第43卷 第6A期
2016年6月

计 算 机 科 学
Computer Science

Vol. 43 No. 6A
June 2016

卫星网络中一种改进的TCPW算法

于 冉 张 栋 邹启杰

(大连大学信息工程学院通信与网络重点实验室 大连 116622)

摘 要 针对卫星网络通信路径改变会引起往返时延剧烈变化,以及长延时环境会引起TCPW校准拥塞窗口精度下降的问题,提出了一种TCPW的改进方案——TCPW-CC。该算法减小了空间链路传播时延对算法性能的影响,利用星上拥塞系数δ作为调整拥塞窗口的依据,同时将窗口调整从每丢包后进行一次修改为每一个RTT进行一次,使得窗口的增长不再激进。仿真实验表明,所提改进方案提高了网络吞吐量,降低了丢包率。

关键词 卫星网络,拥塞控制,拥塞系数,TCPW

中图法分类号 TP393 文献标识码 A

Advanced TCPW Algorithm over Satellite Networks

YU Ran ZHANG Dong ZOU Qi-jie

(Key Laboratory of Communication Networks, College of Information Engineering, Dalian University, Dalian 116622, China)

Abstract Aimming at the dramatic change of round trip delay caused by high dynamic changes of satellite network's communication path and the alignment precision degradation when TCPW is used in long-delay network, a advanced TCPW algorithm named TCPW-CC was proposed, which uses the congestion coefficient of satellites as the adjusted basis of congestion windows. The new algorithm is able to avoid the influence of transmitting delay of space link. At the same time, the window adjustability is changed to be done after each RTT, which used to happen after dropping, thus resulting in that window grows less aggressive than the original. The results of simulation indicate that the TCPW-CC obviously improves the system throughput, decreases the rate of lost packets.

Keywords Satellite network, Congestion control, Congestion coefficient, TCPW

1 引言

拥塞控制算法[1]一般分为两类:基于窗口的和基于位率的调整方式。基于窗口的拥塞控制算法是通过源端限制数据的发送,根据目的端返回的ACK调整源端的发送速率;基于位率的拥塞控制算法要对发送的数据流进行整形,以防止数据流的爆发。地面互联网应用最广泛的传输控制协议TCP是一种基于窗口的端到端拥塞控制机制。

随着卫星组网技术的不断发展,卫星网络在通信领域中占据着越来越重的位置。由于卫星网络具有高误码率[2]、大延迟、拓扑结构高动态变化以及由其导致的传播时延动态变化和链路易中断等特点,使得原有TCP协议应用起来相对困难。目前,适用于卫星网络的拥塞控制算法[3]已成为世界各国的研究热点,如TCP SACK[4],TCP Hybla[5],TCP Vegas[6],TCP Peach[7],TCPW等[8]研究成果大量涌现。其中TCPW[9]是针对无线网络高误码率环境而设计的。该算法通过连续检测接收端返回的ACK速率得到单位时间发送的分组数目,然后与分组的大小相乘,再与时间间隔相除,就可获得发送端实际的传送带宽,并通过低通滤波器平滑样本序列以得到更加精确的值。该估计算法在拥塞发生或丢包后利用带宽估计值设置ssthresh和cwnd,避免了在发生丢包后原有TCP协议盲目地将发送速率减半,从而提高了链路利用率。

卫星网络是一种具有高误码率的无线网络,如何在卫星网络中更有效地使用TCPW算法成为近年来的研究热点。文献[10]提出了一种基于选择性确认丢包恢复的改进算法TCPW-SACK,该算法通过SACK机制进一步提高了网络利用率和吞吐量,同时仍具有很好的公平性和友好性;文献[11]提出了一种TCP增强算法,该算法通过在慢启动阶段尽早确定合适的门限值,在拥塞避免阶段延长从拥塞避免开始到再次发生拥塞的时间,提高了网络利用率;文献[12]提出了一种改进算法TCPW-S,该方案在每次带宽计算后,确定当前拥塞窗口是否适宜,一旦当前窗口大于估计带宽,就迅速降低窗口,不仅提高了TCPW的吞吐量,而且使丢包数大幅度下降。但是上述算法都没有考虑到卫星节点的移动对往返时延的影响。

本文在TCPW的基础上提出了一种基于拥塞系数的拥塞控制算法TCPW-CC(TCPW Based on Congestion Coefficient)。该算法首先在ACK的时间间隔中通过计算传播时延的方法去掉了星间距离变化产生的影响。进一步引入了拥塞系数,在慢启动阶段和拥塞避免阶段,根据拥塞系数和门限值的比较结果来确定窗口变化值,使拥塞即将发生时,窗口能及时下降到适宜的水平。仿真实验表明,TCPW-CC有效提高了卫星网络的吞吐量。

