Performance Evaluation and Improvement of Hybrid TCP Congestion Control Mechanisms in Wireless LAN Environment

Masafumi Hashimoto, Go Hasegawa and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka Suita, Osaka, 560–0871 Japan
{m-hasimt, murata}@ist.osaka-u.ac.jp, hasegawa@cmc.osaka-u.ac.jp

***Abstract*—In this paper, we first evaluate the performance of recent TCP variants for high-speed and long-delay networks in IEEE 802.11 wireless LAN environment through simulation experiments. We show that some of them still have well-known TCP unfairness property between uplink and downlink flows, and that there is another unfairness problem among uplink flows caused by the loss-based behavior of hybrid TCP variants. We then propose an end-to-end-basis modification to TCP congestion control mechanisms to alleviate the unfairness problems, which activates the congestion control when detecting ACK packet losses. Through simulation experiments, we present that the proposed method is effective not only for TCP fairness among uplink flows but also for fairness between uplink and downlink flows, while keeping the total throughput to be large enough.**

I. Introduction

With recent development on networking technologies and wide spread of the Internet environment, the available bandwidth-delay product of end-to-end network paths is rapidly increasing. In such high-speed and long-delay networks, it is known that TCP Reno, which is mainly used in the current Internet, cannot obtain sufficient throughput due to its congestion control mechanisms [1].

In recent researches, many congestion control mechanisms have been proposed to address such problem [1-6]. Generally, those TCP variants can be categorized into three types, that are loss-based TCP [1, 2], delay-based TCP [3, 7] and hybrid TCP [4, 5], from the viewpoint of the network congestion indications a TCP sender utilizes to regulate its congestion window size. These TCP variants have been evaluated by simulation and implementation experiments mainly for the wired high-speed and long-delay networks.

On the other hand, accessing the Internet through wireless networks is becoming common situation as the wireless network technologies improve. IEEE 802.11b [8], IEEE 802.11a [9] and so on are standardized by IEEE as the wireless LAN (WLAN) environment. 802.11b and 802.11a have a maximum data rate of 11 Mbps and 54 Mbps, respectively. Moreover, 802.11n [10], which has a maximum data rate of 600 Mbps, is now being standardized, and other wireless networking technologies will increase the wireless network bandwidth. Therefore, we should consider the situation where the TCP variants for high-speed and long-distance networks would be transparently utilized in such high-speed wireless networks. In these WLANs, CSMA/CA is used as a channel access mechanism. In CSMA/CA, if the channel is sensed as busy before a transmission, the transmission is deferred for a random backoff interval for reducing the probability of frame collisions. This is one of the main contributing factor for the fluctuation of the transmission time in WLANs.

In the network environment where round trip times (RTTs) change by increasing and decreasing transient delay such as CSMA/CA mechanisms, we can expect that TCP variants using RTT as network congestion indications, such as delay-based and hybrid TCP, cannot detect the network congestion accurately. However, it is not reasonable situation where a TCP sender would switch its TCP stack according to the access network technologies to which it connects for the Internet access. Therefore, it is important to evaluate the performance of TCP variants for high-speed and long-delay networks in WLAN environments.

It has been pointed out in the past researches that per-flow unfairness in TCP throughput can be found in WLAN environment [11]. There are various solutions proposed for alleviating such unfairness [11-15]. These solutions diminish TCP throughput unfairness by modifying MAC protocol parameters or queue management mechanisms in an access point. However, MAC protocols of wireless access points are generally implemented in hardware, so it takes large cost to change them. Furthermore, in some of those methods, it is necessary to estimate the number of flows and the throughput of the flows.

IEEE 802.11e [16] is the standardization for supporting different Quality of Service (QoS) requirements in WLANs. It realizes different QoS requirements among flows which are generated from the station by utilizing per-flow queueing at the station's network interface. Therefore, it cannot help improve per-station fairness. Furthermore, the difficulty in parameter settings is another problem.

