Journal of Network and Computer Applications 34 (2011) 1416–1428

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Hybrid congestion control for high-speed networks

Wenjun Xu [a,b,*], Zude Zhou [a], D.T. Pham [b], C. Ji [b], M. Yang [b], Quan Liu [a]

[a] School of Information Engineering, Wuhan University of Technology, Jian Hu Campus, 122 Luo Shi Road, Hongshan District, Wuhan 430070, China
[b] Manufacturing Engineering Centre, School of Engineering, Cardiff University, Cardiff, UK

ARTICLE INFO

Article history:
Received 6 September 2010
Received in revised form
18 February 2011
Accepted 10 March 2011
Available online 17 March 2011

Keywords:
High-speed networks
Hybrid congestion control
Delay-based approach
Loss-based approach
Simulation
Performance analysis

ABSTRACT

Nowadays, more and more applications require fast transfer of massive data over networks, and the emergence of high-speed networks provides an ideal solution to this challenge. Due to the limitations of the conservative congestion control algorithm, the standard TCP is no longer appropriate for high-speed networks to efficiently utilize the bandwidth resources. Therefore, several high-speed TCP variants have been suggested to conquer the problem. However, although these protocols perform successfully to improve the bandwidth utilization, they still have the weakness on the performance such as RTT-fairness, TCP-friendliness, etc. In this paper, we propose HCC TCP, a hybrid congestion control algorithm using the synergy of delay-based and loss-based approach for the adaptation to high speed and long distance network environment. The algorithm uses queuing delay as the primary congestion indicator, and adjusts the window to stabilize around the size which can achieve the full utilization of available bandwidth. On the other hand, it uses packet loss as the second congestion indicator, and a loss-based congestion control strategy is utilized to maintain high bandwidth utilization in the cases that the delay-based strategy performs inefficiently in the networks. The two approaches in the algorithm are dynamically transferred into each other according to the network status. We finally perform simulations to verify the properties of the proposed HCC TCP. The simulation results demonstrate HCC TCP satisfies the requirements for an ideal TCP variant in high-speed networks, and achieves efficient performance on throughput, fairness, TCP-friendliness, robustness, etc.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

The rapid evolution of high-speed networks is significantly supporting the international collaborations with massive data transfer and computing resource sharing, and the networks, e.g. StarLight (2010), UKLight (2010), NetherLight (2010), CERN (2010), etc. integrated with 1–10 Gbps bandwidths have been developed and deployed over numbers of research institutions. In order to efficiently utilize the large bandwidths at the physical layer, researchers have focused on the developments of protocols at transport and network layers.

The standard TCP has been remarkably successful in performing congestion avoidance and control to prevent severe congestion in the current low-speed networks. However, it is well-known that the standard TCP is not appropriate for high-speed networks in terms of the additive increment multiplicative decrement (AIMD) algorithm is too conservative to rapidly achieve full bandwidth utilization while is too drastic to recover from per packet loss event. In order to conquer the poor performance problem, the standard TCP together with the AIMD algorithm should be modified in high-speed networks. So far, a number of high-speed TCP variants have been proposed, including the end-to-end approaches, e.g. HighSpeed TCP (HSTCP) (Floyd, 2003), Scalable TCP (STCP) (Kelly, 2003), HTCP (Leith and Shorten, 2004), BIC TCP (Xu et al., 2004), CUBIC TCP (Ha et al., 2008), FAST TCP (Wei et al., 2006), Compound TCP (CTCP) (Tan et al., 2006), TCP-Illinois (Liu et al., 2008) and the router-based approaches, e.g. XCP (Katabi et al., 2002), VCP (Xia et al., 2005). In addition, some researches focus on the application-level schemes on top of UDP to realize the congestion control functions for high-speed networks, such as UDT (Gu and Grossman, 2007). Although these approaches achieve higher throughput over the standard TCP in high-speed networks, most of them also have shortcomings in various aspects such as fairness, TCP-friendly, responsiveness, robustness, etc. Since none of the existing approaches is overwhelmingly better than the other protocols and has the convincing evidence that could be generally deployed, the development of new high-speed TCP variants is still needed.

In this paper, we introduce a new congestion control protocol, named hybrid congestion control TCP (HCC TCP), for high-speed networks. The protocol utilizes the delay information as the primary congestion indicator and utilizes the loss information as the second congestion indicator to jointly adjust the window

* Corresponding author at: School of Information Engineering, Wuhan University of Technology, Wuhan, China. Tel.: +86 139 71 69 3463.
E-mail address: xuwenjun@whut.edu.cn (W. Xu).

size so as to satisfy the design requirements on efficiency, fairness, TCP-friendliness and robust, and outperforms the standard TCP and other TCP variants in high-speed networks. Due to the synergy of the delay-based strategy and loss-based strategy, HCC TCP is a hybrid scheme of congestion control.

The paper is organized as follows. In Section 2, a brief overview of related work is presented. Section 3 describes the main mechanisms and development of the proposed protocol. Then the experiment results are presented and discussed in Section 4. Finally Section 5 concludes this paper.

2. Related work

As the aforementioned content, numbers of new protocols have been developed to replace the standard TCP and achieve efficient bandwidth utilization in high-speed networks. The router-based protocols, such as XCP and VCP, require the explicit feedback information from routers to guide their control strategies. However, it is impractical to modify all the existing routers in a real world. Therefore, a majority of the existing protocols focus on the end-to-end method rather than the router-based method for the performance improvement of high-speed networks.

The end-to-end protocols can be mainly classified into two categories: loss-based congestion control algorithms, e.g. HSTCP, STCP, HTCP, BIC TCP, CUBIC TCP, etc. and delay-based congestion control algorithms such as FAST TCP. The loss-based congestion control algorithms utilize packet loss as the congestion measure, the window size increases for each ACK and decreases per packet loss. HSTCP and STCP are the early works along the loss-based methods. To quickly catch up the available bandwidth, HSTCP uses step-wise functions for the increase and decrease of window size while STCP sets the increasing and decreasing values proportional to the current window size. However, both the two protocols have a serious problem on RTT-fairness performance. Using these protocols, as the multiple flows competing for the bottleneck bandwidth have different RTT delays, the fair utilization of the bandwidth cannot be achieved. HTCP sets a function of the elapsed time since last packet loss for increase parameter and uses an adaptive backoff strategy at congestion events so as to achieve a perfect performance on responsiveness and efficiency in high-speed networks. For the above three protocols, the increment of window size is still fast even the network is close to the congestion event, thus the congestion in network will be caused more easily among the competing flows and result in the degradation of throughput. BIC TCP is an effective protocol that has drawn much attention in research area. The protocol adjusts the window size using a binary search method to reach a reference value. When updating the window size, it sets the reference value as the midpoint between the maximum reference value W_{max} and the minimum reference value W_{min}. If the length between W_{min} and the midpoint is larger than a maximum value S_{max}, the window size increases linearly by the value S_{max}. Hence, the increment of the window size is linear at the initial stage and then becomes logarithmic when approaching to the reference point. BIC TCP performs better than the earlier approaches. However, it also suffers the RTT unfairness problem. Subsequently, an enhanced version, CUBIC TCP, is developed to improve the RTT-fairness performance of BIC TCP.

On the other hand, fundamentally different from loss-based congestion control algorithms, delay-based congestion control algorithms use queuing delay as the congestion measure. FAST TCP is a typical delay-based high-speed TCP variant derived from TCP Vegas (Brakmo and Peterson, 1995). The protocol maintains queue occupancy at routers for a small but not zero value so as to lead the network around full bandwidth utilization and achieve a higher average throughput. On the contrary, the throughput of loss-based algorithms oscillates between full utilization and under utilization in terms of the probing action purposely generates packet losses. In addition, FAST TCP is able to rapidly converge to the equilibrium state and does not suffer the RTT unfairness problem. However, despite the unique advantages mentioned above, it also has some inherent limitations. Since FAST TCP is a delay-based approach and uses the RTTs for congestion measure, its throughput performance is significantly affected by the reverse traffic, and the throughput of the source traffic decreases as the queuing delay increases on the reverse path. Some works have focused on the reverse traffic problem in delay-based congestion control algorithms and uses a variety of schemes that relies on the measurement of one-way delay to conquer this problem, e.g. in Parsa and Garcia-Luna-Aceves (1999), Fu and Liewm (2003), Kuzmanovic and Knightly (2003) and Chan et al. (2004). However, these schemes are not designed for high-speed networks. In addition to the reverse traffic problem, FAST TCP requires the buffer size to be larger than the specified value which indicates the total packet amount maintained in routers along the flow's path.

