因为 TCP Vegas 流的带宽竞争力不及 TCP Reno 流,Reno 和 Vegas 共存时,Reno 会占用更多的带宽,同时 Vegas 算法本身也存在一些问题,如不公平性问题[9],导致了 TCP Vegas 并没有在互联网上大规模使用,所以在仿真时我们仅考虑 TFRC 协议和 TCP Reno、TCP New Reno、TCP Sack 协议竞争时,TFRC 协议改进前、后主要性能的比较.

4.1　丢包率

通过在慢启动过程中使用式(3)动态调整真实的发送速率,可以降低慢启动结束时发生的第一次丢包量,其中与图 1 相同场景改进后的结果如图 3.

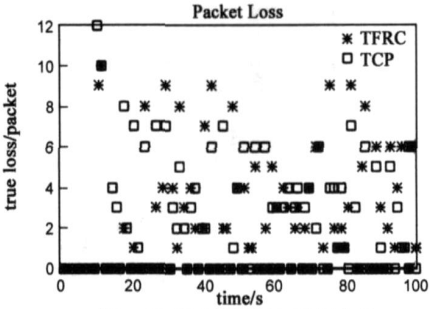

图3　改进后每秒丢包数量曲线

比较图 1 与图 3,图 1 中未改进前第一次丢包开始的 1 秒内共计丢包 22 个,而改进后在连续的两个 1 秒内分别丢包 10 个和 9 个,无论是该时间段内丢包总量和丢包集中程度都较改进前有所提高. 这也证实了前述理论论证得到的结果,即避免一定拥塞程度下过快增长发送速率可以降低慢启动结束时的连续丢包数.

基本场景采用表 1 所示,我们针对有代表性的场景改变做了同样实验,结果如下:

表 2　改进前后慢启动结束时丢包情况多场景对比

场景变化	慢启动结束时的丢包数	
	改进前	改进后
100ms 瓶颈延迟	16	19
200ms 瓶颈延迟	16	15
2Mb 瓶颈带宽	40	30
RED 队列管理	20	16
RED-ECN 队列管理	18	14
vs. TCP New Reno	22	8 + 10
vs. TCP Sack	22	8 + 10

表 2 中改进后“8 + 10”表示连续两个 1 秒内都有丢包,数量分别是 8 和 10,而其他情况则是仅在 1 秒内有丢包. 根据表 2 中所示的改进前后情况对比,大部分网络情况下该方法降低了慢启动结束时丢包数.

4.2 吞吐量

另一方面,为了确定数据是由慢启动改进而产生变化的,实验仅运行 120 秒,TCP 和 TFRC 协议均在 20 秒时开始发送数据. 分别改变网络延迟和逐渐添加 TCP long-term 背景流以增加网络拥塞的传输吞吐量,实验结果如图 4、图 5 所示:

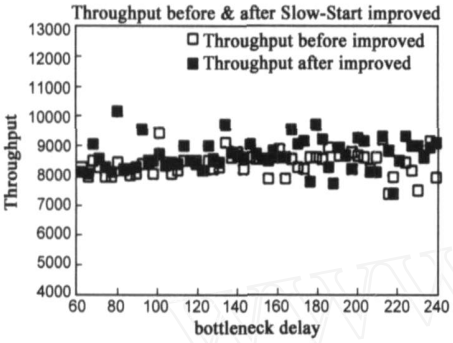

图4 瓶颈带宽不同时TFRC的吞吐量变化

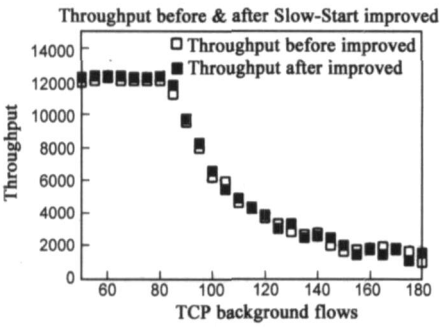

图5 TCP背景流数不同时TFRC的吞吐量变化

图 4 及图 5 反映了改进后效果并不是在所有场景中都优于改进前的,但在大部分情况下吞吐量有所提升. 图 4 中改进前所有场景共计发送 552480 个数据包,而改进后共计发送 580433 个数据包,约提高 5 %;图 5 中改进前所有场景共计发送 286571 个数据包,而改进后共计发送 291631 个数据包,约提高 1.8 %.

4.3 平稳度

对协议传输速率平稳性的评价主要采用基于标准差的公式:

$$CoV = \sqrt{\sum_{i=1}^{n} (Xi + \bar{X})} / \bar{X}$$

此公式得出的 *CoV* 值的意义是传输速率标准差(Standard Deviation)与均值的比值,是一个无量纲的值,此值反映出传输速率的波动,值越小表示平稳性越好,也是 TFRC 所需要达到的目标之一.

仿真结果表明,改进 TFRC 的慢启动速率增长方式同样提高了传输平稳度(即 *CoV* 值下降).

