PAPER

A Transport-layer Solution for Alleviating TCP Unfairness in a Wireless LAN Environment

Masafumi HASHIMOTO[†a)], ***Nonmember***, **Go HASEGAWA**[†], ***and*** **Masayuki MURATA**[†], ***Members***

SUMMARY Per-flow unfairness of TCP throughput in the IEEE 802.11 wireless LAN (WLAN) environment has been reported in past literature. A number of researchers have proposed various methods for alleviating the unfairness; most require modification of MAC protocols or queue management mechanisms in access points. However, the MAC protocols of access points are generally implemented at hardware level, so changing these protocols is costly. As the first contribution of this paper, we propose a transport-layer solution for alleviating unfairness among TCP flows, requiring a small modification to TCP congestion control mechanisms only on WLAN stations. In the past literature on fairness issues in the Internet flows, the performance of the proposed solutions for alleviating the unfairness has been evaluated separately from the network bandwidth utilization, meaning that they did not consider the trade-off relationships between fairness and bandwidth utilization. Therefore, as the second contribution of this paper, we introduce a novel performance metric for evaluating trade-off relationships between per-flow fairness and bandwidth utilization at the network bottleneck. We confirm the fundamental characteristics of the proposed method through simulation experiments and evaluate the performance of the proposed method through experiments in real WLAN environments. We show that the proposed method can achieve better a trade-off between fairness and bandwidth utilization, regardless of vendor implementations of wireless access points and wireless interface cards.

key words: *Transmission Control Protocol (TCP), wireless LAN, fairness, congestion control, fairness index*

1. 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 WLAN environment, IEEE 802.11 families [1] are standardized. The current IEEE 802.11 implementations primarily use the Distributed Coordination Function (DCF) as the 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 [1]. Therefore, fairness among stations in the WLAN is ensured at the MAC layer. However, the fairness at the MAC layer does not mean the fairness at upper layers. When TCP is used as a transport-layer protocol, two types of unfairness among TCP flows (TCP unfairness) occur [2]–[4]: unfairness between upstream and downstream TCP flows and unfairness among upstream TCP flows. 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, resulting in unfairness between upstream and downstream traffics. On the other hand, the unfairness among upstream flows is caused by their congestion control mechanisms, i.e., TCP activates the congestion control against data packet losses, but not against ACK packet losses [5].

Various solutions have been proposed to alleviate unfairness in WLANs [2]–[4], [6]–[10]. 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, some 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 problems in the transport layer. For example, a priority-based solution at the AP [6] can improve fairness among TCP flows, but this solution may cause unfairness among UDP flows by assigning higher priority to traffic from the AP.

As the first contribution of the present paper, we propose a transport-layer approach to alleviate the above unfairness. 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 by software, modifying transport-layer protocols is easier than modifying MAC-layer protocols. The proposed method alleviates unfairness among TCP flows by detecting TCP ACK packet losses as an indication of congestion at an AP. It requires a small modification of the TCP congestion control mechanisms only on WLAN stations.

Generally, Jain's fairness index [11] 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 solu-

[†]The authors are with Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871, Japan.

a) E-mail: m-hasimt@ist.osaka-u.ac.jp

DOI: 10.1587/transcom.E0.B.1

tions, 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, as the second contribution of the present paper, we propose a novel performance metric that considers the trade-off relationships between per-flow fairness and bandwidth utilization at a network bottleneck. The proposed metric can simultaneously evaluate both fairness and utilization with a single metric value.

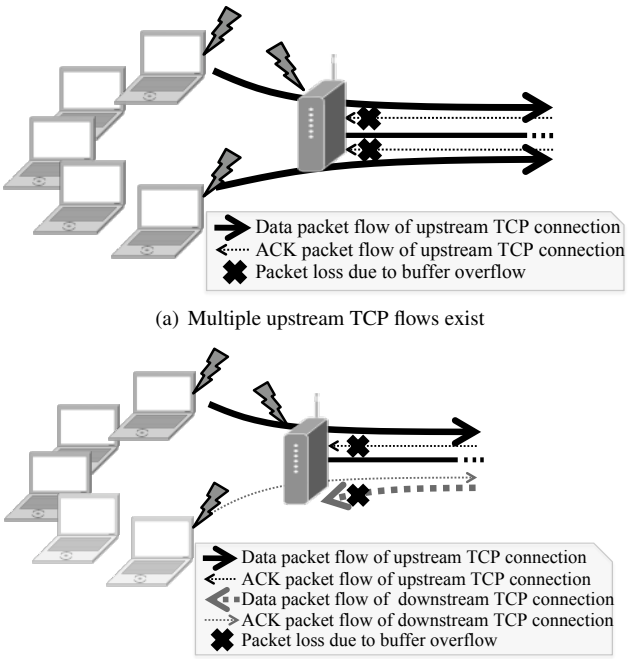
We conducted ns-2 [12] simulation experiments in order to confirm the fundamental characteristics of the proposed method. Through the simulation results, we show that the proposed method is effective not only for fairness among upstream flows but also for fairness between upstream and downstream flows. We then confirm the applicability and product-dependent characteristics of the proposed method through experiments using real environments with WLAN products from several vendors.