Although any of the loss-based and delay-based approaches can achieve higher throughput than the standard TCP, both of them have their pros and cons. In order to perform more efficiently and effectively in high-speed networks, some approaches, like CTCP and TCP-Illinois, focus on the synergy of loss-based and delay-based approach. CTCP utilizes loss and delay information as the primary congestion indicators in different stages to determine direction of window size change, and keeps the traditional Slow-Start at the start-up period while uses a delay-based component derived from TCP Vegas in congestion avoidance phase. TCP-Illinois uses loss information as the primary congestion indicator and uses delay information to be the second congestion indicator. During the operation, it utilizes the loss information to determine the direction of window change and the delay information is used for adjusting the pace of window size change. In order to achieve a concave window size curve and a high throughput, TCP-Illinois set two parameters, α and β, in the protocol operation. When network is far from congestion, it set α to be large and β to be small. Otherwise, α is small and β is large when network is close to congestion. These approaches inherit the advantages from both the loss-based and delay-based approaches. However, due to the delay-based components still uses RTTs to measure the congestion, their throughput performance is also affected by the reverse traffic. In Xu et al. (2010), we presented an end-to-end Enhanced FAST (EEFAST) congestion control algorithm, which dynamically adjusting the window size according to the measurements of one-way delay, to remove the effect of reverse traffic for delay-based congestion measurement in high-speed networks. However, EEFAST is a delay-based congestion control algorithm and suffers the same problem as FAST in which the packet amount maintained in routers should be smaller than the router buffer size.

3. HCC TCP protocol: mechanisms and development

The synergic methods, as CTCP and TCP-Illinois, inherit the advantages from both the loss-based and delay-based approaches. Although these approaches still suffer some limitations, they are able to effectively overcome the weakness which is difficult to be remedied by either loss-based methods or delay-based methods themselves. Therefore, HCC TCP also adopts the method that uses the synergy of the loss-based and delay-based approach to realize the congestion control for high-speed

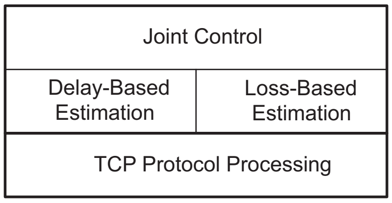

Fig. 1. HCC TCP architecture.

networks. Since a measurement of delay provides multi-information related to congestion but a measurement of packet loss only provide one bit information, we uses the delay information as the primary congestion indicator and uses the loss information as the second congestion indicator. This mechanism fundamentally differentiates HCC TCP from CTCP and TCP-Illinois.

3.1. Architecture

As illustrated in Fig. 1, the congestion control mechanism of HCC TCP can be separated into three components: *delay-based estimation*, *loss-based estimation* and *joint control*. The *delay-based estimation* component determines the congestion measure using queuing delay and the *loss-based estimation* component determines the congestion measure using packet loss. The window control strategy relying on the measurements of the delay and loss information is realized by the *joint control* component.

3.2. Delay-based congestion control

From the perspective of a delay-based congestion control approach, such as FAST TCP, if the queuing delay on the reverse path is heavy, the full utilization of available bandwidth will never be achieved and thus lead to potentially serious degradation of throughput on the forward path. In Xu et al. (2010), we presented EEFAST congestion control algorithm to remove the effect of the reverse traffic in high-speed networks. For the design of the *delay-based estimation* component, the mechanisms of the EEFAST algorithm are used to estimate the congestion in a network. In addition, based on this algorithm, we also adopt a set of new control strategies for adjusting the window size in order to achieve a further performance improvement.

At equilibrium state, a number of packets α_i should be queued in routers by a delay-based congestion control approach. Without the effect from the queuing delay on the reverse path, the source i can achieve full utilization of available bandwidth on the forward path and the window size can be written as

$$w_i = \frac{\alpha_i}{Q_i^f} D_i = \frac{\alpha_i}{Q_i^f}(Q_i^f + Q_i^r + d_i) \tag{1}$$

From Eq. (1), if the source i attempt to maintain α_i packets in the queue along the path at time t, the anticipated window size $w_i^a(t)$ should be

$$w_i^a(t) = \frac{\alpha_i}{Q_i^f(t)} D_i(t) = \alpha_i\left(1 + \frac{Q_i^r(t) + d_i}{Q_i^f(t)}\right) \tag{2}$$

Consider an ideal network model that a source sends data through a path. During the initial phase of protocol operation, since the forward queuing is usually a small value, the anticipated window size $w_i^a(t)$ is extremely larger than the current window size $w_i^a(t)$. With the rise of the current window size, the forward queuing delay increases thus lead to the decrease of the anticipated window size. The changes of the two kinds of window size are shown in Fig. 2. If the distance between the anticipated window size and current window size locates within a threshold

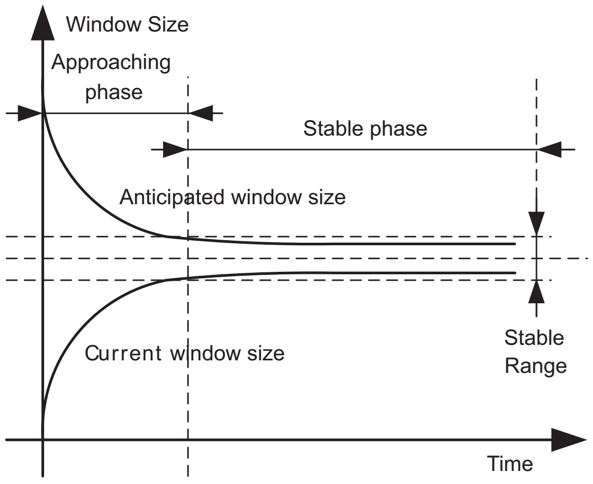

Fig. 2. Window size changes by the delay-based strategy.

range, it can be considered that the current window size almost reaches the anticipated value and the full utilization of the available bandwidth is achieved. Hence, the window size should stabilize around the anticipated value.

According to the aforementioned analysis, we design a delay-based control strategy in HCC TCP so as to let the window size efficiently converge to the anticipated value and achieve a high utilization of network bandwidth. The details of the delay-based control algorithm are described as below.

3.2.1. Calculating the values of window size change

HCC TCP utilizes delay information as the primary congestion indicator. From the startup, the timestamp option described in Kuzmanovic and Knightly (2003) and Alonso et al. (2007) is used to measure the forward one-way delay D_i^f. Let $baseD_i^f$ be the minimum D_i^f observed so far, the forward queuing delay of source i is $Q_i^f = D_i^f - baseD_i^f$.

Each source computes its anticipated window $w_i^a(t)$ periodically according to

$$w_i^a(t) = \frac{\alpha_i}{avgQ_i^f(t)} avgD_i(t) \tag{3}$$

where $avgQ_i^f(t)$ and $avgD_i(t)$ denote the average values of the forward queuing delay and the round-trip time, respectively. Then the distance Δw_i between the anticipated window and the current window can be calculated as $\Delta w_i(t) = w_i^a(t) - w_i^c(t)$. If Δw_i is above zero, the current window should increases. Otherwise, it should decreases to approach the anticipated window size.

3.2.2. Adjusting the window size

We take the size of the anticipated window as a *delay-reference* in HCC TCP. As the *delay-reference* is determined, we adjust the window according to the value Δw_i so that enable the current window size approach the *delay-reference*. However, if the distance between the anticipated window and the current window is too large, directly changing the window size to the *delay-reference* will cause so much stress to a network. Hence, as suggested in Xu et al. (2004), a maximum value I_{max} is set for the window size change when $|\Delta w_i|$ is larger than I_{max}, and the window size increases/decreases by I_{max} until the distance $|\Delta w_i|$ is less than I_{max}.

In order to rapidly converge to the full bandwidth utilization and also to avoid the burst change of window size, proper parameter I_{max} should be set in the delay-based control strategy. As the analysis in Xu et al. (2010), the ideal maximum increment of window size in FAST TCP is $\gamma\alpha_i$, and therefore the maximum value I_{max} is also set as $\gamma\alpha_i$ for window size change in HCC TCP. The protocol periodically updates the window size per RTT. When the value for window size change is obtained, we do not directly adjust the change value in the RTT. In order to enable the congestion window gradually approach the anticipated window size, only a half of the change value determined by Δw_i and I_{max} is used for adjusting the window in the update phase. Note that BIC TCP set the midpoint of the current window size and the maximum window size to be the target window size. Due to the anticipated window size that can cause the packet loss must locates in the distance between the two values, BIC TCP uses the binary increase method to search the anticipated value within a determined area. On the other hand, HCC TCP relies on Eq. (3) to calculate the anticipated window size, and a half of the value determined by Δw_i and I_{max} is used to enable the current window approach to the finally anticipated window size, which locating in the stable range shown in Fig. 2. This makes HCC TCP fundamentally different from the mechanisms in BIC TCP, which use the half of a determined value to increase the window size.

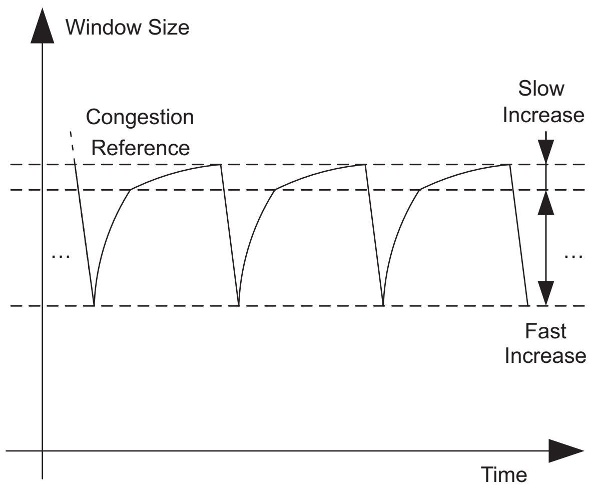

Fig. 3. Linear to logarithmic increase scheme.

