Trade-off evaluation between fairness and throughput for TCP congestion control mechanisms in a wireless LAN environment

Masafumi Hashimoto, Go Hasegawa and Masayuki Murata
Graduate School of Information Science and Technology, Osaka University
1–5 Yamadaoka Suita, Osaka, 565–0871, Japan
Telephone: +81-6-6850-6863
Fax: +81-6-6850-6868
Email: {m-hasimt, murata}@ist.osaka-u.ac.jp, hasegawa@cmc.osaka-u.ac.jp

***Abstract*—Per-flow unfairness of TCP throughput in the IEEE 802.11 wireless LAN environment has been reported. Although a number of researchers have proposed various methods for alleviating the unfairness, they evaluated the fairness of their methods separately from the network bandwidth utilization, meaning that they did not consider the trade-off relationships between fairness and bandwidth utilization. In this paper, we first propose a novel performance metric considering both per-flow fairness and bandwidth utilization at network bottlenecks. We then propose a transport-layer solution for alleviating TCP unfairness. We finally evaluate the performance of the proposed method through experiments in a real wireless LAN environment. We demonstrate that the proposed method can achieve better a trade-off between fairness and throughput, regardless of vendor implementations of wireless access points and wireless interface cards.**

***Index Terms*—Transmission Control Protocol (TCP), wireless LAN, fairness, performance metric, experimental evaluation**

I. Introduction

With recent developments in wireless networking technologies, accessing the Internet through a wireless LAN (WLAN) is becoming common, and WLAN-based Internet access environments are often available at public areas, such as railway stations and airports. As such, it is important to consider fairness among coexisting users.

As the wireless LAN environment, IEEE 802.11 families [1] are standardized. The current IEEE 802.11 implementations primarily use the Distributed Coordination Function (DCF) as a medium access control protocol. In the DCF, a medium access mechanism is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA enables stations, including an access point (AP), to fairly access a wireless channel. Therefore, fairness among stations in the WLAN is ensured at the MAC layer. Here, the traffic in a WLAN consists of the traffic flowing from client stations to wired networks via an AP (called upstream traffic) and vice versa (called downstream traffic). Downstream traffic is transmitted only from an AP, whereas upstream traffic is generated from multiple client stations. Therefore, upstream traffic obtains more access opportunities to the wireless channel than downstream traffic. This means that the fairness among stations realized at the MAC layer does not mean the fairness between the two flow directions at the upper layers. In fact, it has been reported that TCP flows experience severe unfairness in WLANs [2]–[4]. When TCP is used as a transport-layer protocol, the unfairness among upstream flows is caused by its congestion control mechanisms, i.e., TCP activates the congestion control against data packet losses, but not against ACK packet losses.

Various solutions have been proposed to alleviate unfairness [2]–[5]. These solutions diminish TCP throughput unfairness by modifying the MAC protocol parameters or queue management mechanisms in APs. However, the MAC protocols of APs are generally implemented at the hardware level, so changing these protocols is costly. Furthermore, a number of methods need to estimate the number of flows and the throughput of each flow at the AP. In addition, although certain solutions can alleviate the unfairness by changing the MAC layer, they may also cause other unfairness in the transport layer. For example, although a priority-based solution at the AP [5] can significantly improve TCP fairness, this solution may cause UDP unfairness by assigning too an overly high priority to traffic from the AP.

In this paper, we propose a transport-layer solution to deal with the above problems. The basic concept of the proposed solution was presented in [6]. The reasons for using a transport-layer solution are as follows. First, MAC-layer solutions may cause other unfairness issues at the transport layer, as described above. Second, the unfairness is mainly caused by the behavior of the transport-layer protocols. Third, since transport-layer protocols are generally implemented through software, modifying transport-layer protocols is easier than modifying MAC-layer protocols. The proposed method alleviates the unfairness by detecting TCP ACK packet losses as an indication of congestion at an AP. The proposed method requires a small modification of the TCP congestion control mechanisms only on stations in the WLAN. We have confirmed the fundamental characteristics of the proposed method in [6]. Through simulation experiments in [6], we demonstrated that the proposed method is effective not only for TCP fairness among upstream flows but also for fairness between upstream and downstream flows. In this paper, we develop the proposed method, including additional functions that are important for implementation in the actual environment.

Generally, Jain's fairness index [7] has been used as an evaluation metric of fairness among users, flows, etc. Since this index depends only on the variation of allocated values, the index values intuitively indicate whether the allocated values are fair. Here, assume that we have two solutions, both of which are identically effective for alleviating unfairness, but one of them degrades the total amount of allocated values, whereas the other does not decrease the total value. Although the latter should be judged to be superior to the former, both solutions are identical from the viewpoint of Jain's fairness index. In other words, Jain's fairness index cannot evaluate such situations accurately.

Therefore, in this paper, we propose a novel performance metric that consider the trade-off relationships between per-flow fairness and bandwidth utilization at network bottlenecks. The proposed metric can simultaneously evaluate both fairness and utilization with a single metric value. We then evaluate the performance of the proposed method through experiments using real WLAN environments with products from several vendors. We demonstrate that the proposed method can significantly improve TCP fairness regardless of the number of upstream and downstream TCP flows.

