HFCC: An Adaptive Congestion Control Algorithm based on Explicit Hybrid Feedbacks

Xianliang Jiang, Guang Jin and Haiming Chen
Faculty of Electrical Engineering and Computer Science
Ningbo University, Ningbo, China
{jiangxianliang, jinguang, chenhaiming}@nbu.edu.cn

Abstract*—The high-throughput, low-latency, and reliable data delivery are fundamental demands of many networked applications, e.g. BitTorrent and Skype. But the inappropriate congestion control of TCPs, caused by the reactive and coarse-grained congestion feedbacks, brings the low link utilization, high queuing delay and frequent packet loss in high bandwidth-delay product network. To mitigate this issue, TCP variants have been developed. Thereinto, the load factor based congestion control (LFCC), e.g. VCP, BMCC, have shown the powerful capabilities to achieve better performances in terms of high link utilization, low persistent queue length, negligible packet loss, and fairness. However, due to the conservative increase and synchronized feedbacks, LFCC faces the slow convergence of the link utilization and inter-flow fairness. This could incur the large flow completion time of new-coming flows indirectly. To solve the issue of existing LFCCs, an asynchronous congestion control based on hybrid feedbacks, called HFCC, is proposed to achieve the faster convergence while keeping the features of LFCCs in this paper. Specifically, HFCC decreases the congestion window when the bottleneck link is in the high-load region and the flow rate exceeds the fair share of the bottleneck bandwidth, or the bottleneck link is in overload region. Otherwise, HFCC increases the congestion window. Note that an overlay coding method is developed in HFCC. To reduce the flow completion time, HFCC adopts an available bandwidth estimation method to speed up the data delivery in low-load region. The simulation results indicate that HFCC has the better performance and faster convergence than VCP, MLCP, and BMCC.

***Index Terms*—congestion control, hybrid feedbacks, convergence, fairness**

I. Introduction

Most networked applications adopt the transmission control protocol (TCP) [1], dominated by additive increase and multiplicative decrease (AIMD) [2], as underlying service to exchange vital data. But with the advancement of communicating techniques, e.g. optical fiber communication, high-speed wireless networks, and the prevalence of high bandwidth delay product (HBDP) networks, traditional TCPs are ill-suited. They could result in the low link utilization, high packet loss rate, unacceptable packet delay, *etc*. Hence, how to develop a high-efficiency and robust congestion control protocol gains in importance to improve the performance of TCPs in HBDP networks.

Although TCP variants have been proposed to solve above issue, they usually face a dilemma between the efficiency and applicability. Generally, the existing protocols can be classified into two categories: *end-to-end* and *network-based*. Each category has their own merits and deficiencies.

The *end-to-end* congestion control (ECC) [3] rely on implicit feedbacks, e.g. packet loss [1] [4], end-to-end delay [5] [6], and synthesis of them [7] [8], to discern the network congestion. They can be easily deployed without modifying the Internet architecture. But, the packet loss and queuing delay emerging when the network is congested, ECC protocols could lead to the unnecessary packet losses and high packets delay in HBDP networks. Conversely, the *network-based* congestion control (NCC) utilize explicit feedbacks [9] [10], which are generated by intermediate routers and returned to senders from receivers by modifying the IP header, to adjust the congestion window ($cwnd$). They can provide more information of the network congestion than ECCs. However, NCCs face two deployed issues in the current Internet. On one hand, they demand more bits for congestion feedbacks than available in the IP header, which requires either changing the IP header or the addition of a *shim* layer. On the other hand, the intricate mechanisms in intermediate routers violate the inter-flow fairness when the traditional TCP flows arise. Therefore, the *limited feedback* schemes, e.g. VCP [11], MLCP [12], and BMCC [13], were proposed. They require changes at end-hosts with incremental support from routers.

This paper investigates the convergence speed of multiplicative increase (MI)-AIMD mechanisms in existing load factor based congestion control (LFCC) protocols, which adopt the explicit congestion notification (ECN) [14] [15] to convey the **L**oad **F**actor (LF). To this end, a novel protocol, named hybrid feedback-based congestion control (HFCC), is proposed. It adopts asynchronous feedbacks for different flows and is compatible with the existing ECN. In addition, we present the experimental analysis that provide insights into the convergence properties and performance of HFCC. Different from VCP, HFCC rely on the overlay coding of the fair-**S**hare **F**actor (SF) and LF to convey the network condition. This could result in the better convergence and the fairer bandwidth allocation in HBDP networks while keeping the low packet delay and negligible packet loss rate. Furthermore, HFCC

This work was supported by the National Natural Science Foundation of China (No.61601252), Public Technology Projects of Zhejiang Province(GG18F020020), Zhejiang Provincial Natural Science Foundation of China (LY18F020011), Ningbo Natural Science Foundation (2017A610129/2017A610116), Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications (SKLNST-2016-2-13).

outperforms VCP and is slightly better than other schemes in the flow completion time (FCT).

The rest of this paper is organized as follows. We summarize related works in section II. In section III, we provide the motivation of HFCC. In section IV, the HFCC protocol is designed based on the overlay coding of SF and LF. To evaluate the HFCC protocol, we conduct extensive experiments using a canonical network simulator (NS2) in section V. Finally, the conclusion is presented in section VI.

II. Related Work

In recent decades, TCP variants have been largely proposed to enhance the standard TCP in HBDP networks. They focus on how to control $cwnd$ effectively. Actually, the detection of network congestion is a key point to adjust the congestion window. Hence, how to convey the feedbacks of network congestion is one of the most important thing for congestion control in HBDP networks. In section I, we have mentioned that TCP variants can be roughly classified into ECC and NCC.

The ECCs rely on the packet loss or delay to detect the network congestion and can be divided into the packet loss-based congestion control (PLCC), the packet delay-based congestion control (PDCC) and the synergy of both of them. To improve the performance of PLCCs in HBDP networks, most TCP variants, such as HSTCP [16], STCP [17], and CUBIC [4], modify the AIMD parameters and the controlling function of $cwnd$ to achieve the high link utilization rapidly. However, the aggressiveness to adjust the congestion window causes the throughput oscillation of TCP flows.