In this paper, we first evaluate the performance of delay-based and hybrid TCP variants, which utilize RTT information as network congestion indications, in WLAN environment through ns-2 [17] simulation experiments. The original papers on delay-based and hybrid TCPs describe how to update the congestion window size according to the RTT information.

However, they do not clearly explain how to use measured RTT values, including statistical filtering methods and the update interval of the congestion window size. Therefore, in this paper, we estimate the behavior of TCP variants from implementation codes in Linux (kernel 2.6.22) and simulation codes of ns-2. From simulation results, we confirm that TCP variants can successfully absorb the RTT fluctuation in WLAN by simple filtering mechanisms which are deployed in the implementation/simulation codes. In addition, we show that pure delay-based TCPs do not cause TCP unfairness in many cases, while loss-based and hybrid TCP face two kinds of serious unfairness problems because of numerous drops of ACK packets at the access point buffer.

We then propose a transport-layer solution to the above unfairness problems. The proposed mechanism activates the congestion control when detecting ACK packet losses. From simulation results, we show that the proposed method can alleviate unfairness among TCP flows while keeping the total throughput to be large enough, and that the flexibility of the proposed method is higher than that of existing methods.

The rest of the paper is organized as follows. In Section II, we discuss the impact of WLAN environment on the performance of TCP variants through simulation experiments. In Section III, we introduce our solution and evaluate it through simulation experiments. We finally conclude this paper and discuss future work in Section IV.

II. Impact of Wireless LAN Environment on TCP performance

In this section, we investigate the impact of WLAN environment on the performance of TCP variants by using ns-2 simulation experiments. We use TCP Vegas [7] and FAST TCP [3] as delay-based TCP variants, and Compound TCP (CTCP) [4] and TCP-Adaptive Reno (AReno) [5] as hybrid TCP variants. Note that the hybrid TCPs use both of loss-based and delay-based behaviors in their congestion control mechanisms. TCP Vegas, FAST TCP, and CTCP update the congestion window size once per RTT, whereas AReno updates it every time an ACK packet is received. In the Linux implementation, TCP Vegas and CTCP utilize the minimal RTT that has been observed in each RTT cycle, denoted as $minRTT$, in the congestion window updating algorithm. FAST TCP and AReno utilize the RTT value, denoted as $avgRTT$, that is smoothed by the exponential weighted moving average and the smoothed RTT value which is managed by the traditional TCP, respectively. For the detailed mechanisms of each TCP variant refer to [3-5, 7]. The above behaviors regarding RTT information are taken into account in the following simulation experiments.

A. Simulation Setting

Fig. 1 shows the network topology using IEEE 802.11a WLAN. In the network, multiple stations share one access point, which is connected to a wired node through a wired link with 100 Mbps capacity and one-way propagation delay of 100 ms. Except as otherwise noted, all stations are located at 4 m from the access point and the buffer size is set to 100 packets. The buffer size of the wired link, the sender buffer size of the station, and the advertised receiver window are set to large enough not to limit the TCP performance. The queue mechanism of the access point and the wired link is the Drop Tail discipline. The parameters of IEEE 802.11a are shown in Table I. In each TCP variant, we use the corresponding ns-2 module when it exists. Otherwise, we use the ns-2 module which is converted from the Linux implementation codes by using NS-2 TCP-Linux [18]. The parameters of the TCP variants are set to as follows.