The remainder of the present paper is organized as follows. In Sect. 2, we discuss unfairness problems among TCP flows in WLANs. Section 3 describes a solution for alleviating TCP unfairness in WLANs. In Sect. 4, we introduce a performance metric considering both per-flow fairness and bandwidth utilization. In Sect. 5, we confirm the fundamental characteristics of the solution proposed in Sect. 3 through simulation experiments. Section 6 presents experimental results. Finally, conclusions and a discussion of future research are presented in Sect. 7.

2. 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 the multiple client stations and downstream flows transmitted from the 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, when the amount of the upstream and downstream traffic is roughly the same, the AP is likely to become a congestion point, and unfairness occurs between upstream and downstream traffic. It means that the fairness at the MAC layer protocol does not always contribute to the fairness at upper layer protocols such as UDP and TCP. Since the many kinds of applications such as P2P file sharing and audio/video conference applications generate both upstream and downstream flows, this problem should be resolved.

Furthermore, 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. In what follows, we specifically explain the two types of the unfairness. Figure 1 depicts the situations in which the unfairness occurs when the wireless network bandwidth is fully utilized.



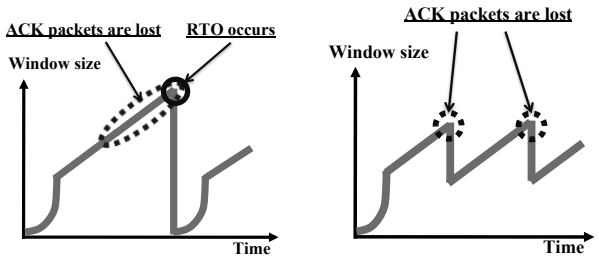
(a) Multiple upstream TCP flows exist

(b) Upstream and downstream TCP flows coexist

Fig. 1 Congestion at the AP

Suppose that each client station has an upstream TCP flow, as depicted in Fig. 1(a). In this situation, ACK packets of the upstream flows are discarded at the AP buffer. 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 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 as shown in Fig. 1(b), ACK packets of upstream flows and data packets of downstream flows are discarded at the AP buffer. In this situation, the upstream TCP flows continue to grow the congestion window size, whereas downstream TCP flows decrease the congestion window size, since TCP activates the congestion control mechanism only against data packet losses. Consequently, the downstream TCP flows maintain the low transmission rate, whereas the upstream TCP flows maintain the high transmission rate. Therefore, serious throughput unfairness occurs between upstream and downstream TCP flows.



(a) Without modification (b) Proposed method

Fig. 2 Behaviors of TCP Reno with and without the proposed method

Various solutions have been proposed for alleviating the above TCP unfairness [2]–[4], [6]–[10]. 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 [6], the author proposed shortening the carrier sense duration of the WLAN APs. A solution proposed in [9] improves TCP unfairness among upstream and downstream flows by controlling the rate of upstream flows such that total throughput should be divided equally between upstream and downstream flows. However, the cost of changing existing hardware devices using these methods is high because the MAC protocols or queue management mechanisms must be modified.

On the other hand, since transport-layer protocols are generally implemented through software, changing transport-layer protocols are easier than changing MAC-layer protocols. In addition, the above unfairness is mainly caused by the behaviors of TCP. Therefore, in the present paper, we propose a transport-layer solution for alleviating TCP unfairness.

