

# 一种改进 ARED 拥塞控制算法的实现

薛 礼

(湖北汽车工业学院 电信学院,湖北 十堰 442002)

**摘 要:**随着互联网的迅速发展,无论是网民人数还是上网设备数都呈现高速增长的态势。虽然带宽等互联网基础资源相比二十年前有了质的飞跃,但是由于网络规模的增加还是带来了一系列的问题,其中网络拥塞是比较典型的一个。RED 作为路由器主动队列管理策略中的重要算法已经在网络拥塞控制方面起到了很好的效果,成为 IETF RFC2309 建议的唯一候选算法。与队尾丢弃算法 DropTail 相比,RED 算法具有网络链路利用率较高、吞吐量较大、网络时延和丢包率较小的优点,但其存在参数配置无法适应网络动态变化的缺陷,因而改进的 ARED 算法增加了自适应的功能,但也存在瞬时队列长度振荡等稳定性问题。对此,研究了 RED 及 ARED 拥塞控制算法,并提出了一种改进算法 QARED,希望通过优化最大丢包概率计算函数来达到提高平均队列长度稳定性以及降低丢包率的目的。

**关键词:**拥塞控制;ARED;主动队列管理;NS2 网络模拟;队列长度振荡

**中图分类号:**TP393.06

**文献标识码:**A

**文章编号:**1673-629X(2020)03-0117-05

doi:10.3969/j.issn.1673-629X.2020.03.022

## Implementation of an Improved ARED Congestion Control Algorithm

XUE Li

(School of Electrical and Information Engineering, Hubei University of  
Automotive Technology, Shiyan 442002, China)

**Abstract:** With the rapid development of the Internet, both the Internet users and the Internet devices have been growing rapidly. Although bandwidth and other basic Internet resources have made a qualitative leap compared with 20 years ago, the increase of network size still brings a series of problems, among which the network congestion is a typical one. As an important algorithm in the active queue management strategy of routers, RED has played a role in network congestion control, becoming the only candidate algorithm recommended by IETF RFC2309. It has higher link utilization, larger throughput and lower network latency and packet loss rate compared with DropTail algorithm, but its parameter configuration can't adapt to the dynamic change of network. So the improved ARED algorithm increases the adaptive function, but there are also stability problems such as instantaneous queue length oscillation. We study the RED and ARED congestion control algorithm and propose an improved algorithm QARED, aiming to improve stability and reduce the packet loss rate by optimizing the biggest packet loss probability calculation function.

**Key words:** congestion control; ARED; active queue management; NS2 network simulation; queue length oscillation

## 0 引言

进入二十一世纪以来,基于互联网的应用越来越广泛,网络数据流量急剧增加,带来的一个严重问题就是网络拥塞。当网络上出现拥塞时,会造成通信双方数据包传输时延增大、数据包容易丢失并重发、网络吞吐量降低等问题,严重的情况下会导致网络死锁和瘫痪,因此 TCP/IP 网络体系结构实现的过程中在运输层中引入了拥塞控制策略。目前运输层中最重的 TCP 协议各版本实现机制中均引入了慢启动、拥塞避免、快

速重传等一系列算法来实现可靠的拥塞控制<sup>[1]</sup>。这种拥塞控制机制是建立在端到端的基础上,在早期互联网运行过程中能够很好地保障网络的可靠性和稳定性,提升服务质量 QoS,因此传输控制协议 TCP 成为一种可靠的运输层协议<sup>[2]</sup>。但随着互联网规模的激增,通信子网的结构越来越复杂,大量数据包涌入到通信子网的节点路由器上,造成网络层的数据拥塞,这样仅仅依靠端到端的拥塞控制机制已经无法满足需求<sup>[3]</sup>。于是近些年来路由器上的拥塞控制机制也逐渐

收稿日期:2019-03-10

修回日期:2019-07-12

网络出版时间:2019-12-05

基金项目:湖北省教育科学研究计划项目(B2016086)