本文受国家自然科学基金(61301151,91338104)资助。

于 冉(1981—),男,博士,讲师,CCF会员,主要研究方向为计算机网络;张 栋(1991—),男,硕士生,主要研究方向为卫星网络拥塞控制技术,E-mail:1216707274@qq.com;邹启杰(1978—),女,博士,副教授,主要研究方向为智能控制、卫星通信与网络技术。

2 TCPW 算法简介

TCPW 算法通过在发送端持续不断地检测确认包 ACK 的到达速率进行带宽估计。带宽的动态估算公式如下：

$$\begin{cases} B_k = \alpha_k B_{k-1} + (1-\alpha_k)\left(\dfrac{b_k + b_{k-1}}{2}\right) \\ \alpha_k = \dfrac{2\tau - \Delta t_k}{2\tau + \Delta t_k} \\ b_k = \dfrac{d_k}{t_k - t_{k-1}} \end{cases} \tag{1}$$

其中，B_k 是 t_k 时刻的带宽估计值，α_k 是过滤权值，d_k 是第 k 个 ACK 所响应的数据量，t_k 是第 k 个 ACK 的到达时间，$\Delta t_k = t_k - t_{k-1}$，$1/\tau$ 是滤波器的截止频率。

当检测到分组丢失时，根据带宽估计值调整拥塞窗口和慢启动阈值。具体来说：

当接收端检测到 3 个重复 ACK 时，

$$\begin{cases} W_{ssthresh} = (B \times T_{rtt-min})/L_{seg} \\ W_{cwnd} = \min(W_{cwnd}, W_{ssthresh}) \end{cases} \tag{2}$$

当发生超时重传时，

$$\begin{cases} W_{ssthresh} = (B \times T_{rtt-min})/L_{seg} \\ W_{cwnd} = 1 \end{cases} \tag{3}$$

其中，$W_{ssthresh}$ 为慢启动阈值，W_{cwnd} 为拥塞窗口的大小，L_{seg} 为 TCP 报文段的比特大小，$T_{rtt-min}$ 为网络负载最小时探测到的往返时延。

3 卫星网络中 TCPW 算法的改进

3.1 链路传播时延的计算

传播时延随星间链路长度的变化而变化，星间链路长度的计算任务由信源端完成。首先，将卫星星座的星历表下载到信源端上，星历表按卫星星座轨道周期（取决于具体的星座）进行定时更新；然后，在计算星间通信链路长度时，从星历表中读取通信路径上的卫星信息。需从 TCP 首部信息的选项字段中明确获知通信路径中所经过的卫星，选项字段记录了所经过的卫星编号信息。信源端可以从返回的 ACK 中得到通信路径中所经过的卫星，由此可以知道在动态路由中每个数据包所经过的通信路径。

卫星间距离关系如图 1 所示。

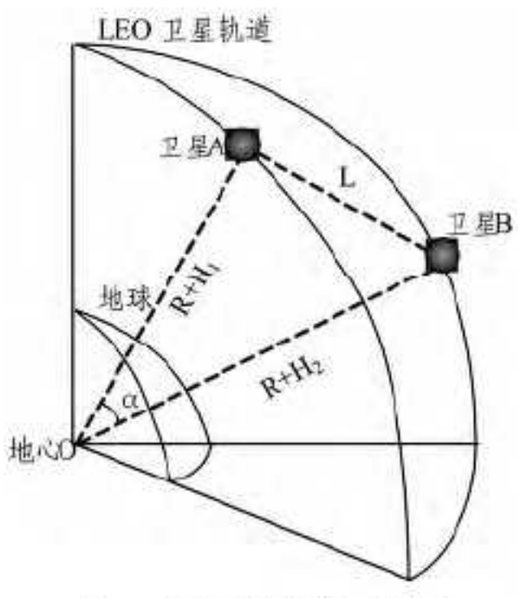

图 1 卫星间距离示意图

利用卫星间的瞬时地心角与卫星的轨道高度计算卫星之间的距离，而瞬时地心角可由卫星星下点的经纬度计算得出。

瞬时地心角的计算公式为：

$$\alpha = \arccos[\sin\varphi_1 \sin\varphi_2 + \cos\varphi_1 \cos\varphi_2 \cos(\lambda_1 - \lambda_2)] \tag{4}$$

式中，(λ_1, φ_1) 和 (λ_2, φ_2) 分别为卫星 A 和卫星 B 的星下点经纬度。

星间链路长度由卫星间的瞬时地心角与卫星的轨道高度计算得出，计算公式如下：

$$l = \sqrt{(R+H_1)^2 + (R+H_2)^2 - 2(R+H_1)(R+H_2)\cos\alpha} \tag{5}$$

其中，R 为地球半径，H_1 为卫星 A 的轨道高度，H_2 为卫星 B 的轨道高度。

链路的总长度为卫星两两之间的距离之和，即 $D = l_1 + l_2 + \cdots + l_n$，式中 $l_1, l_2, \cdots, l_n$ 为链路上相邻卫星间的链路长度。