Different from PLCCs, PDCCs assume that the increase in RTTs indicates the coming of the network congestion and attempt to proactively adjust $cwnd$ based on the variation of RTTs. In [18], Jain *et al.* proposes the first PDCC. After that, Vegas [5] was proposed to improve Reno's throughput. To adapt to HBDP networks, FAST [6] was proposed to grab the network bandwidth rapidly. However, the inaccuracy of the RTT measurement often makes PDCCs inefficient. Moreover, PDCCs also suffer from significant low throughput if the competing flows are PLCCs, e.g. Reno. Synthesizing the advantages of PLCCs and PDCCs, the synergy schemes, such as CTCP [7] and ACCF [8], were proposed. To reduce the retransmission upon packets loss, Cui *et al.* [19] believes that the coding can improve TCPs. Broadly speaking, the weakness of ECCs is that actions are taken only after the network congestion is detected. And the aggressive nature of them results in unwanted packets loss and queuing at bottleneck routers. Existing studies also show that *end-to-end* feedbacks are difficult to achieve the high link utilization and fairness while maintaining the low bottleneck queue size and the near-zero packet loss rate. Hence, the router-assisted feedbacks are required to break through the limitations of ECCs.

The NCCs utilize the link load [11] [12] [13], fair share rate [10] [20], or changes in window size [9] to adjust the congestion window and can be subdivided into load-based congestion control (LCC) and rate-based congestion control (RCC). The advantages include that they can rapidly achieve the efficient (e.g. full link utilization, fast convergence, stability) and fair bandwidth allocation among different flows in HBDP networks. In terms of RCC schemes, Falk *et al.* [9] generalized the ECN proposal and introduced the concept of decoupling the utilization control and the fairness control. To achieve the fair share of bottleneck bandwidth and finish flows quickly, Dukkipati [10] proposed a faster congestion control scheme based on the fair share rate. As mentioned in section 1, two deployment problems make RCC schemes difficult to be used on current Internet.

Differ from RCC schemes, LCC schemes measure the link load and use the ECN fields [14] [15] in the IP header to convey the encoded link load. The advantage of LCC is that it doesn't need to modify the IP header and is compatible with the current network architecture. TCP+RED/ECN [21] [22] uses two ECN bits to notify senders whether the network is congested or not. It makes TCP able to reduce the packet loss rate and the queue length. However, due to the binary indicator, senders cut $cwnd$ by half, which is too aggressive in HBDP networks, when a ECN-marked ACK packet arrives. In [11], Xia *et al.* proposed a simple, low-complexity protocol which leverages only the existing ECN bits for network congestion feedback, and yet achieves comparable performance (e.g. high utilization, negligible packet loss, low persistent queue length, and reasonable fairness) to XCP [9]. However, due to up to two bits resolution, the convergence speed of VCP is significantly slower than that of XCP. In [12], Qazi *et al.* uses experimental and theoretical analyses to prove that the convergence speed improves significantly when the load is estimated to 4 bits resolution. But the IP header of a single packet has insufficient space. Hence, the Adaptive Deterministic Packet Marking (ADPM) [23] was adopted to obtain 16 bits high-resolution congestion estimation in [13]. Recently, Wang *et al.* [24] proposed a bandwidth estimation based VCP scheme that can provide more accurate network state information by estimating available bandwidth with an adaptive bandwidth estimator.

III. Motivation

In LFCC, VCP is a seminal scheme and inherits from the standard TCP (Reno). Specifically, a VCP router measures and encodes the LF (ρ_l) values of its output links periodically (t_ρ). When a packet leaves the VCP router, the encoded LF value is inserted into ECN fields of the packet's IP header. A VCP receiver is similar to the standard TCP receiver except the transfer of ECN values from data packets to acknowledgement (ACK) packets. Using the LF value, a VCP sender can employ different strategies to control $cwnd$ as listed in Table I. Roughly, the VCP sender inflates $cwnd$ using MI and AI strategies when the link load is in the low-load and high-load regions, respectively. In the over-load region, the VCP sender deflates $cwnd$ using the MD strategy immediately and then freezes $cwnd$ for one t_ρ (200ms in the implementation of VCP) regardless of the remaining feedbacks.

TABLE I
ECN MARKING AND CONGESTION WINDOW ADJUSTMENT OF VCP

Load Region	ECN	Action
Low-load ($0 \leq \rho_l < 0.8$)	$(01)_2$	MI: $cwnd \leftarrow cwnd \cdot (1+\alpha)$
High-load ($0.8 \leq \rho_l < 1.0$)	$(10)_2$	AI: $cwnd \leftarrow cwnd + \beta$
Overload ($\rho_l \geq 1.0$)	$(11)_2$	MD: $cwnd \leftarrow cwnd \cdot \gamma$

Using the following equation, the VCP router can estimate ρ_l [11]

$$\rho_l = \frac{\lambda_l + \kappa_q \cdot \bar{q}_l}{\xi_l \cdot C_l \cdot t_\rho}, \tag{1}$$

where λ_l and $\bar{q}_l$ are the total input traffic (using a packet counter) and the average queue length of VCP router observed in t_ρ, respectively. κ_q is a coefficient to control the draining speed of the queue backlog and is fixed to 0.5 in [11]. The larger κ_q means the faster speed to drain the persistent queue backlog. ξ_l denotes the target link utilization (e.g. 0.98), and C_l denotes the link capacity. To measure $\bar{q}_l$, VCP adopts a low-pass filter to sample the instantaneous queue length, *q(t)*, every t_q (e.g. 10ms). To eliminate the RTT unfairness, VCP scales α and β according to the current RTT, rtt, as

$$\alpha_s = (1+\alpha)^{\frac{rtt}{t_\rho}} - 1, \tag{2}$$

$$\beta_s = \beta \cdot (\frac{rtt}{t_\rho})^2. \tag{3}$$

Problem 1: In the low-load region, VCP adopts a conservative MI factor (α=0.0625), which is even lower than the slow start (SS) of standard TCP, to adjust $cwnd$. This results in the slow increasing of $cwnd$ to achieve the high link utilization. The reason is that, to avoid the overflow of the bottleneck queue, VCP senders always regard ρ_l as 0.8 when the ECN value is $(01)_2$ while the real ρ_l varies in [0, 0.8). Suppose that the bandwidth of the bottleneck link is 100Mbps and the round-trip propagation delay (RTPD) is 80 ms, a VCP flow needs 21s or so to achieve the high link utilization as shown in Fig. 1. The slow convergence speed is also the root cause of the long FCT of short flows.

