

一种适应于异质网络的 RED 改进算法

高玲珊, 王芳, 郑明春

(山东师范大学 信息管理学院, 山东 济南 250014)

摘要:随着 Internet 的发展, 异质网络结构成为发展的趋势。由于 RED 算法在异质网络环境下会引起队列长度波动较大、丢包率较高等现象, 所以针对这些特点文中提出了一种改进算法, 即通过在限定时间内的数据报重传次数进行数据传输控制。这种算法在指定的异质网络环境下能够有效地提高网络性能。最后, 经过仿真试验表明, 改进后的算法较 RED 算法在性能方面有明显的提高。

关键词:异质网络; RED 算法; 拥塞控制

中图分类号: TP301.6

文献标识码: A

文章编号: 1005-3751(2006)02-0173-03

An Improved RED Algorithm in a Heterogeneous Network

GAO Ling-li, WANG Fang, ZHENG Ming-chun

(Dept. of Computer Science, Shandong Normal University, Jinan 250014, China)

Abstract: With the development of Internet, heterogeneous network has become the tendency. Random Early Detection (RED) algorithm is an effective algorithm for active queue management in general networks. But it can't fit well for heterogeneous networks. So the paper analyzes RED algorithm about its virtues and shortcomings, and proposes an enhanced algorithm. At last, the simulation experiments indicate that the proposed algorithm provides better performance in heterogeneous networks.

Key words: heterogeneous network; RED algorithm; congestion control

0 引言

Internet 的发展使网络结构由以固定有线网络为主向移动无线网络发展, 因此端到端的数据传输需要适应有线和无线两种传输媒体^[1~3]。无线网络的传输特性在很多方面不同于有线网络。特点是带宽较窄, 同时由于突发噪声干扰和越区切换等原因, 容易引起分组丢失, 即数据传输的错误率较高, 因此在异质网络传输的整个路径中容易成为瓶颈。而有线链路的数据传输率相对较高, 抗干扰性强, 传输错误率很低, 发生分组丢失或延迟主要是由于网络拥塞。当由于无线干扰或切换而发生分组丢失时, 解决的方法应尽快重发分组。如果这时源端 TCP 监测到传输超时而放慢发送速率, 只能降低吞吐量, 浪费带宽。对于跨越有线和无线网络的端到端连接, 传统的 TCP 对超时不作区分地按照拥塞来处理, 所以传统的主动队列管理算法已不能完全适应异质网络传输的特点。

1 RED 算法及改进算法

1.1 RED 算法介绍

为了提高网络性能, Internet 工程任务组 (IETF) 推荐

在路由器中使用活动队列管理算法 RED^[4], 该算法的基本思想是: 通过监控路由器输出端口平均队列的平均长度来探测拥塞, 一旦发现接近拥塞, 就随机地选择连接来通知拥塞, 使它们在队列溢出导致丢包之前减少发送窗口, 降低数据发送速度, 从而缓解网络拥塞。

算法具体描述如下:

数据包到达路由器后, 需要在不同的输出端口缓冲区中进行排队, 每一个输出端口维护一个队列, 当有新的数据包到达时, 采用指数加权滑动平均的方法计算平均队列长度 avg , 并把 avg 与两个预先设定的门限 (最小门限 min_{th} 和最大门限 max_{th}) 相比较:

若平均队列长度 avg 小于 min_{th} , 则不丢弃分组;

若 avg 大于或等于 max_{th} , 则丢弃所有分组;

若 avg 大于或等于 min_{th} 而小于 max_{th} , 则根据平均队列长度 avg 计算概率 p_b , 以概率 p_b 丢弃分组。

如图 1 所示, 其中 q_{time} 表示队列为空的起始时刻; $count$ 表示上次丢弃分组后受到的分组数目; w_q 表示计算队列的平均长度是采用的权重; max_p 表示 p_b 能够取得的最大值; p_a 表示当前分组被丢弃的概率; q 表示当前队列的实际长度; $time$ 表示当前时间; $f(t)$ 表示时间 t 的线性函数。