根据星间通信链路长度计算当前时刻传播时延，计算公式如下：

$$T_{PD} = 2 \times D/c \tag{6}$$

其中，T_{PD} 为传播时延，c 为光速，D 为当前时刻通信链路的长度。

3.2 ACK 间隔时间的重新计算

在卫星网络中，式(1)的 Δt_k 值里面包含着链路状态改变的信息。卫星之间距离的变化以及星际路由的改变都会导致 Δt_k 的值发生动态变化。在由低、高轨卫星组成的简单网络拓扑中，星间的位置是不固定地，这种影响会明显地投射到 Δt_k 的值中，即使在卫星相对固定的卫星星座（如铱星星座）中，也会由于卫星的运动而存在路由的变化，而这些最终都将归结于星间距离的改变。当卫星之间距离变远时，尽管网络的拥塞状况没有变化，但计算出来的带宽会降低，而此时，进行拥塞控制是一种不合理的做法。因此需要对 Δt_k 进行重新计算，重新计算所得的 ACK 间隔时间记为：

$$\Delta t_k' = \Delta t_k - (T_{PD(k)} - T_{PD(k-1)}) \tag{7}$$

其中，$T_{PD(k)}$ 为 t_k 时刻的传播时延，$T_{PD(k-1)}$ 为 t_{k-1} 时刻的传播时延。

3.3 最小往返时延的计算

根据传播时延计算非传播时延，计算公式如下：

$$T_{NPD} = RTT - T_{PD} \tag{8}$$

其中，RTT 为往返时延，T_{NPD} 为非传播时延。T_{NPD} 仅由发送时延和排队时延组成，发送时延基本不变，T_{NPD} 的变化主要由排队时延引起，可以较准确地反映卫星网络当前的拥塞状况。每经过一个往返时延 RTT，就可以计算得到一个对应的 T_{NPD}，取 T_{NPD} 的历史最小值作为 T_{NPDmin}，T_{NPDmin} 的值保持更新，一旦出现了更小的，就用新值代替旧值，而卫星网络最小往返时延 $RTT_{\min}$ 不再是网络最小的一个往返时延，而是网络中非传播时延最小的往返时延，公式如下：

$$RTT_{\min} = T_{NPDmin} + T_{PD} \tag{9}$$

由于在 $RTT_{\min}$ 的计算过程中考虑了距离变化的问题，使得 $RTT_{\min}$ 可以准确地反映卫星网络的排队时延变化的状况。

3.4 拥塞系数的计算

考虑到一次通信过程中所历经卫星的星上最大处理时延可以反映整个网络的拥塞情况，TCPW-CC 算法以此为基础来构建拥塞系数。

为了计算星上最大处理时延，需要获得每颗卫星的星上处理时间，这就需要对时间戳选项的功能做适当的修改，即时间戳不仅记录报文段的发送和接收时间，还需记录该报文段每一次经卫星转发时的入队列时间（*in-queue-time*）和出队列时间（*out-queue-time*）。同时，在选项部分增加 4 字节的处理

时间记录字段 $processing\_time$，记录最大处理时间 $max\_processing\_time$。当该报文段被转发至另一颗卫星时，由当前卫星计算出上一颗卫星的星上处理时间并由处理时间记录字段进行记录。

$$processing\_time = out\_queue\_time - in\_queue\_time \tag{10}$$

与此同时，时间戳选项记录下该卫星的入/出队列时间，在报文段下一次转发时再次计算星上处理时间，若该处理时间大于上一颗卫星的处理时间，则处理时间记录字段更新为当前处理时间，否则保持不变。当报文段最终被转发至目的终端时，由接收端计算出最后一颗卫星的处理时间并与处理时间记录字段的记录值进行比较。这样就可以保证当数据报文段从发送端成功发送至接收端整个过程中处理时间记录字段始终记录的是最大星上处理时间，该处理时间随接收端的返回 ACK 发送至发送端，发送端依据网络中卫星的 $max\_processing\_time$ 判断网络当前拥塞状况。

考虑卫星的星上处理时间主要是报文段的排队时间，可以利用当前报文段的排队时间与卫星最大排队时间的比值作为拥塞系数，记为 δ。

假设星上缓存大小为 sat_buffer，卫星转发速率为 v_sat，则星上的最大排队时延为

$$max\_queue\_time = \frac{sat\_buffer}{v\_sat} \tag{11}$$

一个报文段从发送端传输至接收端的过程中测得的最大处理时延为 $max\_processing\_time$，则拥塞系数为

$$\delta = \frac{max\_processing\_time}{max\_queue\_time} \tag{12}$$

3.5 窗口增长策略的调整

TCPW 依靠分组丢失来调整窗口的策略容易造成资源的浪费，从而导致网络性能的下降。考虑到 TCPW 的慢启动和拥塞避免算法仍沿用 TCP-Reno 算法，较为激进，本文将增加拥塞状态门限值，并与拥塞系数相比较，从而得到更为保守的窗口调整机制。

