Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

CW-HSTCP: Fair TCP in high-speed networks

PAN Xue-zeng (潘雪增)[1], SU Fan-jun (苏凡军)[†1], LÜ Yong (吕 勇)[2], PING Ling-di (平玲娣)[1]

([1]*School of Computer Science, Zhejiang University, Hangzhou 310027, China*)

([2]*School of Electrical Engineering, Zhejiang University, Hangzhou 310027, China*)

[†]E-mail: suwang@zju.edu.cn; sufanjun@sina.com

Received Oct. 11, 2005; revision accepted Dec. 28, 2005

Abstract: The congestion control mechanisms of the current standard TCP constrain the congestion windows that can be achieved by TCP in high-speed networks, which leads to low link utilization. HSTCP is one solution to solve this problem by modifying the congestion control mechanism to have the characteristics of TCP friendliness in high loss rate environment and high scalability in low loss rate environment. However, experiments revealed that HSTCP has severe RTT unfairness. After analyzing the RTT unfairness in HSTCP with a model, we proposed CW-HSTCP, which added a fair factor to decrease the difference of congestion window caused by different RTT. Fair factor of long RTT flows can cause a sharp window increment that is easy to cause a bursty traffic, so a method called block-pacing was adopted. Simulation results showed that our new proposal could alleviate the RTT unfairness while keeping advantages of HSTCP.

Key words: High-speed networks, HSTCP, Congestion control
doi:10.1631/jzus.2006.A0172 **Document code:** A **CLC number:** TP393

INTRODUCTION

The development of network technology led to the appearance of many high-speed networks with bandwidth larger than 1 Gbps, or even 10 Gbps. Through high-speed networks, applications like scientific collaboration, telemedicine, and real-time environment monitoring can transfer high-bandwidth real time data, images, and video captured from remote sensors such as satellite, radars, and echocardiography. What is more, data intensive grid application (Foster *et al*., 2001) and SAN (Phillips, 1998) network can benefit from high-speed networks too.

TCP widely adopted as a data transfer protocol in Internet now works well when transfer rates are in the range of 100 bps to 10^7 bps and round-trip delays are in the range of 1 ms to 100 s (Jacobson *et al*., 1992), but it performs badly in high-speed networks. TCP increases its congestion window by one at every round trip time (RTT) and reduces it by half at a loss event. In order for TCP to increase its window for full utilization of 10 Gbps with 1500 byte packets, it requires over 83333 RTTs. With 100 ms RTT, it takes approximately 1.5 h, and for full utilization in steady state, the loss rate cannot be more than 1 loss event per 5×10^9 packets which is less than the theoretical limit of the network's bit error rates (Floyd, 2003).

Some efforts have been made to improve the performance of TCP. The proposal in (Semke *et al.*, 1998) gets better TCP performance by auto-tuning the buffer limit of the sender and receiver, although limited effects can be achieved in high-speed networks. The proposal in (Sivakumar *et al.*, 2000) uses parallel TCP to transfer bulk data, but the number of the connections is set by users, greedy action may result. XCP (Katabi *et al*., 2002) is a router-assisted protocol, which limits its scalability. Currently, the research focus is modifying congestion control mechanisms of TCP. Some recently proposed protocols are High-Speed TCP (HSTCP) (Floyd, 2003), Scalable TCP (STCP) (Kelly, 2003), BIC (Xu *et al*., 2004), LTCP (Bhandarkar *et al.*, 2004). HSTCP is a modification of TCP's current congestion control mechanisms for high-speed links. HSTCP's modification of the re-

sponse function is realized by change of additive increase and multiplicative decrease parameters. Therefore, HSTCP has high scalability in high-speed networks. STCP has similar idea. BIC views congestion control as a searching problem in which the system gives yes/no feedback through packet loss. It consists of two parts: binary search increase and additive increase. LTCP uses two dimensional congestion control: at the macroscopic level, the layers are added/dropped based on dynamic network conditions and at the microscopic level, the congestion window behavior is defined for operating at any given layer.