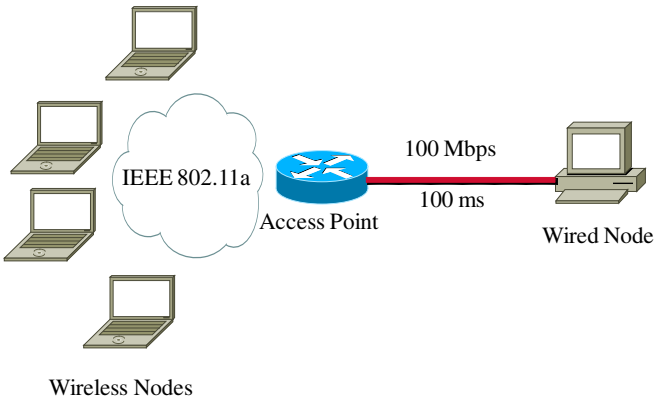


Fig. 1. Network topology

TABLE I
THE PARAMETERS OF IEEE 802.11A

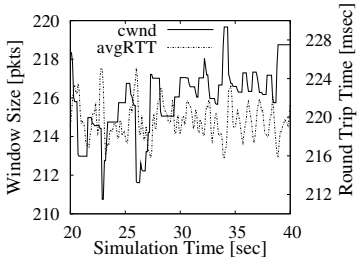
Data Rate	54 Mbps
Slot Time	9 μs
SIFS	16 μs
DIFS	34 μs
CW_{min}	15
CW_{max}	1023

TCP Vegas	$\alpha_v = 2,\ \beta_v = 4$
FAST TCP	$\alpha_f = 20,\ \gamma_f = 0.5$
CTCP	$\alpha_c = 0.125,\ \gamma_c = 2,\ \zeta_c = 1$
AReno	$a = 0.5,\ \alpha_a = 16,\ \beta_a = 1$

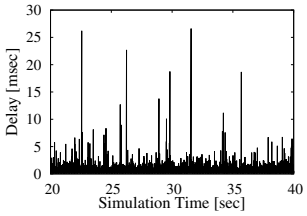
Only one flow is generated for each station, meaning that we increase the number of stations for increasing the number of concurrent flows in the network. The data transmission is started when the simulation starts. The simulation time is 200 s. Here, an uplink flow denotes a TCP flow that transmits data from the station to the wired node, and a downlink flow denotes a TCP flow that transmits data from the wired node to the station.

B. Simulation Results and Discussions

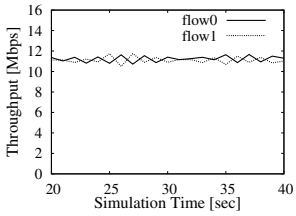
1) Impact of delay fluctuations in wireless channel: Fig. 2 shows simulation results where FAST TCP is used and the number of uplink flows is two. Fig. 2(a) shows the change in the congestion window size ($cwnd$) and $avgRTT$ of one flow. Fig. 2(b) exhibits delay fluctuations at wireless channel from



(a) Congestion window size and RTT

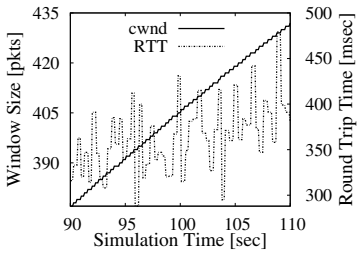


(b) Transmission delay in wireless channel

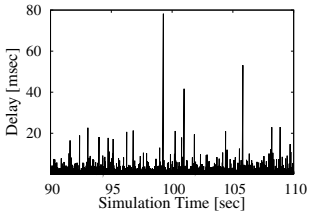


(c) Throughput

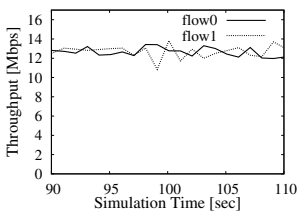
Fig. 2. FAST TCP (2 uplink flows)



(a) Congestion window size and RTT



(b) Transmission delay in wireless channel

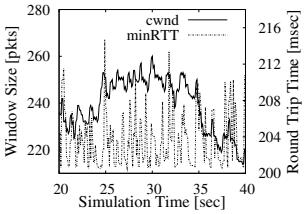


(c) Throughput

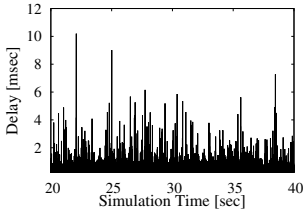
Fig. 3. TCP Reno (2 uplink flows)

the station to the access point. Fig. 2(c) presents the change in the throughput of both flows.