3. TCP Congestion Control for ACK Packet Losses

Figure 2(a) depicts the behavior of TCP Reno when ACK packets are discarded at the AP buffer. The main reason for unfairness among TCP flows is that the TCP continues to increase the congestion window size even when the AP is highly congested and numerous ACK packets are discarded, as shown in Fig. 2(a). Therefore, we propose a simple modification to TCP congestion control mechanisms to alleviate these unfairness. The proposed method 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. Figure 2(b) shows the behavior of the proposed method on TCP Reno, corresponded to Fig. 2(a). The proposed method activates congestion control when detecting ACK packet losses, as shown in Fig. 2(b).

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:

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

where ack_loss is the number of ACK packet losses, ack is the sequence number (bytes) of the current received ACK packet, $prev\_ack$ is the sequence number (bytes) of the previous received ACK packet, 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 (*cwnd*) and slow-start threshold ($ssthresh$) is set to the halved congestion window size. Note that before halving the window size, the TCP sender waits for another RTT to avoid false detection caused by the disorder of data packet reception at the TCP receiver. Additionally, when detecting data packet losses, a TCP sender stops checking the ACK packet losses for one RTT.

When using Eq. (1) to detect ACK packet losses, the effect of the delayed ACK option [13] should be considered. Note that the delayed ACK option has been implemented in both Windows and Linux [14], [15]. Let b be the number of data packets acknowledged by a received ACK packet. Several TCP receiver implementations with the delayed ACK send one ACK packet for two consecutive received packets, so b is two typically. When the delayed ACK option is enabled, Eq. (1) cannot correctly estimate the number of ACK packet losses. Therefore, TCP senders must determine the value of b.

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, the TCP sender can obtain an accurate value of b, but this method requires modifications at TCP receivers. For this reason, we use the second method, which does not require TCP receiver-side modification.

In the proposed method, the value of b is estimated as follows:

$$b_{est} = \left\lfloor sb_i + \frac{1}{2} \right\rfloor \tag{2}$$

$$sb_i = (1-\beta)\, sb_{i-1} + \beta\left(\frac{ack - prev\_ack}{MSS}\right) \tag{3}$$

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. (3) is a continuous value, but b_{est} in Eq. (2) should be a discrete value because b should be a discrete value. Thus,

Algorithm 1 TCP congestion control for ACK packet losses

Initialization:
$prev\_ack \leftarrow$ the smallest sequence number of the unacknowledged packets (snd_una)
$sb \leftarrow MSS \times 2$, $b \leftarrow MSS \times 2$
$cnt\_ack\_loss \leftarrow 0$
$data\_loss \leftarrow 0$, $wait\_state \leftarrow 0$
On each ACK:
$sb \leftarrow (1-\beta)\, sb + \beta \left(\frac{ack - prev\_ack}{MSS}\right)$
$b_{est} \leftarrow \left\lfloor sb + \frac{1}{2} \right\rfloor$
$cnt\_ack\_loss \leftarrow \max\left(\left\lfloor \frac{ack - prev\_ack}{b_{est} \times MSS} \right\rfloor - 1 + cnt\_ack\_loss,\ 0\right)$
$prev\_ack \leftarrow ack$
On each RTT:
if $wait\_state$ **then**
  if $cnt\_ack\_loss \geq thresh\_ack\_losses$ and not $data\_loss$ **then**
    $cwnd \leftarrow \max\left(\frac{cwnd}{2},\ 1\right)$
    $ssthresh \leftarrow cwnd$
  end if
  $data\_loss \leftarrow 0$
  $cnt\_ack\_loss \leftarrow 0$
  $wait\_state \leftarrow 0$
else
  if $cnt\_ack\_loss \geq thresh\_ack\_losses$ or $data\_loss$ **then**
    $wait\_state \leftarrow 1$
  end if
end if
Packet loss:
$data\_loss \leftarrow 1$
Packet disorder:
$cnt\_ack\_loss \leftarrow \max(cnt\_ack\_loss - 1,\ 0)$

b_{est} is calculated by half-adjust rounding. Using Eq. (2), Eq. (1) is transformed into:

$$ack\_loss = \max\left(\left\lfloor \frac{ack - prev\_ack}{b_{est} \times MSS} \right\rfloor - 1,\ 0\right). \tag{4}$$

Algorithm 1 shows the pseudo-code of the proposed method. Note that the proposed method can be utilized with arbitrary TCP modifications because the proposed method can be applied to several TCP variants without any ill-effect.

In addition, because the proposed method is one of a TCP modification which detects early congestion of networks such as TCP Reno, the proposed method is ineffective when the proposed method and that TCP modifications coexist. Thus, the proposed method has the same issues of TCP modifications which have been reported in several researches [16], [17]. The deployment of the proposed method is future work.