However, in (Xu *et al*., 2004), the author points out that HSTCP has more severe RTT unfairness. We define the RTT unfairness of two competing flows to be the throughput ratio, which equals to the ratio of bottleneck link utilization. Lakshman and Madho (1997) studied RTT unfairness of standard TCP. It is reported that under a DT (Drop Tail) queue TCP throughput is inversely proportional to RTT^{α}, where $1\leq\alpha\leq2$.

In this paper, we analyze the RTT unfairness with a model proposed in (Chiu and Jain, 1989). To resolve the RTT unfairness, a fair protocol named CW-HSTCP is proposed, and we give a relative fair criterion to evaluate the algorithm in Section V. CW-HSTCP adds a fair factor to eliminate the difference of congestion window between flows with different RTT. Block-pacing scheme is adopted to avoid bursty traffics caused by fair factor of long RTT flows.

Our proposal can greatly alleviate the severe RTT unfairness, and keep the RTT unfairness below the inverse ratio of RTT. With ns-2 simulator, we show CW-HSTCP can get a stable fairness between different RTT flows, and have friendliness to standard TCP. What is more, CW-HSTCP only needs small modification of the HSTCP algorithm. Compared with BIC, our proposal is simple and keeps the advantages of HSTCP.

BACKGROUND: TCP AND HSTCP

Because our proposal builds on HSTCP, in this section, we briefly introduce TCP and HSTCP (Floyd, 2003).

TCP uses a sliding window mechanism and end-to-end acknowledgments to provide reliable data transfer across a network (Stevens, 1994). Congestion window (cwnd for short) represents the size of sliding window used by the sender. In congestion avoidance phase, TCP and HSTCP use the following algorithm to adjust its congestion window:

In response to a new acknowledgement (ACK):

$$w\leftarrow w+a(w)/w. \tag{1}$$

In response to a congestion event:

$$w\leftarrow w-b(w)\times w, \tag{2}$$

where w denotes congestion window size. The above congestion control mechanisms are also called AIMD (additive increase and multiplicative decrease), where $a(w)$ and $b(w)$ are additive increase and multiplicative decrease parameters respectively.

Therefore, RTT has a significant influence on TCP performance. We can deduce from Eq.(1) that:

$$w(t+1)=w(t)+a(w)\times\Delta t/RTT. \tag{3}$$

This means that connection increases its sending rate by $a(w)$ packets per round-trip time. For standard TCP, $a(w)=1$, $b(w)=0.5$, which is not sufficient for high-speed networks, so HSTCP makes $a(w)$ and $b(w)$ become the function of current congestion window size.

In HSTCP, some parameters such as High_Window, Low_Window, High_P, Low_P are defined. We express them as H_W, L_W, H_P, L_P, where H_W and L_W are the congestion window, and H_P, L_P are the corresponding drop rate. When $w\leq L\_W$, HSTCP uses the same values of $a(w)$ and $b(w)$ as standard TCP. While when $w>L\_W$, they will be calculated as follows:

$$b(w)=(b(H\_W)-0.5)(\log w-\log(L\_W))/(\log(H\_W)-\log(L\_W))+0.5,$$
$$a(w)=w^2\times p(w)\times2\times b(w)/(2-b(w)).$$

Then the value of $a(w)$ and $b(w)$ can be calculated from the default value $H\_W=83000$, $L\_W=38$, $H\_P=10^{-7}$, $L\_P=10^{-3}$ and $b(H\_W)=0.1$. We show some of them in Table 1. STCP (Kelly, 2003) has similar idea of setting $a(w)=0.01w$, $b(w)=0.125$.

HSTCP and STCP are realized in sender without needing modifications of network devices, so they are easy to implement in current TCP stacks.

Table 1 The value of $a(w)$ and $b(w)$ in HSTCP

w	$a(w)$	$b(w)$
38	1	0.50
118	2	0.44
221	3	0.41
347	4	0.38
...	...	...
84035	71	0.10
...	...	...

SIMULATION TOPOLOGY AND RTT UNFAIRNESS ANALYSIS

Simulation topology and configuration

We adopted ns-2 simulator (version 2.26). The topology and configuration are shown in Fig.1. The buffer size of the router is set to the product of bandwidth and the delay of the bottleneck link.