作者简介:薛 礼(1978-),男,硕士,副教授,研究方向为计算机基础教育、网络安全与应用。

网络出版地址: <http://kns.cnki.net/kcms/detail/61.1450.TP.20191205.1104.004.html>

部署起来,称为节点拥塞控制机制,对于这类研究主要以主动队列管理 AQM 算法为主<sup>[4]</sup>。主动队列管理是一种基于先进先出(first input first output)调度策略的队列管理,能够控制路由器丢弃数据包的时间和数量。其核心思想是通过对网络拥塞实施早期监测,一旦发现拥塞的可能性,便主动向发送端发出拥塞信号,这时发送端便会降低数据包的发送速率,从而减少网络中数据包的数量,达到降低数据包丢失率和提高通信链路吞吐量的目的。在 AQM 算法控制下,路由器缓冲区出现溢出之前以一定的策略丢弃数据包,避免了队列溢出,解决了满队列问题。从拥塞控制的角度看,传统的队列管理算法只包含拥塞恢复机制,而 AQM 算法既包含拥塞恢复机制又包括拥塞避免机制,因此 AQM 算法对于提高拥塞控制的性能具有比较重要的作用,在路由器中使用 AQM 算法可以带来的好处包括:

(1)降低路由器中被丢弃数据包的数量。互联网中数据包的突发性从根本上是不可避免的,而 AQM 算法通过使路由器保持较短的平均队列长度(average queue length),从而能够解决吸收突发数据包的问题,大大降低了突发数据包丢失的可能性。

(2)交互式应用实现更低的延时。AQM 算法中路由器可以保持较短的平均队列长度,因此能够降低数据包的排队时延(queuing delay),而排队时延是端到端时延的重要组成部分,这对交互式应用(如 WEB 浏览、Telnet 和视频会议等)有重要帮助。

(3)避免网络“死锁”现象。“死锁”是指当拥塞严重到网络无吞吐量的现象,AQM 算法能够确保路由器几乎总有可用的队列空间来容纳到达的数据包,从而可以防止“死锁”情况的发生。同时因为这个原因,AQM 算法也能防止路由器对突发数据流的偏见。

随机早期丢弃算法(random early detection, RED)是 AQM 算法的一个代表,相比队尾丢弃算法 DropTail,RED 算法具有网络链路利用率较高、吞吐量较大、网络时延和丢包率较小的优点<sup>[5-6]</sup>。因此互联网工程任务组 IETF 将其作为唯一拥塞控制候选算法极力推荐使用,希望达到良好的节点拥塞机制的效果。

## 1 RED 及 ARED 算法原理

### 1.1 RED 算法概述

当节点路由器实施了 RED 算法,它会实时监控缓冲区平均队列长度,因为其作为一个重要指标用来判断网络是否出现拥塞。如果发现网络有出现拥塞的可能性,那么路由器便会以一定的概率丢弃接收到的数据包,并向发送端发送通知,发送端收到通知后会降低数据包的发送速率,这样进入网络的数据包便会减缓,

使路由队列长度维持在一个良性的状态,从而降低了网络传输延迟和丢包率,提高了网络资源的利用率,达到有效缓解网络拥塞的目的<sup>[7-8]</sup>。与传统 DropTail 算法相比,RED 算法改进点主要体现在两个方面:第一,路由器不是等队列全部填满后再丢弃随后到来的数据包,而是在有可能出现拥塞之前通过计算概率随机丢弃部分数据包来预防可能出现的拥塞;第二,计算数据包随机丢弃概率时采纳平均队列长度而不是瞬时队列长度,因此能尽量满足突发数据流对路由器缓冲区空间的需求,降低产生网络全局同步现象的可能性。

