系数自调整的 PD-RED 算法*

毛银宁[1]，姚旭寅[2]，张小贝[1]，李伟杰[2]，杨　融[2]
（1. 上海大学 通信与信息工程学院，上海 200444；2. 上海飞机设计研究院 电子集成部，上海 201315）

摘　要： 随着技术发展，民机上的网络环境渐渐趋于日常生活中的网络环境。为了避免拥塞，需要对飞机上的链路数据进行主动队列管理。在比例微分控制的随机早期检测（random early detection based on proportional derivative control principle，PD-RED）算法基础上，提出一种系数自调整的 PD-RED（improved PD-RED，IPD-RED）算法。在 IPD-RED 算法中，引入比例系数和微分系数的调整函数来减小参数选取对算法的影响。考虑队列偏差，将其归一化处理后，作为调整函数的参数来使系数动态变化，根据系数变化规律设计函数。实验通过改变上下门限值组合、路由节点间延时和端节点数量来分析算法改善效果，通过改变比例系数和微分系数的初值来探究初值对算法性能的影响。NS2 仿真结果表明，IPD-RED 算法使平均队列长度更接近期望值，提高了吞吐量，减小了丢包率。初值影响表明，一定范围内增大比例系数可使平均队列长度更快更接近期望值且减小丢包率，但会使延时和振荡增加。微分系数的动态范围很大，对几个网络性能参数的影响很小。实际应用中适当选取系数初值，可使算法更好地适配飞机上的业务。

关键词： 主动队列管理；早期检测；调整函数；IPD-RED
中图分类号： TP393　　**文献标志码：** A　　**文章编号：** 1001-3695(2022)06-014-1683-06
doi:10.19734/j.issn.1001-3695.2021.10.0610

PD-RED algorithm with self-adjusting coefficients

Mao Yinning[1], Yao Xuyin[2], Zhang Xiaobei[1], Li Weijie[2], Yang Rong[2]
(1. *Communication & Information Engineering Institute, Shanghai University, Shanghai* 200444, *China*; 2. *Electronic Integration Dept., Shanghai Aircraft Design & Research Institute, Shanghai* 201315, *China*)

Abstract: With the development of technology, the network environment on civil aircraft gradually tends to it in daily life. In order to avoid congestion, link data on aircraft needs active queue management. This paper proposed an IPD-RED algorithm with self-adjusting coefficients, which based on the PD-RED algorithm. IPD-RED algorithm introduced the adjustment functions of proportional coefficient and derivative coefficient to reduce the influence of parameter selection on the algorithm. Considering the queue deviation after normalization, the adjustment function used it as the parameter in order to make the coefficient change dynamically. IPD-RED designed the adjustment function according to the law of coefficient change. The experiment analyzed the improvement effect of the algorithm by changing the combination of upper and lower threshold values, the delay between routing nodes and the number of end nodes, and explored the impact of the initial value on the performance of the algorithm by changing the initial value of proportional coefficient and derivative coefficient. NS2 simulation results show that IPD-RED algorithm makes the average queue length closer to the expected value, improves the throughput and reduces the packet loss rate. The influence of initial value shows that increasing the proportional coefficient in a certain range can make the average queue length faster and closer to the expected value and reduce the packet loss rate, but it increases the delay and oscillation. The dynamic range of the derivative coefficient is very large and has little effect on several network performance parameters. In practical application, the appropriate selection of the initial value of the coefficient can make the algorithm better adapt to the business on the aircraft.

Key words: active queue management; early detection; adjustment function; IPD-RED

0　引言

随着时代发展和科技进步，飞机上的接口和链路越来越多，与外界的通信需求越来越多。随着国产大飞机事业的蓬勃发展，机场无线通信、机组无线通信、宽带卫星通信、空地无线宽带通信（ATG）、航空机场场面移动通信系统（AeroMACS），这些已有的或者未来将有的通信链路将逐渐地部署到位。这些链路日趋复杂与庞大，链路所涉及到的用户需求与通信数据量也在与日俱增。此时，飞机上的网络环境逐渐趋于日常生活中的网络环境。它也会如日常生活中使用因特网一般，出现拥塞与延迟，网络波动与不稳定等情况，给用户带来不愉快的体验感。设想到未来的这些情形，现考虑对于飞机上的链路数据进行队列管理，采取合适的队列管理算法来提高性能。

早在 1984 年，美国加州的一位学者 Nagle 就提出了针对网络拥塞控制[1]的思想，他指出，TCP、IP 协议一起使用时，由于数据报文层和传输层之间的相互作用，会导致异常拥塞问题，特别是 IP 网关特别容易受到拥塞崩溃。基于此思想，后来的学者逐渐提出了拥塞控制的策略及算法，归根结底可分为两大类。第一类是针对源端而设计的拥塞控制机制，这类算法的思想是根据发送端接收到来自接收端返回的是否接收到超时

收稿日期： 2021-10-19；**修回日期：** 2021-12-16　　**基金项目：** 上海飞机设计研究院课题项目（MJZ-2016-S-45）

作者简介： 毛银宁（1997-），男，浙江宁波人，硕士研究生，主要研究方向为民机无线链路管理系统（359322025@qq.com）；姚旭寅（1983-），男，安徽寿县人，高级工程师，主要研究方向为民机机载软件与适航验证工作；张小贝（1982-），男，湖北宜城人，教授，博士，主要研究方向为特种光纤器件、光学谐振腔、光纤传感研究；李伟杰（1985-），男，山东威海人，高级工程师，硕士，主要研究方向为机载信息系统的集成与适航验证；杨融（1984-），男，陕西西安人，高级工程师，硕士，主要研究方向为机载信息系统的集成、民机网络安保.