一般认为[13]，当队列长度超过最大队列长度的 2/3 时，网络负载较重，发生拥塞的概率较大；当队列长度小于最大队列长度的 1/3 时，网络负载较轻，发生拥塞的概率较小。引进上述拥塞系数的概念，当拥塞系数大于 2/3 时，认为发生拥塞概率较大；当它小于 1/3 时，认为发生拥塞概率较小；否则，网络处于正常工作状态。因此，取门限值 $\alpha = 1/3, \beta = 2/3$，改进算法中，拥塞窗口的调整策略如下。

在慢启动阶段：

$$cwnd(t+1) = \begin{cases} cwnd(t)+1, & \delta \leqslant 1/3 \\ cwnd(t), & 1/3 < \delta < 2/3 \\ cwnd(t)-1, & \delta \geqslant 2/3 \end{cases} \tag{13}$$

在拥塞避免阶段：

$$cwnd(t+1) = \begin{cases} cwnd(t)+1/cwnd(t), & \delta \leqslant 1/3 \\ cwnd(t), & 1/3 < \delta < 2/3 \\ cwnd(t)-1/cwnd(t), & \delta \geqslant 2/3 \end{cases} \tag{14}$$

当发生快重传和超时重传时，依然采用式(2)、式(3)对慢启动门限值和窗口值进行调整。

3.6 TCPW-CC 算法的流程

TCPW-CC 算法的流程图如图 2 所示。

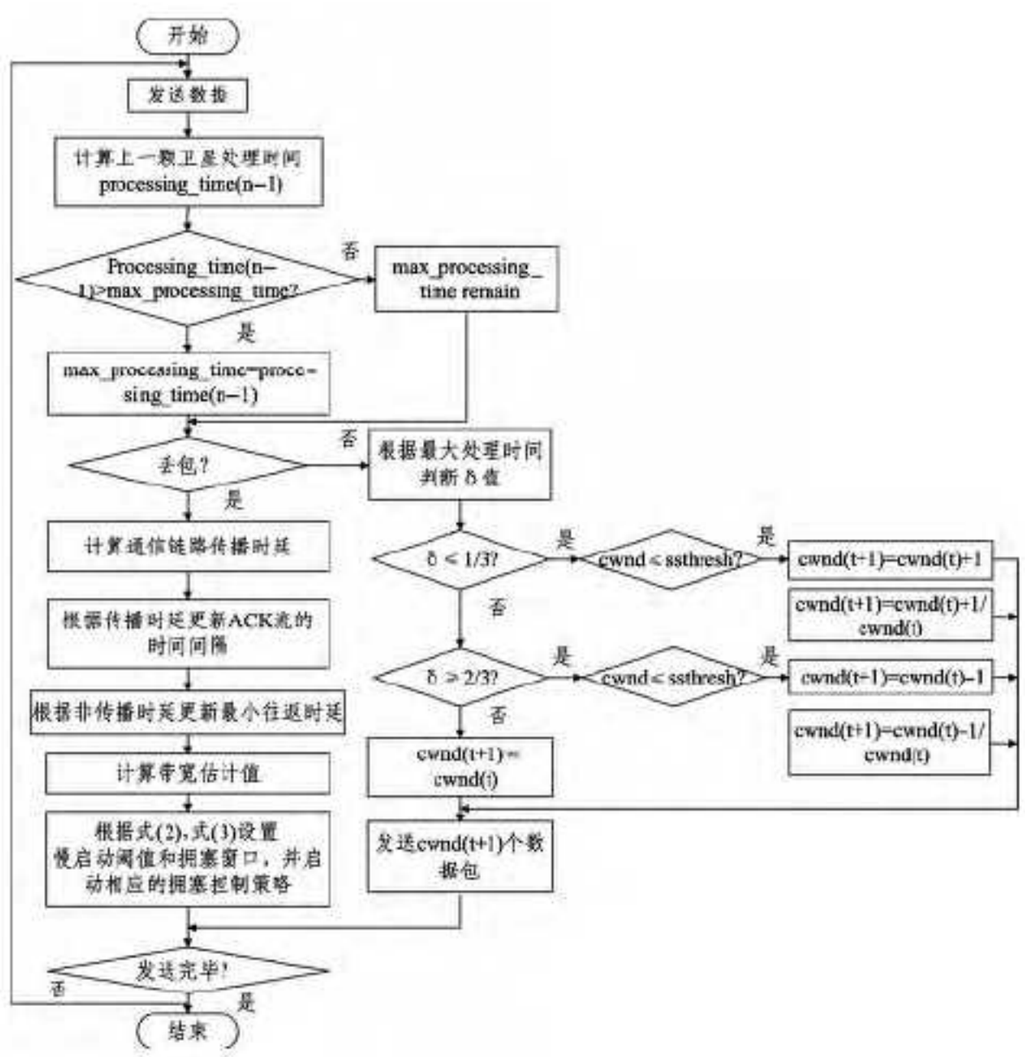

图 2 TCPW-CC 算法的基本流程