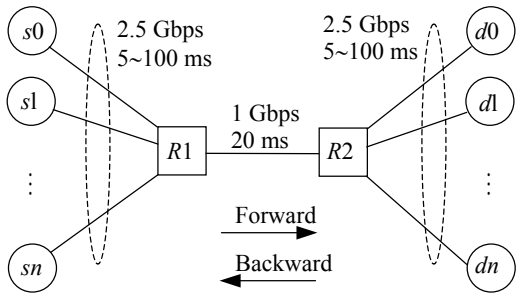

Fig.1 Simulation topology and configuration

We use TCP SACK for the simulation, and packet size is set to 1000 bytes. The maximal congestion window is set to 10^6. FTP is the application used to transmit data through the TCP connections.

To avoid phase effect (Floyd and Jacobson, 1992), some web flows and short-lived TCP flows are used, together with 3~5 standard long-lived TCP flows. They act as background traffic for the simulation. The short-lived flows start and end randomly, and the start times of other flows are set randomly in the range of 20 s.

Simulations were run for 200 s, and we get the data during the steady state. Two router queue management policies, DT (Drop Tail) and RED (Random Early Detection) were used respectively.

RTT unfairness

To measure the RTT unfairness, simulations involving five HSTCP flows with RTTs varying from 60 ms to 240 ms were run. We found that RTT unfairness under RED router was not severe. On the contrary, when using DT router, the unfairness was very severe, so we only emphasize the case of DT router in this paper.

In Fig.2, the evolution of the congestion window is given. Subsequently, in Table 2, we list the per-flow bottleneck bandwidth utilization at different time interval calculated as follows:

$$Utilization=(Transferred\ bits\ within\ \Delta t)/(\Delta t\times Bandwidth).$$

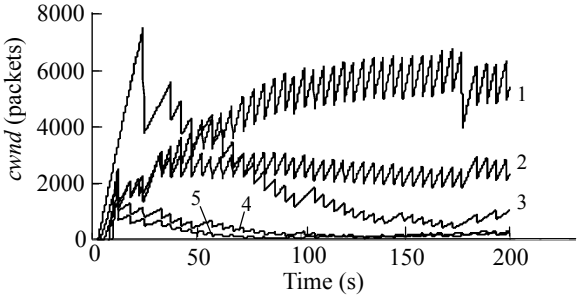

Fig.2 The congestion window evolution of HSTCP under DT router

Per-flow utilization reflects RTT unfairness of HSTCP. From Fig.2 and Table 2, we can easily find the severe RTT unfairness. Flow 1 with RTT of 60 ms, increases congestion window rapidly. As time increases, it gets more and more bandwidth. On the other hand, long RTT flows such as Flow 4 or Flow 5 evolves their windows slowly and bandwidth utilization decreases as time increases. On the whole, short

Table 2 Per-flow bottleneck link utilization (%)

Time intervals Δt (s)	Bandwidth utilization (%)				
	Flow 1 (RTT=60 ms)	Flow 2 (RTT=80 ms)	Flow 3 (RTT=140 ms)	Flow 4 (RTT=200 ms)	Flow 5 (RTT=240 ms)
50~100	38.98	20.96	18.24	2.38	1.06
100~150	48.10	21.35	14.10	1.77	0.84
150~200	52.97	21.39	11.56	1.52	0.80

RTT Flow 1 and Flow 2 get most of the bandwidth.

The RTT unfairness of STCP is more severe. Three STCP flows with RTT of 120 ms, 160 ms, 240 ms get 91%, 2.73%, 0.77% of the bandwidth respectively.

Analysis with model

We analyze the congestion window evolution using the model proposed in (Chiu and Jain, 1989), which can easily deduce RTT unfairness.

The model assumed synchronized loss which means multiple competing flows simultaneously encounter loss events. When DT routers are used, synchronized loss will happen frequently as shown in Fig.2.