4. Performance Metric for Evaluating Trade-off between Fairness and Utilization

4.1 Jain's Fairness Index

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]–[4], [6], [9], 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:

$$F_j(X) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \tag{5}$$

where n is the number of contending users, $X = \{x_1, x_2, \cdots, x_n\}$ is a set of allocations for n users such that x_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.

4.2 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. (5) 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 average squared distance between the each flow's throughput and the fair and fully-utilized throughput (x_f):

$$f(X, C) = \frac{1}{n} \sum_{i=1}^{n} (x_i - x_f)^2. \tag{6}$$

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

$$g(X, C) = \frac{\sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - x_f)^2}}{x_f}. \tag{7}$$

Table 1 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

According to [11], Jain's fairness index can be transformed into

$$F_j(X) = \frac{1}{1 + COV^2} \tag{8}$$

$$COV = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}}{\bar{x}} \tag{9}$$

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. (7) and (9), we obtain the following definition of the proposed index:

$$F(X, C) = \frac{1}{1 + g(X, C)^2} = \frac{C^2}{n\sum_{i=1}^{n} x_i^2 - 2C\sum_{i=1}^{n} x_i + 2C^2}. \tag{10}$$

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.

4.3 Comparison with Jain's Fairness Index

In Table 1, 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 1 have fair throughput distributions, but different total throughputs. The proposed index can differentiate these cases, whereas Jain's index cannot distinguish among them. As in 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 obtain much 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 Subsect. 4.2. 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 Subsect. 4.2.

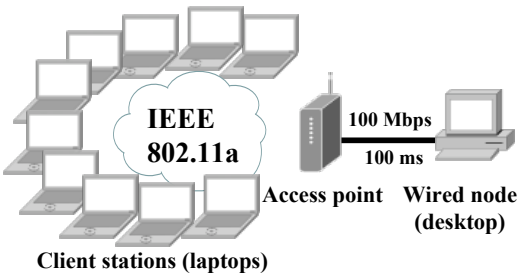


Fig. 3 Simulation environment

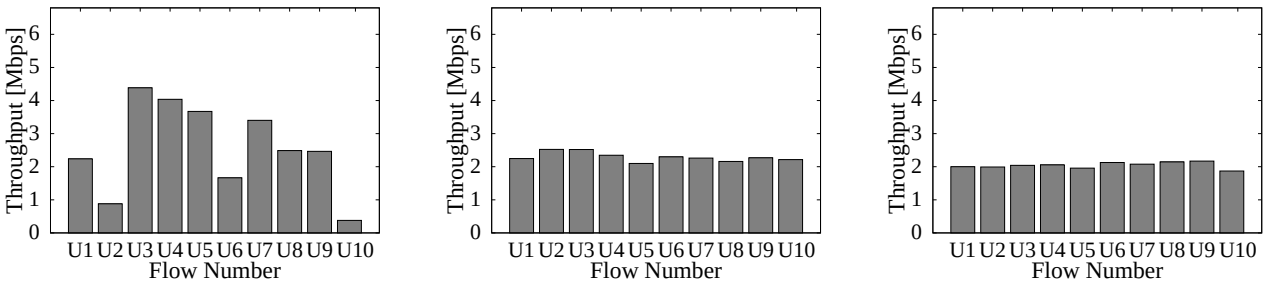
5. Evaluation with Simulation Experiments

In this section, we show simulation results in order to confirm the basic characteristics of the proposed method.