当路由器实施 RED 算法控制机制时,其针对缓冲区队列事先设置了两个重要参数,即最小队列长度阈值  $TH_{min}$  和最大队列长度阈值  $TH_{max}$ 。路由器接收到一个新到达的数据包时,首先计算缓冲区队列平均长度  $L_{av}$ ,如果  $L_{av}$  小于阈值  $TH_{min}$ ,则丢弃概率为 0,允许新到的数据包进入队列排队;如果  $L_{av}$  大于最大阈值  $TH_{max}$ ,则丢弃概率为 1,将新到的数据包丢弃;如果  $L_{av}$  在阈值  $TH_{min}$  和  $TH_{max}$  之间,则会依据控制策略中的相关机制计算丢包概率  $P$ ,并将新收到的该数据包按此概率实施丢弃。RED 控制策略中计算平均队列长度  $L_{av}$  的方法是将瞬时队列长度  $L_{sa}$  和旧的平均队列长度  $L_{av}$  进行加权计算,其中权值  $W_q$  根据实际情况取  $[0,1]$  之间的值<sup>[9]</sup>,  $L_{av}$  的计算如式(1)所示。

$$L_{av} = (1 - W_q) \times L_{av} + W_q \times L_{sa} \quad (1)$$

根据式(1)计算出平均队列长度  $L_{av}$ ,与预先设定好的最小队列长度阈值  $TH_{min}$ 、最大队列长度阈值  $TH_{max}$  进行比较,通过比较结果来判定网络拥塞的程度,以此作为决定数据包丢弃概率  $P$  的依据。具体来说就是:如果  $L_{av} < TH_{min}$ ,则丢弃概率  $P$  为 0,允许新到的数据包进入路由器缓冲区队列中排队;如果  $L_{av} > TH_{max}$ ,则丢弃概率  $P$  为 1,不允许新到的数据包进入缓冲区排队,直接丢弃;若  $TH_{min} \leq L_{av} \leq TH_{max}$ ,会利用线性函数计算出丢弃概率  $P$ ,并以此概率丢弃新到的数据包。其中计算丢弃概率  $P$  会涉及到预先设定的最大丢弃概率  $P_{max}$ 、上一次丢包之后新进入队列的数据包数量 count 以及相关的计算函数,式(2)和式(3)为采用线性函数计算所得从 0 到  $P_{max}$  变化的  $P_b$  和丢弃概率  $P$ 。

$$P_b = P_{max} \times \frac{L_{av} - TH_{min}}{TH_{max} - TH_{min}} \quad (2)$$

$$P = \frac{P_b}{1 - \text{count} \times P_b} \quad (3)$$

从以上公式可以看出,丢弃概率  $P$  的值不仅和路由器缓冲区平均队列长度  $L_{av}$  有关,同时也会随着进入缓冲区队列数据包数量 count 的增加而平缓增大,这样会使新到数据包的丢弃时间间隔相对平滑,避免密

集地产生丢包现象<sup>[10]</sup>。

算法RED的劣势在于前面提到的各种参数的预置上,网络拥塞情况影响因素很多,是动态变化的,而相对固定的参数值不能很好地满足网络拥塞情况动态变化的需求<sup>[11]</sup>,因此对于RED算法的各种改进研究便集中在这点上面。

1.2 ARED 算法概述

针对上面提到的RED算法中固定参数无法适应动态网络需求的缺点,相关研究提出了一些改进方法,其中最具有代表性的就是自适应RED算法,简称为ARED。在ARED算法中,最大丢弃概率 $P_{max}$ 不是使用事先设定好的固定值,而是根据平均队列长度的动态变化来调整 $P_{max}$ 。具体检测策略就是如果缓冲区平均队列长度在 $TH_{min}$ 附近波动,说明数据包丢失过多,拥塞控制策略过于激进,应减小 $P_{max}$ ;如果缓冲区平均队列长度在 $TH_{max}$ 附近波动,说明数据包丢失过少,拥塞控制策略有点保守,应增大 $P_{max}$ 。动态调整算法的具体伪代码如下:

```
every intertime
if(  $L_{av}$  >targetl &&  $P_{max} \leq 0.5$  )
 $P_{max} = P_{max} + \alpha$  ;
else if(  $L_{av}$  <targetl &&  $P_{max} \geq 0.01$  )
 $P_{max} = P_{max} * \beta$  ;
```