Let w_i and RTT_i denote the congestion window size and the RTT of Flow i (i=1, 2) respectively. As shown in Fg.3, axes w_1 and w_2 denote congestion window of Flow 1 and Flow 2. Point denotes the congestion window size of the two flows. Efficiency line denotes the bandwidth is fully utilized by Flow 1 and Flow 2. Throughput of Flow i equals w_i/RTT_i, so the point (e.g., point A (w_1,w_2)) on efficiency line denotes the state of bandwidth fully utilization. In others words, $w_1/RTT_1+w_2/RTT_2$=bottleneck bandwidth. RTT fair line denotes the state that standard TCP with different RTT can achieve, namely, $w_1:w_2=RTT_2:RTT_1$. The throughput ratio of the two flows is $(RTT_2:RTT_1)^2$.

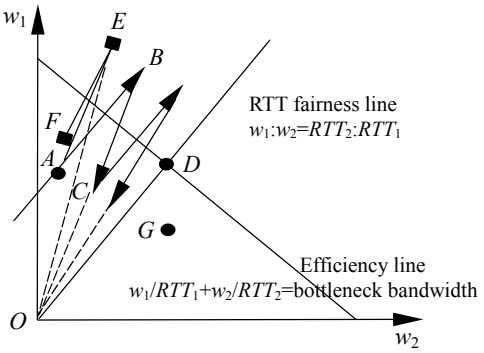

Fig.3 The analysis with model

We assume $RTT_1>RTT_2$, and begin with a state of $w_1>w_2$ denoted by point A. If the initial state is $w_1<w_2$ (denoted by point G), the same result will be obtained.

For standard TCP, $a(w)$=1, $b(w)$=0.5, because point A is under the efficiency line, which means that there is available bandwidth, so the congestion window will increase as line AB increases. According to Eq.(3), within the same time interval, the ratio of congestion window increment is $\Delta w_1/\Delta w_2=RTT_2/RTT_1$, therefore line AB is parallel to fairness line OD. Point B is above the efficiency line, so congestion will happen. According to Eq.(2), $w_1(t+1)/w_2(t+1)=w_1(t)/w_2(t)$, the congestion window will decrease with line BC, whose extension line will pass point O. Several iterations later, the congestion window of the two flows will converge to a value (denoted by point D).

However, for HSTCP, $a(w)$ and $b(w)$ become the function of the congestion window. When $w_1>w_2$, we will get $a(w_1)>a(w_2)$ and $b(w_1)<b(w_2)$. Therefore, $\Delta w_1/\Delta w_2>RTT_2/RTT_1$, the congestion window of HSTCP will increase with line AE (whose slope is greater than that of line OD). And when congestion happens, $w_1(t+1)/w_2(t+1)>w_1(t)/w_2(t)$, so the congestion window will decrease with line EF (whose extension line will not pass point O). The trend is far from point D as time elapses. In other words, w_1 will become bigger and bigger, but w_2 will become smaller and smaller.

Let $v(t)$ denote the throughput of TCP, then $v(t)=w(t)/RTT$, so the RTT unfairness will become more and more severe as time increases.

From the above analyses, we get the conclusion that the high scalability of HSTCP in low loss rate environment and synchronized loss when DT routers are used aggravate the RTT unfairness.

In fact, when the window of a flow is small, it will meet less congestion than flows with larger window. For example, as shown in Fig.2, at 102 s, Flow 1 and Flow 2 are affected by congestion, while Flow 3 avoids synchronized loss. Therefore, to some extent, the unfairness is alleviated. RED router can avoid synchronized loss, so there is no severe RTT unfairness when RED routers are adopted.

CW-HSTCP ALGORITHM

Based on our above analyses, we propose CW-HSTCP (Constant window HSTCP) as a solution.

The main idea is adding a factor to compensate the congestion window increment difference caused

by different RTT. Namely, $a(w)'=\eta\times a(w)$, where $a(w)$ was explained in Section II. The fair factor is based on the value of RTT, calculated as $\eta=c\times RTT$, where c has the similar meaning as that in CR (Constant Rate) algorithm (Floyd, 1991). For example, when c=10, and RTT=100 ms=0.1 s, η=1.

