嵌入式网络通信中 RED 算法改进

邹国霞[1],黄廷磊[2],唐建清[1]

1. 桂林航天工业高等专科学校计算机系,广西桂林 541004
2. 桂林电子科技大学计算机科学与工程学院,广西桂林 541004

摘要 在嵌入式网络通信中,主要采用 RED 算法解决网络拥塞。由于 RED 算法中丢包率与平均队列长度成线性关系,导致网络在拥塞并不严重时丢包率较大,在拥塞比较严重时丢包率较小,拥塞控制能力较低。经研究,发现 IMPRED 算法能解决这个问题,当平均队列长度在最小阈值附近时丢包率增长速度较小,在最大阈值附近时丢包率增长速度较大,避免了网络的全局同步。利用时间复杂度和空间复杂度对 IMPRED 算法和 RED 算法进行比较,IMPRED 算法没有增加 RED 算法的复杂度。通过 NS 2.30 仿真证实,IMPRED 算法可以提高网络吞吐量,减少延时抖动,使网络比较稳定。

关键词 随机早期检测;网络拥塞;吞吐量

中图分类号 TP301.6 **文献标识码** A **doi** 10.3981/j.issn.1000-7857.2011.12.008

Improvement of RED Algorithm in the Embedded Network Communication

ZOU Guoxia[1], HUANG Tinglei[2], TANG Jianqing[1]

1. *Department of Computer Science and Technology, Guilin College of Aerospace Technology, Guilin 541004, Guangxi Zhuang Autonomous Region, China*
2. *School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, Guangxi Zhuang Autonomous Region, China*

Abstract The RED algorithm is used to solve the network congestion problem. In the embedded network communication, the drop packet ratio is proportional to the average queue length, and,as a result, the drop packet ratio is high when the network congestion is not serious and the drop packet ratio is low when the network congestion is serious, so the congestion control is not effective. The density function IMPRED (Improved Random Early Detection) can solve that problem. In the IMPRED algorithm, there are two curves, one is 3 times density function, the other is 1/3 times density function. With the IMPRED algorithm, the drop packet ratio is smaller around the minimum threshold, and larger around the maximum threshold, to avoid the global synchronization of the network. Using the time complexity and space complexity of the algorithm to compare the RED algorithm and the IMPRED algorithm, it is found that the time complexity of IMPRED and RED is $O(n)$, the space complexity of IMPRED is as large as the RED's. The NS 2.30 simulation results show that when the network congestion does not appear, the efficiency of IMPRED algorithm and RED algorithm is the same. When the network is congested, IMPRED algorithm can improve the network throughput and reduce the delay jitter, making the network more stable.

Keywords random early detection; network congestion; throughput

0 引言

嵌入式系统正日益得到越来越广泛的应用,并随着研究和应用的进一步深入,将朝着网络化、智能化、规范化、集成化方向发展。在嵌入式网络通信中,网络拥塞容易造成数据传输、抖动和吞吐量等 QoS(Quality of Service)性能指标下降[1],从而引起网络带宽的资源浪费。目前在嵌入式网络通信中主要采用一种主动队列管理(Active Queue Management)技术 RED(Random Early Detection,随机早期检测)解决网络拥塞,但该算法对参数的设置很敏感,其性能的优劣在某种程度上由参数决定,为此已产生不少的 RED 变种算法,较有影响力

收稿日期:2010-11-16;修回日期:2011-04-01
基金项目:广西教育厅科研项目(201010LX609);桂林航天工业高等专科学校课题(200910)
作者简介:邹国霞,讲师,研究方向为嵌入式、中间件技术,电子信箱:zouguoxia@163.com

的有 Stabilized-RED[3]、Self-configuration RED[4]、Adaptive RED[5]、FRED 和 Balanced-RED，其中 FRED 和 Balanced-RED 侧重解决 RED 存在的公平性问题，其余均意在增强 RED 的稳定性[2]。

在上述比较有影响的改进 RED 算法中,采用的均是线性 RED 算法,即当平均队列长度的估算值介于最小和最大缓冲阈值之间时，临时丢包率随平均队列长度的增加而线性增加,但这种线性关系并不合适。因为当平均队列长度在最小缓冲阈值附近时,容易产生较高的丢包率,而此时网络并不处于严重拥塞状态,需要较低的丢包率;当平均队列长度在最大缓冲阈值附近时,容易产生较低的丢包率,而此时网络已经处于严重的拥塞状态,需要较高的丢包率。