The remainder of this paper is organized as follows. In Section II, we discuss unfairness problems among TCP flows in WLANs. In Section III, we introduce a novel performance metric considering both per-flow fairness and bandwidth utilization. Section IV describes a solution for alleviating TCP unfairness in WLANs. In Section V, we present experimental results obtained using the solution proposed in Section IV. Finally, conclusions and a discussion of future research are presented in Section VI.

II. Fairness among TCP flows in WLAN environments

In a typical WLAN environment, all stations including an AP use the same parameters of CSMA/CA. Thus, the contending stations obtain fair access opportunities to wireless channel in the long term. This also means that upstream flows, transmitted from multiple client stations, and downstream flows, transmitted from an AP, share the same wireless channel. Therefore, when n client stations share a single AP in the WLAN network, the access opportunities to the wireless channel of upstream traffic is $n/(n+1)$, whereas that of downstream traffic is $1/(n+1)$. Hence, an AP is likely to become a congestion point, meaning that the fairness at the MAC layer protocol does not contribute to the fairness at upper layer protocols such as UDP and TCP. Such a difference in the ability to access the wireless channel between upstream and downstream traffic causes serious problems because many applications, such as P2P file sharing, used in WLANs generate both downstream traffic and upstream traffic.

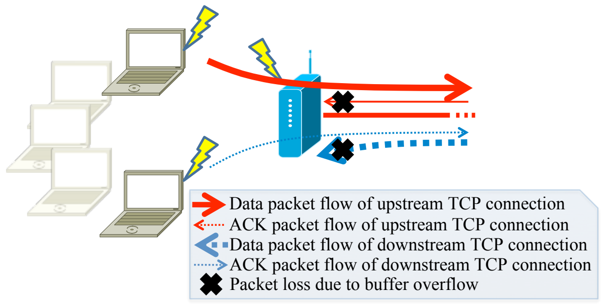

Data packet flow of upstream TCP connection
ACK packet flow of upstream TCP connection
Data packet flow of downstream TCP connection
ACK packet flow of downstream TCP connection
Packet loss due to buffer overflow

Fig. 1. Congestion at the AP

Moreover, when TCP is used as a transport protocol, serious per-flow unfairness occurs not only between upstream and downstream flows but also among upstream flows. When the congestion occurs as a result of the above-mentioned situation, TCP ACK packets of upstream flows and TCP data packets of downstream flows are discarded at an access point buffer (Fig. 1). Note that TCP activates the congestion control against data packet losses, but not against ACK packet losses. Therefore, the congestion window size of upstream flows continues to grow until Retransmission Time Out (RTO) occurs and *all* ACK packets in a window are lost. When all ACK packets of a certain upstream flow are discarded, the congestion window size is set to one packet. At this time, the buffer at the AP is still fully utilized because the access point congestion is not resolved. That is, once a certain flow experiences an RTO, the flow cannot increase its congestion window size for some time, which causes throughput unfairness among upstream TCP flows.

On the other hand, when upstream and downstream TCP flows coexist, upstream TCP flows do not activate the congestion control mechanisms and increase the congestion window size, whereas downstream TCP flows activate the congestion control mechanisms and decrease the congestion window size. This causes serious throughput unfairness between upstream and downstream TCP flows because upstream TCP flows obtain more access opportunities to the wireless channel than downstream TCP flows.

Various solutions have been proposed for alleviating the above TCP unfairness [2]–[5]. A solution proposed in [2] improves fairness among upstream and downstream flows by rewriting the advertised receiver window size at the AP. A solution proposed in [4] alleviates unfairness between upstream and downstream flows by dividing the buffer in an AP into a buffer for data packets and a buffer for ACK packets. In [3], TCP unfairness among upstream flows is diminished by filtering ACK packets in an AP. In [5], the author proposed shortening the carrier sense duration of the WLAN APs. However, the cost of changing existing hardware devices using these methods is high because the MAC protocols or queue management mechanisms must be modified.

We therefore proposed an end-to-end solution for alleviating serious TCP unfairness [6], focusing on numerous

ACK packets discarded at the buffer of an AP when serious TCP unfairness occurs. The proposed method regards ACK packet losses as an indication of congestion and activates the congestion control, whereas the normal TCP does not activate the congestion control against ACK packet losses. The proposed method can improve TCP fairness not only among upstream flows but also between upstream and downstream flows.

III. Performance Metric for Trade-off between Fairness and Throughput

A. Existing Methods

The definition of fairness is important when we discuss fairness among flows, because the improvement of fairness is sometimes achieved at the expense of total bandwidth utilization. In previous researches [2]–[5], [8], fairness is defined such that all flows contending on a wireless channel in a WLAN achieve the same throughput, and the effect on the total network throughput is not considered.