Here we give a more detailed explanation. Assume there are two flows whose RTT are 100 ms, 200 ms respectively. Suppose at a time $t1$, the sizes of their congestion window are both w. Before fair factor is added, 100 ms later the window of Flow 1 will become $w+a(w)$, and will become $w+a(w)+a(w+a(w))$ at time $t1$+200 ms. $w+a(w)+a(w+a(w))\approx w+2\times a(w)$. While for Flow 2, 200 ms later, congestion window will be $w+a(w)$. After fair factor is added, we can reduce the difference of the congestion window. If we set c=10, for Flow 1, η=1, while for flow 2, η=2. So after an RTT time, the congestion window of Flow 2 will be $w+2\times a(w)$ too. Therefore, Flow 1 and Flow 2 will have the same congestion window.

Because throughput $v(t)=w(t)/RTT$, under the condition of the same congestion windows, $v(t)_1/v(t)_2=RTT_2/RTT_1$. Namely, CW-HSTCP can keep RTT unfairness to be inversely proportional to the RTT ratio. When congestion windows are the same, the value of $a(w)$ and $b(w)$ will be the same, so stable fairness can be achieved.

After fair factor is added, long RTT flows will have sharp congestion window increment after an RTT, and because the packets are sent back-to-back, so it is easy to cause a burstiness traffic that may cause congestion and more packets loss. To solve this problem, a block-pacing method is adopted. The congestion window will be divided into several "blocks". After the packets in one block have been sent out, other packets in another block will be sent after a time interval. This can counteract the negative effects caused by adding fair factor.

We set the number of the block based on fair factor η. The details of the algorithm are shown in Fig.4 [the details of $a(w)$ and $b(w)$ were introduced in Section II, so we omit the code of them here], and give a simple illustration of block-pacing algorithm in Fig.5.

```
// Initial value
k=1; // the number of block
block=0; // the packets number in a block
number=0; // the number of packets that have been sent
On receiving a new ACK in congestion avoidance state:
    increment=c*RTT*a(w)/cwnd;
    // cwnd is the congestion window size
     if (increment>1)
        increment=1;
        // to avoid the increment larger than slow start
    cwnd=cwnd+increment;
    k=(int)c*RTT+1;
    block=cwnd/k;
On congestion happening:
    cwnd=cwnd*(1–b(w));
    number=0;
    time_0=now;
On sending date:
    if (number>block)
        time_1=now;
        // now is the value of current time
        if (k>cwnd)
            k=cwnd;
        delay=RTT/k–(time_1–time_0);
    // delay is the time interval between two blocks
    // (time_1–time_0) is the sending time of one block
        output(delay); // send data after a time of delay
        number=0; // begin a new count
        time_0=time_1+delay;
    else
        output( ); //send data directly
        number++;
```

Fig.4 Pseudo-code of CW-HSTCP

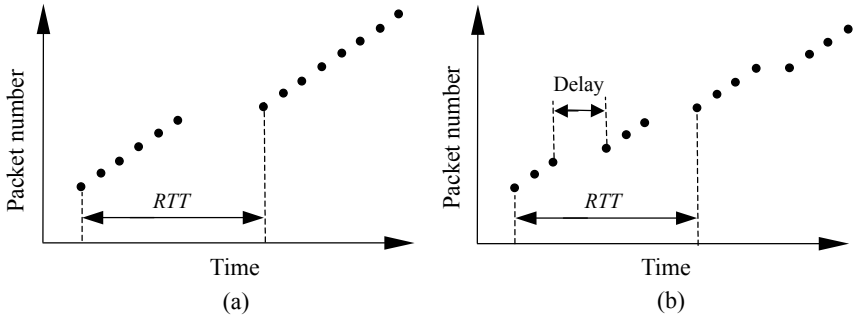

Fig.5 A simple explanation of block-pacing (block number is 2)
(a) The sending process without block-pacing; (b) The sending process with block-pacing;

A challenge in the implementation is the need for the fine-grained timer to pace out the packets. This problem was studied in (Aron and Druschel, 2002), whose work showed that the soft timer technique allowed the system timer to achieve 10 μs granularity without significant system overhead.