其中intertime为检测缓冲区平均队列长度的间隔时间,一般取值为0.5秒;targetl表示缓冲区平均队列长度的目标值,其取值范围是 $[TH_{min} + 0.4 * (TH_{max} - TH_{min}), TH_{min} + 0.6 * (TH_{max} - TH_{min})]$ ;  $\alpha$ 是加法增大 $P_{max}$ 的因子,一般取值为 $\alpha = \min(0.01, 0.25 * P_{max})$ ,  $\beta$ 是乘法减少 $P_{max}$ 的因子,一般取值为 $\beta = 0.9$ 。

1.3 算法对比

以下将通过仿真网络环境来模拟两种算法,模拟环境的组建采用网络仿真软件NS2<sup>[12-13]</sup>,对应的网络拓扑结构如图1所示。

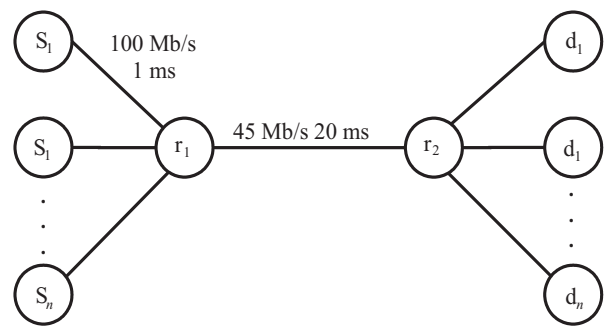


图1 仿真实验网络拓扑结构

路由器 $r_1$ 和 $r_2$ 之间的链路为瓶颈链路,带宽设置为45 Mb/s,往返时延设置为20 ms,在路由器 $r_1$ 和 $r_2$ 上依次实施网络拥塞算法RED和ARED。 $s_1$ 到 $s_n$ 为一定数量的发送实体, $d_1$ 到 $d_n$ 为对应数量的接收实

体,发送实体与路由器 $r_1$ 连接,接收实体与路由器 $r_2$ 连接,所有链路的带宽均设置为100 Mb/s,往返时延设置为1 ms。在发送实体 $s_n$ 及对应的接收实体 $d_n$ 之间建立TCP连接,使用FTP协议流量。在两种拥塞控制算法中所用到的参数分别设置为 $W_q = 0.002$ ,  $P_{max} = 0.1$ ,intertime = 0.5 s,  $TH_{min} = 40$  packets,  $TH_{max} = 180$  packets, Buffersize = 200 packets。建立80个发送实体和接收实体的并发连接,模拟运行时间100 s,在此过程中分别跟踪实施RED和ARED算法的路由器平均队列长度<sup>[14]</sup>,并绘制其变化如图2和图3所示。通过比较可以发现,ARED算法的平均队列长度振荡平缓,稳定性优于RED算法。