4 算法性能仿真

为了验证新算法的性能，本文采用了以下两种仿真软件：①卫星仿真软件 STK，主要用来仿真卫星网络拓扑结构，版本为 8.1.1；②网络仿真软件 NS2[14,15]，主要用来仿真算法性能，版本为 2.34。仿真运行环境为 Windows XP。

本文仿真实验采用的网络拓扑结构如图 3 所示。

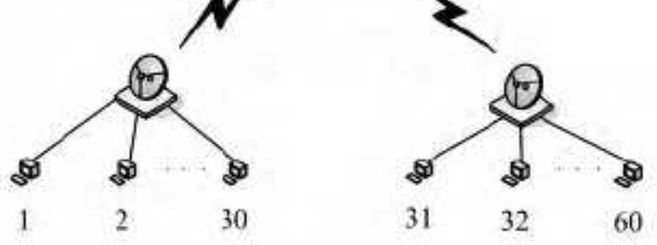

图 3 仿真拓扑结构图

其中 LEO 卫星通信网采用具有全球覆盖能力的 Iridium 极轨道卫星星座，具体参数设置如表 1 所列。地面站采取 NS2 的默认配置，选取伯克利(37.9，−122.3)和波士顿(42.3，−71.1)作为两个地面站。每一个地面站与 30 个终端节点相连，与伯克利相连的是源端节点，与波士顿相连的是终端节点，共有 30 条链接，每条链接都经过 Iridium 卫星网络。所有源端向目的端发送数据。主要网络参数中卫星缓存为 50packets，星上转发速率为 10Mbit/s。

表 1 Iridium 卫星星座参数

参数	参数取值
轨道高度	780 km
轨道数	6
每个轨道平面的卫星数	11
轨道倾斜度	86.4°
轨道平面间隔	31.6°
缝间隔	22°
允许最小覆盖的仰角	8.2°
每个卫星的星际链路数	4
星际链路带宽	25 Mbps
上下行带宽	1.5 Mbps
星际链路纬度阈值	60°

使用上述 Iridium 卫星网络拓扑结构和仿真参数配置，分别对传统 TCP Reno 算法、TCPW 算法和基于拥塞系数的 TCPW 算法进行了仿真，得到如图 4—图 8 所示的仿真结果。