3.3. Loss-based congestion control

Delay-based congestion control algorithms require a specified number of packets queued in routers so as to keep the average throughput around the full utilization. Therefore the buffer size of routers should be larger than the specified value in the delay-based algorithms, and the specified value for a network increases as the increment of source numbers. However, if the buffer size of the routers is not large enough for the specified value, packet loss might happen in the networks. To tackle this, we use packet loss as the second congestion indicator and design a loss-based congestion control strategy for the operation of HCC TCP.

For the loss-based congestion control, when the network is close to the congestion status, the fast increment of window size could lead to the congestion event more easily and cause a heavy oscillation of window size so that degrade the throughput performance for each traffic source. The linear to logarithmic increase function, as described in Xu et al. (2004) and Liu et al. (2008), is an efficient way to avoid the heavy congestion induced by fast increment of window size. The approaches increase the window linearly at the initial stage and then increase logarithmically to get close to the reference point that a congestion event may happen. Fundamentally, the change of the window size is from fast to slow. Therefore, using such mechanisms, the traffic source can rapidly catch up the available bandwidth and also prolong the time interval between two successive congestion events so as to achieve better performance on average throughput.

To rapidly increase the window size while keep it around the congestion reference point for a longer time, similar to the linear to logarithmic increase functions in Xu et al. (2004), we use the binary increase scheme associated with the delay-based strategy described above to adjust the window size. As shown in Fig. 3, once a packet is detected to be lost, the source immediately halves the window and records the window size that the congestion event happens as a *loss-reference*. When the source starts to recover its window size to obtain the bandwidth utilization, it sets the *loss-reference* as the maximum size w_i^{max} and the current window is treated as the minimum size w_i^{min}, then the target increment of window size is

$$w_i^{tar} = (w_i^{max} - w_i^{min})/2 \tag{4}$$

The source increases the window size by the target value. After the target increment of window size is achieved and if there is no packet lost, the current window size is set to be the new minimum size and a new target increment of window size is computed according to Eq. (4). Using the binary increase scheme, the window grows rapidly if the current size is far from the *loss-reference* and grows slowly if the current size is close to the *loss-reference*. On the other hand, if a packet loss is detected during the growth of the window, the source updates the *loss-reference* as well as the maximum size to be the current window size. As the mechanisms in the delay-based strategy, the maximum value I_{max} is also set for the window size change if the increment value is larger than I_{max}. Meanwhile, a minimum increment value I_{min} is used for increasing the window if the increment value falls below I_{min}. A half of the increment value calculated by the above mechanisms is utilized to adjust the window size in the loss-based control strategy.

3.4. Implementation of window control algorithm

In this subsection, we show how the delay-based strategy and the loss-based strategy cooperate in the operation phase of HCC TCP, and the detailed implementation of window control algorithm is presented as follows.

At startup, HCC TCP relies on the delay-based algorithm to increase the window size. Firstly, like FAST TCP, we set a threshold value, *mi_threshold*, to estimate the congestion on the forward path. If $avgQ_i^f(t) < mi\_threshold$, it indicates that the forward queuing is light, the multiplicative increase (MI) scheme can be used to rapidly increase the window size. Otherwise, the protocol should periodically update the congestion window using the delay-based algorithm in terms of the queuing along the forward path becomes heavy. Secondly, we set the *loss-reference* to be a default maximum value until it is updated by a congestion event. During the startup period, the congestion window is adjusted by the delay-based congestion control algorithm and can lead to a rapid growth in congestion window size.

Figure 4 depicts the window growth function of HCC TCP. If there is no detection of packet loss when the source reaches the equilibrium state, it indicates that the *loss-reference* is above the *delay-reference* that the source can achieve the full utilization of bandwidth, as shown in Fig. 4(a). Therefore the congestion window finally stabilizes around the *delay-reference*, and the window size changes according to the dynamics of the *delay-reference*. On the other hand, before reaching the *delay-reference*, if a packet loss is detected, it indicates that the *loss-reference* is less than the *delay-reference*, as shown in Fig. 4(b). At that time, the source updates its *loss-reference*, halves the size of congestion window and then uses the loss-based congestion control

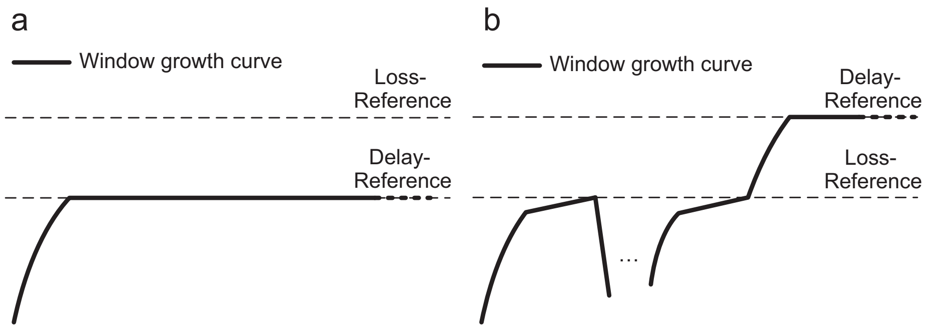

Fig. 4. The window growth function of HCC TCP. (a) *Loss-reference* is above *delay-reference* and (b) *loss-reference* is less than *delay-reference*.

Table 1
Pseudo-code of window control algorithm in HCC TCP.

```
Variables:
  $W$ Current window size;
  $W_{old}$ Previous window size;
  $w_i(t)$ Window size of source $i$, at time $t$;
  $\alpha_i$ Packet numbers that should be maintained at routers for source $i$;
  $mi\_threshold$ Threshold value for forward congestion estimation.

Startup ( ):
  Set loss-reference to be a default maximum value;
  Start to calculate average queuing delay: $avgQ_i^f(t)$;
  Start to calculate average RTT: $avgD_i(t)$;
  If $avgQ_i^f(t) < mi\_threshold$, then $W = 2 \cdot W_{old}$ per RTT;
  If $avgQ_i^f(t) \geq mi\_threshold$, then turn to delay-based congestion control
function ( );

Delay-based congestion control function ( ):
  Estimate anticipated window size (delay-reference):
    $w_i^a(t) = \alpha_i \cdot avgD_i(t)/avgQ_i^f(t)$;
  Calculate the change for current window size: $\Delta w_i(t) = w_i^a(t) - w_i^c(t)$;
  If $\Delta w_i(t) > I_{max}$, then $\Delta w_i(t) = I_{max}$;
  If $\Delta w_i(t) < -I_{max}$, then $\Delta w_i(t) = -I_{max}$;
  Update congestion window: $W = W_{old} + (\Delta w_i/2)$ per RTT;
  Upon detected loss packet:
    update loss-reference;
    back off to $W = W/2$;
    turn to loss-based congestion control function ( );

Loss-based congestion control function ( ):
  Calculate increment for current window size:
    $w_i^{tar} = (w_i^{max} - w_i^{min})/2$;
  If $w_i^{tar} > I_{max}$, then $w_i^{tar} = I_{max}$;
  If $w_i^{tar} < I_{min}$, then $w_i^{tar} = I_{min}$;
  Update congestion window: $W = W_{old} + (w_i^{tar}/2)$ per RTT;
  If $W >$ loss-reference & no packet loss is detected,
    reset loss-reference;
    turn to delay-based congestion control function ( );
  Upon detected loss packet:
    update loss-reference;
    back off to $W = W/2$;
```

algorithm to adjust the window. After the congestion window grows past the *loss-reference*, if no packet loss is given, it indicates that the congestion in the network is mitigated and there is more available bandwidth for the source. To rapidly catch up the available bandwidth, after passing the *loss-reference*, the source resets the *loss-reference* as the default maximum value and turns to the delay-based congestion control algorithm so that enable the window size converge to the *delay-reference*. Note that if a packet loss is detected before reaching the *delay-reference*, the source will update its *loss-reference* and turn to the loss-based algorithm again for adjusting the window size. The corresponding pseudo-code of window control algorithm in HCC TCP is illustrated as Table 1.

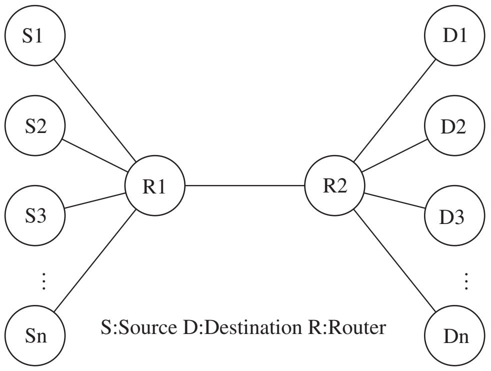

Fig. 5. The dumb-bell network topology.

4. Performance evaluation

Using ns-2 (Cui and Andrew, 2007; NEC-Labs, 2007; Wei and Cao, 2007; USC/ISI, 2010), we conducted extensive simulation experiments to evaluate the HCC TCP protocol and compare its performance with TCP Reno, TCP Vegas, HSTCP, STCP, HTCP, BIC-TCP, TCP-Illinois and FAST TCP. As shown in Fig. 5, a dumb-bell network topology in which one or multiple of source and destination nodes share a single bottleneck link is used in simulation. The forward and reverse links each with 200 Mbps bandwidth and 20 ms propagation delay, and the routers are equipped with the first-in-first-out (FIFO) service discipline for queuing. The size of data packets is 1000 bytes in all experiments.