(C)1994-2023 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

信息以及是否接收到确认信息,以此来调整发送端的数据发送速率,这一类算法难以适应网络负载的变化,且是被动调节。第二类算法是基于链路中间节点的拥塞控制机制,这类算法的思想是链路的中间节点比目的端节点对拥塞情况的感知更加及时与准确,因此算法通过监测中间节点的拥塞情况并通知发送端来调节发送端的数据发送速率,以此来达到拥塞控制的目的。而针对中间节点的拥塞控制机制,又可分为队列调度算法[2]和队列管理算法[3]两类相关算法。队列调度算法是根据数据流所属的业务类型来进行带宽资源的分配,队列管理算法则通过不同形式地丢弃包来避免拥塞的发生。目前队列管理算法可分为被动式队列管理(passive queue management,PQM)和主动式队列管理(active queue management,AQM)[4,5]两大类。被动式队列管理主要采用先进先出(first in first out,FIFO)机制进行处理,队列发生溢出时对末尾的包进行丢弃,简单的处理机制使它具有固有的一些缺点[6],这里不再赘述。本文主要对主动式队列管理算法进行研究。

1 主动队列管理算法简述

主动式队列管理算法 AQM 是拥塞控制算法的其中一类[7],也可归于链路管理算法。它主动检测进入队列的数据包大小,并通过预设的算法来通知源端相应地调整源端数据的发送速率[8,9]。随机早期检测算法(random early detection,RED)是 AQM 算法中的一种典型算法[10]。RED 算法的核心思想就是保持缓冲区队列长度较低。当进入队列的数据包较大时,会按照一定的算法进行随机丢包。RED 算法在参数选取及不同的拥塞情况下并不能保持良好的性能,且对于目标队列长度的期望值也不理想。因此基于 RED 算法,又有许多学者提出了改进算法[11,12]。比例微分控制的 RED(RED based on proportional derivative control principle,PD-RED)就是其中一种能提高性能的改进算法[13]。它引入了比例微分控制的思想来调整最大丢弃概率,通过算法来动态调节最大丢弃概率,使得队列保持在期望值附近,丢弃概率仍然按照 RED 算法的方式来进行处理。它能使 RED 算法的一部分性能得到改善,但同时也引入了新的参数,即比例系数和微分系数。新参数会涉及参数初值选取问题,不同参数的选择会使算法最终呈现出来的性能有所差异。考虑到这个问题,为了尽量减小参数选取对于算法性能的影响,本文提出一种改进的 PD-RED 算法(improved PD-RED,IPD-RED),引入比例系数和微分系数的调整函数,通过设定初值,然后动态调整这两个参数,以此来减小参数选取对于算法性能的影响。通过仿真实验,比较吞吐量和丢包率,发现 RED、PD-RED、IPD-RED 算法在这两个参数上保持着逐步优化的趋势。比较队列图,IPD-RED 算法相较 PD-RED 算法,平均队列长度更接近期望值。

2 算法及其思想介绍

2.1 RED 算法

RED 算法的思路[14]可分为两个步骤。

a)计算平均队列长度。

$$\mathrm{avg}(i+1)=(1-w_q)\cdot\mathrm{avg}(i)+w_q\cdot q \tag{1}$$

其中:avg(i)是每采样时刻(i 为采样时刻,本文所述每时刻的数据即为采样时刻的数据)计算的平均队列长度;w_q 是权值,一般取 0~1 的一个值;q 是每时刻的瞬时队列长度。

RED 算法使用平均队列长度是为了使整体的队列长度更趋于平滑状态,剔除一些数据瞬态变化造成的影响。权值 w_q 可以理解为低通滤波器长度,该值的选取可影响到瞬时队列长度 q 变化时对平均队列长度 avg 的改变程度。在设置初值时,w_q 通常取一个较小的值,此时平均队列长度会与瞬时队列长度反映出来的队列情况有所滞后,但也正是该滞后性降低了队列对瞬态变化的敏感度。

b)根据最大丢弃概率进行丢包。RED 算法引入了 min_{th} 和 max_{th} 两个阈值,它们分别是设定的最小队列长度和最大队列长度,算法希望队列长度能保持在这两个值之间。当平均队列长度大于最大阈值时,进行数据包的随机丢弃;当平均队列长度小于最小阈值时,不进行包的丢弃;当平均队列长度在两个阈值之间时,按照以下的公式进行计算与丢弃包。

$$P_b=max_p\cdot(avg-min_{th})/(max_{th}-min_{th}) \tag{2}$$

$$P_a=P_b/(1-count\cdot P_b) \tag{3}$$

其中:$count$ 表示当前进入队列中连续的没有被丢弃的包;max_p 是最大丢弃概率,取 0~1 的一个值。RED 算法把 P_b 作为一个中间量,不让包丢弃概率直接取该值,而是如该算法的核心思想一样,希望能够使包丢弃的情况更加均匀化,不对瞬态突发情况进行大面积的丢弃。随着 $count$ 的增加,到达的数据包被丢弃的概率也越来越大,最终以 P_a 作为实际的丢弃概率进行丢包。上述步骤的目的是为了避免连续大面积丢包,使瞬态的影响被均衡到其他时刻,让队列更能反映传输过程中整个趋势而不是直接剔除瞬时数据流变化所造成的影响。

RED 算法引入了平均队列长度这个参数,试图尽量减小突发数据拥塞的情况对于整个数据流传输的影响,用这个参数来反映整个过程中的队列拥塞情况。RED 的稳定性主要由最大包丢弃概率 max_p 所决定,因此考虑通过调整最大丢弃概率使队列长度稳定在期望值,使算法性能进一步优化。

2.2 PD-RED 算法

PD-RED 算法通过调节最大丢弃概率 max_p 来对 RED 算法进行优化,对于 max_p 的调节函数如下[13]:

$$\max_p(i+1)=\max_p(i)+K_{p0}\frac{e(i+1)}{B_s}+K_{d0}\frac{e(i+1)-e(i)}{B_s} \tag{4}$$

$$e(i)=\mathrm{avg}(i)-Q_T \tag{5}$$

$$Q_T=(max_{th}+min_{th})/2 \tag{6}$$

其中:K_{p0} 为比例系数,K_{d0} 为微分系数,$e(i)$ 是队列误差长度,如式(5)所示,为每时刻的平均队列长度和期望队列长度的差值。Q_T 为期望的队列长度,如式(6)所示表现为最大阈值和最小阈值的均值。B_s 为时延带宽积,即链路延时和带宽相乘的结果。