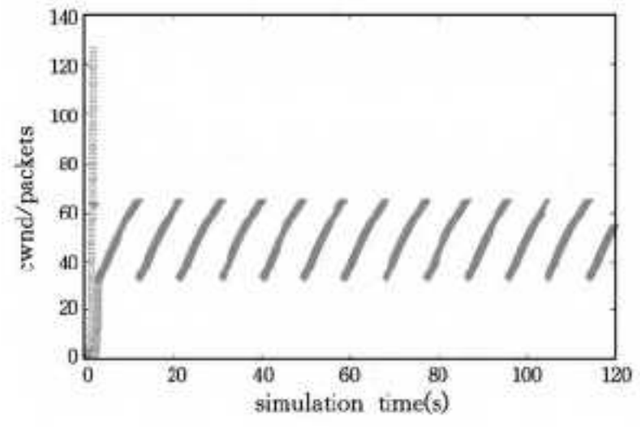

图 4 Reno 算法的拥塞窗口

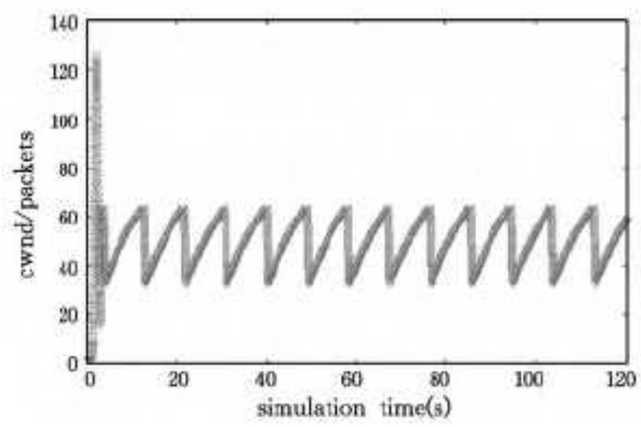

图 5 TCPW 算法的拥塞窗口

由图 4 和图 5 可以看出，Reno 算法和 TCPW 算法在仿真不久后都呈现出一个周期性的变化。在这个变化过程中，两种算法所得到的拥塞窗口的最值和变化趋势几乎相同，不同的是在每一个丢包发生的时候，Reno 算法的拥塞窗口存在一个突然的降低，它的变化是不连续的，而 TCPW 算法的拥塞窗口总是缓慢地降低，它的变化是连续的。之所以出现这种现象，是由于 Reno 算法在每次丢包后执行了快恢复算法，将拥塞窗口降到和慢启动门限值相同，然后执行拥塞避免算法，而 TCPW 算法引入了带宽估计，在每次丢包后，不是盲目降低窗口，而是把窗口恢复到和带宽相适应的水平，于是在 Reno 算法间断的地方，TCPW 算法是连续的。因此 TCPW 算法在整个过程中维持了一个较高的拥塞窗口水平，最终导致吞吐量的上升。

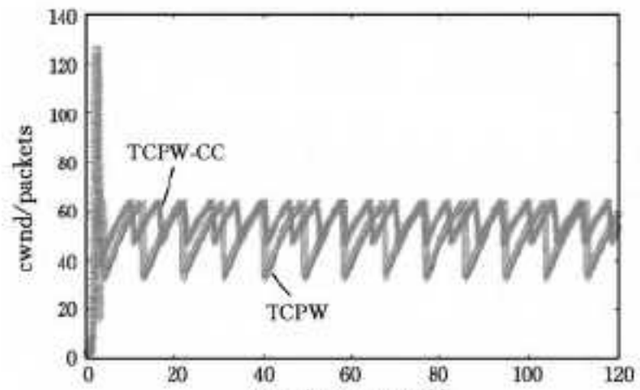

图 6 TCPW 算法与 TCPW-CC 算法的拥塞窗口比较

由图 6 可知，TCPW 算法和 TCPW-CC 算法在每次丢包后，拥塞窗口都会缓慢降低，呈现一个连续的变化，这是由于两种算法都内含了带宽估计机制。但是 TCPW 算法的窗口调整策略仍比较激进，作为改进，TCPW-CC 算法利用本文引入的拥塞系数来合理调整窗口，使得算法在接近拥塞的时候表现出一定的优势，从图 6 可以看出，TCPW-CC 在整个过程中维持一个更高的拥塞窗口水平，这会导致它有一个更高的吞吐量。

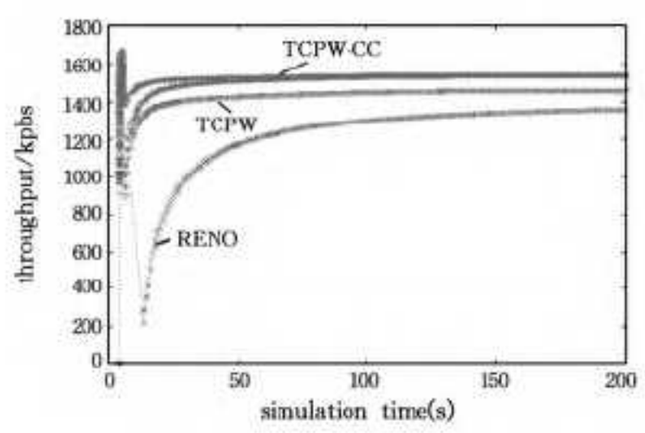

图 7 3 种算法的吞吐量比较

由图 7 可知，Reno 算法由于受拥塞窗口减半的影响，波动较大，而 TCPW 算法和 TCPW-CC 算法的吞吐量波动较小，体现了这两种算法在系统稳定性上的优越性。从图中还可以看出，相比 TCPW 算法，TCPW-CC 算法的吞吐量性能平均提高了 5.6%。

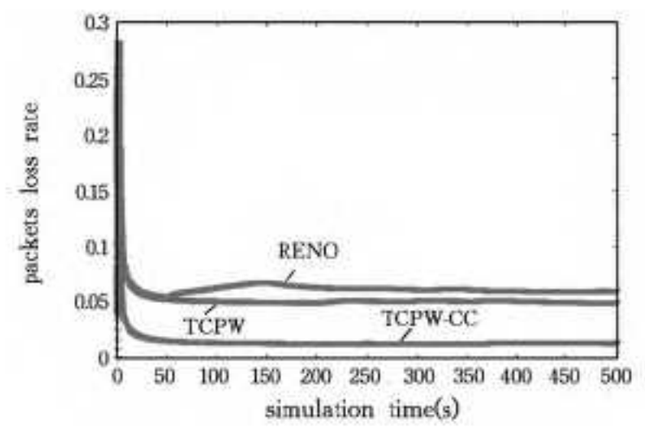

图 8 3 种算法的丢包率比较

由图 8 可看出，TCPW 算法的丢包率要低于 Reno 算法，而 TCPW-CC 算法的丢包率要明显低于 TCPW 算法。这说明 TCPW-CC 算法的性能要优于 TCPW，而这与图 7 得出的吞吐量性能的结论正好是相对应的。