Jain's fairness index, which defined as follows, has been used to evaluate the fairness:

$$F_j(A) = \frac{\left(\sum_{i=1}^{n} a_i\right)^2}{n\sum_{i=1}^{n} a_i^2} \quad (1)$$

where n is the number of contending users, $A = \{a_1, a_2, \cdots, a_n\}$ is a set of allocations for n users such that a_i is an allocation for user i. The index value approaches one as the variation of allocations decreases, and the index value approaches $1/n$ as the variation of allocations increases. Note that Jain's index is independent of the scale of allocations. For example, consider fairness when allocating 10, 30, and 40 dollars, respectively, to three persons, and fairness when allocating 100, 300, and 400 dollars, respectively, to three persons. Both cases are equivalent from the viewpoint of Jain's index (0.82).

However, the total amounts of allocated values are different. That is, Jain's index is not suitable for comparing fairness and considering the total amount of allocations. The total amount of allocated values corresponds to the network bandwidth utilization in the context of network bandwidth sharing. Therefore, when we have a solution for alleviating unfairness while slightly degrading the total throughput, Jain's index cannot accurately evaluate such a performance trade-off.

B. Definition of the Proposed Metric

Given a throughput set $X = \{x_1, x_2, \cdots, x_n\}$, where x_i is the throughput of the i th flow, and the network bandwidth at the bottleneck, C, where $\sum_{i=1}^{n} x_i \leq C$, we define *fair and fully-utilized throughput* $x_f = \frac{C}{n}$, where all flows achieve the same throughput and the network bandwidth is fully utilized. Using the relationship between Jain's fairness index $F_j(X)$ in Eq. (1) and total throughput $\sum_{i=1}^{n} x_i$, we define the desired properties for proposed fairness index $F(X, C)$ as follows:

1) If $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \leq C$ and $F_j(X) < F_j(Y)$, then $F(X, C) < F(Y, C)$.
2) If $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \leq C$ and $F_j(X) = F_j(Y)$, then $F(X, C) = F(Y, C)$.
3) If $\sum_{i=1}^{n} x_i = C$, then $F_j(X) = F(X, C)$.

where $Y = \{y_1, y_2, \cdots, y_n\}$.

We start from the index $f(X, C)$, which represents the degree to which the throughput of each user is not fair and fully utilized throughput (x_f):

$$f(X, C) = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_f)^2. \quad (2)$$

We then normalize $f(X, C)$ by x_f and obtain $g(X, C)$ as follows:

$$g(X, C) = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x_f)^2}}{x_f}. \quad (3)$$

According to [7], Jain's fairness index can be transformed into

$$F_j(X) = \frac{1}{1 + COV^2} \quad (4)$$

$$COV = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}}{\bar{x}} \quad (5)$$

where COV is the coefficient of variance of allocations to the users, and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the average of the allocations. Finally, by comparing Eqs. (3) and (5), we obtain the following definition of a novel index:

$$F(X, C) = \frac{1}{1 + g(X, C)^2} = \frac{C^2}{n\sum_{i=1}^{n} x_i^2 - 2C\sum_{i=1}^{n} x_i + 2C^2}. \quad (6)$$

Note that the index satisfies the above-mentioned properties. The index approaches one when the bandwidth utilization of the network bottleneck approaches 100 % and the throughput variance of each flow is small. In contrast, the index approaches $1/n$ when the throughput variance is large.

C. Comparison with Jain's Fairness Index

In Table I, Jain's fairness index and the proposed index are compared using simple examples in which the network capacity bandwidth is 30 Mbps. Cases 1, 2, and 3 in Table I have fair throughput distributions, but different total throughputs. The proposed index can differentiate these cases, whereas

TABLE I
Comparison between Jain's fairness index and the proposed index ($C = 30$ Mbps)

Case	Throughput distribution [Mbps]	Total [Mbps]	Jain's index	Proposed index
1	{ 3, 3, 3, 3, 3, 3, 3, 3, 3, 3}	30	1.00	1.00
2	{ 2, 2, 2, 2, 2, 2, 2, 2, 2, 2}	20	1.00	0.90
3	{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}	10	1.00	0.69
4	{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 6}	15	0.50	0.67
5	{ 1, 1, 1, 1, 1, 1, 1, 1, 6, 6}	20	0.50	0.64
6	{ 2, 2, 2, 2, 2, 2, 2, 2, 2, 12}	30	0.50	0.50
7	{ 1, 1, 1, 1, 1, 1, 2, 3, 3, 6}	20	0.62	0.73
8	{ 1, 1, 1, 3, 3, 3, 3, 4, 5, 6}	30	0.78	0.78

Jain's index cannot distinguish among them. Like Cases 1, 2, and 3, Cases 4, 5, and 6 have different total throughputs, but the throughput distributions have the same variations. The proposed index becomes small for Cases 4, 5, and 6, since the variance of throughput distribution of which are 2.25, 4.0, and 9.0, respectively. Thus, the values of the proposed index become small when a small number of flows acquire almost all of the bandwidth, even if the total throughput is high. In Cases 2, 5, and 7, the total throughputs are identical, but the throughput distributions have different variations. These cases corresponded to property 1) in Subsection III-B. In this situation, the order of the three cases in Jain's index and that in the proposed index are identical. Moreover, when the total throughput is equal to the network bandwidth capacity, as in Cases 1, 6, and 8, Jain's index and the proposed index are identical. This situation is related to property 3) in Subsection III-B.