业界的规则是路由器应能存储与时延带宽积相同大小的数据,这里可以假设缓冲区在传输速率上呈线性增长。由于队列长度波动与时延带宽积呈线性增长,所以它们也与所表示的缓冲区大小呈线性增长。那么,实际可以使用归一化的误差信号来代替误差信号,即式(4)。PD-RED 通过调节最大丢弃概率来使队列均值更靠近期望值,同时也改进了 RED 的部分性能参数,实现了优化目的。

2.3 IPD-RED 算法

本文提出的 IPD-RED 算法在 PD-RED 的基础上进行了改进,其核心是根据比例系数和微分系数引入相应的调整函数,因此要把 PD-RED 算法变形为

$$\max_p(i+1)=\max_p(i)+K_p\frac{e(i+1)}{B_s}+K_d\frac{e(i+1)-e(i)}{B_s} \tag{7}$$

$$B_s=B\cdot linkdelay \tag{8}$$

$$K_p=\begin{cases}-\dfrac{400}{9}K_{p0}\cdot x^2+5K_{p0} & 0\leqslant x<0.3\\ 20K_{p0}\cdot(x-0.5)^2+0.2K_{p0} & 0.3\leqslant x<0.7\\ -\dfrac{400}{9}K_{p0}\cdot(x-1)^2+5K_{p0} & 0.7\leqslant x<1\\ 5K_{p0} & x\geqslant 1\end{cases} \tag{9}$$

$$K_d=\begin{cases}1.5K_{d0}\cdot(x-1)^2+0.5K_{d0} & 0\leqslant x<1\\ 0.5K_{d0} & x\geqslant 1\end{cases} \tag{10}$$

(C)1994-2023 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

$$x = |\mathrm{avg}(i) - Q_T| / Q_T \cdot 10 \tag{11}$$

其中:B 为链路带宽,$linkdelay$ 为链路延时,K_{p0} 和 K_{d0} 为设定的比例系数和微分系数的初值。x 为调整函数的参数,表示队列长度的波动情况。本文设计的调整函数参考了文献[15,16]中 K_p、K_d 的变化规律。变化规律为

a)当队列长度与期望值波动较大时,即 $|e(i)|$ 较大时,K_p 应取较大值;当队列长度与期望值波动中等大小时,K_p 应取较小值;当队列长度与期望值波动较小时,K_p 应增大。

b)当队列长度与期望值波动较大时,K_d 应取较小值;当队列长度与期望值波动较小时,K_d 应取较大值。

根据以上规律,本文选取平均队列长度与期望值之间的偏差作为检测量来实时调整比例系数和微分系数。首先本文设计队列的波动期望保持在 $1/10Q_T$ 之间,将偏差取绝对值并对其作归一化处理,乘以 10 得到调整函数的参数 x。接着以 x 为参照量,将它作为调整函数的横坐标,然后将[0,1]这个区间划分为[0,0.3]、[0.3,0.7]、[0.7,1]。此区间内变化规律为:比例系数 K_p 在[0,0.3]单调递减,在[0.7,1]单调增加,在[0.3,0.7]有减有增,在 $x=0.5$ 取得最小值,在 $x=0$ 和 $x=1$ 两点取得最大值,调整函数均取为二次函数,保证了调整函数的连贯性;微分系数 K_d 在[0,1]单调递减,在 $x=0$ 时取得最大值,在 $x=1$ 时取得最小值。比例系数和微分系数在[0,1]以变化规律依照调整函数进行动态调整,超过 1 就保持一恒定倍率,即维持 $x=1$ 时的取值。将 K_{p0} 的倍率设置在[1/5,5],K_{d0} 的倍率设置在[1/2,2],即 K_{p0} 和 K_{d0} 设定完初值后在这个倍率之间进行波动调整,具体的调整值遵循上述公式中的函数。IPD-RED 控制流程如图 1 所示。

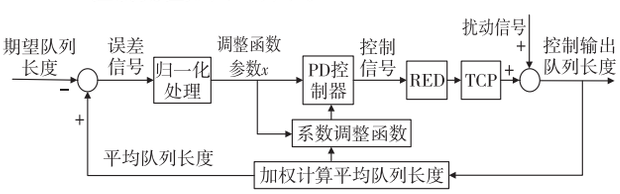


图 1　IPD-RED 控制流程

Fig. 1　Control flow chart of IPD-RED

图 1 中,平均队列长度和期望队列长度计算得到误差信号,归一化处理得到调整函数参数 x,然后将 x 输入系数调整函数中。PD 控制器完成比例微分控制的作用,其受到系数调整函数的影响,通过控制信号调整最大丢弃概率 max_p,实际丢包概率按照 RED 算法的方式进行计算,从而达到 TCP 拥塞控制的作用。本文设计的 IPD-RED 算法与模糊逻辑控制、应用隶属度函数调整的算法差异在于,应用的调整函数比较简单、易设计,引入了调整函数的参数 x,该变量由误差信号计算归一化后得出。参数 x 计算式中对误差信号即偏差取绝对值是为了在设计调整函数时能更加简便,取绝对值可以使负数域的波动范围映射到正数区间,从而在设计调整函数时只需考虑设计正半轴的函数即可,减小设计复杂性。算法设法改变最大丢弃概率 max_p,通过调整 max_p 来使队列长度保持稳定,并尽量地靠近期望队列长度附近,从而使网络链路利用率、网络延时、丢包率等方面能有更好的表现。