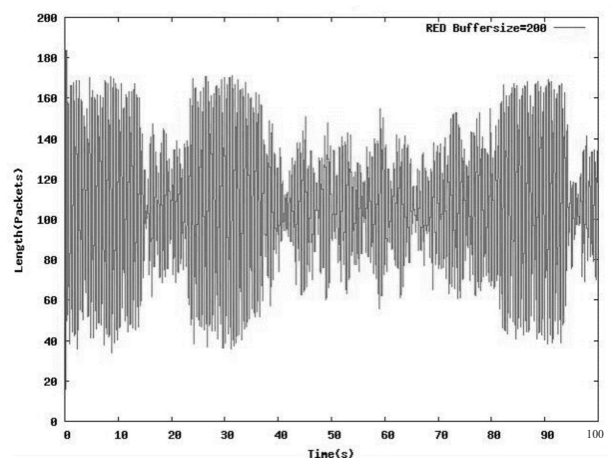


图2 RED 算法平均队列长度变化

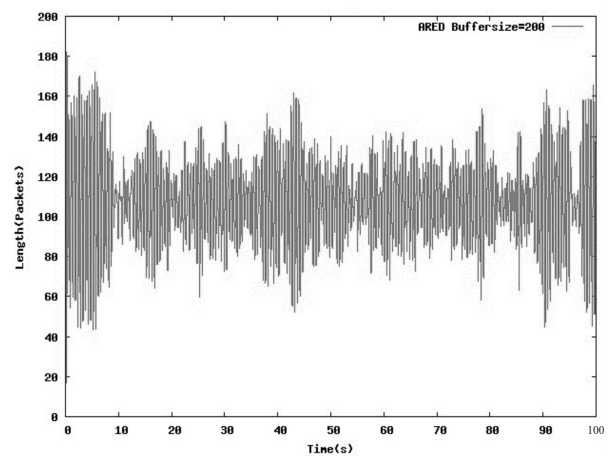


图3 ARED 算法平均队列长度变化

2 ARED 算法的优化

2.1 QARED 算法概述

ARED算法在控制策略上增加了对 $P_{max}$ 参数的动态修订,这使得丢包概率能够随着网络数据流量的变化而动态调整,但随之又引入了三个新的参数intertime、 $\alpha$ 、 $\beta$ ,造成ARED算法与RED算法一样存在着预置参数值选取的问题,不同参数值的选取会影响 $P_{max}$ 动态调整的效果。因此以下提出一种改进算



法,称为 QARED 算法,主要目的是希望通过优化丢弃概率  $P_b$  的计算方式来提高路由器缓冲区平均队列长度的稳定性,以减少算法中固定参数值设定所带来的影响<sup>[15]</sup>。

在式(2)中计算  $P_b$  所采用的是线性函数,而 QARED 算法提出采用平方函数来进行计算,如式(4)所示,两种计算函数的曲线如图 4 所示。

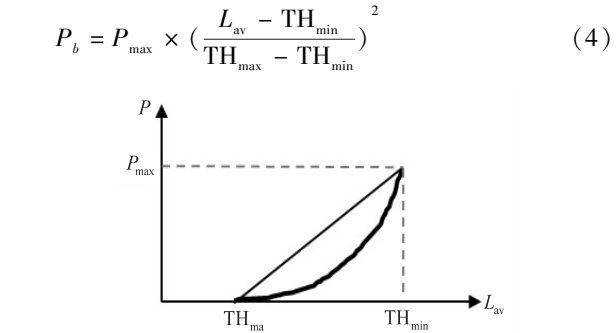


图 4 两种算法丢弃概率计算函数曲线

通过平方函数曲线可以看到,当缓冲区平均队列长度比较接近  $TH_{\min}$  时,丢弃概率  $P_b$  变化比较平缓;而当超过  $0.5 * (TH_{\min} + TH_{\max})$ ,丢弃概率  $P_b$  变化加剧。这种处理方式带来的好处是在缓冲区平均队列长度较小时降低丢包率,从而提高包转发效率;而接近  $TH_{\max}$  时加大丢包率,减少拥塞的可能,最终维持缓冲区平均队列长度在一个稳定值,保证了网络的吞吐量。

2.2 QARED 算法模拟实验及对比

在 NS2 网络仿真软件中,通过路径 ns-allinone-2.29\ns-2.29\queue 打开 ARED 算法实现 C++源程序 red.cc,其中 calculate\_p\_new()便是  $P_b$  的计算函数,对应的语句为:

```
p=v_a * v_ave + v_b;
p *= max_p;
将计算方式改为:
p=(v_a * v_ave+v_b) * (v_a * v_ave+v_b);
p=0.5 * max_p+0.5 * max_p * p;
然后重新编译 NS2:
cd ns-allinone-2.29
cd ns-2.29
make clean
make
```