Problem 2: To achieve the intra-flows fairness, VCP adopts the modified AIMD mechanism to adjust $cwnd$ according to the synchronized feedbacks (the identical feedback for different flows). Balancing the responsiveness and the overall link utilization, VCP fixes the MD factor to 0.875. This setting makes new flows having less chances to achieve the fair bandwidth allocation quickly in the high-load or overload regions. It also results in the long FCT of later-coming flows indirectly. Fig. 1 shows that the second flow starts at 50s after the first flow has entered its steady state ($\rho_l > 0.8$). Until 225s or so, the two flows get intra-flows fairness. In other words, the convergence speed in obtaining fairness is very slow.

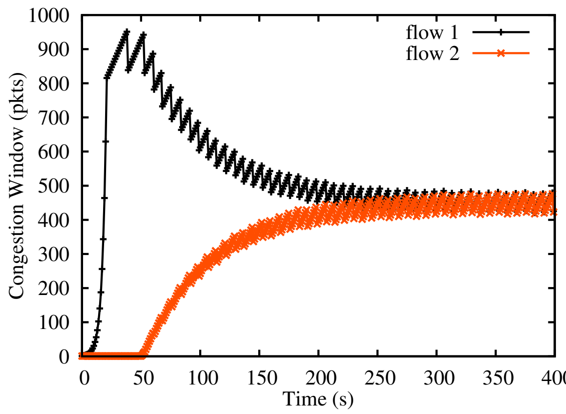


Fig. 1. The evolution of the congestion window of VCP flows

TABLE II
ECN MARKING AND CONGESTION WINDOW ADJUSTMENT OF HFCC

Load Region	Flow Rate	ECN	Action
Low-load($0 \leq \rho_l < 0.8$)	-	$(01)_2$	RI
High-load($0.8 \leq \rho_l < U_{th}$)	-	$(10)_2$	A^{2}I
High-load($U_{th} \leq \rho_l < 1.0$)	$r_i \leq r_f$	$(10)_2$	A^{2}I
High-load($U_{th} \leq \rho_l < 1.0$)	$r_i > r_f$	$(11)_2$	AMD
Overload($\rho_l \geq 1.0$)	-	$(11)_2$	AMD

To address these issues, Qazi *et al.* proposed two enhanced protocols in [12] [13] using high-resolution feedbacks. However, the feedbacks of them are probabilistic and can not reflect the network condition timely. In [24], Wang *et al.* proposed the bandwidth estimation based VCP (VCP-BE). Using the available bandwidth to judge the network state, VCP-BE can adjust $cwnd$ more precisely than VCP. But the inaccuracy of bandwidth estimation may cause the throughput fluctuation of VCP-BE flows. To improve the performance while maintaining the applicability of VCP, we need a novel mechanism which generates different congestion feedbacks according to the network condition and the partial flow states. Hence, a congestion control scheme, named HFCC, is proposed based on the overlay coding of SF and LF to achieve above goals.

IV. HFCC Protocol

In this section, we provide an overview of HFCC firstly. And then we introduce the estimation of the link load, flow rate, and available bandwidth. After that, the implementation of HFCC is presented. Finally, several key parameters are discussed.

A. HFCC Overview

In HFCC, the sender adjusts $cwnd$ adaptively according to the overlay coding of LF and SF. The detailed information is shown in Table II and explained as follows. In low-load region, HFCC adopts the rapid increase (RI) to adjust $cwnd$, which can achieve the high link utilization rapidly and avoid unwanted packet losses. In high-load region, HFCC adopts the adaptive AI (A^{2}I) and adaptive MD (AMD), which is dominated by the SF value, to accelerate the convergence of intra-flow fairness. In overload region, HFCC uses AMD to reduce $cwnd$ reasonably. Note that HFCC employs a similar method like [25] to adjust the AMD factor, γ. r_i is the rate of i_{th} flow and r_f is the fair-share rate of the bottleneck link. U_{th} is a threshold to balance the efficiency and fairness.

To illustrate the operation of HFCC, a network scenario is provided in Fig. 2. It has three components: HFCC sender,

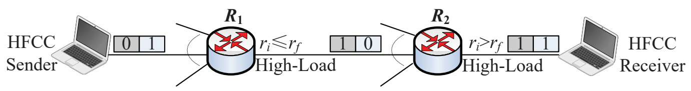


Fig. 2. The operation of HFCC with identical link load and different flows rate

HFCC receiver, and HFCC router. Two ECN bits, indicated by different blocks, are adopted to convey the overlay coding of LF and SF. Therein, the dark-grey block indicates that the bottleneck link is either in low-load region (0) or high-load and overload regions (1). The light-grey block represents whether the flow rate (e.g. r_i) is lower than the fair share (e.g. r_f) of the bottleneck bandwidth (0) or not (1) when the link load is in high-load region. In addition, the light-grey block is always 1 when the link load is in overload region. When packets leave the HFCC sender, their ECN bits in the IP header are initialized to $(01)_2$ ($(00)_2$ for the backward compatibility). Passing the link between the HFCC sender and R_1, packets arrive at R_1 where several related variables are updated. When packets leave R_1, their ECN fields are updated to $(10)_2$ because the link load is in high-load region and $r_i \leq r_f$. However, due to the high-load of the R_2's output link and $r_i > r_f$, the ECN bits of packets are updated to $(11)_2$ when they leave R_2. When packets are accepted by the HFCC receiver, the corresponding ACK packets, whose ECN bits in the TCP header are the same as the accepted data packets, are sent back to the HFCC sender. Thus, the HFCC sender can adjust $cwnd$ according to ECN bits. In addition, the HFCC router can regain LF and SF values agilely when the link capacity, the flows RTT, the queue backlog, and the flow rate change dynamically. The HFCC sender can adjust $cwnd$ and other parameters adaptively.