IV. TCP Congestion Control for ACK Packet Losses

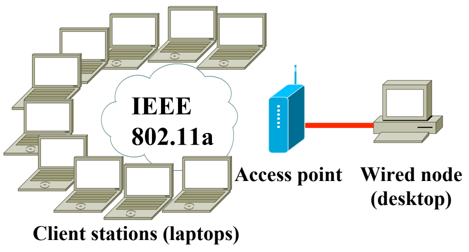
The main reason for unfairness among TCP flows is that a TCP continues to increase the congestion window size even when the AP is highly congested and numerous ACK packets are discarded. Therefore, we propose a simple modification to TCP congestion control mechanisms to alleviate these unfairness. This proposal is based on a simple concept: TCP should activate congestion control when ACK packet losses are detected, whereas the traditional TCP activates congestion control only when data packet losses are detected.

More specifically, the proposed method activates congestion control when the number of ACK packet losses in a window exceeds a pre-determined threshold (*thresh_ack_losses*). The TCP sender detects ACK packet losses by monitoring the sequence number of the received ACK packets. When the TCP sender observes abnormal jumps in the ACK sequence numbers, these jumps are regarded as ACK packet losses in the network and are calculated as follows:

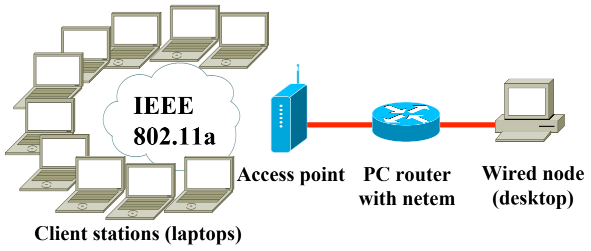
$$ack\_loss = \max\left(\left\lfloor \frac{ack - prev\_ack}{MSS} \right\rfloor - 1,\ 0\right) \quad (7)$$

where ack_loss is the number of ACK packet losses, ack is the sequence number of the current received ACK packet (bytes), $prev\_ack$ is the sequence number of the previous received ACK packet (bytes), and MSS is maximum segment size (bytes). When the number of ACK packet losses in a Round Trip Time (RTT) exceeds *thresh_ack_losses*, a TCP sender halves the congestion window.

When using Eq. (7) to detect ACK packet losses, the effect of the delayed ACK option [9] should be considered. Note that the delayed ACK option has been implemented in both Windows and Linux [10], [11]. Let b be the number of data packets acknowledged by a received ACK packet. Several TCP receiver implementations send one ACK packet for two consecutive received packets, so b is typically set to two. When the delayed ACK option is enabled, Eq. (7) cannot correctly estimate the number of ACK packet losses. Therefore, we



(a) Simple experimental environment



(b) Experimental environment with a PC router

Fig. 2. Experimental environments

add a function which determine the value of b to the method proposed in [6].

There are two possible methods by which determine the value of b. In the first method, a TCP receiver explicitly informs a TCP sender of the value of b. In the second method, a TCP sender estimates the value of b without any explicit information from the TCP receiver. In the first method, a TCP sender provides an accurate value of b, but this method requires modifications at a TCP receiver. For this reason, we use the second method, which does not require a TCP receiver-side modification.

In the proposed method, b is estimated as follows:

$$b_{est} = \left\lfloor sb_i + \frac{1}{2} \right\rfloor \quad (8)$$

$$sb_i = (1-\beta)\, sb_{i-1} + \beta \left(\frac{ack - prev\_ack}{MSS} \right) \quad (9)$$

where b_{est} is the estimated value of b, and sb_i is the i th smoothed value for b with smoothing factor β. Note that sb_i in Eq. (9) is a continuous value, but b_{est} in Eq. (8) should be a discrete value because b should be a discrete value. Thus, b_{est} is calculated by half-adjust rounding. Using Eq. (8), Eq. (7) is transformed into:

$$ack\_loss = \max\left(\left\lfloor \frac{ack - prev\_ack}{b_{est} \times MSS} \right\rfloor - 1,\ 0\right). \quad (10)$$

V. Performance Evaluation

A. Experimental Settings and Methods

Two experimental environments are shown in Fig. 2. In both environments, ten client stations share one AP. All client stations are located within 50 cm of the AP in order to avoid packet losses due to wireless link error. In Fig. 2, a wired node is directly connected to the AP through a wired link. On the other hand, the experimental environment in Fig. 2(b)

TABLE II
WIRELESS DEVICES

(a) Wireless Interface Cards

Vendor	Product name
Buffalo	WLI-CB-AGHP
NEC	Aterm WL54AG

(b) Access Points

Vendor	Product name
Buffalo	WAPS-HP-AM54G54
NEC	Aterm WR8500N
Corega	CG-WLR300NNH

introduces a PC router between the AP and the wired node for the purpose of evaluation in long delay environments. DELL Latitude E5500 laptops and a DELL Precision 390 desktop are used as the client stations and the wired node, respectively. All nodes, including the wired node, use Ubuntu 8.10 (Linux kernel 2.6.28) as the OS. As shown in Fig. 2(b), another DELL Precision 390 desktop is used as the PC router with netem [12] for generating a 50 ms delay to the wired link between the AP and the wired node. We use Web100 [13] patch to collect the TCP connection information, such as the congestion window size and the RTT from the Linux kernel. We used TCP Reno and implemented the proposed method on the Linux code of TCP Reno.