For TCP Reno, TCP Vegas, HSTCP, STCP, HTCP, BIC-TCP and TCP-Illinois, their default parameter settings are used in the simulations. For FAST TCP and HCC TCP, we set $\alpha = 400$ and $\gamma = 0.5$. In addition, the propagation delay of packets is used in the window control algorithm of FAST TCP and HCC TCP and the estimation error induced by queuing delay can result in certain unfairness among the flows. So far, some effective solutions have been provided in Low et al. (2002), Tan et al. (2005) and Cui et al. (2006) to eliminate this error. Due to the unfairness caused by propagation delay measurements is not the problem we focused on in this paper, it is assumed that each flow of FATS TCP and HCC TCP knows its true propagation delay in the simulations.

4.1. *Efficiency of a single traffic flow*

We perform simulations to evaluate the efficiency of a single traffic flow with different values of buffer size. The simulation period for one simulation run is 300 s. Figure 6 shows the average throughput for the buffer sizes varied from 100 to 4000 pkts. From the graph, it can be seen that the average throughput of all the protocols increases as the buffer size grows, and the high-speed TCP variants uniformly achieve a better throughput

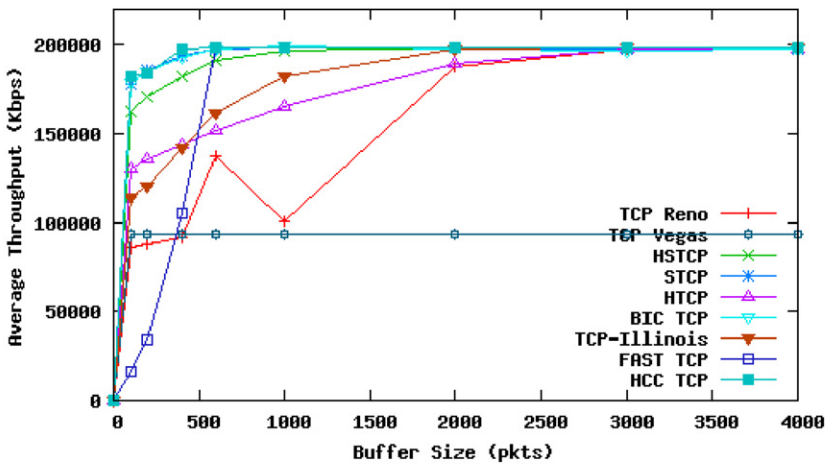

Fig. 6. Average throughput versus buffer size.

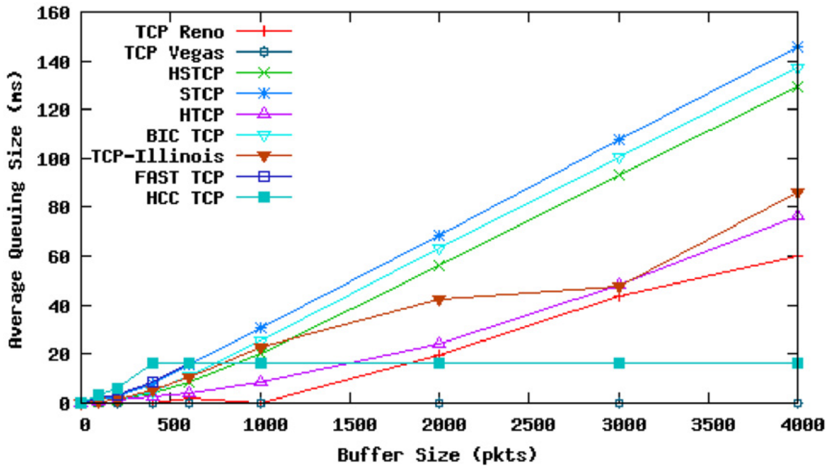

Fig. 7. Average queuing size versus buffer size.

performance than TCP Reno and TCP Vegas which are not designed for high-speed networks. However, the average throughput of FAST TCP falls rapidly when the buffer size is less than 400 pkts. HCC TCP throughput does not change significantly for the varied buffer size. For the higher buffer sizes, its throughput slightly increases, and then approximately achieves the full utilization when the buffer size becomes higher than 400 pkts. Moreover, HCC TCP performs better on throughput performance than other high-speed TCP variants almost in all cases of buffer size. The average queuing size of the protocols for the varied buffer sizes is shown in Fig. 7. It can be observed that the average queuing size of the loss-based congestion control protocols, e.g. TCP Reno, HSTCP, STCP, HTCP and BIC TCP as well as TCP-Illinois which uses the packet loss as the primary congestion indicator, increases rapidly with the growth of buffer size and then becomes extremely higher than that of TCP Vegas, FAST TCP and HCC TCP. Therefore, the protocols, which using the queuing delay as the primary congestion indicator, bring fewer overloads to routers than the loss-based protocols, especially when the buffer size is large.

Then the average throughput of the protocols is evaluated under a network with different packet loss probabilities, e.g. 1×10^{-5}, 0.7×10^{-5}, 0.5×10^{-5}, 0.3×10^{-5}, 1×10^{-6}, which are according to a Bernoulli process. The results obtained are presented in Fig. 8. It can be seen that when the router buffer size is 200 pkts, as shown in Fig. 8(a), the average throughput of all the protocols decreases as the value of packet loss probability increases. Most of the high-speed TCP variants achieve higher average throughput than TCP Reno and TCP Vegas in all cases. HCC TCP is able to maintain a high average throughput amongst the protocols. However, the average throughput of FAST TCP is much lower than other high-speed TCP variants. As shown in Fig. 8(b), when the router buffer size is 2000 pkts, the average throughput of all the high-speed TCP variants is higher than TCP Reno and TCP Vegas in all packet loss probability cases, and HCC TCP can also achieve higher average throughput than most of other protocols.

From the simulation results, we demonstrate that HCC TCP outperforms the current high-speed TCP variants in the performance of average throughput and queuing size. This is mainly because of the hybrid nature of HCC TCP. Since HCC TCP should maintains α packets at routers when reaches the equilibrium state. If the buffer size is higher than 400 pkts and it gives no packet loss, the delay-based strategy is used for congestion control, as shown in Fig. 9(b), HCC TCP rapidly converges to the equilibrium state and achieves full bandwidth utilization. Moreover, a number of packets α are maintained at the routers along the path so that results in a fix average queuing size for the traffic flow. When the buffer size is less than 400 pkts and it gives packet loss events for the delay-based strategy, then HCC TCP turns to

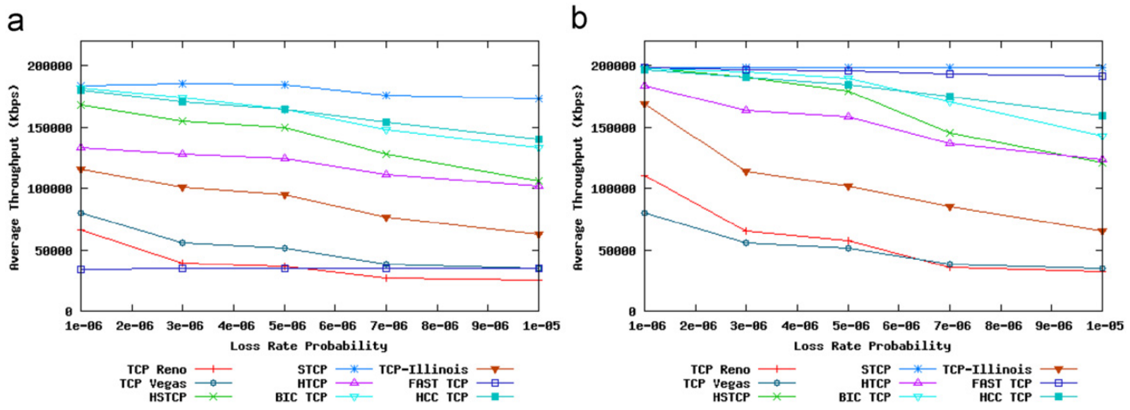

Fig. 8. Average throughput versus packet loss probability. (a) Buffer size is 200 pkts and (b) buffer size is 2000 pkts.

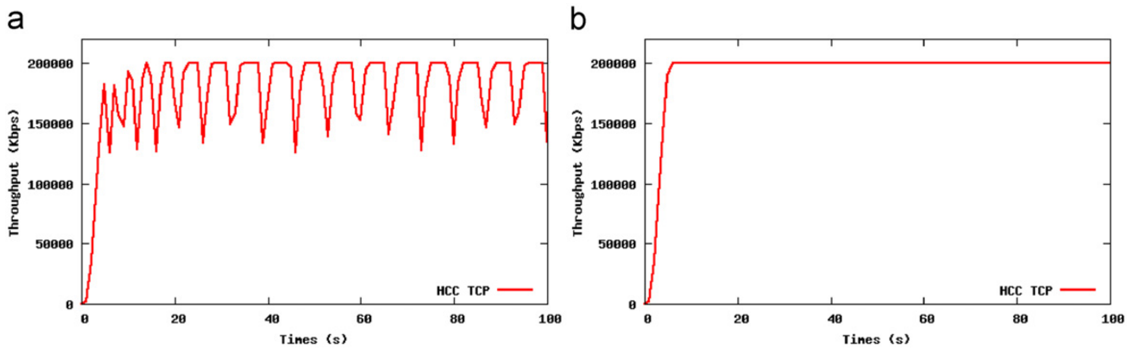

Fig. 9. Rate dynamics of HCC TCP. (a) Buffer size is 200 pkts and (b) buffer size is 600 pkts

the loss-based strategy for congestion control. The linear to logarithmic increase of window size realized by the loss-based strategy keeps the window close to the congestion point for a longer time and decreases the probability of packet loss event, thus it still achieves high bandwidth utilization in the high-speed networks, as shown in Fig. 9(a).