The key goal of HFCC is to achieve the faster convergence of efficiency and fairness than other LFCCs when the link load is in high-load region. If the link load is in high-load region, high-bandwidth flows must decrease their sending rate positively while low-bandwidth flows can increase their sending rate gracefully. In addition, the high-bandwidth and low-bandwidth flows can increase their sending rate when the link load is in low-load region and must decrease their sending rate when the load of the bottleneck link is in over-load region. To improve the performance of short flows, we adopt the similar method like [26] [24] to estimate the available bandwidth and then adjust $cwnd$ of HFCC flows when the link load is in low-load region.

B. Network State Estimation

To support HFCC, the network state estimation is necessary. Next, we will introduce three methods to estimate the link load, flow rate, and available bandwidth.

1) Link Load Estimation: To realize the low persistent queue length, the high link utilization and intra-flow fairness, HFCC adopts the similar method like [11] to compute the link load. The HFCC router divides time into intervals with length t_ρ (e.g. 200ms) and computes the link load in each interval as equation (1). After that, the HFCC router generates LF based on the computed link load (ρ_l) and updates the dark-grey block in Fig. 2 when packets leave the router. As an extension, we can also adopt some active queue management algorithms [21] to measure LF. Due to space limitations, we leave them for future study and do not discuss them in this paper.

2) Flow Rate Estimation: To estimate the flow rate and compute SF, we adopt the similar method like [27]. Next, we provide the detailed procedure.

In [27], *Pan et al.* proposed an approximate fair dropping (AFD) algorithm to achieve the fair bandwidth allocation among flows. To reduce the spatial complexity, AFD adopts a shadow buffer and a flow table to estimates the flow rate r_i and the fair share rate r_{fair}. Therein, the shadow buffer can be used to sampling incoming packets. The flow table contains the packet count of different flows. r_i is estimated by m_i (the amount of traffic from i_{th} flow during an interval). r_{fair} is estimated dynamically by m_{fair}, which is determined by

$$m_{fair} \leftarrow m_{fair} + \nu \cdot (Q_{pre} - Q_{target}) - \varrho \cdot (Q_{cur} - Q_{target}), \tag{4}$$

where Q_{cur} is the instantaneous queue length in current interval. Q_{pre} is the queue length in previous interval. Q_{target} is the target queue length. ν and ϱ are two configurable coefficients. If $r_i < r_{fair}$, the SF is 0. Otherwise, it equals 1. In implementation, m_i can be calculated through the exponentially weighted moving average method. Therefore, m_f of u flows can be denoted as $\frac{1}{u}\sum_{i=1}^{u} m_i$.

We can also use an approximate method, e.g. $sample$-$match$ in [28] [29], to estimate SF. When a packet leaves the HFCC router, it makes a comparison among the dequeued packet and n randomly selected packets from the router's queue. If they belong to the same flow, the SF equals 1. Otherwise, it equals 0. Therein, n is a user-defined parameter.

3) Available Bandwidth Estimation: Like [26], HFCC estimates the available bandwidth at sender by observing returning ACKs. In [26], the available bandwidth estimator is as follows

$$B_n = \frac{RTT \cdot B_{n-1} + L_n}{(t_n - t_{n-1}) + RTT}, \tag{5}$$

where B_n is the estimated available bandwidth at time t_n when n_{th} ACK arrives. L_n is the acknowledged data size of nth ACK. *RTT* is the estimated round-trip time of TCP packet at time t_n. To balance the responsiveness and accuracy of the bandwidth estimator, a dynamic coefficient, π_n, is introduced to control the length of observation time. Hence, the equation (5) can be rewritten to

$$B_n = \frac{\pi_n \cdot RTT \cdot B_{n-1} + L_n}{\pi_n \cdot RTT + (t_n - t_{n-1})}, \tag{6}$$

where π_n can be calculated by

$$\pi_n = \frac{B_{n-1}}{(cwnd_{n-1} \cdot P_s)/RTT}, \tag{7}$$

where $cwnd_{n-1}$ is previous congestion window. P_s is the packet size. Note that HFCC uses the estimated bandwidth to adjust the congestion window when the bottleneck link is in low-load region.

C. Implementation of HFCC

In section IV-B, we have discussed how to estimate the network state. Next, we explain the implementation of HFCC.

1) Sender: To adjust $cwnd$, the HFCC sender employs the RI-A^{2}I-AMD, which determined by two ECN bits of received ACK packets. To facilitate discussions, suppose that all flows share a single bottleneck link and the RTT of all flows equals t_ρ. Therefore, the control interval synchronizes with the load computation. At any time t, when the ECN values are $(01)_2$, the HFCC sender increases $cwnd$ rapidly as

$$cwnd \leftarrow max\left[cwnd + cwnd \cdot \alpha_s, \frac{RTT_{min} \cdot B_n}{P_s}\right], \quad (8)$$

where α_s is calculated by equation (2). α in α_s is the increased ratio of $cwnd$ in one RTT inherited from the MI stage of [11] and its default value is 0.0625. RTT_{min} is the minimum RTT observed by sender. When the ECN values is $(10)_2$, $cwnd$ is increasing linearly by

$$cwnd \leftarrow cwnd + \beta_s/cwnd, \quad (9)$$

where β_s is calculated by equation (3). β in β_s is the increased value of $cwnd$ in one RTT and its default value is 1.0. When the ECN bits is $(11)_2$, the HFCC sender decrease $cwnd$ prudently by

$$cwnd \leftarrow max(1.0, \gamma \cdot cwnd), \quad (10)$$

where γ is the decreased ratio of $cwnd$ in one RTT. Note that $cwnd$ must be frozen by one RTT after the AMD action. In section IV-D, we will discuss how to set β and γ.