模拟 QARED 算法运行时设置与 1.3 小节中相同的参数,跟踪获得路由器缓冲区平均队列长度变化如图 5 所示。通过与 ARED 算法进行对比可以发现,在改进算法中缓冲区平均队列长度振荡更趋于平缓,其长度值差不多维持在  $0.5 * (TH_{\min} + TH_{\max})$  附近,这进一步证实了其稳定性要优于 ARED 算法,从而更有效地保证路由器缓冲区队列有空间来处理突发数据流,不会造成数据包的大量丢失,提高了数据包转发的

效率。

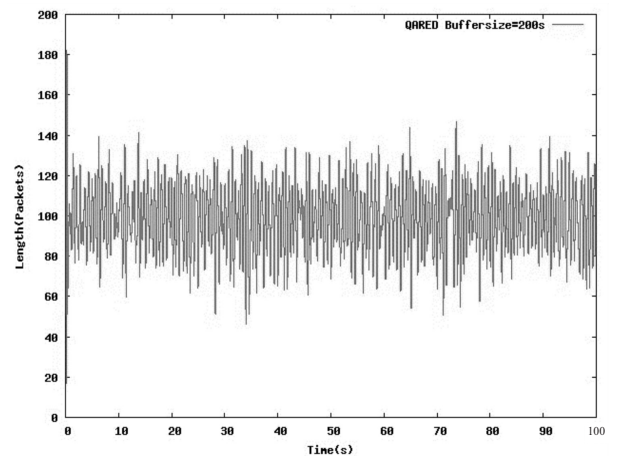


图 5 QARED 算法平均队列长度

2.3 三种算法的时延和丢包率对比

在 QARED 算法中,由于缓冲区平均队列长度更加平稳,从而达到了降低数据包丢失率和网络平均往返时延的目的。在模拟实验定量分析中,通过 NS2 系统中的数据流监控对象 Flowmon 和分类器对象 Classifier/Hash/Fid 来对各种算法下链路的平均往返时延、路由器到达数据包数量和被丢弃数据包数量分别进行跟踪并记录,相关程序如下:

```
set fmon [ $ns makeflowmon Fid ];
$ns attach-fmon $link_r1r2 $fmon;
set redq [ [ $ns link $r1 $r2 ] queue ];
set traceq [ open red-tra( $minthresh ). tr w ];
$redq trace curq;
$redq trace ave;
$redq attach $traceq;
然后利用绘图工具 plot 基于上面的跟踪数据绘制平均队列长度变化图。
plot 'red-cur(15).tr' using 2:3 title 'RED Buffersize = 48'
with lines
```

最后分别统计三种算法模拟实验中的平均往返时延、到达数据包数量、被丢弃数据包数量,以及计算丢包率,如表 1 所示。通过表中数据可以看出,QARED 算法是最优的,证明了这种改进思路的有效性。

表 1 三种算法数据对比

指标	RED	ARED	QARED
平均往返时延/s	0.019 818	0.019 221	0.014 592
被丢弃数据包数量	22 518	22 327	22 210
到达数据包数量	562 346	562 739	563 097
丢包率/%	4.54	3.97	3.94

3 结束语

RED 及其改进算法是更为有效的节点拥塞控制策略,通过模拟实验对比可以发现,当网络流量动态变

化时,RED 算法在控制平均队列长度稳定性上是成功的,因为当路由平均队列长度超过了最大阈值后就采用均匀随机丢弃数据包的策略,有效控制了平均队列长度,降低了平均时延,消除了对突发数据流的偏见;同时也在很大程度上缓解了 TCP 数据流的“全局同步”现象。由于 RED 算法在不降低吞吐量的情况下可以大大降低传输延时,因此其综合性能要优于传统的 DropTail 算法。