4.2. Fairness

In order to evaluate the fairness performance of HCC TCP, we consider two different scenarios, e.g. homogeneous RTT and heterogeneous RTT, respectively. Furthermore, the Jain's fairness index (*FI*) is used to quantitatively evaluate the fairness performance of the protocols (Jain et al., 1984; Gu and Grossman, 2007)

$$FI=\frac{(\sum_{i=1}^{n}\overline{x}_i)^2}{n\cdot\sum_{i=1}^{n}\overline{x}_i^2} \tag{5}$$

where n is the number of the concurrent flows and $\overline{x}_i$ is the average throughput of the flow *i*. The value of *IF* is always no more than 1. The larger *IF* value means the better fairness performance, and as the value equal to 1, the competing flows in a network achieve definitely equivalent throughput. The efficiency index (*EI*), which is the amount of the concurrent flows' throughput, is also used to evaluate the throughput performance of the protocols.

For homogeneous RTT scenarios, three connections pass through the same bottleneck path and the RTT is 120 ms for all users. The three sources start sending data at 0 s and terminate at 300 s simultaneously. Figure 10 depicts the average throughput of three users using the high-speed TCP variants. We see that when the buffer size is 2000 pkts, as shown in Fig. 10(b), all the protocols enable the users fairly share the link resources, and HCC TCP obtains approximately full utilization of bandwidth as other high-speed protocols. As the buffer size decreases to 600 pkts, the results in Fig. 10(a) show that the fairness performance of STCP, TCP-Illinois and FAST TCP degrades significantly. However, HCC TCP not only achieves fair sharing of link resources among the three users, but also maintains high utilization of bandwidth. This is again because of the hybrid nature of HCC TCP. When the buffer size is larger than the predefined number of packets that need to be maintained at routers, the delay-based strategy is implemented in HCC TCP will not cause packet loss. Hence, HCC TCP performs well as FAST TCP to fairly share the link capacity. When the buffer size is less than the predefined number of packets and there are packet loss be detected, the delay-based protocol, FAST TCP, performs inefficiently in the network environment. At that time, HCC TCP utilizes the loss-based strategy using the binary increase scheme to adjust the window size so that still achieves an efficient fairness performance as well as the high utilization of bandwidth.

For heterogeneous RTT scenarios, the fairness performance of theses high-speed TCP variants is demonstrated with a different RTT. We first conduct the experiments that two high-speed flows with different RTT ratios share the bottleneck link. The RTT of flow 1 is 60 ms, and the RTT of flow 2 varies among 90, 120 and 360 ms, thus the RTT ratio for the two high-speed flows is 1.5, 2 and 6, respectively. Table 2 illustrates the results for the runs with the buffer sizes of 600 and 2000 pkts, respectively. It can be observed that when the buffer size is 600 pkts, as shown in Table 2(a), the fairness performance of all the protocols is significantly affected by the RTT ratio, and the flow 1 with a short RTT obtains much higher throughput than the flow 2 with a long RTT. Compare to HSTCP, STCP, HTCP, BIC TCP and FAST TCP,

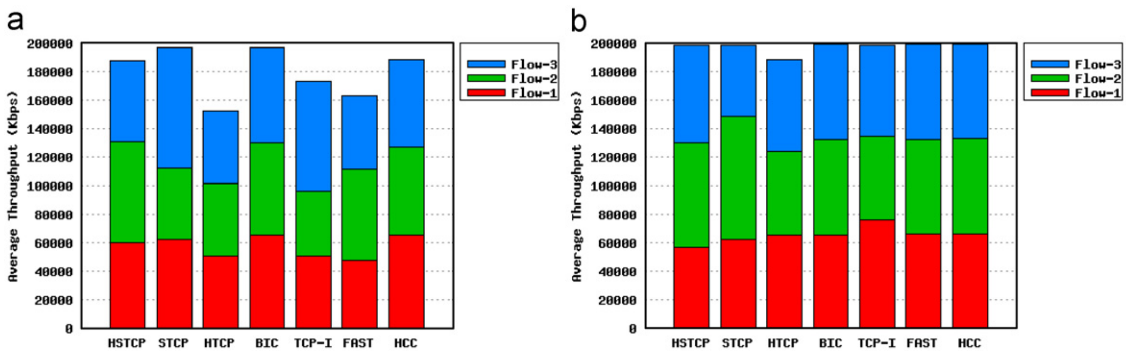

Fig. 10. Average throughput of 3 homogeneous RTT (120 ms) users. (a) Buffer size is 600 pkts and (b) buffer size is 2000 pkts.

Table 2
The simulation results of *EI* and *FI*.

(a) Buffer size is 600 pkts

RTT ratio	**1.5**				**2**				**6**			
Protocols	**T1**	**T2**	***EI***	***FI***	**T1**	**T2**	***EI***	***FI***	**T1**	**T2**	***EI***	***FI***
HSTCP	197.91	1.68	199.59	**0.5085**	196.75	2.76	199.51	**0.5140**	198.57	0.66	199.23	**0.5033**
STCP	199.53	0.08	199.61	**0.5004**	199.24	0.29	199.53	**0.5015**	199.54	0.02	199.56	**0.5001**
HTCP	175.96	9.38	185.34	**0.5532**	184.64	6.09	190.73	**0.5329**	166.89	9.31	176.20	**0.5556**
BIC TCP	199.02	0.62	199.64	**0.5031**	199.00	0.62	199.62	**0.5031**	199.31	0.34	199.65	**0.5017**
TCP-Illinois	156.56	42.28	198.84	**0.7517**	173.75	23.15	196.90	**0.6309**	195.81	2.93	198.74	**0.5150**
FAST TCP	187.13	12.43	199.56	**0.5661**	190.11	9.44	199.55	**0.5495**	186.76	6.21	192.97	**0.5332**
HCC TCP	177.03	22.45	199.48	**0.6248**	179.19	20.26	199.45	**0.6116**	182.33	16.18	198.51	**0.5880**

(b) Buffer size is 2000 pkts

RTT ratio	**1.5**				**2**				**6**			
Protocols	**T1**	**T2**	***EI***	***FI***	**T1**	**T2**	***EI***	***FI***	**T1**	**T2**	***EI***	***FI***
HSTCP	196.82	2.84	199.66	**0.5144**	199.35	0.23	199.58	**0.5012**	199.01	0.32	199.33	**0.5016**
STCP	198.74	0.93	199.67	**0.5047**	199.45	0.12	199.57	**0.5006**	197.69	1.26	198.95	**0.5064**
HTCP	189.87	9.79	199.66	**0.5514**	189.55	10.00	199.55	**0.5526**	168.39	30.28	198.67	**0.6742**
BIC TCP	199.33	0.31	199.64	**0.5016**	198.17	1.33	199.50	**0.5067**	199.03	0.63	199.66	**0.5032**
TCP-Illinois	130.19	69.27	199.46	**0.9147**	161.90	37.56	199.46	**0.7201**	168.30	31.13	199.43	**0.6788**
FAST TCP	100.18	99.40	199.58	**1.0000**	100.35	99.22	199.57	**1.0000**	102.77	96.77	199.54	**0.9991**
HCC TCP	99.94	99.55	199.49	**1.0000**	100.13	99.33	199.46	**1.0000**	103.31	96.11	199.42	**0.9987**

T1: average throughput of user 1 (Mbps); T2: average throughput of user 2 (Mbps).

TCP-Illinois and HCC TCP is fairer to the heterogeneous RTT users. Moreover, HCC TCP does not significantly deteriorate as the RTT ratio changes. We also note that when the buffer size is 2000 pkts, as shown in Table 2(b), FAST TCP and HCC TCP achieves fairly share of the bottleneck link and perform much better than HSTCP, STCP, HTCP, BIC TCP and TCP-Illinois. This is because in the scenario, FAST TCP and HCC TCP rely on the delay-based strategy to realize the congestion control function, and the buffer size is higher than the number of packets that need to maintained at routers, thus the delay-based strategy can normally operate under the high bandwidth environment and avoid the unfairness caused by the heterogeneous RTTs.

To further demonstrate the fairness performance of the high-speed protocols, we run the simulations that multiple users start sending data at different times to share the bottleneck link. In Wei et al. (2006), Li et al. (2007) Ha et al. (2008) and Liu et al. (2008), extensive experiments have been conducted to evaluate the fairness performance of HSTCP, STCP, HTCP, BIC TCP, TCP-Illinois and FAST TCP in this scenario, thus here we mainly focus on the performance of HCC TCP. Figure 11 illustrates the rate dynamics of HCC TCP with buffer size of 600 pkts. The results in Fig. 11(a) show that when two homogeneous RTT users of HCC TCP coexist in the scenario, the two flows converge to a fair share of the bottleneck link after 80 seconds. On the other hand, when two heterogeneous RTT users of HCC TCP compete for the link bandwidth, as shown in Fig. 11(b), the flow with shorter RTT achieves a higher throughput than the other one. The simulation results in Fig. 11 present the features of loss-based strategy in HCC TCP.