在有线链路中 RED 算法能有效地降低丢包率, 而且由于它是随机地通知发送方调整速率, 能够有效地消除全局同步。但是在异质网络环境中, 由于有线和无线链路在

收稿日期: 2005-05-26

作者简介: 高玲珊 (1980—), 女, 山东潍坊人, 硕士研究生, 研究方向为异质网络性能; 郑明春, 教授, 研究方向为异质网络性能、网络服务质量。

带宽大小上的差异、无线链路的高误码率、移动主机的位置切换的原因,会使得 RED 算法产生高的丢包率、引起网络大的抖动等问题,不能很好地发挥它的在有线上的优势,所以,要在 RED 算法的基础上,做一定的改进。

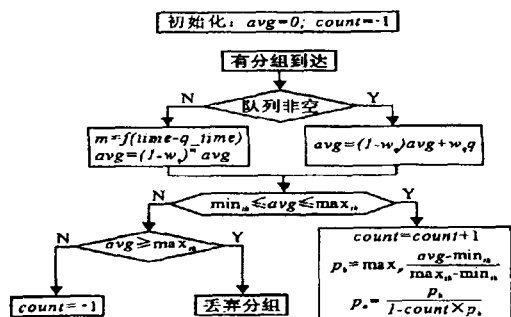


图 1 RED 算法

当移动主机与网络中的一台固定主机通信时,通常是通过一段无线链路与本站相连,再由本站通过固定的有线网络与固定主机连接。基站极少出错,但有时阻塞的有线网与可靠性较低而容易导致数据段丢失的无线网连接起来,假设基站到移动主机的无线链路是整个 TCP 连接的最后一跳(见图 2)。

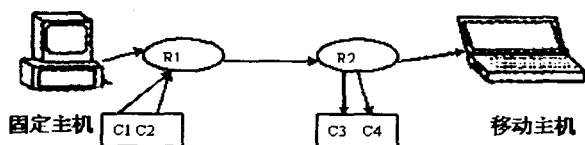


图 2 固定主机与移动主机的连接模型

在实际的移动计算环境中,这种结构最为常见。它的突出特点是带宽的不对称性,即有线部分的带宽比无线部分的带宽大很多,改进算法的具体思想如下。

1.2 RED 的改进算法

在初始阶段通过记录每个连接的反馈信息来区别对待不同的数据流。由于无线链路的高误码率,所以重传的机率较之有线网络较高,所以对数据重传的信息作记录,如果重传的次数超过 m ,则为此类数据流建立一个新的缓冲队列,对此缓冲队列以低丢包概率保护。

算法具体描述如下:

1) 首先设置参数值, M 表示启用预备缓冲队列的阙值,可根据不同的网络动态设置。在给定时间内重传次数超过 M ,就启用预备缓冲队列。 M_{cou} 表示每个数据流重传次数; \max_q 表示除预备队列以外的其它连接允许的最大队列长度; M_{avg} 表示预备缓冲队列的长度。

2) 当有新的数据包到达时,根据对数据包所属连接分两种情况处理:

a. 如果该数据包所属的连接是发送到无线链路上的,即在给定时间内,此类数据流的 $M_{cou} \geq M$,计算 M_{avg} ;

如果 $M_{avg} \geq \beta \times \max_{th}$,则丢弃该数据包(其中 $\beta > 1$,具体值根据不同的网络确定);

如果 $M_{avg} < \beta \times \min_{th}$,则不丢弃分组,放入预备队列;

如果 $\beta \times \min_{th} \leq M_{avg} < \beta \times \max_{th}$,采用与 RED 算法相同的策略。

b. 如果在给定时间内该数据包的 $M_{cou} < M$,采用与 RED 算法相同的策略。

2 仿真试验与性能比较

这里采用 ns2^[5]进行仿真试验,实验的网络拓扑结构如图 1 所示。实验中用到的参数如下: \min_{th} 80, \max_{th} 280, 权值 w_q 0.002, 最大丢包概率 \max_p 0.1, 在给定时间 100 秒内比较 M_{cou} 与 M 。该实验对文中提出的算法性能加以分析,并与 RED 算法在队列长度、丢包概率、丢包率等方面进行性能比较。