结束语 与 TCPW 算法相比，TCPW-CC 算法有效降低了卫星网络高动态变化的拓扑结构对算法性能的影响，利用拥塞系数作为拥塞窗口调整的依据更加精准地反映了当前网络状态变化的情况，同时，在计算新窗口的时候，采取了一种更加保守的窗口增长方式，让窗口的增长更加柔和。仿真实验证明，相比 TCPW 算法，TCPW-CC 提升了系统的吞吐量，并大幅度地降低了丢包率，节约了宝贵的星上资源。

参 考 文 献

[1] 刘秋让，倪红波. TCP 拥塞控制解决方法分析及评价[J]. 计算机工程，2001，27(6)：65-66

[2] 王平，顾学迈. LEO 卫星网络中 TCP 协议性能及路由策略研究[J]. 南京理工大学学报，2007，31(1)：87-91

[3] 刘光华，王辉. LEO 卫星网络中 TCP 协议性能研究[J]. 计算机工程，2010，36(14)：96-98

[4] 罗大军. 在 Ad Hoc 网络中 TCP-SACK 性能研究及改进[J]. 无线互联科技，2012(12)：85-86

[5] Caini C, Firrincieli R. TCP Hybla: a TCP enhancement for heterogeneous networks[J]. International Journal of Satellite Communications and Networking, 2004, 22(5): 547-566

[6] Brakmo L S, Perterson L L. TCP Vegas: end-to-end congestion avoidance on a global Internet[J]. IEEE Journal on Selected Areas in Communication, 1995, 13(8): 1465-1480

[7] 王斌，陈元琰，冯伟，等. TCP Vegas-b: TCP Vegas 改进算法[J]. 计算机工程与设计，2011，32(2)：438-441

[8] Akyildiz I F, Morabito G, Palazzo S. TCP- peach: a new congestion control scheme for satellite IP Networks[J]. IEEE/ACM Transactions on Networking, 2001, 9(3): 307-321

[9] Mascolo S, Casetti C. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links[C]// Proc. of MOBICOM. 2001: 287-297

[10] 赵东平, 郑卫斌. 高误码率无线环境下的 TCP 改进算法[J]. 计算机工程, 2006, 32(9): 96-98

[11] 彭华, 邓亚平. 基于 TCP-Westwood 的一种 TCP 增强算法[J]. 计算机应用, 2006, 26(S1): 221-223

[12] 黄蕾, 刘立祥. TCP-Westwood 针对卫星网的改进方案[J]. 计算机工程, 2007, 33(8): 103-105

[13] 戴帅, 肖楠, 梁俊, 等. 基于处理时延的卫星网络 TCP 拥塞控制算法[J]. 现代防御技术, 2014, 42(3): 127-134

[14] Gurer G, Alparslan O, Karasan E, et al. nOBS: an ns2 based simulation tool for performance evaluation of TCP traffic in OBS networks[J]. Annals of Telecommunications, 2007, 62(5/6): 618-637

[15] Shivkumar S, Umamaheswari G. Certificate Authority Schemes Using Elliptic Curve Cryptography, Rsa and Their Variants-Simulation Using NS2[J]. American Journal of Applied Sciences, 2014, 11(2): 171-179

(上接第 267 页)

考虑信道环境发生突变的情况，此时混合矩阵发生变化，算法需要重新迭代收敛。将初始分离矩阵优化应用到信道突变的场景中，当检测到信道环境发生突变时，进行初始分离矩阵优化，使算法获得较好的初始值，可有效加快算法的再次收敛。假设混合矩阵 A 在 16001 点处发生变化，突变为 $[3\ -0.5;0.6\ \ 2]$。图 4 所示为相邻时刻分离矩阵之差的 Frobenius 范数，由图可知在突变点处，相邻分离矩阵之差的 Frobenius 范数发生突变，可作为信道环境变化的检测条件，并设检测阈值为 0.15。

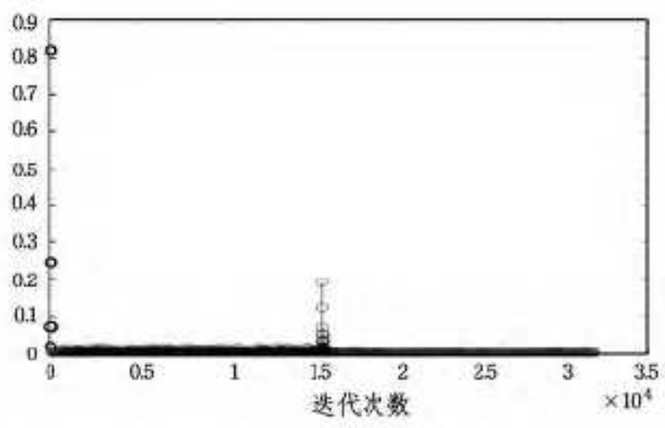

图 4 相邻时刻分离矩阵之差的 Frobenius 范数曲线

图 5 所示为信道环境突变时，初始分离矩阵优化和未优化的在线动量项 EASI 算法的性能曲线。由图可知，信道环境变化时，运用初始分离矩阵优化能够有效改善 PI 性能指数和收敛速度。