We also run the simulations in which three HCC TCP users with heterogeneous RTTs share the bottleneck link. The three sources start sending data at 0, 100 and 200 s, respectively and terminate at 300 s. The buffer size is set to be 2000 pkts which means the buffer size is above the number of packets that need to be maintained at routers by HCC TCP. The results in Fig. 12 show that the three flows rapidly converge to a fair share of the bottleneck link, thus HCC TCP achieves better RTT-fairness than the other loss-based protocols. These simulation results present the features of delay-based strategy in HCC TCP. In addition, we increase the link bandwidth to 1 Gbps. In this environment, the buffer size is large enough for the three users and the parameter α of HCC TCP is set as 2000. The simulation results in Fig. 13 show that when the link bandwidth increases to 1 Gbps, HCC TCP can still achieve an efficient performance with proper parameter

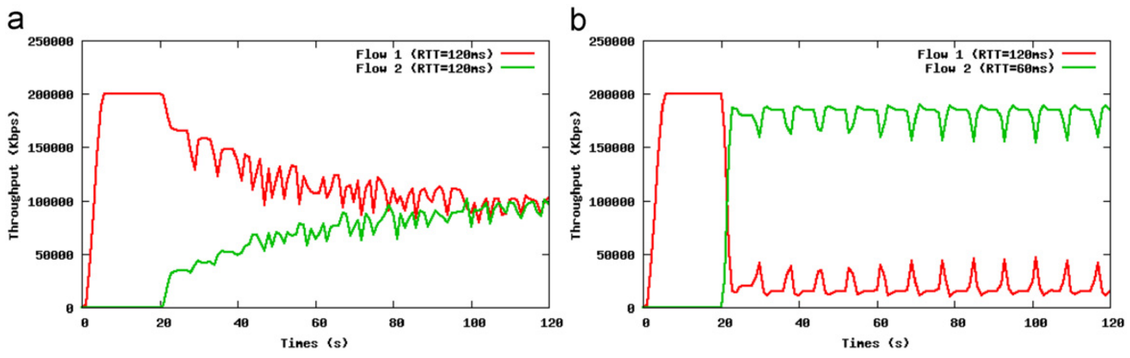

Fig. 11. Rate dynamics of HCC TCP with buffer size of 600 pkts. (a) Rate dynamics of 2 homogeneous RTT users and (b) rate dynamics of 2 heterogeneous RTT users.

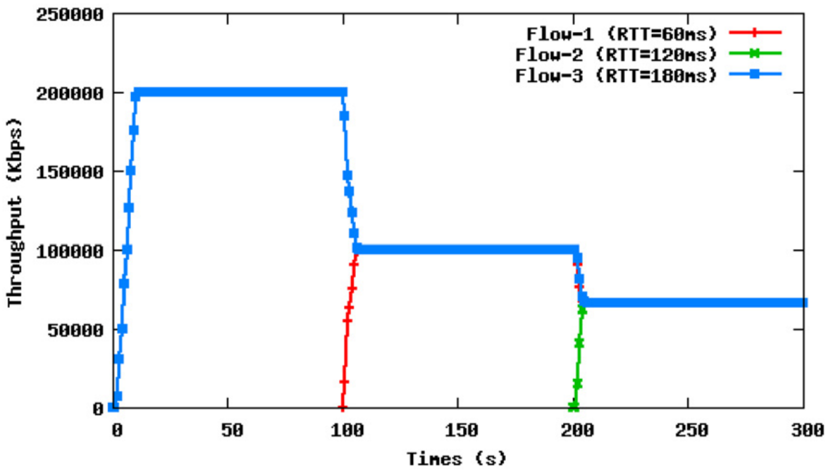

Fig. 12. Rate dynamics of HCC TCP with buffer size of 2000 pkts.

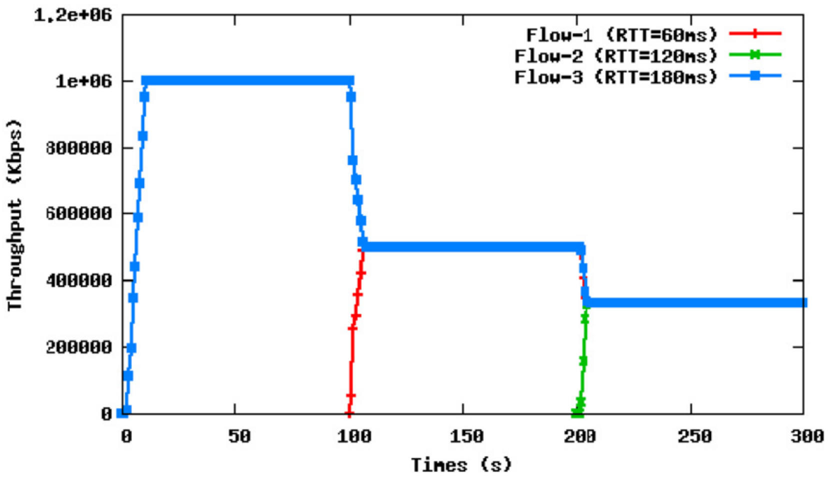

Fig. 13. Rate dynamics of HCC TCP in a 1 Gbps bandwidth environment.

settings. The three HCC TCP users not only rapidly converge to a fair share of the 1 Gbps bottleneck link, but also maintain perfect RTT-fairness amongst the users.

Overall, HCC TCP shows a better fairness performance among the high-speed TCP variants.

4.3. *TCP-friendliness*

For the evaluation of TCP-friendliness performance, we run the simulations that two sources are specified to run TCP Reno while two sources implement the high-speed TCP variants in a homogeneous RTT scenario. Figure 14 shows the average throughput of the four flows with different buffer sizes in which the number 1 and 2 denote the flows using high-speed TCP variants and the number 3 and 4 denote the flows using TCP Reno. From the results, it can be seen that even the buffer size changes, the loss-based protocols, e.g. HSTCP, STCP, HTCP and BIC TCP, always perform unfair and significantly reduce the average throughput of the TCP Reno flows. TCP-Illinois performs better than the loss-based protocols. However, as the buffer size grows, the TCP Reno flows achieve higher throughput than the TCP-Illinois flows. For the delay-based protocol FAST TCP and the

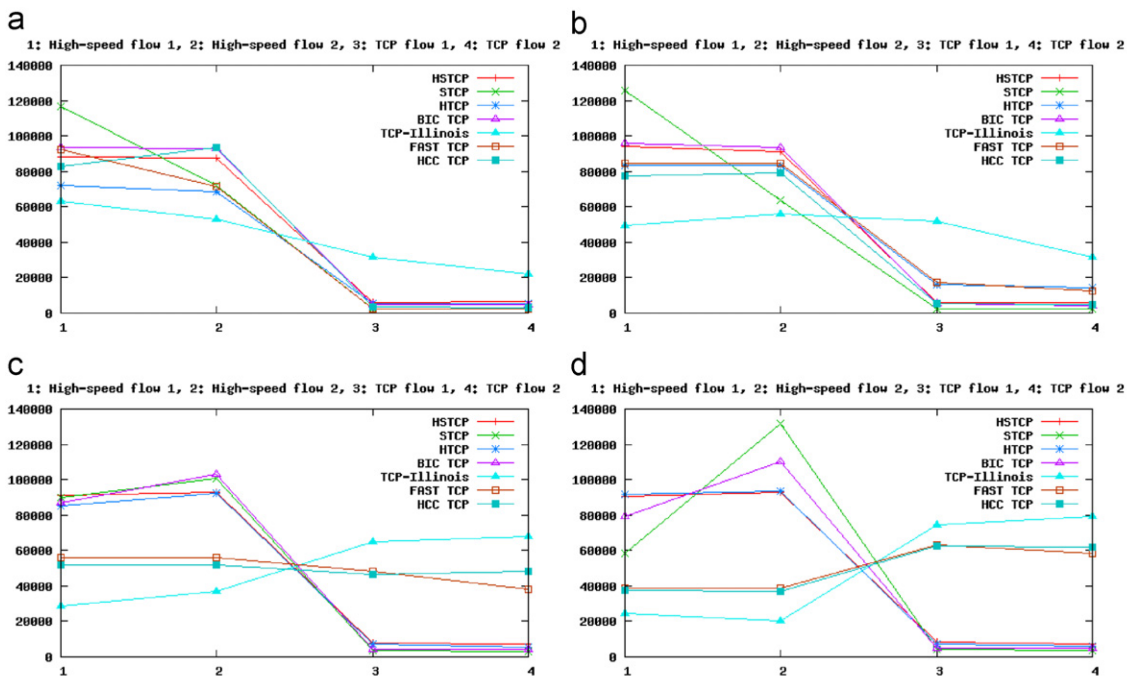

Fig. 14. Average throughput of four flows, two use high-speed TCP variants, two use TCP Reno. (a) Buffer size is 600 pkts, (b) buffer size is 1000 pkts, (c) buffer size is 2000 pkts and (d) buffer size is 3000 pkts.