3　算法仿真

为了比较三种主动队列管理算法的性能差异,本文采用的仿真平台及仿真工具为 VMware Workstation Pro 16.1.1、Ubuntu20.04和 NS2(ns-2.35)[17~19]。仿真网络的拓扑结构如图 2 所示。

r_1 和 r_2 作为路由器节点,数据的传输都要经过这两个节点。$n_1 \sim n_{10}$为端节点,可作为数据的源端与终端。它们之间可以任意传输数据,实验中设置的数据传输方向为 n_1 到 n_6,n_2 到 n_7,n_3 到 n_8,n_4 到 n_9,n_5 到 n_{10}。$n_1 \sim n_5$ 连接着 r_1 节点,它们链路的带宽与延时设置为 10 Mbps、3 ms;$n_6 \sim n_{10}$连接着 r_2 节点,它们链路的带宽与延时设置为 10 Mbps、3 ms;r_1 和 r_2 链路之间的带宽与延时设置为 10 Mbps、5 ms。在此情况下对链路配置服务,对于五条传输链路配置 FTP 业务,并在数据层使用 TCP 协议。此外,对于队列管理算法的配置,$n_1 \sim n_5$ 连接 r_1 节点之间的链路配置 DropTail(队尾丢弃队列),$n_6 \sim n_{10}$连接 r_2 节点之间的链路配置 DropTail,r_1、r_2 节点之间的链路依次配置 RED、PD-RED、IPD-RED。算法中配置的基本参数为:RED 中 $thresh=20$,$maxthresh=80$,$w_q=0.002$,$max_p=0.1$,$mean\_pktsize=500$;PD-RED 和 IPD-RED 中 $K_{p0}=0.002$,$K_{d0}=0.05$;仿真时间设置为 60 s,从 0 时刻开始传输 TCP 流。上述配置中,$thresh$ 和 $maxthresh$ 为门限值的包大小,乘上后面的 $mean\_pktsize$(单位为 Byte)便是 min_{th} 和 max_{th} 两个门限值,因此期望队列长度 Q_T 以包大小表示为 $(thresh+maxthresh)/2=50$。

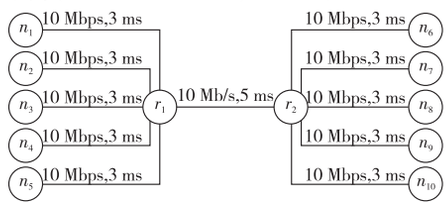


图 2　仿真网络拓扑

Fig. 2　Simulation network topology

3.1　队列图

设置上述的基本配置信息后,首先通过网络仿真软件来观察队列图的情况,用 NS2 仿真器对三个算法分别进行仿真,得到如图 3 所示的队列。

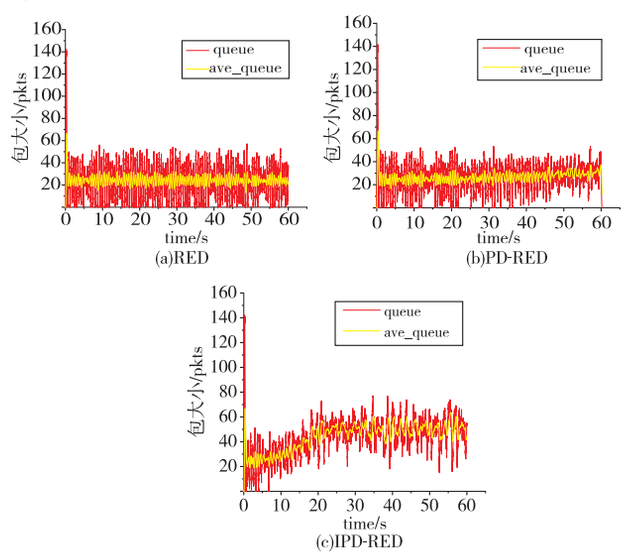


图 3　算法的队列

Fig. 3　Queue graph of algorithms

图 3 中,红色表示的是瞬时队列长度数据,黄色表示的是平均队列长度数据(见电子版)。图(a)可以看出,RED 算法的期望队列长度是 50 个包大小,但实际仿真出来的结果在 20~30,这是 RED 算法的一个缺陷。图(b)可以看出,PD-RED 算法的平均队列长度在一定时间后上升,从这幅图中看到,上升的幅度并不是很大。图(c)可以看出,IPD-RED 算法的平均队列长度在一定时间后上升,上升幅度较 PD-RED 算法明显,最终趋于期望值,在期望值附近上下波动。

从仿真的结果暴露出来,RED 算法的缺陷之一在于该算法并不能达到设计预期的期望队列长度 Q_T,它离期望队列长度还有显著差距。PD-RED 算法能在 RED 基础上有所改善,但该初始条件下改进效果不显著,IPD-RED 较 PD-RED 算法对

(C)1994-2023 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

于平均队列长度的改进程度较明显，经过一定时间后能接近期望值。从与期望队列长度的接近程度来看，RED、PD-RED、IPD-RED 呈逐步改良与优化的关系。

3.2 thresh 组合的影响分析

为了进一步说明三个算法之间的性能差异，实验中引入更多网络仿真中的参数来观察它们之间的改进情况。接下来，通过吞吐量和丢包率这两个参数来观察各个算法之间的差异。吞吐量和丢包率通过运行 awk 脚本文件[20,21]来得到。NS2 首先运行 tcl 脚本文件，以此来调用到队列管理算法，仿真软件运行结束后在文件夹中生成 tr 数据流文件。然后运行 awk 脚本文件，它可以分析仿真过程中生成的 tr 数据流，对于实验需要的列数据进行读取并加载到 awk 脚本文件中，由不同脚本逻辑得出不同的网络仿真参数。此外，考虑到参数选取对于算法性能的影响，因此在分析吞吐量和丢包率这两个参数时，通过控制变量的方法，分别改变 *thresh*/*maxthresh*、路由节点间延时 *delay* 和端节点数量 *node* 的数值来分析算法性能差异。首先保持 *delay*、*node* 不变，通过仿真来观察 *thresh*/*maxthresh* 不同组合的影响。