Fig. 2(b) shows that most of delay fluctuations are within 5 ms, and the delay larger than 10 ms cannot be found so often. When focusing on around 23 s of the simulation time in Fig. 2(b), we can observe that the sudden increase of $avgRTT$ and the corresponding decrease of $cwnd$ by roughly four packets. However, the throughput remains stable within 1 Mbps fluctuation (Fig.2(c)). For comparison, we show the simulation results with two uplink TCP Reno flows in Fig. 3. Since the congestion window size of TCP Reno is not affected by RTT fluctuation, the results in Fig. 3(c) can be regarded as the throughput fluctuation without the effect



(a) Congestion window size and RTT



(b) Transmission delay in wireless channel

Fig. 4. CTCP (2 uplink flows)

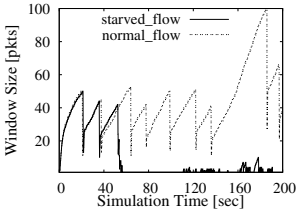
of delay fluctuation. By comparing Figs. 2(c) and 3(c), we conclude that the throughput of FAST TCP are not affected by the delay fluctuation in this environment. The main reason of this is that the delay fluctuation in the wireless channel would be absorbed at the sending buffer of the WLAN interface of the stations.

Fig. 4 shows the corresponding simulation results of CTCP. Fig. 4(b) shows that most of the delay fluctuation is less than 2 ms, and it rarely exceeds 5 ms. Furthermore, by comparing Figs. 4(a) and 4(b), we can observe that the delay spikes found in wireless channel do not always affect $minRTT$. This is due to the RTT filtering mechanism deployed in the Linux implementation of CTCP. Therefore, the increase in $minRTT$ found in Fig. 4(a) is caused mainly by the queueing delay at the sending buffer of WLAN interface of the stations.

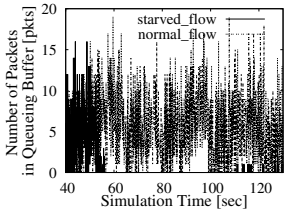
In addition, we confirmed through extensive simulation experiments that the delay fluctuation in the wireless channel have little effect on the performance of TCP Vegas and AReno as well as FAST TCP and CTCP. It is due to RTT filtering mechanism implemented at Linux implementation and ns-2 simulation codes of those TCP variants.

2) TCP fairness among uplink flows: We observed a phenomenon that, as the number of uplink flows increases, some flows occupy the WLAN bandwidth and other flows are starved for TCP throughput. Fig. 5 shows the simulation results of the change in the congestion window size (Fig. 5(a)), the change in the number of ACK packets buffered at the access point (Fig. 5(b)), and the change in the number of discarded ACK packets at the access point (Fig. 5(c)), when the number of uplink CTCP flows is 16. In the figure, we only plot the behavior of one of non-starved flows (labelled as *normal_flow*) and that of starved flows (*starved_flow*). Fig. 5(a) shows that the starved flow does not increase its congestion window size after 55 s, while the normal flow keeps the congestion window size at a certain level. We also observe from Figs. 5(b) and 5(c) that there are many ACK packet losses at the access point, and that the starved flow sends almost no data and ACK packets.