2) Router: The structure of HFCC router is shown in Fig. 3 and has four components. Therein, the Load Computation adopts the same method as [11] to compute LF of the router's output links periodically. The Packet Queue holds packets that can not be timely forwarded. The Rate Estimation uses the similar methods as [27] to estimate SF. As a matter of fact, the flow table is a good choice to compute SF in some novel network architectures, such as software-defined network. The Packet Marking adopts the overlay coding method to encode LF and SF, and then updates two ECN bits in IP header of data packets when they leave the router.

The detailed operation of HFCC router is shown in algorithm 1. Therein, l_f and s_f are LF and SF respectively. ECN means two ECN bits in IP header. Note that, to protect ACK packets, the HFCC router has two priority queues. The high priority queue is for ACK packets and the low priority queue is for other packets. As an alternative, we can adopt the method like [30] to achieve the weighted fairness.

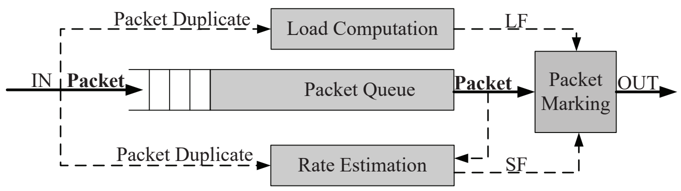


Fig. 3. The structure of a HFCC-enabled router

3) Receiver: The function of the HFCC receiver is similar to the Reno receiver except that two ECN bits must be copied from data packets to ACK packets.

D. Parameters Setting

To strengthen HFCC, we provide the guideline to configure key parameters including t_ρ, β, γ and U_{th} in this section.

Parameter t_ρ: To eliminate the burstiness of congestion responses, HFCC should keep t_ρ greater than most RTTs. Meanwhile, to avoid the queue backlog, HFCC should keep t_ρ as small as possible. In VCP [11], t_ρ is 200ms. But the experimental analysis shows that the fixed t_ρ is ill-suited when RTT is too large or the link bandwidth is too small. To ensure the performance, HFCC adopts a similar method like [12] to update t_ρ. The detailed procedure is as follows.

(1)Suppose that the control interval T=10ms and the RTT estimation in i_{th} packet is rtt_i. The number of packets in T is n_T. Hence, the HFCC router estimates the average RTT, $\overline{RTT_T}$, smoothly in T by

$$\overline{RTT_T} = \frac{1}{n_T}\sum_{i=1}^{n_T} rtt_i. \quad (11)$$

(2)Using $\overline{RTT_T}$ and its previous smoothing estimation $\overline{RTT_d}$, the HFCC router computes the current smoothing gain θ. If $\overline{RTT_T} \geq \overline{RTT_d}$, $\theta = T/\overline{RTT_d}$. Otherwise, $\theta = T \cdot \overline{RTT_T}/(\phi \cdot \overline{RTT_d})$. In [12], $\phi = 50$.

(3)According to θ in (2), the HFCC router computes the new $\overline{RTT_d}$ and then updates t_ρ using the new $\overline{RTT_d}$. Specifically, $\overline{RTT_d}$ should be firstly calculated by

$$\overline{RTT_d} = \overline{RTT_d} + \theta \cdot (\overline{RTT_T} - \overline{RTT_d}). \quad (12)$$

If $\overline{RTT_d} < 1400$, $t_\rho = min_{\forall i \in |S|}\left\{s_i | s_i \in S, s_i \geq \overline{RTT_d}\right\}$. Otherwise, $t_\rho = 1400$. Therein, S={80, 200, 400, 600, 800, 1000, 1200, 1400}. Note that when RTT lies in [2.0, 2.5] times of t_ρ, the queue backlog is avoidable.

Algorithm 1 Overlay Coding of LF and SF

```
1: compute $l_f$ using equation (1);
2: estimate $s_f$ like section IV-B2;
3: if $l_f$ not in [0.8, 1.0] then
4:     if $l_f$ in [0, 0.8] then
5:         $ecn = (01)_2$;
6:     else
7:         $ecn = (11)_2$;
8:     end if
9: else
10:    if $s_f$ equals to 1 then
11:        $ecn = (11)_2$;
12:    else
13:        $ecn = (10)_2$;
14:    end if
15: end if
16: if $ECN < ecn$ then
17:     $ECN = ecn$;
18: end if
```

Parameter β: Intuitively, more residual bandwidth needs a larger β. To avoid unnecessary packets loss, β should be limited to a fixed range. To identify the range, we conduct different experiments with different β values. The results show that HFCC can obtain better performance when β varies in [1, 5]. Using the estimated available bandwidth, HFCC can tune the β value adaptively as follows.

$$\beta = max\left[1, \frac{B_n - B_{min}}{B_{max} - B_{min}} \cdot \beta_{max}\right], \tag{13}$$

where B_n is n_{th} estimated available bandwidth. B_{max} and B_{min} are the global maximum and minimum values of B_n. β_{max} is an upper bound of β. As shown in equation (13), the larger available bandwidth means a larger β. It can accelerate the convergence speed of HFCC.

Parameter γ: To balance the convergence speed of the fairness and link utilization. HFCC uses the similar method like [25] to estimate the congestion degree (c_l) of the bottleneck link and then tune γ as follows.

$$\gamma = \gamma_{max} - c_l \cdot (\gamma_{max} - \gamma_{min}), \tag{14}$$

where c_l is in [0, 1]. γ_{min} and γ_{max} are the upper bound and lower bound of γ. In this paper, $\gamma_{min} = 0.675$ and $\gamma_{max} = 0.875$. Actually, equation (14) has two merits. Firstly, the frequent overload indicates that the network congestion is forthcoming. To avoid the network congestion, $cwnd$ should be reduced (large γ). Secondly, $cwnd$ of high-bandwidth flows (large γ) should be reduced. More available bandwidth should be provided to low-bandwidth flows (small γ). The experimental results show that the adaptive γ can improve the performance of HFCC. Due to the space limitation, the results are not listed here.