3.2.1 吞吐量

在 *delay* = 5 ms，*node* = 10 的条件下，保持期望队列长度 Q_T 的数值为 50 个包大小，但改变 *thresh* 和 *maxthresh* 的数值，由于建议的 *maxthresh* 数值要在 *thresh* 的两倍以上[22]，所以选取以下几种组合情况（由于仿真结束时刻为 60 s，所以选取 60 s 时的吞吐量作对比，吞吐量单位为 Kbps）。以 case 表示各种组合，case1：*thresh* = 10，*maxthresh* = 90；case2：*thresh* = 15，*maxthresh* = 85；case3：*thresh* = 20，*maxthresh* = 80；case4：*thresh* = 25，*maxthresh* = 75；case5：*thresh* = 30，*maxthresh* = 70。通过运行 awk 脚本文件可得到仿真过程中的吞吐量数值，选取 60 s 时刻的数据进行汇总，如表 1 所示。

表 1 不同 *thresh* 组合情况下的吞吐量

Tab. 1 Throughput under different *thresh* combinations

算法	case1	case2	case3	case4	case5
RED	10 293.62	10 314.45	10 228.61	9 674.23	9 298.56
PD-RED	10 340.23	10 334.44	10 318.02	10 258.29	9 970.16
IPD-RED	10 348.58	10 344.82	10 357.66	10 303.50	10 279.48

通过表 1 可以看出，无论哪一种组合情况下，在吞吐量方面，RED、PD-RED、IPD-RED 呈现逐步增大的趋势。在 case1、case2、case3 下，算法的改善程度比较小；在 case4、case5 下，算法的改善程度较显著。这是由于在前面几种情况下，*thresh* 和 *maxthresh* 差值更大，对 RED 算法来说能取得更优异的性能，改善效果不显著，随着上下门限值差值变小，RED 算法性能变差，改善效果渐渐明显。在 case1、case2、case3 下，三种算法的吞吐量都可以保持一个较高的值，尤其是 RED 算法与 case4、case5 对比。PD-RED 与 IPD-RED 在这几种 case 下均可以保持较高的吞吐量且保持着逐步提升的关系，说明实现了这一参数方面的改善效果。

3.2.2 丢包率

同样在上述条件下，运行 awk 脚本文件得到丢包数量和传输的包总量，两者相除得到丢包率，具体如表 2 和图 4 所示。

表 2 不同 *thresh* 组合情况下的丢包率

Tab. 2 Packet loss rate under different *thresh* combinations /%

算法	case1	case2	case3	case4	case5
RED	3.145	2.868	2.737	2.740	2.753
PD-RED	2.765	2.685	2.552	2.458	2.496
IPD-RED	1.841	1.820	1.810	1.848	1.888

通过表 2 和图 4 可以看出，无论哪一种组合情况下，在丢包率方面，RED、PD-RED、IPD-RED 呈现逐步减小的趋势，且改善的效果均比较明显。在几种情况下，RED 算法在 case3 下取得了最小的丢包率，PD-RED 算法在 case4 下取得了最小的丢包率，IPD-RED 在 case3 下取得了最小的丢包率。IPD-RED 算法在每一种 case 下都能保持较低的丢包率且波动不大。

3.3 延时影响分析

综合上述两个表格的结果来看，case3 是一个较优的选择，即 *thresh* = 20，*maxthresh* = 80 的阈值条件下，RED 在丢包率和吞吐量这两个方面的综合性能最优，PD-RED、IPD-RED 也能有接近最优的性能。因此，这里选择 *thresh* = 20，*maxthresh* = 80 这个条件，通过改变路由节点间延时 *delay* 的大小，同样也是通过观察吞吐量和丢包率这两个参数来进行分析。吞吐量和丢包率也是通过上述同样的 awk 脚本文件分析得出。

3.3.1 吞吐量

选取 *delay* 为 5 ms、10 ms、15 ms、20 ms、25 ms、30 ms 进行仿真。整理 60 s 时吞吐量的数据进行汇总，如表 3 所示。

表 3 不同延时下的吞吐量对比

Tab. 3 Throughput comparison under different delays

算法	5 ms	10 ms	15 ms	20 ms	25 ms	30 ms
RED	10 228.61	9 893.56	9 527.72	9 256.88	9 043.19	8 950.67
PD-RED	10 318.02	10 034.03	9 733.54	9 451.92	9 272.94	9 148.58
IPD-RED	10 357.66	10 269.02	10 185.00	9 976.26	9729.70	9424.51

通过表 3 可以看出，随着延时的增大，不同算法的吞吐量均表现出下降的趋势。在每个不同的延时下，RED、PD-RED、IPD-RED 算法三者表现出吞吐量逐渐增加的情况，表明改进算法的确实现了吞吐量性能的改进。

3.3.2 丢包率

接下来同样利用 awk 脚本文件得到包数据，计算丢包率与汇总数据，具体如表 4 和图 5 所示。

表 4 不同延时下的丢包率对比

Tab. 4 Comparison of packet loss rates under different delays /%

算法	5 ms	10 ms	15 ms	20 ms	25 ms	30 ms
RED	2.737	2.259	1.914	1.671	1.482	1.226
PD-RED	2.552	2.191	1.830	1.590	1.389	1.200
IPD-RED	1.810	1.674	1.507	1.398	1.202	1.083

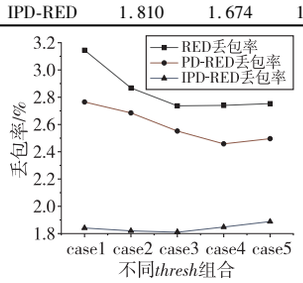


图 4 不同 *thresh* 组合情况下的丢包率对比

Fig. 4 Comparison of packet loss rate under different *thresh* combinations

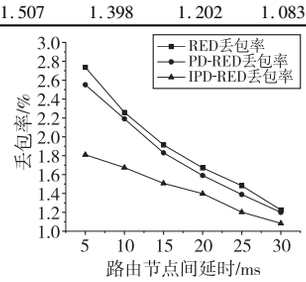


图 5 不同延时下的丢包率对比

Fig. 5 Comparison diagram of packet loss rate under different delays

通过表 4 和图 5 可以看出，在一定的延时范围内，随着延时的增加，各个算法的丢包率均表现出减小的趋势，且在不同的延时下，RED、PD-RED、IPD-RED 的丢包率均逐次减小。在延时较小的情况下，RED、PD-RED、IPD-RED 之间的改善效果较为显著，随着延时的增大，改善效果渐渐变小。