为此,本文在基于 RED 算法思想基础上,提出了非线性 IMPRED(Improved Random Early Detection)算法,通过对 RED 算法使用的临时丢包率 P_b 公式进行改进,当平均队列长度在最小缓冲阈值附近时,产生较低的丢包率;当平均队列长度在最大缓冲阈值附近时,产生较高的丢包率,从而降低网络的延时抖动和提高网关的吞吐量。通过仿真软件 NS 2.30 进行仿真，结果显示,IMPRED 算法具有较好的拥塞控制能力,验证了算法的有效性。

1　RED 算法简介

随机早期检测 RED 算法的基本思想是在拥塞发生的早期,对到达分组按某个概率进行丢弃,以避免拥塞的发生。算法的基本思想如图 1 所示。

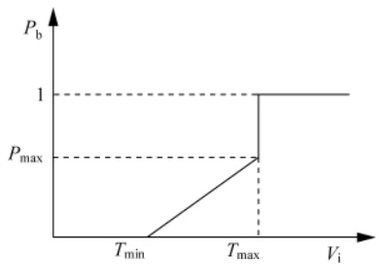


图 1　RED 算法的 P_b 函数图

Fig. 1　Relationship curve of V_i and P_b in RED

由于 RED 算法中丢包率与临时丢包概率呈正比关系,因此,在某种程度上,丢包率的控制可看作是对临时丢包概率 P_b 的控制[6-8],图 1 反映了 P_b 与分组平均队列长度 V_i 之间的关系,其中 T_{min}、T_{max}、P_{max} 分别为最小缓冲阈值门限、最大缓冲阈值门限、最大临时丢包率。

(1) 分组到来时,首先计算平均队列长度 V_i,然后将 V_i 与 T_{min}、T_{max} 两阀门值进行比较并计算随机丢包率。

(2) 随机丢包率的计算。

① 当 $V_i<T_{min}$,此时网络没有发生拥塞现象,即丢包率 P_b 为 0,所有到达的数据包都被插入到当前队列,进行正常的分组转发;

② 当 $V_i>T_{max}$ 时,所有的到达包都被丢弃;

③ 当 $T_{min}<V_i<T_{max}$ 时,所到达的数据包被丢弃的临时概率即丢包率为 P_b,其计算函数为

$$P_b=P_{max}\cdot\frac{V_i-T_{min}}{T_{max}-T_{min}} \tag{1}$$

由式(1)和图 1 可知,当 $T_{min}<V_i<T_{max}$ 时,P_b 随平均队列长度 V_i 的增加而线性增加,这种线性关系并不合适。因为在 V_i 较小时,缓冲队列相对较空,此时可以尽少量的丢包;而在 V_i 较大时,缓冲队列比较满,此时丢包少,容易造成全局同步现象。因此单纯的线性变化并不能很好的控制网络的拥塞。

2　RED 算法的改进

2.1　RED 改进算法——IMPRED

当队列比较空时，应该可以允许容纳较多的到达分组,这时丢包率应该尽量小,从而增加网络的吞吐量;而当 V_i 很大时,可以容纳的相应的到达分组变得很少,此时应该控制到达分组的平均队列长度,丢包率应该很大。为了达到这种效果,P_b 与 V_i 应该呈曲线形式的关系,V_i 较小时,P_b 值应该在直线下方;V_i 较大时,P_b 值应该在直线上方。为此,本文在 RED 算法基础上进行改进,设计了 IMPRED 算法,主要是对 P_b 表达式进行改进,当平均队列长度在最小阈值附近时丢包率增长速度较小,在最大阈值附近时丢包率增长速度较大。如图 2 所示,当 V_i 在 T_{min} 附近时,P_b 比 RED 中 P_b 小;当 V_i 在 T_{max} 附近时,P_b 比 RED 中 P_b 大,可以很好地解决丢包率分配问题。

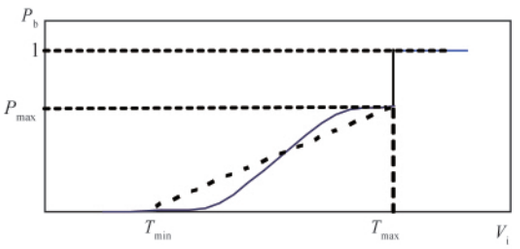


图 2　IMPRED 算法的 P_b 函数图

Fig. 2　Relationship curve of V_i and P_b in IMPRED

当 $T_{min}<V_i<T_{max}$ 时,所到达的数据丢包率为 P_b,表达式为

$$P_b=\frac{P_{max}\cdot(V_i-T_{min})^3}{(T_{max}-T_{min})\cdot T_{min}^2} \tag{2}$$