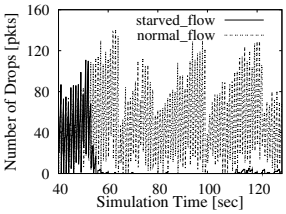
From the above results, the reason for serious TCP unfairness among uplink flows can be explained as follows. When the number of uplink flows increases, the wireless channel between stations and the access point is highly congested. Since IEEE 802.11 provides a half-duplex wireless channel and the transmission opportunity of a station and that of an



(a) Congestion window size



(b) The number of ACK packets buffered at the access point



(c) The number of drops of ACK packets at the access point

Fig. 5. Unfairness using CTCP (16 uplink flows)

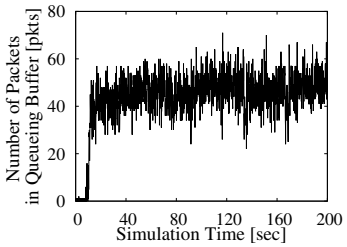
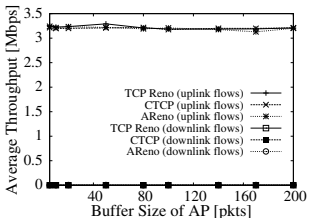
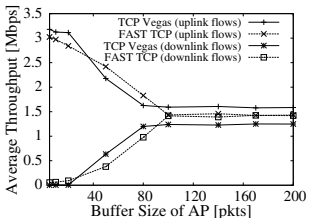


Fig. 6. The total number of ACK packets buffered at the access point using TCP Vegas (16 uplink flows)



(a) loss-based and hybrid TCP variants



(b) delay-based TCP variants

Fig. 7. Average throughput (8 uplink flows, 8 downlink flows)

access point are equal, the access point becomes congested by ACK packets. It causes the full-utilization of the access point buffer and numerous ACK packet losses as shown in Fig. 5(c). Note that TCP does not activate the congestion control mechanisms against ACK packet losses, except for occurring Retransmission Time Out (RTO) when *all* ACK packets in a window are lost. Therefore, when we use loss-based and hybrid TCPs which utilize loss-based mechanisms, the congestion window size keeps growing up regardless of the congestion at the access point. Eventually, when all ACK packets in a window are discarded for a certain flow, RTO occurs and the congestion window size is set to one packet. At this time, the buffer of the access point is still fully-utilized because the access point congestion is not resolved. Therefore, the ACK packets from the RTOed flow do not arrive at the station due to buffer overflow at the access point. That is, once a RTO occurs at a certain flow, the flow cannot increase its congestion window size for a long time. This causes the serious throughput unfairness among uplink flows.

This phenomenon is caused when numerous ACK packets are buffered and discarded at the access point. Therefore, we can expect that this phenomenon occurs when a new flow starts transmitting in the WLAN environment after the access point has been congested.

Fig. 6 shows the change in the total number of ACK packets buffered at the access point when the number of uplink TCP Vegas flows is 16. We can see that the number of buffered ACK packets keeps around 45 packets and there is no ACK packet loss. This is because that delay-based TCP variants such as TCP Vegas regulate the congestion window size such that it keeps the number of backlogged packets in bottleneck routers to be constant. Therefore, the unfairness problem as shown in Fig. 5 does not occur. On the other hand, loss-based TCP, as well as hybrid TCP variants which end up using loss-based mechanisms [4, 5], fills up the access point buffer with ACK packets and the unfairness problem would occur due to numerous ACK packet losses.

3) TCP fairness between uplink and downlink flows: We next present the evaluation results when uplink and downlink flows coexist in the network. Fig. 7(a) shows the average throughput of loss-based and hybrid TCP variants as a function of the buffer size at the access point, when the number of uplink and downlink flows is set to eight respectively. Fig. 7(b) shows the corresponding results with delay-based TCP variants. Note that the average throughput is calculated from observed throughput during the latter 160 s simulation time to avoid transient effects.

Fig. 7(a) exhibits that the loss-based and hybrid TCP cause significant unfairness between uplink and downlink flows regardless of the access point buffer size. The reason of this is as follows. When the access point is congested, ACK packets of uplink flows and data packets of downlink flows are discarded at the access point due to the congestion at the access point. The downlink flows activate the congestion control with data packet losses and decrease the congestion window size. On the other hand, the uplink flows do not activate the congestion control with ACK packet losses and continue increasing the congestion window size. This causes serious throughput unfairness between uplink and downlink flows.