Parameter U_{th}: In HFCC, U_{th} is adopted when the link load is in high-load region. To balance the link utilization and convergence speed, the U_{th} should be configured properly. If U_{th} is too small, the convergence speed of fairness will accelerate. But it also results in the lower link utilization, especially in high BDP networks. In this paper, we set U_{th} according to the output link capacity dynamically.

V. Performance Evaluation

To evaluate HFCC, we conduct several experiments using the packet-level simulator ns-2 [31], which has been extended with HFCC and integrated with other LFCC protocols including VCP, MLCP and BMCC. We focus on the comparison between HFCC and other protocols including TCP Reno, TCP SACK, CUBIC [4], CTCP [7], VCP [11], MLCP [12], BMCC [13] in this paper. Therein, TCP Reno is a standard TCP and widely employed in most operating systems. CUBIC and CTCP are default transport protocols in the latest Linux and Windows operating systems, respectively. The parameters of all compared protocols are the default values of ns-2 or optimal values of original papers. And the key parameters of HFCC are tuned automatically according to the network states.

To obtain the fair support from routers, TCP Reno, TCP SACK, CUBIC, and CTCP are always employed in conjunction with the ECN-enabled RED [22]. Unless other statements, the bottleneck queue limit is one BDP, or two packets per-flow, whichever is larger. The size of data packets is 1000 bytes, while the size of ACK packets is 40 bytes. To prove HFCC, we consider the convergence, average FCT, robustness, *etc.* in different scenarios. All experiments last at least 200s and the data in the first ten seconds are ignored.

A. Single Bottleneck

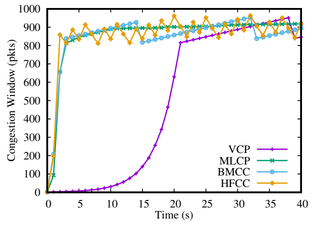
In this topology, all flows, which are generated by HFCC or other compared protocols, share the same bottleneck link. The default values of the bottleneck bandwidth and RTPD are 150 Mbps and 80 ms respectively.

1) Convergence: HFCC inherits from VCP and has the better convergence in efficiency and fairness. Differ from existing protocols, HFCC can achieve the rapid convergence of the efficiency and fairness because of the RI phase, hybrid feedback and self-adaptive parameters.

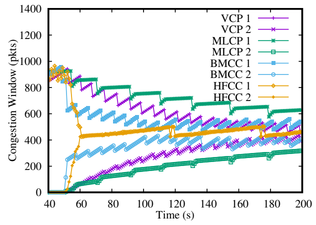
Converges to Efficiency: Different from VCP, MLCP and BMCC, HFCC possesses the RI phase and uses the available bandwidth estimation to accelerate the occupation of residual bandwidth in low-load region. Theoretically, the bandwidth estimator requires a few RTTs to measure the available bandwidth of bottleneck links and achieve the rapid increase of $cwnd$. In practice, considering the stability and accuracy, the observation interval (constant) of the bandwidth estimator is relatively longer. In HFCC, we employ an adaptive method to adjust it dynamically and the low bound should not too large, which may affect the convergence speed of the efficiency.

To illustrate the convergence speed of the efficiency of HFCC, we conduct an experiment with two long-lived flows. The compared protocols include VCP, MLCP and BMCC. And t_ρ, β, γ and U_{th} are all adaptive as explained above. We collect the result data of the first 40 seconds. As shown in Fig. 4(a), the convergence speed of the efficiency of HFCC is close to that of MLCP and BMCC and faster than that of VCP.

Converges to Fairness: The synchronized feedbacks of previous LFCCs result in the sluggish convergence of intra-flow fairness. Unlike previous LFCCs, HFCC has the faster convergence because of hybrid feedbacks. Low-bandwidth flows have fewer MD phases while high-bandwidth flows would undergo more MD action. Using the same setting as previous, we conduct an experiment and collect the result data after 40 seconds. As can be seen from Fig. 4(b), HFCC has the faster convergence of intra-flow fairness than other protocols



(a) Efficiency



(b) Fairness

Fig. 4. The convergence speed of different LFCC protocols

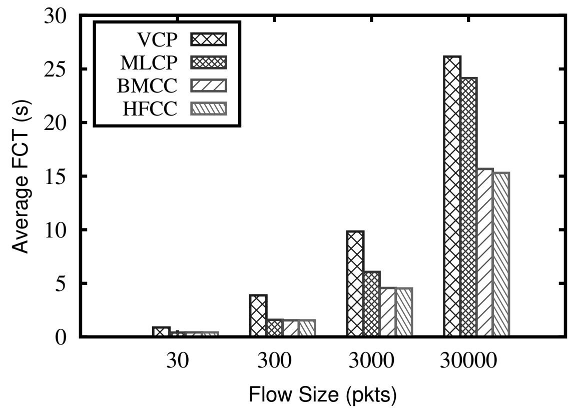


Fig. 5. The average FCT of different LFCC protocols

when a flow starts at 50 second. The inversely-proportional increase of MLCP makes it become the worst one.

2) Flow Completion Time: To investigate the FCT of LFCCs under different flow size, we conduct an experiment with the flow num 100. The flow size obeys the pareto distribution, where the mean value varies in [30, 30000] packets (1000 bytes per packet) and the shape equals to 1.2. In addition, the arrival time of flows obeys the poisson distribution. As shown in Fig. 5, HFCC has the minimal average FCT and is slightly better than BMCC. Compared with VCP and MLCP, the average FCT is reduced by 40% and 36% respectively when the flow size is 30000 packets. The results indicate that HFCC is better for delay-sensitive applications.

3) Impact of Buffer Size: To analyze the influence of the router buffer, we evaluate different protocols when the bottleneck queue limit varies in [0.2, 2.0] BDP. The forward and reverse paths have 10 FTP flows respectively. As shown in Fig. 6, the performance of HFCC is similar to that of MLCP and BMCC and better than that of VCP and other TCPs.