当 $2T_{min}<V_i<T_{max}$ 时,P_b 曲线为

$$P_b=B(V_i-2T_{min})^{\frac{1}{3}}+C \tag{3}$$

将点 $\left(2T_{min},\frac{T_{min}\cdot P_{max}}{T_{max}-T_{min}}\right)$ 和点 (T_{max},P_{max}) 代入式(3)得 C 和 B 的表达式:

$$C=\frac{T_{min}\cdot P_{max}}{T_{max}-T_{min}} \tag{4}$$

$$B=\frac{P_{max}\cdot(T_{max}-2T_{min})^{\frac{2}{3}}}{T_{max}-T_{min}} \tag{5}$$

2.2 IMPRED 算法的复杂性分析

算法的复杂性分析主要是从时间复杂度和空间复杂度进行分析。

假设有 N 个分组,同等条件下执行一条减法执行时间为 T_1,乘法执行时间为 T_2,除法执行时间为 T_3,开立方执行时间为 T_4,加法执行时间为 T_5。在 IMPRED 中,设 $T_{min}<V_i<T_{max}$ 的概率为 P_1,$2T_{min}<V_i<T_{max}$ 的概率为 P_2,且 $P_1+P_2<1$。

由于算法改进的关键在表达式 P_b,为此,分析时间复杂度时主要分析执行 P_b 所需要的时间。RED 算法和 IMPRED 算法的 P_b 执行时间分别为

$$T_{RED}(N)=N(2T_1+T_2+T_3) \quad (6)$$

$$T_{IMPRED}(N)=N[P_1(T_1+7T_2+T_3)+P_2(T_1+2T_2+T_4+T_5)] \quad (7)$$

由式(6)和式(7)可知,对于所有 N,当 $N\geqslant n_0$ 时,有 $T_{RED}(N)=O(N)$,$T_{IMPRED}(N)=O(N)$。为此,IMPRED 算法和 RED 算法的时间复杂度是相同的。

空间复杂度上,IMPRED 所需要的常量空间更多一些,但采用的存储方法一样,因此,在空间复杂度上讲也没有变化。

3 算法仿真与分析

3.1 仿真模型和参数设计

采用 NS 2.30 仿真软件对两种不同算法的网关进行仿真实验,由于要对算法进行改进,需要对 NS 2.30 内核中 queue/red.h 和 queue/red.cc 重新编辑和编译。该网络模拟环境的拓扑结构如图 3 所示。

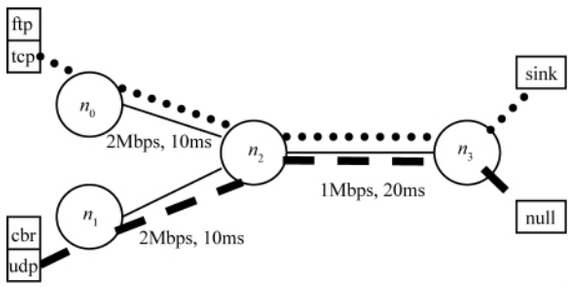


图 3 网络拓扑结构

Fig. 3 Network topology diagram

图 3 中,网络节点 n_0 到 n_2 之间,和节点 n_1 到 n_2 之间的网络频宽为 2Mbps,延迟时间为 10ms。网络拓扑中的频宽瓶颈在 n_2 到 n_3 之间,频宽为 1Mbps,延迟时间为 20ms。每个网络节都采用 RED 方式,且在 n_2 到 n_3 之间的最大队列长度是 8 个封包的长度。在 n_0 到 n_3 之间有一条 FTP 的联机,其架构于 TCP 之上,在 n_0 上使用 TCP agent 产生“tcp”发送 TCP 的封包,在 n_3 上使用 TCPsink agent 产生“sink”接受 TCP 的资料,并产生回复封包(ACK)回传送端,最后将接收的 TCP 封包释放。在 n_1 到 n_3 之间有一条 CBR 的联机,其架构于 UDP 之上,在 n_1 上使用 UDP agent 产生“udp”发送 UDP 封包,在 n_3 上使用 Null agent 产生“null”接收由 n_1 传送过来的 UDP 封包,然后将接收的封包释放。CBR 的传送速度为 1Mbps,每一个封包大小为 1kB。CBR 从 0.1s 开始传送,4.5s 时结束传输;FTP 从 0.5s 秒开始传送,5.0s 时结束传输。仿真时间为 5.5s,包长为 1kB。RED 参数设置如下:$T_{min}=5$,$T_{max}=15$,$P_{max}=0.1$。