The wireless devices listed in Table II are used as wireless interface cards for client stations and the AP. Note that all client stations use the same type of wireless interface card in Table II(a) in each experiment. Due to space limitations, we show only the results obtained using the Buffalo's wireless interface card. However, the tendencies of the results are similar regardless of the type of wireless interface card. In the following, APs are abbreviated as *[vendor name]-AP*, e.g., Buffalo-AP.

The experiments using the environments in Fig. 2 were conducted as follows. Only one TCP flow is generated for each client station using Iperf [14], assuming bulk data transfer. We keep the number of concurrent TCP flows at ten and change the ratio of upstream and downstream TCP flows from (0, 10) to (10, 0). For the purposes of comparison, TCP connections use either the proposed method on TCP Reno or conventional TCP Reno. The *thresh_ack_losses* parameter in the proposed method is set to one. The experiment time is set to 180 seconds, and each TCP connection is generated simultaneously when the experiment starts. We disabled vendor-specific functions implemented at APs. The TCP-delayed ACK option was disabled at TCP receivers, except in experiments using the environment shown in Fig. 2(b). For each experimental setting, the experiments are conducted ten times in order to average the results.

B. Experimental Evaluation Results and Discussion

Figs. 3, 4, and 5 show the snapshot results for the average throughput of ten upstream flows in the experimental environment shown in Fig. 2(a) using Buffalo-AP, NEC-AP, and Corega-AP, respectively. The average throughput is calculated using the amount of data transmitted in 50-180 seconds in the experiment. Figs. 3(a), 4(a), and 5(a) show that serious throughput unfairness occurs, regardless of the vendors of the APs. Note that some of the upstream flows occupy the network bandwidth, whereas other flows are completely starved. From the viewpoint of unfairness, Corega-AP provides better results than the other APs. In other words, the number of starved flows when using Corega-AP is smaller than that when using the other APs. This is because of the difference in the buffer size at the APs. The buffer size of Corega-AP is larger than that of the other APs. On the other hand, Figs. 3(b), 4(b), and 5(b) show that the proposed method successfully alleviates throughput unfairness among upstream TCP flows, regardless of the AP products.

Fig. 6 presents the average throughput of upstream and downstream flows and the total throughput for various ratios of upstream and downstream flows. In the figure, uxdy denotes that the number of upstream and downstream TCP flows are x and y, respectively. Fig. 6 shows that, using TCP Reno, when at least one upstream TCP flow exists in the network, the upstream flows occupy almost all of the network bandwidth, and the downstream flows are starved. On the other hand, the proposed method can significantly improve the throughput fairness between upstream and downstream flows, and starved flows do not occur. Fig. 6(b) reveals that the degree of fairness