5.1 Simulation Settings and Methods

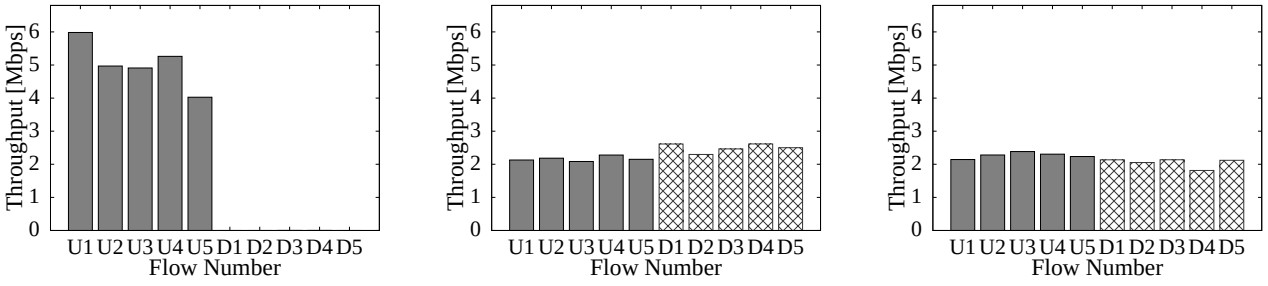
Figure 3 shows the simulation environment using IEEE 802.11a WLAN with the ns-2 simulator. In the environment, multiple client stations share one AP which is connected to a wired node through a wired link with 100 Mbps capacity and one-way propagation delay of 100 ms. In the simulation evaluation, except as otherwise noted, all stations were located at four meters from the AP and the buffer size of the AP was set to 100 packets. The buffer size of the wired link, the sender buffer size of each station, and the advertised receiver window size were set to large enough not to limit the TCP performance. As the queue management mechanisms, the Drop Tail principle was used in the AP and the wired link. TCP connections used either the proposed method on TCP Reno or conventional TCP Reno. In order to use these TCPs in the ns-2 simulator, we used the ns-2 modules which are converted from the Linux implementation codes by NS-2 TCP-Linux [18]. The *thresh_ack_losses* parameter in the proposed method was set to one. Note that we confirmed that we can obtain the best performance of the proposed method when the *thresh_ack_losses* parameter is one. In addition, for comparative purposes, the prioritised AP scheme proposed in [6] was also evaluated. In order to focus on fundamental characteristics, the delayed ACK option was disabled. The effects of the delayed ACK option are investigated in Sect. 6.

In the simulation experiments, only one flow was generated for each station, meaning that we increased the number of stations for increasing the number of concurrent flows in the network. The simulation time was set to 200 seconds and the data transmission of each flow started at random time which was uniformly distributed on [0 s, 10 s]. The simulations ran ten times in order to average the results.



(a) Without modification (b) Prioritized AP scheme [6] (c) Proposed method

Fig. 4 Average throughput of each flow when ten upstream flows exist



(a) Without modification (b) Prioritized AP scheme [6] (c) Proposed method

Fig. 5 Average throughput of each flow when five upstream flows and five downstream flows coexist

5.2 Evaluation Results and Discussion

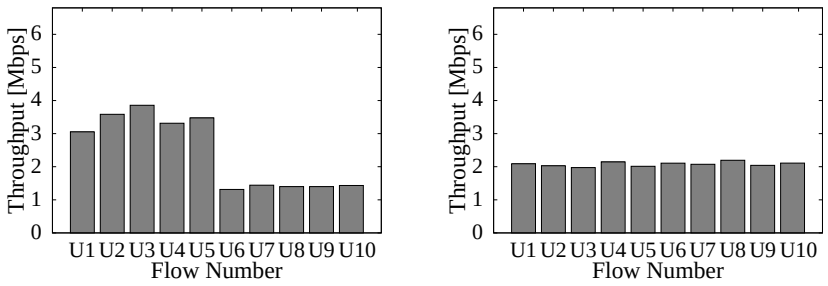
Figures 4 and 5 show the snapshot results for the average throughput when ten upstream flows exist and when five upstream flows and five downstream flows coexist, respectively. Note that the other results of ten times trials are also almost the same as Figs. 4 and 5. In these figures, Ui and Dj in x-axis denote the i th upstream flow and the j th downstream flow, respectively. The average throughput of each flow is calculated using the amount of data transmitted in 50-200 seconds in the simulation. Figures 4(a) and 5(a) shows the serious TCP unfairness occurs among upstream flows and between upstream and downstream flows without modification, whereas the prioritized AP scheme and the proposed method can successfully alleviate TCP unfairness both among upstream flows and between upstream and downstream flows as shown in Figs. 4(b), 4(c), 5(b), and 5(c).

Figure 6 represents the results when there are ten upstream flows and the five stations are located at one meter and the others are located ten meters from the AP. Note that we obtained the similar results when upstream and downstream flows coexist. The prioritized AP scheme degrades the effectiveness for alleviating unfairness among flows dependently on distance of the stations from AP, as shown in Fig. 6(a). This is because that the prioritized AP scheme is sensitive to the wireless channel environment since it is based on MAC parameter tuning regarding the channel access. On the other hand, the performance of the proposed method is independent on the wireless channel since the proposed method is based on a transport-layer approach.