3.2 仿真结果比较

在该瓶颈链路上分别使用原 RED 算法和改进 IMPRED 算法进行仿真,仿真结果如下。

(1) 丢包率随平均队列长度变化曲线如图 4 所示。

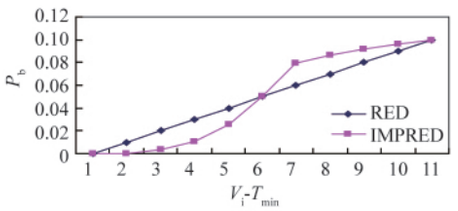
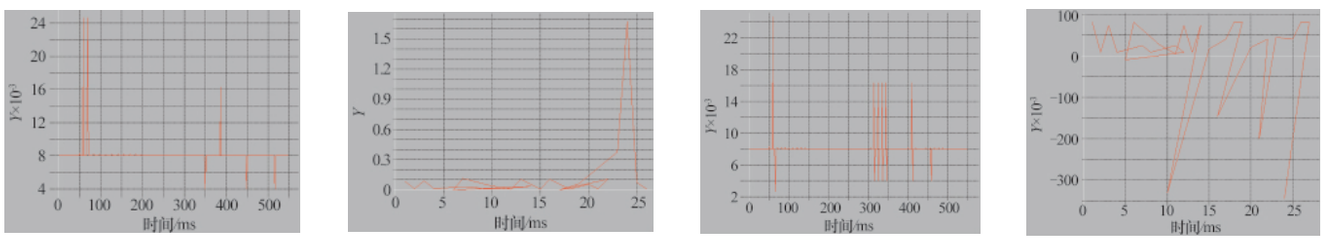


图 4 丢包率随平均队列长度变化曲线

Fig. 4 Curve between P_b and V_i

图 4 中,横轴为平均队列长度减去最小缓冲阈值门限后的值,纵轴为临时丢包率,由图可知,临时丢包率 P_b 在平均队列较小时,$P_{bIMPRED}$ 比 P_{bRED} 小,随着平均队列长度的增加,$P_{bIMPRED}$ 比 P_{bRED} 大得多,这样有利于在网络比较拥塞时进入快速丢包状态,避免全局同步的发生。

(2) 在瓶颈处(n_2—n_3)之间采用 RED 算法、IMPRED 算法时的时延抖动比较如图 5 所示。由图可知,在 IMPRED 算法



(a) RED 算法之 CBR 延时抖动
(a) CBR delay jitter of RED algorithm

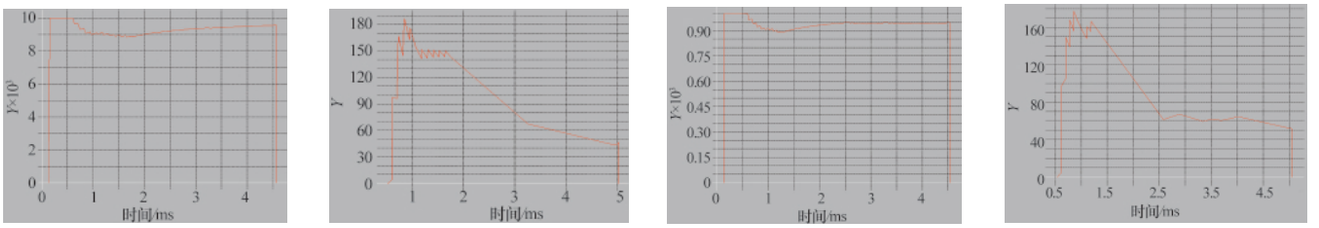
(b) RED 算法之 FTP 延时抖动
(b) FTP delay jitter of RED algorithm