4) Sudden Demand Changes: We also analyze the robustness of different protocols when the number of flows changes suddenly. In this experiment, five long-lived flows start at 0.1 s and stop at 300 s. One hundred flows start at 100 s and stop at 200 s. The parameters are default values. Fig. 7 shows that HFCC can adjust $cwnd$ agilely and ensure the high utilization of bottleneck link when the burstiness occurs.

B. Multiple Bottlenecks

To evaluate HFCC in the multi-bottleneck scenario, we build a topology with seven bottleneck links. The capacity and propagation delay of bottleneck links are 150 Mbps and 20 ms respectively. There are five long-lived flows sharing the bottleneck links in forward and reverse directions. In addition, each bottleneck link has five interfering FTP flows in the forward direction. The RTPD of long-lived, FTP flows is 360 ms, and the RTPD of interfering flows is 60 ms. The results in Fig. 8 indicates that HFCC is better than other protocols.

VI. Conclusion

This paper proposes a high-efficiency LFCC protocol, HFCC, based on hybrid feedbacks in HBDP networks. Using the overlay coding and asynchronous feedbacks, HFCC can regulate $cwnd$ agilely and improve the convergence speed of the efficiency and intra-flow fairness while keeping the advantages of LFCCs. Furthermore, to reduce the average FCT of short flows, HFCC adopts a bandwidth estimator to accelerate the convergence of the MI stage of VCP. Adopting the simulator ns-2, we have proven the viability and validity of HFCC. Furthermore, due to the ECN-enabled IPv6 stack, HFCC can be easily applied into the IPv6 network.

As a future work, we will investigate the relationship between parameters and the performance of HFCC and then optimize parameters, which are suitable to a wide range network conditions, to achieve optimal performances.

References

[1] J. Postel,“Transmission Control Protocol”, RFC 793, http://www.ietf.org/rfc/rfc793.txt, Sep. 1981.

[2] D.M. Chiu, R. Jain, “Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks”, *Journal of Computer Networks and ISDN Systems*, vol. 17 no. 1, pp. 1-14, Jun. 1989.

[3] A. Afanasyev, N. Tilley, P. Reiher, *et al*, “Host-to-Host Congestion Control for TCP”, *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, pp. 304-342, Jun. 2010.

[4] S. Ha, I. Rhee, L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant”, *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64-74, Jul. 2008.

[5] L. S. Brakmo, L. L. Peterson, “TCP Vegas:End to End Congestion Avoidance on a Global Internet”, *IEEE Journal on Selected Areas Communications*, vol. 13, no. 8, pp. 1465-1480, Oct. 1995.

[6] C. Jin, D. X. Wei, S. H. Low, “FAST TCP: Motivation, Architecture, Algorithms, Performance”, in *Proc. IEEE INFOCOM*, 2004, pp. 2490-2501.

[7] K. Tan, J. Song, Q. Zhang, *et al.*, A Compound TCP Approach for High-Speed and Long Distance Networks, in *Proc. of IEEE INFOCOM*, 2006, pp. 1-12.

[8] M. Wang, J. Wang, S. Han, “Adaptive Congestion Control Framework and a Simple Implementation on High Bandwidth-Delay Product Networks”, *Computer Networks*, vol. 64, pp. 308-321, Mar. 2014.

[9] A. Falk, D. Katabi, Y. Pryadkin, “Specification for the Explicit Control Protocol (XCP)”, http://www.isi.edu/isi-xcp/docs/draft-falk-xcp-spec-03.html, Jul. 2007.

[10] N. Dukkipati, “Rate Control Protocol (RCP): Congestion Control to Make Flows Complete Quickly”, Stanford University, Ph.D. Thesis, http://yuba.stanford.edu/ nanditad/thesis-NanditaD.pdf, 2006.

[11] Y. Xia, L. Subramanian, I. Stoica, *et al.*, “One More Bit is Enough”, IEEE/ACM Transactions on Networking, vol. 16, no. 6, pp. 1281-1294, Dec. 2008.

[12] I. A. Qazi, T. Znati, “On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks”, *Computer Networks*, vol. 55, no. 1, pp. 45-60, Jan. 2011.

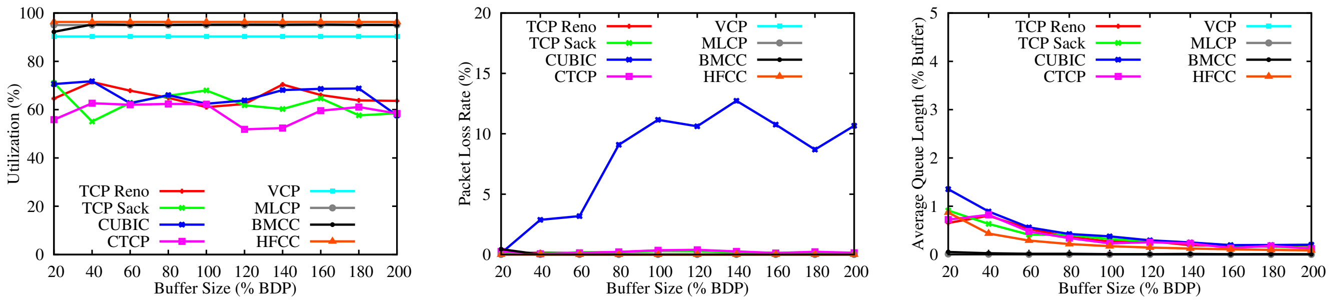
[13] I. A. Qazi, T. Znati, L. L. H. Andrew, “Congestion Control with Multipacket Feedback”, *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1721-1733, Dec. 2012.

[14] A. Kuzmanovic, A. Mondal, S. Floyd, *et al.*, “Adding Explicit Congestion Notification (ECN) Capability to TCP’s SYN/ACK Packets”, RFC5562, https://tools.ietf.org/html/rfc5562, Jun. 2009.