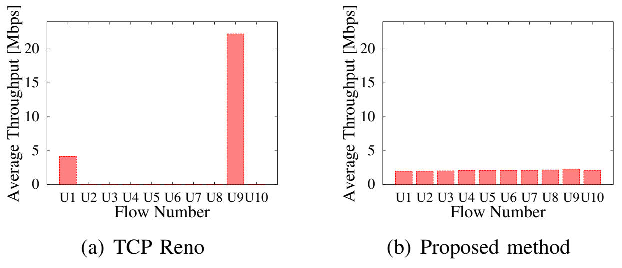


Fig. 3. Average throughput of ten upstream flows when using Buffalo-AP

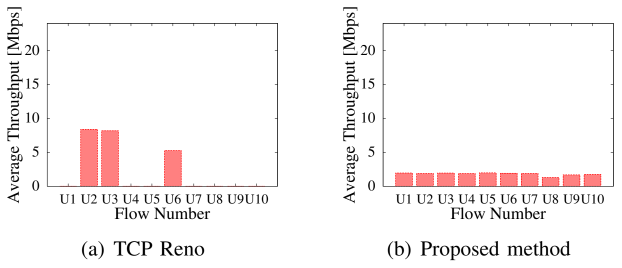


Fig. 4. Average throughput of ten upstream flows when using NEC-AP

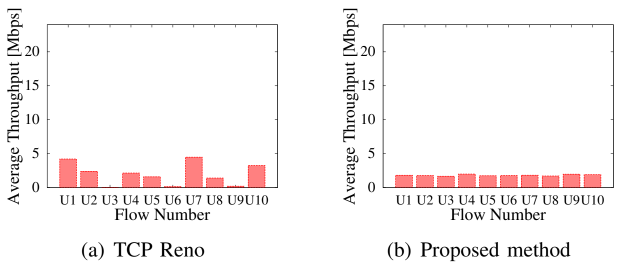
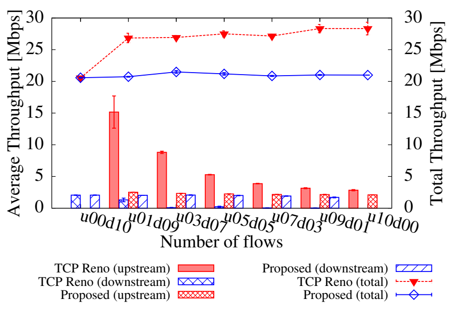
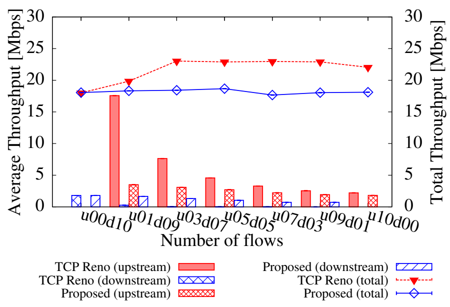


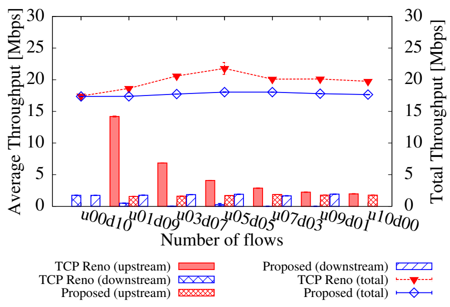
Fig. 5. Average throughput of ten upstream flows when using Corega-AP



(a) Buffalo-AP



(b) NEC-AP



(c) Corega-AP

Fig. 6. Effect of the number of upstream and downstream flows

improvement is small when using NEC-AP. The reason for this is as follows. In this case, the flows experience RTO even when the proposed method is used, whereas the proposed method produces no RTO when using the other APs. We believe that the reason for this is that the buffer size of the NEC-AP is much smaller than that of the other APs. However, even in this case the proposed method can avoid starvation of flows.

In terms of total throughput, the total throughput of ten downstream flows with TCP Reno and the proposed method are equivalent, regardless of AP type. This is because ACK packets are not discarded at the APs and the behaviors of TCP with and without the proposed method are identical. However, when there exist one or more upstream flows, the total throughput of the proposed method degrades while the fairness improves significantly. The reason for this is as follows. When the proposed method is not used, several ACK packets of upstream TCP flows are discarded at the APs. This means that the numbers of data packets and ACK packets in the WLAN are not balanced, and so the number of data packets that are transmitted in the WLAN increases. However, when using the proposed method, the numbers of data packets and ACK packets are balanced because the proposed method activates congestion control against ACK packet losses and the number of ACK packet losses at APs decreases. Note that the total throughput would increase when deactivating the proposed method, but the increased throughput is distributed among non-starved flows, so that the fairness among flows degrades. In other words, in the proposed method, there exists a trade-off relationship between fairness and bandwidth utilization.

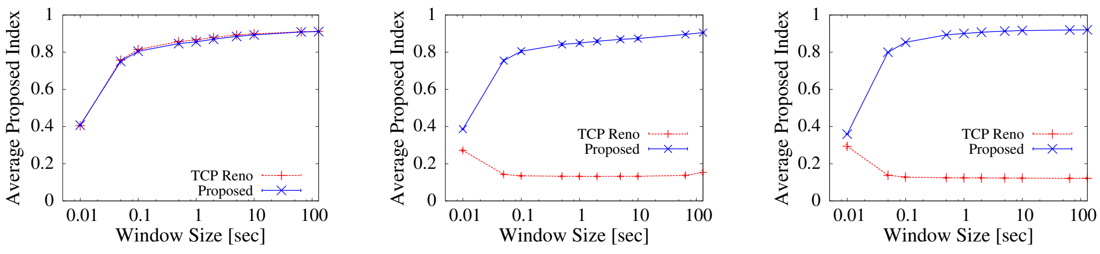
In order to evaluate this relationships, we use the index proposed in Section III with the Sliding Window Method (SWM) [15]. The SWM can provide a quantitative measure of fairness over a wide range of time scales and has the advantage of measuring *short-term fairness* and *long-term fairness* at the same time. Intuitively, the short-term fairness of a data transmission flow refers to its ability to provide equitable access to resources to all contending flows over short time scales. In contrast, long-term fairness measures the average amount of resources assigned over a longer time. The SWM function applied to Eq. (6) is as follows:

$$SWM(w) = \frac{C^2}{n\sum_{i=1}^{n} x_i(w)^2 - 2C\sum_{i=1}^{n} x_i(w) + 2C^2} \quad (11)$$

where w is the time-window size for evaluating the fairness, and $x_i(w)$ is the average throughput of flow i in a time-window w.