综合上述两组改变不同变量的实验最终得出，改变 *thresh*/*maxthresh* 的值以及 *delay* 的值，用相同的脚本分析几个算法之间的吞吐量与丢包率，可以发现，在一定的参数范围内，RED、PD-RED、IPD-RED 之间在吞吐量和丢包率方面保持着一致性的规律：在吞吐量方面，吞吐量大小始终保持着 IPD-RED > PD-RED > RED 这样一个趋势；在丢包率方面，丢包率大小始终保持着 IPD-RED < PD-RED < RED 这样一个趋势。这样的趋势说明了 RED 改进算法的合理性，实现了它们的改进功能，在吞吐量和丢包率方面有明显的改善。此外，在上面两组实验中，也对延时和队列振荡的时间进行了脚本分析。经实验发现，三个算法在延时和振荡这两个方面并无明显的改善关系，而是保

(C)1994-2023 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

持着整体相近的数值。

3.4　端节点数量影响分析

然后研究端节点数量变化会对算法的网络参数造成何种影响。选择 *thresh* = 20,*maxthresh* = 80,*delay* = 30 ms 这一条件,选取端节点为 10、20、40、60 进行仿真,观察高负载下网络参数如何变化。

3.4.1　吞吐量

运行 awk 脚本得到 60 s 时的吞吐量数据,如表 5 所示。

表 5　不同端节点数量下的吞吐量

Tab. 5　Throughput under different number of end nodes

算法	10	20	40	60
RED	8 950.67	9 315.48	10 067.28	10 318.97
PD-RED	9 148.58	9 367.73	10 043.32	10 314.64
IPD-RED	9 424.51	9 652.81	10 187.39	10 331.77

通过表 5 可以看出,随着端节点数量增加,各算法吞吐量都呈增加趋势。PD-RED 算法在负载量小时吞吐量高于 RED 算法,随着负载增加,一定程度后吞吐量小于 RED 算法。IPD-RED 算法的吞吐量一直处于最优,但随着负载增加改善效果渐渐变小,各算法的网络吞吐量渐渐趋于瓶颈容量。

3.4.2　丢包率

计算丢包率数据如表 6 所示。

表 6　不同端节点数量下的丢包率

Tab. 6　Packet loss rate under different number of end nodes　/%

算法	10	20	40	60
RED	1.226	3.759	7.957	11.568
PD-RED	1.200	3.657	7.929	11.585
IPD-RED	1.083	3.388	7.605	11.374

通过表 6 可以看出,随着端节点数量增加,各算法的丢包率都呈增加趋势。在负载量小时,PD-RED 的丢包率小于 RED 算法,在负载大于一定程度后丢包率大于 RED 算法。IPD-RED 算法的丢包率一直处于最优,但随着负载增加改善效果渐渐变小,说明高负载下各算法的丢包率指标变差,渐渐趋同。

此外,仿真得出,随着负载增加,队列长度均值渐渐增加并靠近期望值,但其余的网络性能参数都呈现恶化趋势,总体说明高负载不利于队列管理算法的执行。

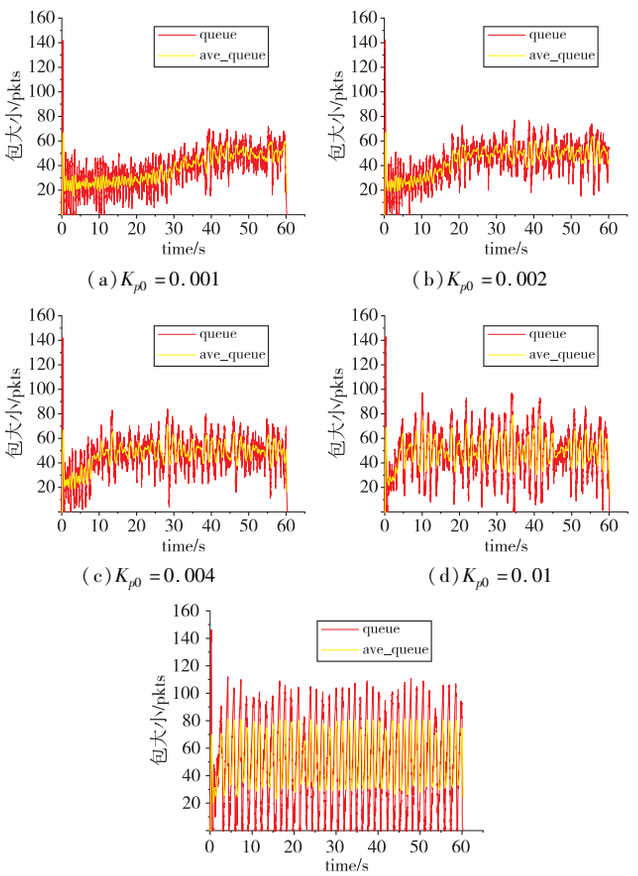
3.5　初值影响分析

对于初值的考虑,是由于其余文献中并未详细提及初值变化会对算法有何影响,也没有研究人员对初值进行改变,并借此来分析可能会对算法的哪些网络参数造成影响。因此本文欲探究比例系数 K_{p0} 和微分系数 K_{d0} 的初值变化对 IPD-RED 算法性能的影响。实验中将初值变为原来的 0.5 倍、2 倍、5 倍、10 倍再次进行仿真比较(保持 *delay* = 5 ms,*node* = 10,*thresh* = 20,*maxthresh* = 80 这一条件)。