On the other hand, from Fig. 7(b), we observe that delay-based TCP does not bring throughput unfairness when the buffer size of the access point is large enough. This is because the converged queue length, as shown in Fig. 6, depends on the number of flows in the network and other network parameters, and it is around 100 packets in this simulation settings.

III. TCP Congestion Control for ACK Packet Losses

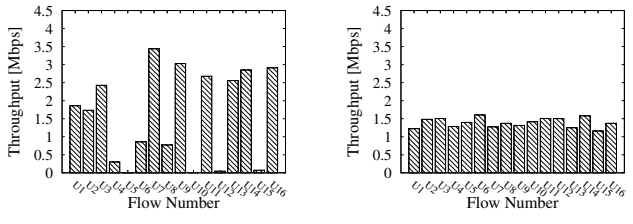
In this section, we propose a transport-layer modification to alleviate the unfairness problem pointed out in Section II.

A. Proposed Method

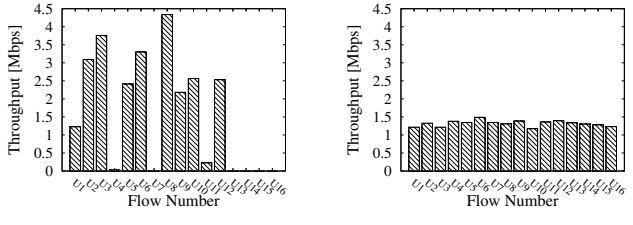
The main reason for the unfairness among TCP flows shown in Section II is that loss-based and hybrid TCP variants keep growing the congestion window size even when the access point is highly congested and numerous ACK packets are discarded. From the results of Subsections II-B2 and II-B3, pure delay-based TCP variants can keep the fairness among TCP flows. However, it is known that the performance of pure delay-based TCP variants significantly degrades where they coexist with loss-based TCP [19, 20]. This is one of the main reasons that pure delay-based TCP variants such as TCP Vegas are not widely used in the current Internet, that encourages recent appearance of hybrid TCP variants. Therefore, we propose an end-to-end basis modification to TCP congestion control mechanisms, which should be applied to loss-based and hybrid TCPs, to alleviate unfairness problems. Our proposal is based on quite a simple idea: TCP should activate the congestion control when detecting ACK packet losses, whereas the traditional TCP activates it only against data packet losses. In detail, the proposed method activates the congestion control when the number of ACK packet losses in a window exceeds the pre-determined threshold (*thresh_ack_losses*). The sender TCP detects ACK packet losses by monitoring the sequence number of received ACK packets. When the sender TCP observes abnormal jumps of the ACK sequence numbers, it is regarded as ACK packet losses in the network. When the number of ACK packet losses in a window exceeds thresh_ack_losses, a sender TCP halves the congestion window size. Note that before halving the window size, the sender TCP waits for one RTT to avoid false detection caused by the disorder of data packet reception at the receiver TCP. Additionally, when detecting data packet losses, a sender TCP stops checking ACK packet losses during one RTT.

B. Simulation Evaluations

We evaluate the performance of the proposed method, which is applied to TCP Reno, through ns-2 simulation experiments. For comparison, we show that the results of two existing solutions, that are a modification of MAC protocol parameters [12] and a modification of queue management mechanisms [13]. The solution in [12] modifies the carrier sense duration of the WLAN access points from Distributed Inter Frame Space (DIFS) to Point Inter Frame Space (PIFS). The solution proposed in [13] divides the buffer in an access point into for data packets and for ACK packets. The advertised window size of the solution in [13] is set to 160 packets. The buffer size for data packets and that for ACK packets at the access point are set to 50 packets, respectively. thresh_ack_losses is set to one for the proposed method. The other simulation settings are the same as in Section II. In what follows, we focus only on the fairness among flows. Note that the difference of the total throughput of all methods is within 10 %.