PERFORMANCE EVALUATION

In this section, we evaluate the performance of CW-HSTCP using simulations. The simulation topology and configuration were described in Section III.

We mainly focus on the property of bandwidth utilization, RTT fairness, and TCP friendliness. We present a relatively fairness criterion:

(1) The bandwidth requirement of standard TCP should be met. In other words, standard TCP flow has the room for bandwidth utilization.

(2) The fairness between different high-speed TCP flows should be guaranteed.

Satisfying Condition (1) also means the protocol has TCP friendliness, because standard TCP only works well in high loss rate environment. The loss rate we choose is 10^{-3}, which is in correspondence to the Low_P of HSTCP. If the total loss rate is lower than 10^{-3}, we say our protocol has the character of TCP friendliness. To evaluate the fairness between high-speed TCP flows, fair index (Chiu and Jain, 1989) is used as follows:

$$f(x)=\left(\sum_{i=1}^{n}x_i\right)^2\Big/n\sum_{i=1}^{n}x_i^2,$$

where x_i ($x_i\geq 0$) is the link utilization of Flow i.

Congestion window evolution

This simulation showed the congestion window evolution after addition of the fair factor. Three high-speed flows with RTT of 80 ms, 140 ms, 200 ms (marked as 1, 2, 3 in Fig.6) were used. We set c=10. As shown in Fig.6, different flows can get the same congestion window, so $a(w)$, $b(w)$ will be the same, and a stable fairness can be achieved. Comparison between Fig.2 and Fig.6 shows our analyses in Section III are correct.

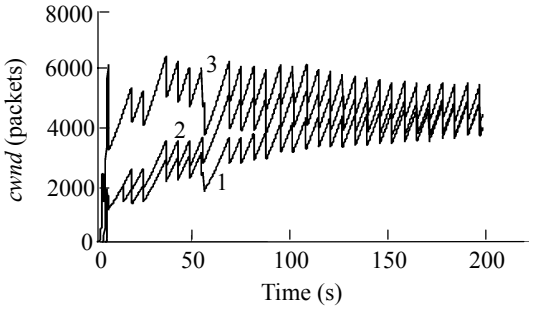

Fig.6 The congestion window evolution of CW-HSTCP

Choice of parameter c and TCP friendliness

This simulation showed the effect of parameter c. We list our simulation results in Table 3. We can find that when c=5, the per-flow bandwidth utilization and total bandwidth utilization of CW-HSTCP flows decrease together. This means that a too small value of c will limit the scalability of HSTCP. However, when c=20, short RTT CW-HSTCP flows will grasp more bandwidth and fairness are decreased too. What is more, background flows can get less bandwidth. In other words, the friendliness of CW-HSTCP decreases. Therefore, c=10 is optimal.

The packet loss rates of background flows are below 10^{-3} in 3 cases, so we say CW-HSTCP is TCP friendly.

Fairness comparison of CW-HSTCP, HSTCP, STCP and standard TCP

In this simulation, we compare the fairness of CW-HSTCP, HSTCP, STCP and standard TCP.

Three flows with RTT being 80 ms, 140 ms, and 200 ms respectively were used, and different algorithms, such as CW-HSTCP (c=10), HSTCP, STCP, and standard TCP were adopted respectively. The fairness index of high-speed flows was calculated at different time scales. As shown in Fig.7, we can find CW-HSTCP has better fairness property.

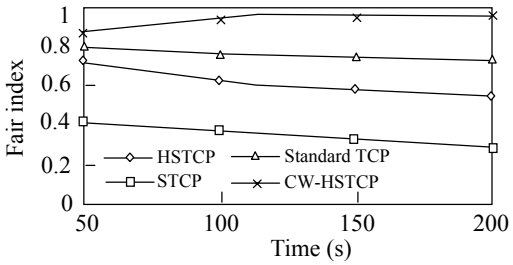

Fig.7 The fairness of different algorithms

Table 3 The effect of *c* and the TCP friendliness

c	Bandwidth utilization (%)					The fairness index of CW-HSTCP	Loss rate of background flows
	CW-HSTCP (*RTT*=80 ms)	CW-HSTCP (*RTT*=140 ms)	CW-HSTCP (*RTT*=200 ms)	Background flows	Total		
5	29.79	16.80	14.38	26.96	87.93	0.90	7×10^{-5}~1×10^{-4}
10	32.22	26.61	19.36	16.35	94.54	0.96	7×10^{-5}~8×10^{-4}
20	44.85	24.69	19.92	5.55	95.01	0.88	5×10^{-4}~1×10^{-3}