3.5.1　比例系数的影响

实验先观察比例系数的初值变化可能会对算法性能造成的影响。保持 K_{d0} 不变,改变 K_{p0} 的初值。仿真队列如图 6 所示。

对于延时、振荡的期望和方差、吞吐量、丢包率数据,如表 7 所示。由图 6 可以看出,随着 K_{p0} 的增大,可以使队列在更短的时间内靠近期望值附近,而且最终平均队列长度离目标期望值(50)更接近。但可以看出从 K_{p0} = 0.01 开始,平均队列长度在期望值附近的抖动明显变大,说明 K_{p0} 初始值到一定程度时会使队列最终稳定性变差。从表 7 可以看出,在选取的范围内,随着 K_{p0} 的增加:a)丢包率先减小后增大;b)一定范围内吞吐量无较大差异,在 K_{p0} 大到一定程度后明显减小;c)延时和振荡的期望与方差均呈现增大趋势。综上可知,K_{p0} 的变化对吞吐量的影响较小,会使队列图、丢包率、延时和振荡发生较大的差异。提高 K_{p0} 可使队列更快更好地接近目标期望值,同时也可以使丢包率减小,但这会带来延时和振荡的增加。因此在实际情况下要适当选择。本实验中,设定侧重于延时和振荡的数据,因此选取 K_{p0} = 0.002,在更侧重丢包率的情况下可选取 K_{p0} = 0.004。



(a) K_{p0} = 0.001　(b) K_{p0} = 0.002

(c) K_{p0} = 0.004　(d) K_{p0} = 0.01

(e) K_{p0} = 0.02

图 6　改变 K_{p0} 后的队列

Fig. 6　Queue diagram after changing K_{p0}

表 7　不同 K_{p0} 下的 IPD-RED 参数

Tab. 7　IPD-RED parameters under different K_{p0}

参数	K_{p0}				
	0.001	0.002	0.004	0.01	0.02
丢包率/%	2.070	1.810	1.670	1.697	2.749
吞吐量/kbps	10 329.34	10 357.66	10 346.53	10 359.94	10 104.98
延时/s	0.028 26	0.030 98	0.032 38	0.033 09	0.034 30
振荡期望/s	0.001 35	0.001 46	0.001 59	0.001 69	0.001 76
振荡方差/s	0.002 06	0.002 25	0.002 46	0.002 60	0.002 69

3.5.2　微分系数的影响

接下来保持 K_{p0} = 0.002,改变 K_{d0} 的初值,来观察微分系数的变化又会使 IPD-RED 算法的性能产生何种变化。同样将初值变为原来的 0.5 倍、2 倍、5 倍、10 倍再次进行仿真比较,如表 8 所示。经实验发现,K_{d0} 的变化对于队列图的影响几乎不可见,因此这里不进行讨论。观察表 8 发现,在选取的范围内,K_{d0} 变化对于其余几个参数的影响也很小,说明该微分系数的取值范围动态空间很大。

表 8　不同 K_{d0} 下的 IPD-RED 参数

Tab. 8　IPD-RED parameters under different K_{d0}

参数	K_{d0}				
	0.025	0.05	0.1	0.25	0.5
丢包率/%	1.827	1.810	1.821	1.810	1.821
吞吐量/kbps	10 357.66	10 357.66	10 357.48	10 357.48	10 357.48
延时/s	0.030 89	0.030 98	0.031 01	0.030 94	0.030 98
振荡期望/s	0.001 48	0.001 46	0.001 46	0.001 56	0.001 45
振荡方差/s	0.002 28	0.002 25	0.002 25	0.002 38	0.002 25

(C)1994-2023 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

4 结束语

本文基于主动式队列管理算法,在 PD-RED 算法的基础上进行改进,提出了 IPD-RED 算法,并通过实验比较了 RED、PD-RED、IPD-RED 算法。经实验发现,改变不同参数的情况下,IPD-RED 算法均能够在吞吐量和丢包率方面对于 PD-RED 算法有所改进,即提高了吞吐量、减小了丢包率。同时实验也验证了 PD-RED 对于 RED 的改进关系。在队列长度方面,IPD-RED 算法能够更快更好地接近期望,PD-RED 次之,RED 这方面性能最差。在实验分析比例系数和微分系数的初值影响后,发现一定范围内比例系数的增大可使平均队列长度更快更接近期望值且减小丢包率,但会使延时和振荡增加。微分系数的动态范围很大,对几个网络性能参数的影响很小。实际应用可以适当选取初值来使队列管理功能更好地实现。

参考文献:

[1] Maheshwari M K. NexGen D-TCP: next generation dynamic TCP congestion control algorithm [J]. IEEE Access, 2020, 8: 164482-164496.

[2] 秦剑秀,宋建霖,戴小文. 改进 WFQ 算法在列车网络调度中的应用研究[J]. 计算机应用研究,2019,36(11):3302-3305. (Qin Jianxiu, Song Jianlin, Dai Xiaowen. Research on application of improved WFQ algorithm in train network scheduling[J]. Application Research of Computers,2019,36(11):3302-3305.)

[3] Khatari M, Zaidan A A, Zaidan B B, *et al*. Multi-criteria evaluation and benchmarking for active queue management methods: open issues, challenges and recommended pathway solutions [J]. International Journal of Information Technology & Decision Making,2019,18(4):1187-1242.

[4] Abdelmoneim A M, Bensaou B. Hysteresis-based active queue management for TCP traffic in data centers[C]//Proc of IEEE INFOCOM. Piscataway, NJ: IEEE Press,2019:1621-1629.

[5] 张义,赵旭,涂华,等. 无线局域网中增强 3D 视频传输的队列管理机制[J]. 中国电子科学研究院学报,2017,12(3):246-250. (Zhang Yi, Zhao Xu, Tu Hua, *et al*. A queue management mechanism to improve 3D video transmission in WLAN[J]. Journal of CAEIT, 2017,12(3):246-250.)