Figure 7 plots the average RTT of each flow corresponded to the results in Figs. 4 and 5. In Fig. 7, RTTs of each flow with the proposed method are smaller than without modification or with the prioritized AP scheme. The reason for this is as follows. Without modification, the congestion window size of upstream flows continues to grow and the packets exceeded the data rate of wireless channel are buffered at the sender buffer of each station. This results in that RTTs of each flow become large. Likewise, with the prioritized AP scheme, RTTs of each flow also become large because the prioritized AP scheme can improve fairness among flows, but the congestion window size of upstream flows remain to grow when the wireless channel is fully utilized. In contrast, the proposed method can also alleviate the congestion at the wireless channel because upstream flows with the proposed method activate the congestion control. Therefore, the proposed method can maintain low RTT.

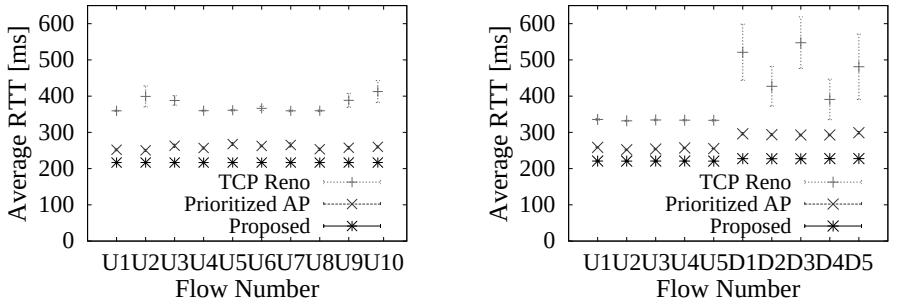
6. Experimental Evaluation

In this section, we present experimental results using real environments with WLAN products from several vendors in order to confirm the applicability and product-dependent characteristics of the proposed method in real environments.



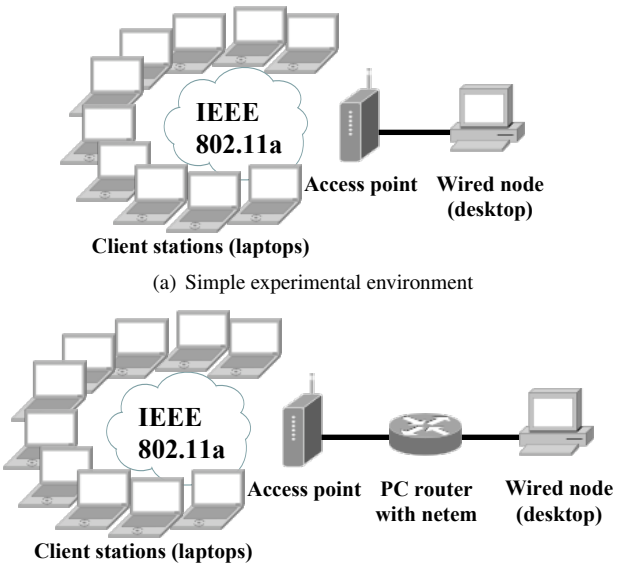
(a) Prioritized AP scheme [6] (b) Proposed method

Fig. 6 Average throughput of each flow when ten upstream flows exist and stations are relocated



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

Fig. 7 Average RTT of each flow



(a) Simple experimental environment

(b) Experimental environment with a PC router

Fig. 8 Experimental environments

Table 2 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

6.1 Experimental Settings and Methods

Two experimental environments are shown in Fig. 8. In both environments, ten client stations share one AP. All client stations were located within 50 cm of the AP in order to avoid packet losses due to wireless link error. In Fig. 8(a), a wired node is directly connected to the AP through a wired link. On the other hand, in the experimental environment in Fig. 8(b), we introduced 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 were used as the client stations and the wired node, respectively. All nodes, including the wired node, used Ubuntu 8.10 (Linux kernel 2.6.28) as the OS. As shown in Fig. 8(b), another DELL Precision 390 desktop was used as the PC router with netem [19] for generating a 50 ms delay to the wired link between the AP and the wired node. We used Web100 [20] 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 2 were used as wireless interface cards for client stations and the AP. Note that all client stations used the same type of wireless interface card in each experiment. Due to space limitation, we

Table 3 Estimated buffer size of each AP

Vendor	$minRTT$ [ms]	T [Mbps]	$cwnd_{overflow}$ [bytes]	B_{est} [packets]
Buffalo	0.975	21.8	118003	76.9
NEC	0.563	19.6	76719	50.2
Corega	0.674	20.2	366826	243.4

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*, respectively, e.g., Buffalo-AP.