[15] M. Kuhlewind, S. Neuner, B. Trammell, “On the State of ECN and TCP Options on the Internet”, in *Proc. PAM*, 2013, pp. 135-144.

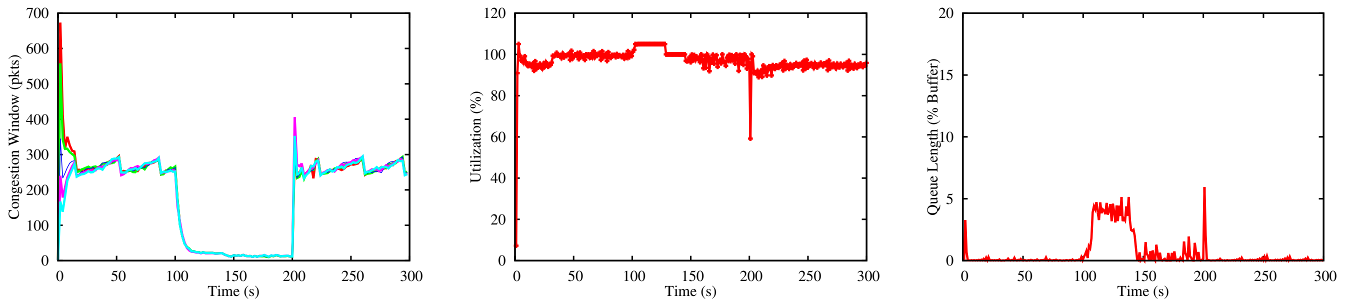
[16] S. Floyd, “HighSpeed TCP for Large Congestion Windows”, RFC3649, http://tools.ietf.org/html/rfc3649, Dec. 2003.

[17] T. Kelly, “Scalable TCP: Improving Performance in Highspeed Wide Area Networks”, *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83-91, Apr. 2003.



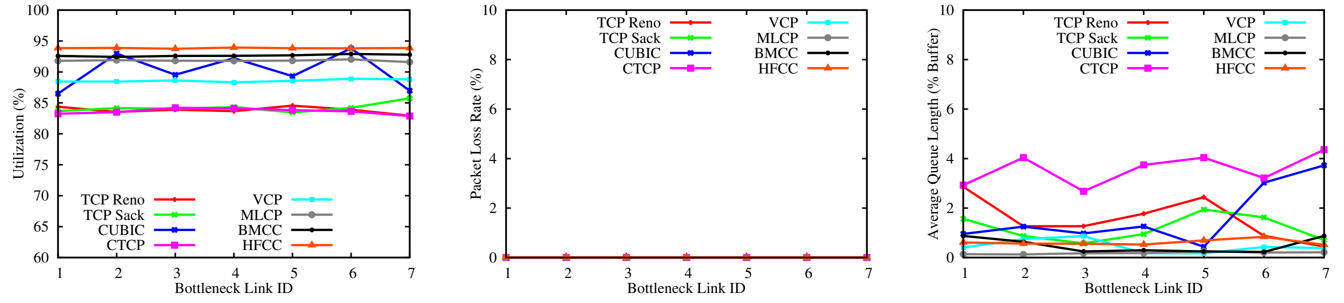
(a) Link Utilization (b) Packet Loss Rate (c) Queue Length

Fig. 6. The buffer size varying from 0.2 BDP to 2.0 BDP



(a) Congestion Window (b) Link Utilization (c) Queue Length

Fig. 7. HFCC is robust against and responsive to sudden, traffic demand changes



(a) Link Utilization (b) Packet Loss Rate (c) Queue Length

Fig. 8. Multiple congested bottlenecks with capacity 150 Mbps

[18] R. Jain, "A Delay-based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks", *ACM SIGCOMM Computer Communication Review*, vol. 19, no. 5, pp. 56-71, Oct. 1989.

[19] Y. Cui, L. Wang, X. Wang, et al, "End-to-End Coding for TCP", IEEE Network, vol. 30, no. 2, pp. 68-73, Mar. 2016.

[20] D. Fesehaye, P. Xia, P. B. Godfrey, *et al.*, "Finishing Flows Faster with A Quick congestion Control Protocol (QCP)", *Unpublished*.

[21] R. Adams, "Active Queue Management: A Survey", *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1425-1476, Jun. 2013.

[22] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, Aug. 1993.

[23] L. L. H. Andrew, S. V. Hanly, S. Chan, *et al.*, "Adaptive Deterministic Packet Marking", *IEEE Communications Letters*, vol. 10, no. 11, pp. 790-792, Nov. 2006.

[24] J. Wang, P. Dong, J. Chen, et al, "Adaptive Explicit Congestion Control based on Bandwidth Estimation for High Bandwidth-Delay Product Networks", *Computer Communications*, vol. 36, pp. 1235-1244, Mar. 2013.

[25] M. Alizadeh, A. Greenberg, D. A. Maltz, et al, "Data Center TCP (DCTCP)", in *Proc. ACM SIGCOMM*, 2010, pp. 63-74.

[26] K. Xu, Y. Tian, N. Ansari, "TCP-Jersey for Wireless IP Communications", IEEE Journal on Selected Areas in Communications, vol. 22, no. 4, pp. 747-756, May 2004.

[27] R. Pan, L. Breslau, B. Prabhakar, et al, "Approximate Fairness through Differential Dropping", *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 23-39, Apr. 2003.

[28] X. Jiang, G. Jin, J. Yang, "LRURC: A Low Complexity and Approximate Fair Active Queue Management Algorithm for Choking Non-Adaptive Flows", *IEEE Communications Letters*, vol. 19, no. 4, pp. 545-548, May 2015.

[29] A. Eshete, Y. Jiang, "Generalizing the CHOKe flow protection", Comput. Netw., vol. 57, pp. 147-161, 2013.

[30] W. Bai, L. Chen, K. Chen, et al, "Enabling ECN in Multi-Service Multi-Queue Data Centers", in *Proc. NSDI*, 2016, pp. 537-549.

[31] -, "The Network Simulator", http://www.isi.edu/nsnam/ns/, 2012.