[6] 张旭,顾乃杰,谷德贺,等. 一种高效通用的 TCP 尾部丢包恢复算法[J]. 小型微型计算机系统,2017,38(9):1921-1927. (Zhang Xu, Gu Naijie, Gu Dehe, *et al*. Efficient universal tail loss recovery algorithm for TCP [J]. Journal of Chinese Computer Systems, 2017,38(9):1921-1927.)

[7] 谭尧木. 自适应队列管理算法研究[D]. 北京:华北电力大学, 2019. (Tan Yaomu. Research on adaptive queue management algorithm[D]. Beijing: North China Electric Power University,2019.)

[8] 刘正飞,孙金生,胡斯乔. 基于模型有效性评价的主动队列管理算法[J]. 控制理论与应用,2019,36(4):570-578. (Liu Zhengfei, Sun Jinsheng, Hu Siqiao. An active queue management algorithm based on model effectiveness evaluation[J]. Control Theory & Applications,2019,36(4):570-578.)

[9] Hotchi R, Chibana H, Iwai T, *et al*. Active queue management supporting TCP flows using disturbance observer and Smith predictor[J]. IEEE Access,2020,8:173401-173413.

[10] 吉晓香. 一种面向网络拥塞的 AQM 算法研究[J]. 现代电子技术,2019,42(14):74-77,82. (Ji Xiaoxiang. Research on AQM algorithm dealing with network congestion[J]. Modern Electronics Technique,2019,42(14):74-77,82.)

[11] Rastogi S, Zaheer H. Comparative analysis of queuing mechanisms: Droptail, RED and NLRED[J]. Social Network Analysis and Mining,2016,6(1):123-232.

[12] 冯思楠,康巧燕,王建峰,等. 一种基于 BP 神经网络预测的 RED 改进算法[J]. 计算机与网络,2019,45(3):68-71. (Feng Sinan, Kang Qiaoyan, Wang Jianfeng, *et al*. An Improved RED algorithm based on BP neural network prediction[J]. Computer & Network, 2019,45(3):68-71.)

[13] Sun Jinsheng, Ko K T, Chen Guanrong, *et al*. PD-RED: to improve the performance of RED[J]. IEEE Communications Letters, 2003, 7(8):406-408.

[14] Hamadneh N, Al-Kasassbeh M, Obiedat I, *et al*. Revisiting the gentle parameter of the random early detection (RED) for TCP congestion control[J]. Journal of Communications,2019,14(3):229-235.

[15] Azami G, Gholizade N H. Bandwidth management with congestion control approach and fuzzy logic[J]. International Journal of Engineering,2021,34(4):891-900.

[16] 魏涛,张顺颐. 一种模糊自调整的 PD-RED 算法[J]. 计算机工程与应用,2007,43(5):124-126,162. (Wei Tao, Zhang Shunyi. A fuzzy self-tuning PD-RED algorithm [J]. Computer Engineering and Applications,2007,43(5):124-126,162.)

[17] 张家钢,王海松,胡洛林. 基于 NS2 的 AODV 路由协议分析及仿真研究[J]. 现代信息科技,2020,4(3):52-54. (Zhang Jiagang, Wang Haisong, Hu Luolin. Analysis and simulation of AODV routing protocol based on NS2[J]. Modern Information Technology,2020, 4(3):52-54.)

[18] Gupta A, Agrawal K, Bansal P, *et al*. Performance evaluation of various transmission control protocols in NS2[M]. Berlin: Springer,2020.

[19] Issariyakul T, Hossain E. Introduction to network simulator NS2[M]. 2nd ed. Berlin: Springer,2009.

[20] Free Software Foundation. Gawk: effective AWK Programming[EB/OL]. (2021)[2021-10-20]. http://www.gnu.org/software/gawk/manual/.

[21] Ahmed K T, Hossen M R, Sana S, *et al*. Estimating a suitable routing protocol in MANET for CBR and FTP traffic for varying pause time [J]. IOSR Journal of Electronics and Communication Engineering,2019,14(6):53-58.

[22] 曹志波. 基于 NS-2 的 RED 算法的优化策略[D]. 焦作:河南理工大学,2010. (Cao Zhibo. Optimization strategy of RED algorithm based on NS-2[D]. Jiaozuo: Henan Polytechnic University,2010.)

(上接第 1682 页)

[18] Basile F, Cabasino M P, Seatzu C. State estimation and fault diagnosis of labeled time Petri net systems with unobservable transitions[J]. IEEE Trans on Automatic Control,2015,60(4):997-1009.

[19] Basile F, Cabasino M P, Seatzu C. Diagnosability analysis of labeled time Petri net systems[J]. IEEE Trans on Automatic Control, 2017,62(3):1384-1396.

[20] 李志武,周孟初. 自动制造系统建模、分析与死锁控制[M]. 北京:科学出版社,2009. (Li Zhiwu, Zhou Mengchu. Modeling, analysis and deadlock control of automated manufacturing system[M]. Beijing: Science Press,2009.)

[21] Li Liang, Basile F, Li Zhiwu. Closed-loop deadlock-free supervision for GMECs in time Petri net systems[J]. IEEE Trans on Automatic Control,2021,66(11):5326-5341.

[22] He Zhou, Li Zhiwu, Giua A, *et al*. Some remarks on "state estimation and fault diagnosis of labeled time Petri net systems with unobservable transitions" [J]. IEEE Trans on Automatic Control, 2019, 64(12):5253-5259.

[23] Liu Baisi, Ghazel M, Toguyéni A. Diagnosis of labeled time Petri nets using time interval splitting [J]. IFAC Proceedings Volumes, 2014,47(3):1784-1789.

[24] 张信哲,张治国,丁晓彬,等. 部分可观时间 Petri 网故障的贝叶斯诊断[J]. 应用科技,2020,47(1):61-67. (Zhang Xinzhe, Zhang Zhiguo, Ding Xiaobin, *et al*. Bayesian diagnosis of partial observable time Petri net faults[J]. Applied Science and Technology,2020, 47(1):61-67.)

(C)1994-2023 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net