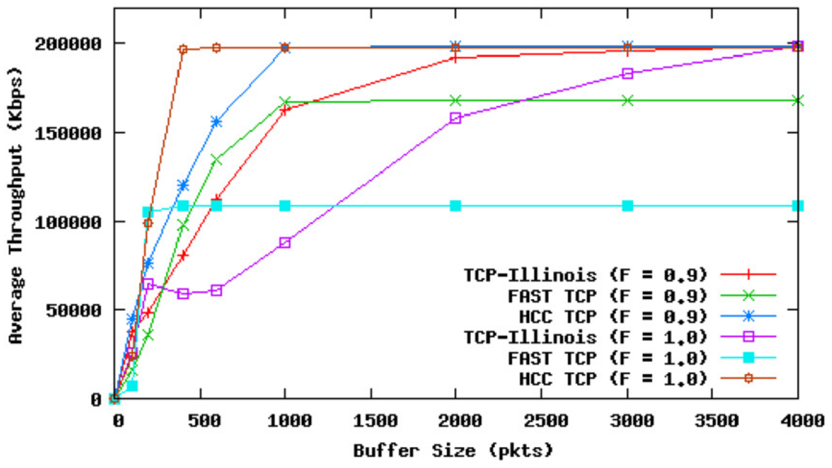

Fig. 15. Average throughput versus buffer size in presence of reverse traffic.

hybrid protocol HCC TCP, if the buffer size is less than the number of packets that need to be maintained at routers, their performance is similar to the loss-based protocols. However, as the buffer size increases, FAST TCP and HCC TCP achieve better friendliness to the TCP Reno flows. We also note that when the buffer size is 2000 pkts, the four flows can even fairly share the bottleneck link. This is because HCC TCP uses the loss-based strategy for window size control when buffer size is less than the specified packet number, and thus lead to its TCP-friendliness performance is similar to other loss-based high-speed protocols. As the buffer size grows larger than the specified packet number, HCC TCP sources turn to the delay-based strategy and always maintain the predefined number of packets at routers, and the redundant buffer is available for the TCP flows. Therefore, the TCP Reno flows could share the bandwidth with the HCC TCP flows. The more available buffer is provided, the higher bandwidth utilization can be achieved by the TCP Reno flows. Obviously, the simulation results demonstrate that HCC TCP does not always suppress the concomitant TCP Reno flows and achieves better TCP-friendliness performance than the loss-based high-speed TCP variants.

4.4. Robustness

The robustness performance of HCC TCP is studied in this subsection. It is well known that the delay-based approaches using RTTs as the congestion measures are significantly affected by the reverse traffic, and as the congestion on the reverse path increases, the throughput of the flows degrades although there is available bandwidth on the forward path. Hence, we perform the simulations to compare the robustness performance of TCP-Illinois, FAST TCP and HCC TCP which utilize the delay-based strategy for congestion control. As suggested in Chan et al. (2004), we adopt a variable-bit-rate (VBR) source which sends data according to Exponential ON–OFF distribution to generate traffic and cause congestion on the reverse path. The buffer size is 2000 pkts for each router along the reverse path, and the size of reverse packet is 500 bytes.

Figure 15 presents the average throughput of the protocols for different buffer sizes when the ON–OFF settings are 0.9 and 1.0, respectively. We run each test in the simulations for 300 s. Compare to the results shown in Fig. 6, it is observed that the throughput performance of TCP-Illinois and HCC TCP is not significantly affected by the reverse traffic. When the buffer size becomes large enough, TCP-Illinois and HCC TCP are able to achieve almost full bandwidth utilization. However, the average

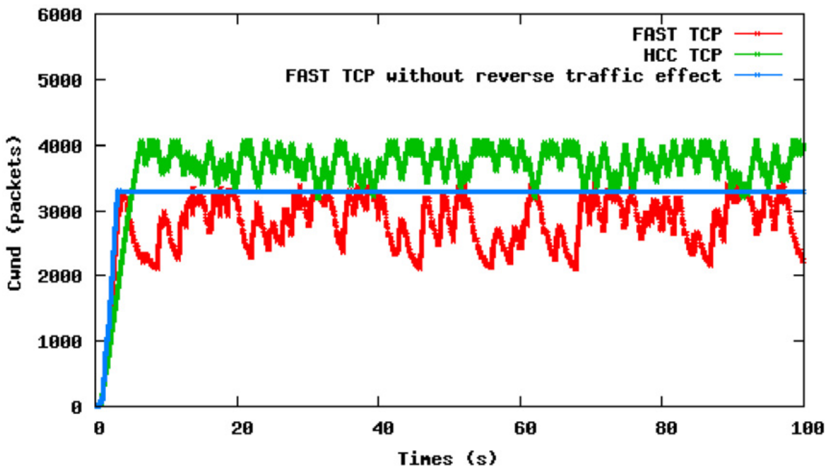

Fig. 16. Window curves comparison.

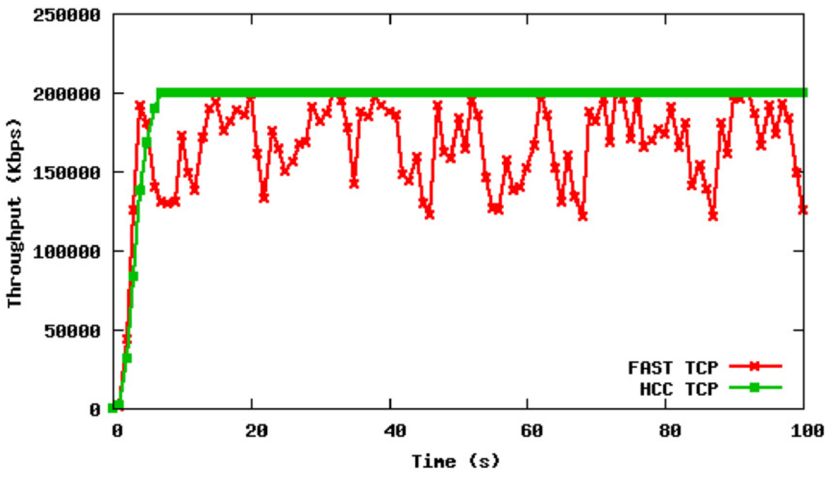

Fig. 17. Rate dynamics in presence of reverse traffic.

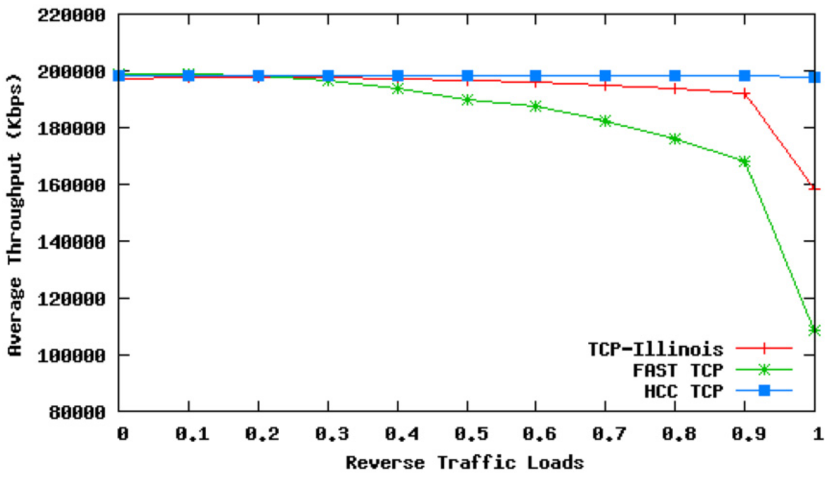

Fig. 18. Average throughput versus varying reverse traffic loads.

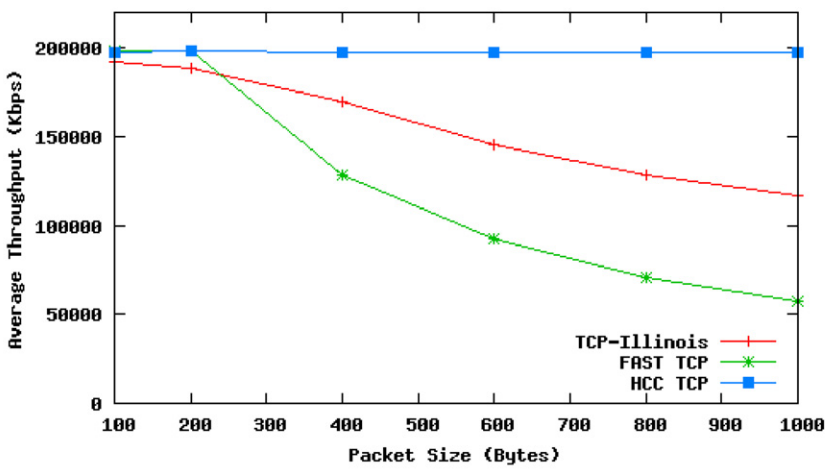

Fig. 19. Average throughput versus varying reverse packet sizes.

throughput of FAST TCP is affected with the reverse traffic and decreases dramatically. This is mainly because FAST TCP uses the round trip queuing delay as the congestion measure, and the reverse queuing delay could result in the inaccurate measurement of FAST TCP for the congestion on the forward path so that lead to the performance degradation of throughput. TCP-Illinois utilizes packet loss information as the primary indicator for congestion and the queuing delay information is utilized for adjusting the window size. This mechanism enables TCP-Illinois avoid the significant effect from the reverse traffic. HCC TCP uses the forward one-way delay as the congestion measure, thus the congestion on the forward path can be exactly measured without the effect of the reverse traffic.