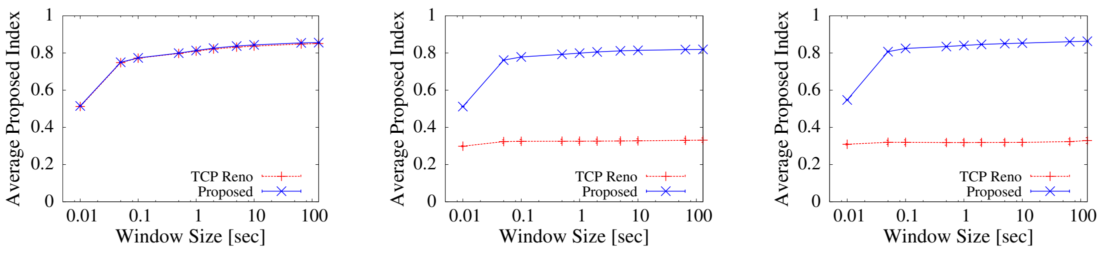
Figs. 7, 8, and 9 show the evaluation results obtained with the proposed metric with SWM when using Buffalo-AP, NEC-AP, and Corega-AP, respectively, under the same conditions as Fig. 6. Parameter C in Eq. (11) is set to 29.60 Mbps according to the theoretical maximum throughput of IEEE 802.11a WLAN with 1460 bytes MTU [16]. The index values are identical for the cases using and without using the proposed method when there is no upstream flow (Figs. 7(a), 8(a) and 9(a)). This implies that the proposed method is not activated because there is no ACK packet loss. On the other hand, when we use Buffalo-AP or NEC-AP, the index values of the proposed method are significantly better than that of TCP Reno in terms of not only long-term fairness but also short-term fairness when one or more upstream flow exists in the network. However, the index value of TCP Reno when using Corega-AP is better than that when using the other APs. This is due to the large buffer size of Corega-AP mentioned above (Fig. 5(a)).

In order to investigate the effect of Eq. (8) in estimating the parameter using the delayed ACK option, we conducted an experiment using and without using the delayed ACK option under the environment shown in Fig. 2(b). Fig. 10 shows the average throughput of upstream and downstream flows and the total throughput for various ratios of upstream and downstream flows in the experimental environment of Fig. 2(b) using Buffalo-AP with a 50 ms delay. The smoothing factor β in Eq. (9) is set to $1/32$. In Fig. 10, the results obtained using and without using the delayed ACK option are labeled as *delack* and *nodelack*, respectively. The total throughput



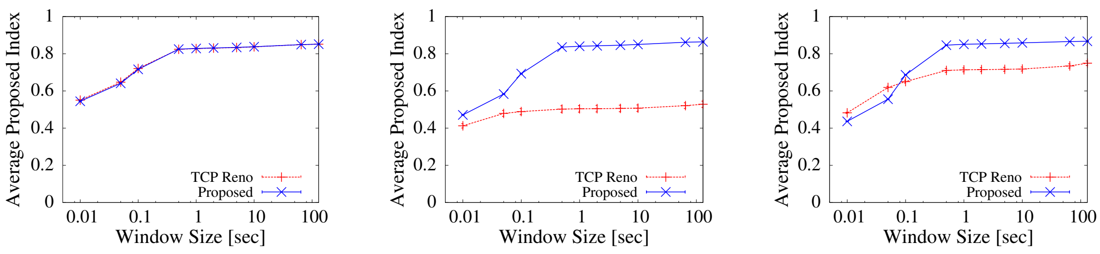
(a) Ten downstream flows (b) Five upstream flows and five downstream flows (c) Ten upstream flows

Fig. 7. Proposed index with SWM when using Buffalo-AP



(a) Ten downstream flows (b) Five upstream flows and five downstream flows (c) Ten upstream flows

Fig. 8. Proposed index with SWM when using NEC-AP



(a) Ten downstream flows (b) Five upstream flows and five downstream flows (c) Ten upstream flows

Fig. 9. Proposed index with SWM when using Corega-AP

increases when using the delayed ACK option with the original TCP Reno or the proposed method. This is because the delayed ACK option decreases the number of ACK packets in the WLAN and consequently increases the number of data packets injected into the WLAN. Furthermore, from the viewpoint of fairness, the original TCP Reno experiences serious unfairness among upstream and downstream flows, regardless of the use of the delayed ACK option, whereas the proposed method can significantly improve fairness using or without using the delayed ACK option. In other words, when using the delayed ACK option, the proposed method can enhance the total throughput without degrading the improvement in fairness.

Fig. 11 shows the evaluation results obtained using the proposed index under the same conditions as Fig. 10. In Fig. 11(a) in which ten downstream flows exist, the index values of the proposed method obtained using the delayed ACK option are better than those obtained without using the delayed ACK option. The reason for this is that the bandwidth utilization is improved by enabling the delayed ACK option, as described above. Furthermore, comparing the index values of the proposed method obtained using and without using the delayed ACK option, the index value obtained using the delayed ACK option is better than that obtained without using the delayed ACK option. In other words, in the proposed method, the improvements in bandwidth utilization leads to the improvement in the index values.