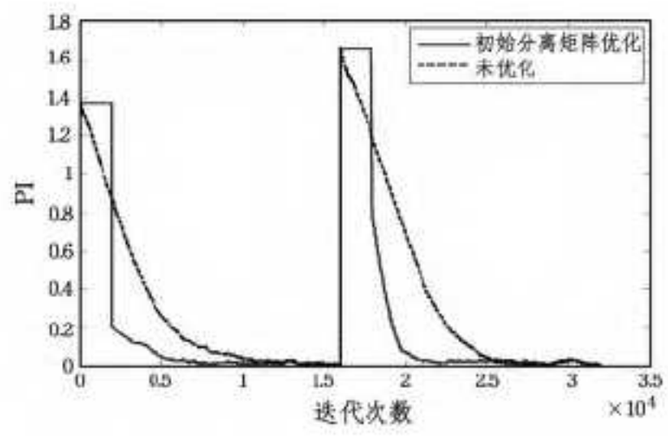

图 5 初始分离矩阵优化和未优化的性能曲线(时变条件)

结束语 针对在线盲源分离算法的收敛速度受分离矩阵初始值影响的问题，本文提出了一种基于人工蜂群算法的初始分离矩阵优化的在线盲源分离算法，即通过运用人工蜂群算法较强的搜索寻优能力，在分离的初始阶段对分离矩阵进行寻优，从而获得较好的初始迭代值，可有效提高盲源分离算法的收敛速度。仿真结果验证了初始分离矩阵优化算法的有效性。对于因信道环境发生变化导致的混合矩阵时变的情形，采用本文提出的算法同样可以提高时变后的盲源分离算法的收敛速度。本文所提算法，初始分离矩阵的优化属于批处理过程，之后为在线迭代过程，属于批处理算法与在线算法的结合，使得算法既具有在线算法的实时性，又可以获得整体分离性能的改善。

参 考 文 献

[1] Jutten C, Herault J. Blind Separation of Sources, Part I: An Adaptive Algorithm based on Neuromimetic [J]. Signal Processing, 1991, 24(1): 1-10

[2] Hu Jing, Fan Le-hao. Application of JADE to separate complex-valued sources[C]// Computer Science and Service System (CSSS). Nanjing, IEEE, 2011: 1127-1129

[3] Hyvarinen A, Oja E A. Fast Fixed-Point Algorithm for Independent Component Analysis [J]. Neural Computation, 1997, 9(7): 1483-1492

[4] 李良敏. 基于遗传算法的盲源分离算法[J]. 西安交通大学学报, 2005, 39(7): 740-743

[5] 张朝柱, 张健沛, 孙晓东. 基于自适应粒子群优化的盲源分离[J]. 系统工程与电子技术, 2009, 31(6): 1275-1278

[6] 张银雪, 田学民, 邓晓刚. 基于改进人工蜂群算法的盲源分离方法[J]. 电子学报, 2012, 40(10): 2026-2030

[7] Tang Y, Li J P. Normalized natural gradient in independent component analysis [J]. Signal Process, 2010, 90(9): 2773-2777

[8] Cardoso J F, Laheld B H. Equivariant adaptive source separation [J]. IEEE Trans. on Signal Processing, 1996, 44(12): 3017-3030

[9] Chambers J A, Jafari M G, Melaughlin S. Variable Step-size EASI Algorithm for Sequential Blind Source Separation[J]. Electronics Letters, 2004, 40: 393-394

[10] Yuan L, Wang W, Chambers J A. Variable Step-size Sign Natural Gradient Algorithm for Sequential Blind Source Separation [J]. IEEE Signal Processing Letters, 2005, 12(8): 589-592

[11] 欧世峰, 高颖, 赵晓辉. 自适应组合型盲源分离算法及其优化方案[J]. 电子与信息学报, 2011, 33(5): 1243-1247

[12] 郑辉. 通信中盲信号处理理论与技术[M]. 北京: 国防工业出版社, 2013: 437-439

[13] Nordhausen K, Ollila E, Oja H. On the performance indices of ICA and blind source separation[C]// IEEE Int. Workshop on Signal Processing Advances in Wireless Communications. Piscataway, 2011: 461-465