When the number of CW-HSTCP flows increase, such as five CW-HSTCP flows with RTT of 60 ms, 80 ms, 140 ms, 160 ms and 200 ms respectively, after simulation and calculation, the fairness index of these high-speed flows is 0.95. Therefore, CW-HSTCP has better fairness property.

CONCLUSION AND FUTURE WORK

Through experiments, we found severe RTT unfairness of HSTCP, and give a theoretic analyses using model. Pointing to the RTT unfairness of HSTCP, we propose CW-HSTCP as an improvement. Fair factor is added to get the equal congestion window increment, and block-pacing scheme is used to avoid the burstiness caused by sharp congestion window increase of long RTT flows. Simulation results showed that our proposal has relatively better fairness, and friendliness while keeping the advantages of HSTCP, such as scalability in high-speed networks.

In the future, we will study using the active queue management mechanism to improve the performance of HSTCP.

References

Aron, M., Druschel, P., 2002. Soft Timers: Efficient Microsecond Software Timer Support for Network Processing. ACM Transactions on Computer Systems, p.232-246.

Bhandarkar, S., Jain, S., Reddy, A.N., 2004. LTCP: A Layering Technique for Improving the Performance of TCP in Highspeed Networks. Internet draft: draft-bhandarkar-ltcp-01.txt.

Chiu, D., Jain, R., 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN*, **17**(1):1-14. [doi:10.1016/0169-7552(89)90019-6]

Floyd, S., 1991. Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic. *ACM SIGCOMM Computer Communications Review*, **21**(5):30-47. [doi:10.1145/122431.122434]

Floyd, S., 2003. High Speed TCP for Large Congestion Windows. RFC 3649.

Floyd, S., Jacobson, V., 1992. On traffic phase effects in packet-switched gateways. *Internetworking*: *Research and Experience*, **3**(3):115-156.

Foster, I., Kesselman, C., Tuecke, S., 2001. The anatomy of the grid: enabling scalable virtual organizations. *International Journal on Supercomputer Applications*, **15**(3):200-222.

Jacobson, V., Braden, R., Borman, D., 1992. TCP Extensions for High Performance. RFC 1323.

Katabi, D., Handley, M., Rohrs, C., 2002. Congestion control for high bandwidth-delay product networks. *ACM SIGCOMM Computer Communications Review*, **32**(4): 89-102. [doi:10.1145/ 964725.633035]

Kelly, T., 2003. Scalable TCP: Improving performance in high-speed wide area networks. *ACM SIGCOMM Computer Communications Review*, **33**(2):83-91. [doi:10.1145/956981.956989]

Lakshman, T.V., Madho, U., 1997. The performance of TCP/IP for networks with high bandwidth delay products and random loss. *IEEE/ACM Transactions on Networking*, **5**(3):336-350. [doi:10.1109/90.611099]

Phillips, B., 1998. Have storage area networks come of age. *Computer*, **31**(7):10-12. [doi:10.1109/2.689672]

Semke, J., Mahdavi, J., Mathis, M., 1998. Automatic TCP Buffer Tuning. Proceedings of ACM SIGCOMM, p.315-323.

Sivakumar, H., Bailey, S., Grossman, R.L., 2000. PSockets: The Case for Application-level Network Striping for Data Intensive Applications Using High Speed Wide Area Networks. Proceedings of Super Computing, p.38.

Stevens, W.R., 1994. TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley.

Xu, L., Harfoush, K., Rhee, I., 2004. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. Proceedings of INFOCOM2004, p.2514-2524.