VI. Conclusion

In this paper, we proposed a novel performance metric for evaluating the trade-off relationship between fairness and bandwidth utilization in a wireless LAN environment. The proposed metric is based on the variations in throughput of concurrent flows and the ideal throughput distribution, in which all flows achieve the same amount of throughput and the

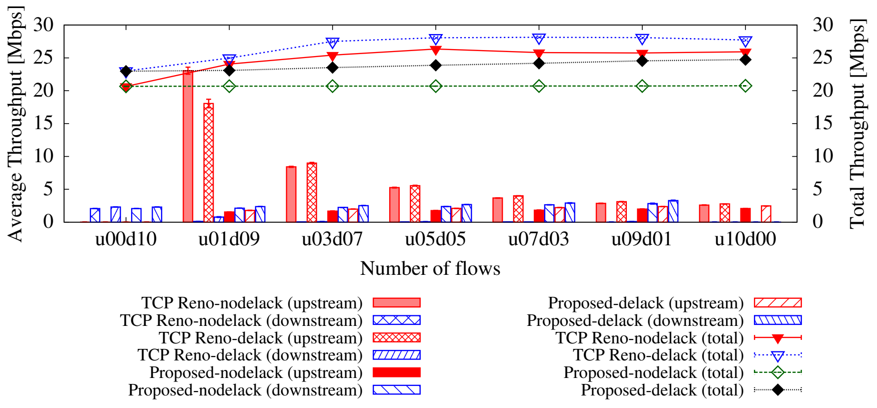


Fig. 10. Effect of the delayed ACK option when using Buffalo-AP with a 50 ms one-way delay

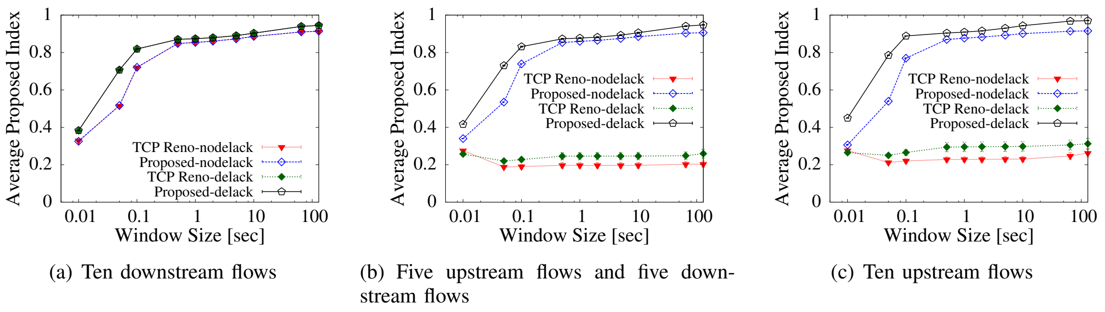


Fig. 11. Proposed index with SWM when using Buffalo-AP with a 50 ms one-way delay

network bandwidth is fully utilized. In addition, we proposed a transport-layer solution for alleviating TCP unfairness in a WLAN environment and evaluated the proposed method. Based on the results of the experimental evaluations, we confirmed that the proposed method alleviates TCP unfairness regardless of the vendor of the APs and wireless interface cards, although the total throughput decreases slightly when the delayed ACK option is disabled. Moreover, the proposed method using the delayed ACK option enhances the total throughput without degrading the effectiveness of fairness improvement. Through trade-off evaluations using the proposed metric, we also demonstrated that the proposed method can achieve a markedly better trade-off between fairness and bandwidth utilization. In the future, we intend to evaluate the proposed method in environments that include wired networks with several traffic scenarios.

References

[1] IEEE 802.11-2007, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, IEEE, Jun. 2007.

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

[3] 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 ICC 2007*, Jun. 2007, pp. 4512–4517.

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

[5] 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, no. 4, pp. 1577–1585, Apr. 2005.

[6] M. Hashimoto, G. Hasegawa, and M. Murata, "Performance evaluation and improvement of hybrid TCP congestion control mechanisms in wireless LAN environment," in *Proceeding of ATNAC 2008*, Dec. 2008, pp. 367–372.

[7] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.

[8] 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," *IEEE Transactions on Wireless Communications*, vol. 6, pp. 1346–1355, Apr. 2007.

[9] R. Braden, "Requirements for internet hosts – communication layers," *Request for Comments 1122*, Oct. 1989.

[10] Microsoft Corporation, *Microsoft Windows Server 2003 TCP/IP Implementation Details*, Jun. 2003.

[11] P. Sarolahti and A. Kuznetsov, "Congestion control in linux TCP," in *Proceedings of 2002 USENIX Ammial Tecnical Conference*, Jun. 2002, pp. 49–62.

[12] netem, available at http://www.linuxfoundation.org/en/Net:Netem.

[13] M. Mathis, J. Heffner, and R. Reddy, "Web100: Extended TCP instrumentation for research, education and diagnosis," *ACM Computer Communications Review*, vol. 33, no. 3, pp. 69–79, Jul. 2003.

[14] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, "Iperf-the TCP/UDP bandwidth measurement tool," available at http://dast.nlanr.net/Projects/Iperf/.

[15] C. E. Koksal, H. Kassab, and H. Balakrishnan, "An analysis of short-term fairness in wireless media access protocols," in *Proceedings of ACM Sigmetrics 2000*, Jun. 2000.

[16] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," in *Proceedings of NCA 2003*, Apr. 2003, pp. 249–256.