The window curves of FAST TCP and HCC TCP in present of reverse traffic as well as FAST TCP without reverse traffic effect is shown in Fig. 16. It can be seen that when the effect from the reverse traffic is not significant, FAST TCP finally stabilizes at a fix window size so that achieves full bandwidth utilization along the forward path. However, when the reverse traffic causes significant queuing delay on the reverse path, the window size of FAST TCP decreases dramatically and thus result in the reduction of throughput on the forward path. Due to the congestion on the forward path can be exactly measured by the one-way delay estimation mechanisms, HCC TCP performs higher window size to conquer the reverse traffic effect and achieves better bandwidth utilization. The rate dynamics of FAST TCP and HCC TCP in presence of reverse traffic is illustrated in Fig. 17.

The average throughput of TCP-Illinois, FAST TCP and HCC TCP with different reverse traffic loads is also studied. The traffic source sending data on the forward path and the VBR source generating congestion on the reverse path start at 0 s and terminate at 300 s. The buffer size is 2000 pkts for either the forward path or the reverse path. The ON–OFF setting of the VBR source varies from zero to one accordingly. As the simulation results shown in Fig. 18, we note that TCP-Illinois and HCC TCP achieve better performance than FAST TCP on average throughput, especially when the reverse traffic load is heavy. As the reverse traffic load increases, the average throughput of TCP-Illinois decreases slightly. However, HCC TCP always maintains full bandwidth utilization for the different reverse traffic loads. Figure 19 presents the average throughput of TCP-Illinois, FAST TCP and HCC TCP with different sizes of reverse packet when the ON–OFF setting is one. From the simulation results, it can be seen that the average throughput of both TCP-Illinois and FAST TCP decreases as the reverse packet size increases, and TCP-Illinois performs better than FAST TCP. On the other hand, HCC TCP always achieves the full utilization of bandwidth with different reverse packet sizes.

Obviously, the obtained results demonstrate the robustness of HCC TCP in the presence of reverse traffic.

5. Conclusion

In this paper, a novel congestion control algorithm using the synergy of delay-based and loss-based strategies is presented for performance enhancement of data transfer in high-speed networks. HCC TCP uses queuing delay as the primary congestion indicator, and packet loss is used as the second congestion indicator. The delay-based strategy relies on the one-way delay measurements to estimate the congestion along the forward path, calculates the anticipated window size which can achieve full bandwidth utilization using the delay information, and adjusts the window that finally stabilizes around the anticipated value. On the other hand, the loss-based strategy using a linear to logarithmic increase function is adopted for the window control when the delay-based strategy performs inefficiently in networks and the loss event is detected. The hybrid solution inherits the advantages from the delay-based and loss-based congestion control strategies so as to achieve efficient performance in high-speed networks.

Extensive simulation experiments are conducted to understand the performance of HCC TCP. Compare to the existing high-speed TCP variants, HCC TCP can rapidly converge to a high utilization of link bandwidth and always achieve higher average throughput than other high-speed TCP variants with different router buffer sizes. For the fairness, HCC TCP not only performs well in a homogeneous RTT scenario, but also maintains a fair share of bottleneck link bandwidth in a heterogeneous RTT scenario. Moreover, HCC TCP performs better than other high-speed TCP variants, especially on the heterogeneous RTT-fairness for the ones using loss-based congestion control. HCC TCP is able to achieve better performance on TCP-friendly than most of other high-speed TCP variants, and does not suppress the throughput of the standard TCP in proper cases. Finally, for the robustness, HCC TCP can still perform well in the presence of significant reverse traffic and efficiently remedy the effect of reverse traffic for the delay-based congestion control. Overall, the experiment results demonstrate HCC TCP satisfies the requirements for an ideal solution to high-speed TCP protocol and achieves better performance on throughput, fairness, TCP-friendliness, robustness, etc. than most of the existing high-speed TCP variants.

Direction for future research is to implement the HCC TCP protocol in a testbed and evaluate its performance in a real environment.

Acknowledgments

This work is supported by the Major International (Regional) Joint Research Program of the National Science Foundation of China under Grant no. 50620130441, the National Science Foundation of China under Grant no. 50905133, the International Science and Technology Collaboration Project from Ministry of Science and Technology of China under Grant no. 2006DFA73180, the self-determined and innovative research funds of WUT under Grant no. 2010-JL-19 and the China Scholarship Council Post-graduate Scholarship Program. We are grateful to the reviewers and editors for their comments and works that help us to enhance the quality of this paper.

References

Alonso SH, Perez MR, Gonzalez AS, Veiga MF, Garcia CL. Improving TCP Vegas fairness in presence of backward traffic. IEEE Communications Letters 2007;11(3):273–5.

Brakmo LS, Peterson LL. TCP Vegas: end to end congestion avoidance on a global internet. IEEE Journal on Selected Areas in Communications 1995;13(8):1465–80.

CERN. 〈http://public.web.cern.ch/public/〉; 2010.

Chan YC, Chan CT, Chen YC. Design and performance evaluation of an improved TCP congestion avoidance scheme. IEE Proceedings: Communications 2004;151(1):107–11.

Cui T, Andrew L. FAST TCP simulator module for ns-2. Version 1.1. 〈http://www.cubinlab.ee.mu.oz.au/ns2fasttcp〉; 2007.

Cui T, Andrew L, Zukerman M, Tan L. Improving the fairness of FAST TCP to new flows. IEEE Communications Letters 2006;10(5):414–6.

Floyd S. HighSpeed TCP for large congestion windows. RFC editor; 2003.

Fu CP, Liew SC. A remedy for performance degradation of TCP Vegas in asymmetric networks. IEEE Communications Letters 2003;7(1):42–4.

Gu Y, Grossman RL. UDT: UDP-based data transfer for high-speed wide area networks. Computer Networks 2007;51(7):1777–99.

Ha S, Rhee I, Xu L. CUBIC: a new TCP-friendly high-speed TCP variant. Operating Systems Review (ACM) 2008;42:64–74.

Jain R, Chiu D, Hawe W. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301 1984.

Katabi D, Handley M, Rohrs C. Congestion control for high bandwidth-delay product networks. Computer Communication Review 2002;32:89–102.

Kelly T. Scalable TCP: improving performance in highspeed wide area networks. Computer Communication Review 2003;33(2):83–91.

Kuzmanovic A, Knightly EW. TCP-LP: a distributed algorithm for low priority data transfer. In: Proceedings of IEEE INFOCOM; 2003.

Leith DJ, Shorten RN. H-TCP protocol for high-speed long-distance networks. In: Proceedings of second workshop protocols fast long distance networks; 2004.

Li YT, Leith D, Shorten RN. Experimental evaluation of TCP protocols for high-speed networks. IEEE/ACM Transactions on Networking 2007;15(5): 1109–22.

Liu S, Basar T, Srikant R. TCP-Illinois: a loss- and delay-based congestion control algorithm for high-speed networks. Performance Evaluation 2008;65(6–7):417–40.

Low SH, Peterson LL, Wang L. Understanding TCP vegas: a duality model. Journal of the ACM 2002;49(2):207–35.

NEC-Labs. An NS2 TCP evaluation tool. 〈http://labs.nec.com.cn/tcpeval.htm〉; 2007.

NetherLight. 〈http://www.science.uva.nl/research/air/projects/optical/〉; 2010.

Parsa C, Garcia-Luna-Aceves JJ. Improving TCP congestion control over internets with heterogeneous transmission media. In: Proceedings of international conference on network protocols; 1999.

StarLight. 〈http://www.startap.net/starlight/〉; 2010.

Tan K, Song J, Zhang Q, Sridharan M. A compound TCP approach for high-speed and long distance networks. In: Proceedings of IEEE INFOCOM; 2006.

Tan L, Yuan C, Zukerman M, FAST TCP. Fairness and queuing issues. IEEE Communications Letters 2005;9(8):762–4.

UKLight. 〈http://www.ja.net/index.html〉; 2010.

USC/ISI. The NS simulator. 〈http://www.isi.edu/nsnam/ns/〉; 2010.

Wei DX, Cao P. A Linux TCP. Implementation for NS2 2007. 〈http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/index.html〉.

Wei DX, Jin C, Low SH, Hegde S, FAST TCP. Motivation, architecture, algorithms, performance. IEEE/ACM Transactions on Networking 2006;14(6):1246–59.

Xia Y, Subramanian L, Stoica I, Kalyanaraman S. One more bit is enough. Computer Communication Review 2005;35:37–48.

Xu L, Harfoush K, Rhee I. Binary increase congestion control (BIC) for fast long-distance networks. In: Proceedings of IEEE INFOCOM; 2004.

Xu W, Zhou Z, Pham DT, Ji C, Yang M, Liu Q. Unreliable transport protocol using congestion control for high-speed networks. Journal of Systems and Software 2010;83(12):2642–52.