基于 RED 及 ARED 算法,文中提出了一种调整丢弃概率计算函数的思路,并利用网络模拟软件 NS2 模拟算法的运行并跟踪收集相关数据,通过三种算法的对比分析验证了改进算法 QARED 相对于 ARED 算法在控制路由器缓冲区平均队列长度稳定性上的优势,QARED 算法能够支持更低的网络时延和丢包率,提升网络动态变化下拥塞控制策略的稳定性。但不管是 ARED 还是 QARED 算法,在其运行过程中相关参数的预置仍然是基于常规的网络情况,如何更好地动态调整相关参数的设置来满足不同的网络情况是这类算法后续改进的研究点。

参考文献:

[1] AFANSYEV A,TILLEY N,REIHER P,et al. Host-to-host congestion control for TCP[J]. IEEE Communications Surveys & Tutorials,2010,12(3):304-342.

[2] 王志明,曾孝平,刘学,等. 一种异构网络 TCP 拥塞控制算法[J]. 电子与信息学报,2016,38(4):780-786.

[3] RASTOGI S,ZAHEER H. Comparative analysis of queuing mechanisms:Drop tail,RED and NLRED[J]. Social Network Analysis and Mining,2016,6(1):123-232.

[4] BRADEN B. Recommendations on queue management and

congestion avoidance in the Internet[S]. [s. l.]:[s. n.], 1998.

[5] 薛礼,陈利. RED 拥塞控制技术在路由器中的应用[J]. 软件导刊,2016,15(11):179-180.

[6] 茹新宇,刘渊. 一种基于正态分布函数的新 TCP 拥塞控制机制[J]. 计算机应用与软件,2017,34(8):245-250.

[7] NASSAR K A,ABDULLAH A A. Fuzzy RED to reduce packet loss in computer network[J]. Journal of Al-Qadisiyah for Computer Science and Mathematics,2016,8(1):107-114.

[8] QUE D,CHEN Z,CHEN B. An improvement algorithm based on RED and its performance analysis[C]//9th international conference on signal processing. Beijing: IEEE, 2008:2005-2008.

[9] 余学杰. 对 TCP/IP 计算机网络拥塞控制的研究[J]. 现代电子技术,2014,37(15):38-40.

[10] 薛礼,陈利. 改进 ARED 拥塞控制算法研究与实现[J]. 软件导刊,2017,16(11):41-43.

[11] GUAN L,AWAN I U,WOODWARD M E,et al. Discrete-time performance analysis of a congestion control mechanism based on RED under multi-class bursty and correlated traffic[J]. Journal of Systems and Software,2007,80(10):1716-1725.

[12] 张登银,张保峰. 新型网络模拟器 NS-3 研究[J]. 计算机技术与发展,2009,19(11):80-84.

[13] 雷宏江,徐鹏,徐勇军. 基于 NS2 仿真的 TCP 教学研究[J]. 科教文汇,2017(18):69-71.

[14] 徐雷鸣,庞博. NS 与网络模拟[M]. 北京:人民邮电出版社,2003.

[15] 饶刚,周井泉. 基于 ARED 的主动队列管理改进算法[J]. 计算机技术与发展,2014,24(5):27-30.

(上接第 103 页)

global lessons for defect prediction and effort estimation[J]. IEEE Transactions on Software Engineering,2013,39(6):822-834.

[37] JURECZKO M,MADEYSKI L. Towards identifying software project clusters with regard to defect prediction[C]//Proceedings of the 6th international conference on predictive models in software engineering. Romania: ACM, 2010:1-10.

[38] BETTENBURG N,NAGAPPAN M,HASSAN A. Think locally, act globally: improving defect and effort prediction models[C]//2012 9th IEEE working conference on mining software repositories (MSR). Zurich:IEEE,2012:60-69.

[39] RAHMAN F,POSNETT D,DEVANBU P. Recalling the "imprecision" of cross-project defect prediction[C]//Proceedings of the ACM SIGSOFT 20th international symposium on the foundations of software engineering. New York: ACM,2012:1-11.