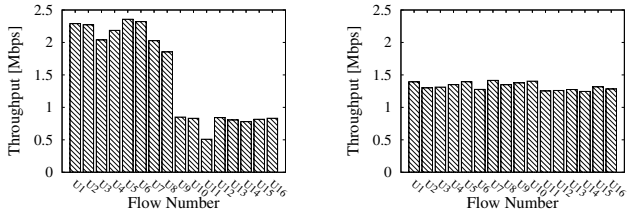


(a) No modification (b) The method in [12]



(c) The method in [13] (d) The proposed method

Fig. 8. Average throughput (16 uplink flows)

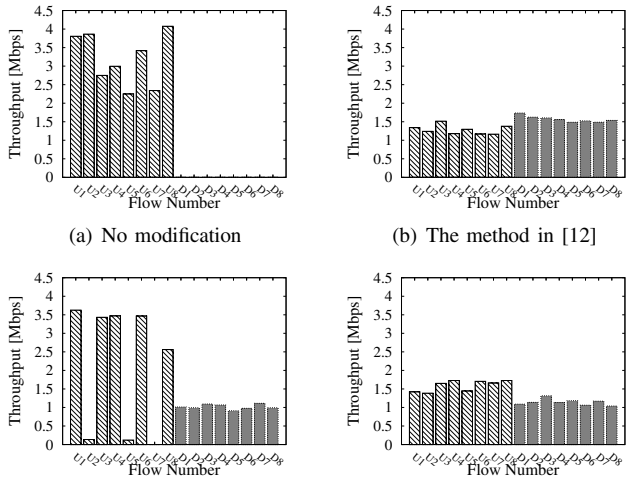


(a) The method in [12] (b) The proposed method

Fig. 9. Average throughput where stations are relocated (16 uplink flows)

Fig. 8 exhibits the simulation results of each method, where the number of uplink flows is set to 16 and the throughput of each flow is plotted. Fig. 8(a) shows that serious unfairness occurs when we do not apply any modifications. Figs. 8(b) and 8(d) show that the solution in [12] and the proposed method successfully alleviate unfairness among uplink flows. On the other hand, the solution in [13] (Fig. 8(c)) causes serious unfairness among uplink flows as well as the case without any modifications (Fig. 8(a)). This is because the solution in [13] divides the access point buffer into for data packets and for ACK packets, which does not contribute on the unfairness problem among uplink flows.

We next present the results in Fig. 9 where there are 16 uplink flows and the eight stations and the other eight stations are located at 1 m and 10 m from the access point, respectively. Here we compare the proposed method and the solution in [12]. Fig. 9(b) shows that the proposed method provides good fairness regardless of the location of the stations. On the other hand, from Fig. 9(a), the solution in [12] cause significant unfairness between flows dependently on the distance of the station from the access point. This is because the solution in [12] is sensitive to the wireless



(a) No modification (b) The method in [12]

(c) The method in [13] (d) The proposed method

Fig. 10. Average throughput (8 uplink flows, 8 downlink flows)

channel environment since it is based on MAC parameter tuning regarding the channel access. It means that the careful tuning is required for the modification to MAC parameters. In contrast, since our proposed method is based on a transport-layer approach, its performance is not dependent on the wireless channel conditions.

Fig. 10 indicates the average throughput of each flow where the number of uplink and downlink flows are eight, respectively. Note that we locate all stations at 4 m from the access point. From Fig. 10(a), we can observe that when we do not apply any modifications, there exists two kinds of unfairness problems, that are the unfairness between uplink and downlink flows and the unfairness among uplink flows. On the other hand, the solution in [12] (Fig. 10(b)) and the proposed method (Fig. 10(d)) improve fairness among uplink and downlink flows. The solution in [13], however, faces unfairness among uplink flows while it keeps fairness among downlink flows. The reason of this is the same as the reason mentioned above for Fig. 8. Fig. 10(d) shows that the throughput of downlink flows is slightly lower than that of uplink flows when the proposed method is applied. On the other hand, proposed method is comparable with the solution in [12] in terms of fairness degradation.