(c) IMPRED 算法之 CBR 延时抖动
(c) CBR delay jitter of IMPRED algorithm

(d) IMPRED 算法之 FTP 延时抖动
(d) FTP delay jitter of IMPRED algorithm

图 5 两种算法下不同流的延时抖动比较

Fig. 5 Delay jitter comparison of different streams between the two algorithms



(a) RED 算法之 CBR 吞吐量
(a) CBR throughput of RED algorithm

(b) RED 算法之 FTP 吞吐量
(b) FTP throughput of RED algorithm

(c) IMPRED 算法之 CBR 吞吐量
(c) CBR throughput of IMPRED algorithm

(d) IMPRED 算法之 FTP 吞吐量
(d) FTP throughput of IMPRED

图 6　两种算法下不同流的吞吐量比较

Fig. 6　Throughput comparison of different streams between the two algorithms

作用下,FTP 延时抖动幅度不超过 0.35，使用 IMPRED 算法时,TCP 延时抖动明显比 RED 算法小。对于 CBR 延时抖动,刚开始时,由于平均队列较小,不会发生拥塞,基本延时抖动值为 8×10^{-3},随着 FTP 分组到来,网络出现了拥塞。在 500ms 附近时,IMPRED 延时抖动值明显小于 RED 延时抖动值;在 3000—5000ms,即网络拥塞最严重时,IMPRED 延时抖动值略大于 RED,说明在 IMPRED 算法作用下,网络比 RED 略为拥塞,这也证明了平均队列大于最小阀值时,在 IMPRED 算法下丢包率比 RED 小。综上可知,IMPRED 算法较 RED 算法在网络稳定性上有所增强。

(3) 在瓶颈处(n_2—n_3)之间采用 RED 算法、IMPRED 算法时的吞吐量比较如图 6 所示。对于 CBR 吞吐量，由于在 500ms 之前,只有 CBR 包,网络没有发生拥塞现象,两种算法的 CBR 吞吐量一样;在 500—4500ms,同时发送 FTP 和 CBR 包，网络出现了拥塞现象，因此 CBR 吞吐量有所下降,但 IMPRED 算法 CBR 吞吐量比 RED 高。对于 TCP 吞吐量,由于 FTP 包从 500ms 之后才开始发送,0—500ms 处没有 FTP 的吞吐量;在 500—4500ms,同时发送 CBR 包和 FTP 包,FTP 包的吞吐量逐渐减少，但 IMPRED 算法 TCP 吞吐量比 RED 高;4500ms 之后,只发送 FTP 包,两个算法的 FTP 吞吐量一样。综上可知，网络在使用 IMPRED 算法时,CBR 与 FTP 的吞吐量均比 RED 算法明显增加。

综上所述,IMPRED 算法在网络延时抖动方面比 RED 算法小,在吞吐量方面比 RED 算法大。这和前面的理论分析是一致的。

4　结论

为了提高嵌入式网络通信的稳定性，充分利用资源,在研究 RED 算法思想的基础上,提出了 IMPRED 算法,通过 NS 2.30 仿真证实,IMPRED 算法可以提高网络吞吐量,减少延时抖动,使网络比较稳定。

参考文献(References)

[1] 潘爱民, 徐明伟. 计算机网络[M]. 北京: 清华大学出版社, 2004.
Pan Aimin, Xu Mingwei. Computer network [M]. Beijing: Tsinghua University Press, 2004.

[2] 刘凤格. 基于 RED 算法的改进研究[J]. 计算机仿真, 2009, 26(5): 118-120, 133.
Liu Fengge. *Computer Simulation*, 2009, 26(5): 118-120, 133.

[3] 黄迎春, 李向丽, 邱保志. 一种改进的 RED 算法[J]. 计算机工程, 2007, 33(1): 117-118, 121
Huang Yingchun, Li Xiangli, Qiu Baozhi. *Computer Engineering*, 2007, 33(1): 117-118, 121.

[4] Verma R, Iyer A, Karandikar A. Towards an adaptive RED algorithm for achieving delay-loss performance [J]. *IEE Proc Commun*, 2003, 150(3): 163-168.

[5] Kim T, Lee K. Refined adaptive RED in TCP/IP networks [C]. SICE-ICASE International Joint Conference, Busan, South Korea, Oct 18-21, 2006.

[6] 封宁, 白光伟. RED 算法的数学模型研究 [J]. 计算机工程与设计, 2008, 29(9): 2179-2180.
Feng Ning, Bai Guangwei. *Computer Engineering and Design*, 2008, 29 (9): 2179-2180.

[7] Li Y Q, Yang S H. Priority Checking RED for Improving QoS in IPv6[C]. IEEE International Conference on Networking, Sensing and Control. ICNSC 2008, Sanya, China, April 6-8, 2008.

[8] 余冠玮, 邢卫, 鲁东明. DF-RED: 一种基于动态公平性的 RED 算法[J]. 制造业自动化, 2010, 32(9): 7-13, 49.
Yu Guanwei, Xing Wei, Lu Dongming. *Manufacturing Automation*, 2010, 32(9): 7-13, 49.

(责任编辑　代丽)