从图 3 中可以看出,RED 算法队列长度波动非常频繁并且范围很大,主要是因为当有大量的活跃的 TCP 连接时,总的流量往往突发性非常高,队长的增减非常迅速。改进算法相对减少了队列长度的波动范围。

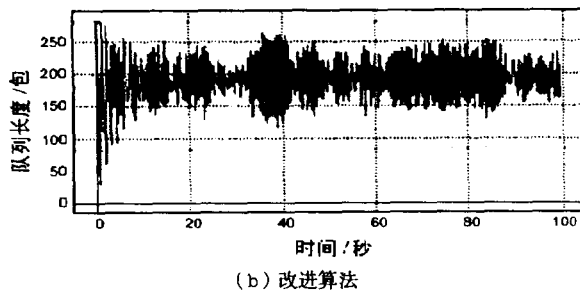
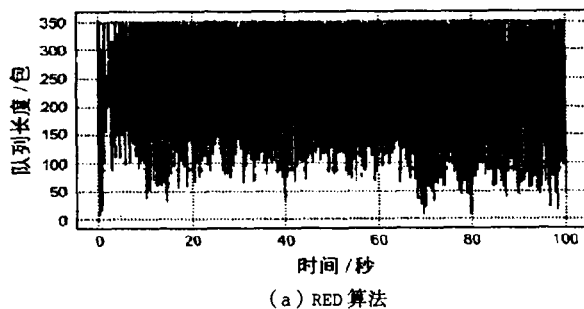


图 3 队列长度变化

从图 4 可以看到,RED 算法的丢包概率波动很频繁,范围很大,这和其队长的波动是一致的。改进算的丢包概率波动较小,说明对于异质网络下的拥塞反应具有一定的缓解作用。

从图 5 中可以看出,RED 算法的丢包率仍然波动范围很大,并且其丢包率也比较高。改进算法有效地降低了丢包率的波动范围,而且减少了丢包率。

3 总结

文中就 RED 算法在异质网络环境下的适应性改进进行了研究。针对其缺点做了相应的改进,并对此算法进行了仿真试验,而且对此算法的性能与 RED 算法性能进行了比较。结果表明该算法可以获得比 RED 算法更好的性能。

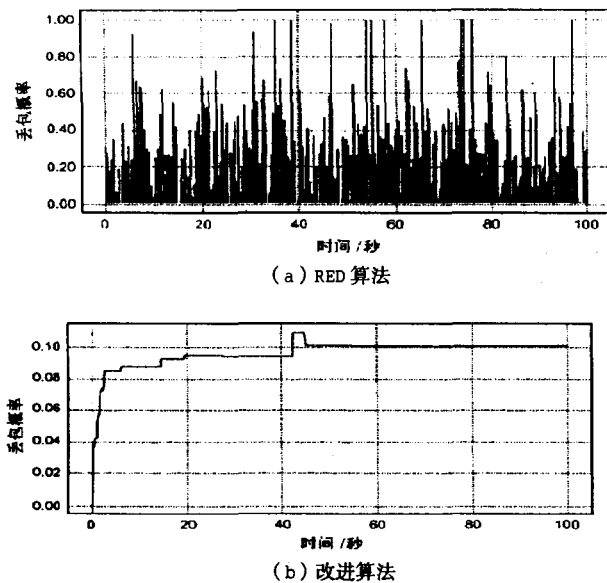


图4 丢包概率变化

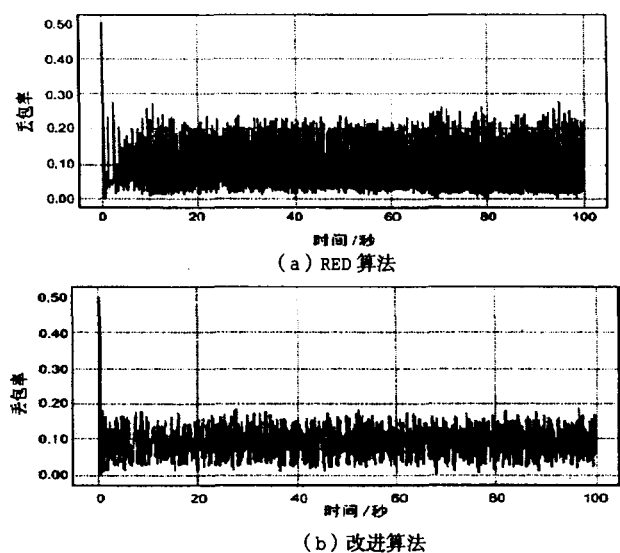


图5 丢包率变化

参考文献:

- [1] Floyd S. A Report on Some Recent Developments in TCP Congestion Control[J/OL]. <http://www.icir.org/floyd/>, 2001.
- [2] Allman M. Ongoing TCP Research Related to Satellites RFC [EB/OL]. <http://roland.grc.nasa.gov/~mallma,2000-02>.

- [3] Balakrishnan H, Padmanabhan V N, Seshan S, et al. A Comparison Of Mechanisms For Improving TCP Performance Over Wireless Links[EB/OL]. <http://nms.lcs.mit.edu/papers/hari-phd/>, 1996.
- [4] Floyd S, Jacobson V. Random early detection gateway for congestion avoidance[J]. IEEE/ACM Transactions on Networking, 1993, 1(4): 397-413.
- [5] McCanne S, Floyd S. ns - LBNL Network Simulator[CP/OL]. <http://www-nrg.ee.lbl.gov/ns/>, 1996.

(上接第172页)

Join points(联结点):在程序结构或者执行过程中明确定义的可以联结附加行为的点,可以是一个方法的调用,或者是一个特别的被抛出的异常。

Advice(通知):在结合点执行的行为,其作用类似拦截器。不同的通知类型包括:around advice, before advice, after advice, throws advice。

Pointcut(切入点):指定一个通知将被引发的一系列 Join points 的集合。

IoC模式和AOP方法的紧密结合首先会在支持IoC模式的容器和框架中得到广泛的应用,然而如何使得两者更加有机而无缝的结合并得到大量实用工具的有力支持仍需要进一步的研究和实践。

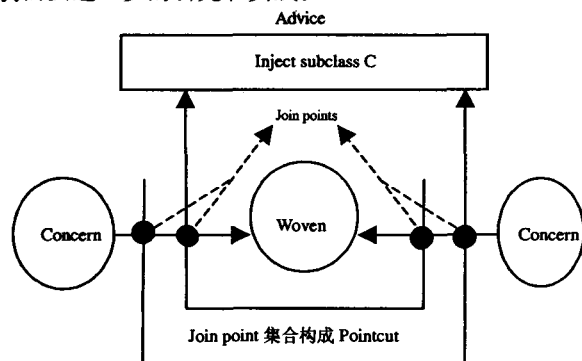


图4 AOP与IoC

4 结论

讨论了IoC及其分类和应用,总结了各种实现IoC方法的优缺点。由于IoC模式概念清晰,对依赖关系有着强大的分解和操作能力,使得IoC成为一种构建轻型容器和框架必要而可行的模式,相信未来其与AOP的结合可以最大限度地发挥功用,从而更简单地构建高质量的软件系统。

参考文献:

- [1] Malarvannan M. Design better software with the inversion of control pattern[EB/OL]. <http://www.devx.com/Java/Article/27583/0/page/1>, 2005-03-18.
- [2] Johnson R, Hoeller J. Expert one-on-one J2EE development without EJB[M]. Indiana: Wiley Publishing Inc., 2004.
- [3] Fowler M. Inversion of control containers and the dependency injection pattern[EB/OL]. <http://www.martinfowler.com/articles/injection.html>, 2004-01-23.
- [4] Johnson R, Hoeller J, Arendsen A, et al. Spring - Java/J2EE Application Framework Reference Documentation[EB/OL]. <http://www.springframework.org/docs/reference/index.html>, 2005.
- [5] Kiczales G, Lamping J, Mendhekar A, et al. Aspect-Oriented Programming[M]. Finland: Springer-Verlag, 1997.