IV. Conclusion

In this paper, we first evaluated the performance of TCP variants for high-speed and long-delay networks in wireless LAN environment by simulation experiments. From the simulation results, we concluded that delay fluctuation in wireless channel have little impact on the throughput of delay-based and hybrid TCP variants. However, loss-based and hybrid TCP variants experiences significant unfairness among uplink flows and that between uplink and downlink flows. On the other hand, pure delay-based TCP variants do not cause such unfairness.

We then proposed the transport-layer solution against unfairness among TCP flows for loss-based and hybrid TCP variants. From the results of the simulation experiments, we presented that the proposed solution is effective not only for TCP fairness among uplink flows but also for fairness between uplink and downlink flows.

For future work, we will investigate the performance of the proposed method in lossy network environments, where there are packet losses caused by wireless link errors. We also plan to implement the proposed method and evaluate it in the actual wireless LAN environment.

References

[1] S. Floyd, "HighSpeed TCP for large congestion windows," *Request for Comments 3649 (Experimental)*, Dec. 2003.

[2] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–91, Apr. 2003.

[3] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[4] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks," in *Proceedings of PFLDnet 2006*, Feb. 2006.

[5] H. Shimonishi, T. Hama, and T. Murase, "TCP-Adaptive Reno: Improving efficiency-friendliness tradeoffs of TCP congestion control algorithms," in *Proceedings of PFLDnet 2006*, Feb. 2006.

[6] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *Proceedings of PFLDnet 2005*, Feb. 2005.

[7] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, Oct. 1995.

[8] IEEE 802.11b, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. IEEE, Feb. 1999.

[9] IEEE 802.11a, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band*. IEEE, Feb. 1999.

[10] IEEE P802.11n/D2.0, *Draft Amendment to STANDARD: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Throughput.* IEEE, Mar. 2007.

[11] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and Prasun, "Understanding TCP fairness over wireless LAN," in *Proceedings of IEEE INFOCOM 2003*, vol. 2, pp. 863–872, Mar. 2003.

[12] Y. Fukuda and Y. Oie, "Unfair and inefficient share of wireless LAN resource among uplink and downlink data traffic and its solution," *IEICE Transactions on Communications*, vol. E88-B, pp. 1577–1585, Apr. 2005.

[13] J. Ha and C.-H. Choi, "TCP fairness for uplink and downlink flows in WLANs," in *Proceedings of IEEE Global Telecommunications Conference 2006*, pp. 1–5, Nov. 2006.

[14] F. Keceli, I. Inan, and E. Ayanoglu, "TCP ACK congestion control and filtering for fairness provision in the uplink of IEEE 802.11 infrastructure basic service set," in *Proceedings of IEEE International Conference on Communications*, pp. 4512–4517, June 2007.

[15] N. Blefari-Melazzi, A. Detti, I. Habib, A. Ordine, and S. Salsano, "TCP fairness issues in IEEE 802.11 networks: Problem analysis and solutions based on rate control," in *Proceedings of IEEE Transactions on Wireless Communications*, vol. 6, pp. 1346–1355, Apr. 2007.

[16] IEEE 802.11e, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Medium Access Control (MAC) Quality of Service Enhancements*. IEEE, Oct. 2005.

[17] The Network Simulator ns-2. available at http://www.isi.edu/nsnam/ns/.

[18] A Linux TCP implementation for NS2. available at http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/.

[19] T. Bonald, "Comparison of TCP Reno and TCP Vegas via fluid approximation," tech. rep., INRIA Research Report 3563, Nov. 1998.

[20] J. Mo, R. J. La, V. Anantharm, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proceedings of INFOCOM '99*, vol. 3, pp. 1556–1563, Mar. 1999.