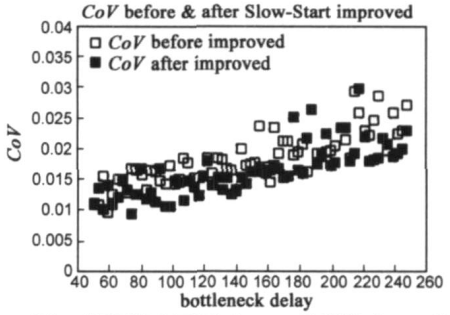

图6 瓶颈带宽不同时TFRC发送速率*CoV*值

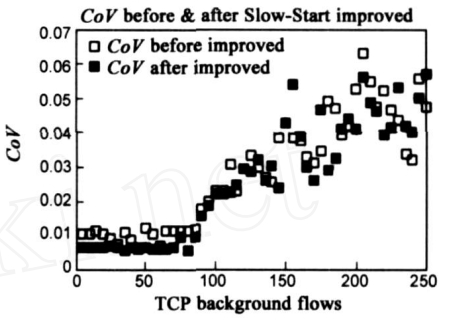

图7 TCP背景流数不同时TFRC发送速率*CoV*值

图 6 及图 7 中改进后效果普遍优于改进前,但也有极少量情况下不如改进前,仍有一定偶然性. 经统计,图 6 中所有使用改进前 TFRC 协议的场景 *CoV* 平均约为 0.01887,而改进后该值约为 0.01671,减小约 11 %;图 7 中使用改进前 TFRC 协议的场景 *CoV* 平均约为 0.0283,而改进后该值约为 0.026173,减小约 8 %.

5 结论

TFRC 协议由于慢启动阶段总是以丢包作为改变当前发送速率和所处的慢启动状态的标志,在经历第一次报文丢失时由于传输速率相对过高,导致短时间内报文丢失的绝对数量明显大于其他时间段,并且影响到拥塞避免阶段的首次速率计算,降低了网络利用率.

本文对 TFRC 慢启动阶段的性能进行了分析,提出了一种改进的方案,该方案在多数情况下对慢启动结束时的第一次丢包数有降低作用,并且由于丢包数能影响首次代入到式(2)中的参数丢包事件率 p,使发送速率从慢启动过渡到拥塞避免阶段更趋平稳,并提高了吞吐率,该方案具有较明显的改善作用.

由于实际网络中的的情况与仿真实验不尽相同,本改进方法需要更好的调整 k 值,以及发送速率与 *RTT* 及其 EWMA 均值之间的关系,未来的工作将着眼于使之更加合理,并研究更优的丢包避免策略提高发送速率平滑度及网络资源探测竞争能力的机制.

参考文献:

[1] Qian Wang, Keping Long, Shiduan Cheng, Runtong Zhang.

TCP-friendly congestion control schemes in the Internet[A]. Proceedings. ICII 2001-Beijing. 2001 International Conferences on[C]. Beijing, ICII, 2001. 211 - 216.

[2] M Handley, S Floyd, J Pahdye, J Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification[S]. IETF RFC3448, January 2003.

[3] J Padhye, V Firoiu, D Towsley, J Kurose. Modeling TCP throughput: A simple model and its empirical validation[A]. Proceedings of ACM SIGCOMM[C]. New York: ACM Press, 1998. 303 - 314.

[4] I Rhee, L Xu. Limitations of equation-based congestion control [J]. IEEE/ ACM Trans. Networking, 2007, 15(4): 852 - 865.

[5] D Bansal, H Balakrishnan, S Floyd, S Shenker. Dynamic behavior of slowly-responsive congestion control algorithms[A]. Proceedings of ACM SIGCOMM[C]. San Diego: ACM Press, 2001. 263 - 274.

[6] Q Li, D Chen. Analysis and improvement of TFRC congestion control mechanism[A]. IEEE/ Wireless Communications, Networking and Mobile Computing 2005[C]. Wuhan, China: IEEE Press, 2005. 2. 1149 - 1153.

[7] **耿福泉, 方敏, 等. 基于 RTCP 反馈的 TCP 友好的实时流媒体拥塞控制机制**[J]. **东北大学学报**, 2006, 27(11): 1204-1207.
Geng Fu-quan, Fang Min, et. al. A TCP-friendly real-time streaming media congestion control based on RTCP feedback [J]. Journal of Northeastern University, 2006, 27(11): 1204 - 1207. (in Chinese)

[8] S McCanne, S Floyd. ns-LBNL Network Simulator. 1996[CP/oL]. http://www-nrg.ee.lbl.gov/ns. 1996.

[9] L Brakmo, L Peterson. End-to-end congestion control avoidance on a global internet[J]. IEEE Journal on Selected Areas in Communications, 1995, 13(8): 1465 - 1480.

作者简介:

蒋　翊　男, 回族, 1982 年出生于湖南常德, 浙江大学计算机学院硕士研究生, 主要从事网络协议分析、拥塞控制、下一代互联网技术等科学研究工作.

吴春明　男, 汉族, 1967 出生于浙江萧山, 博士, 浙江大学计算机学院教授, 博士生导师. 主要从事网络服务质量, 可重构网络技术, 网络虚拟化和覆盖网, 人工智能等领域的科学研究工作.
E-mail: wuchunming@cs.zju.edu.cn