The buffer size of an AP significantly affects on flow's throughput and end-to-end delay [21]. However, AP vendors do not make publish the buffer size in detail. For the reason, we estimated the buffer size through simple experiments with a single TCP connection, with an estimation equation as follows:

$$B_{est} = \left(cwnd_{overflow} - \frac{T \cdot minRTT}{8}\right)/MSS \qquad (11)$$

where B_{est} is the estimated buffer size (packets) of an AP, $cwnd_{overflow}$ is the congestion window size (bytes) of a single TCP connection when TCP detects data packet losses, regarding it as buffer overflow occurs at the AP, T is throughput (bps) just before TCP detects data packet losses, and $minRTT$ is a minimum RTT (seconds) during the experiments. Table 3 presents the results of estimated buffer size of each AP. Note that the experiments were conducted ten times for averaging the results. In Table 3, the buffer size of Corega-AP is the largest of all.

The experiments using the environments in Fig. 8 were conducted as follows. Only one TCP flow was generated for each client station using Iperf [22], assuming bulk data transfer. We kept 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 used either the proposed method on TCP Reno or conventional TCP Reno. As in Sect. 5, the *thresh_ack_losses* parameter in the proposed method was set to one. The experiment time was set to 180 seconds, and each TCP connection was generated almost simultaneously when the experiment starts. More specifically, a computer separated from the experimental network in Fig 8, connected to each stations and the wired node through a separate wired network, simultaneously executed Iperf programs through the separate wired network, in order to generate a TCP connection at each station and the wired node. We disabled vendor-specific functions implemented at APs, such as *Frame Burst* in Buffalo-AP. For each experimental setting, the experiments were conducted ten times in order to average the results.

We first conducted experiments using the environment shown in Fig. 8(a) in which the delayed ACK option was disabled at TCP receivers. In the experiments, the proposed method used the equation in Eq. (1) in order to estimate the number of ACK packet losses. On the other hand, in order to investigate the effects using the equation in Eq. (4), we then conducted experiments using the environment in Fig. 8(b) with and without the delayed ACK option. In the situation, the smoothing factor β in Eq. (3) was set to $1/32$.

6.2 Evaluation Metric

The effectiveness of the proposed method is evaluated from the viewpoint of fairness and bandwidth utilization by assessing the throughput of each flow, and the index proposed in Sect. 4.

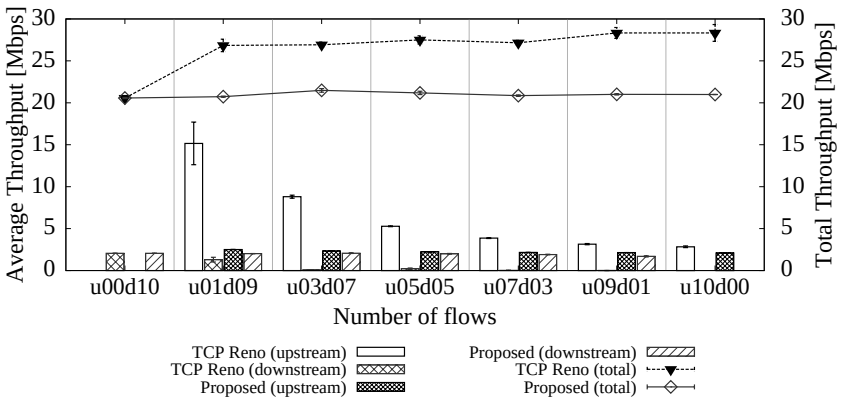
In order to evaluate fairness and trade-off relationships between fairness and bandwidth utilization, Sliding Window Method (SWM) function [23] is applied to the proposed index. The SWM can give a quantitative measure of an arbitrary metric over a wide range of time scales. Especially, when using the SWM to measure fairness, it has an advantage of measuring *short-term fairness* and *long-term fairness* at the same time. Intuitively, short-term fairness of a data transmission flow refers to its ability to provide equitable access to resources to all the 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 which applied to Eq. (10) is as follows:

$$SWM(w) = \frac{C^2}{n\sum_{i=1}^{n} x_i(w)^2 - 2C\sum_{i=1}^{n} x_i(w) + 2C^2} \qquad (12)$$

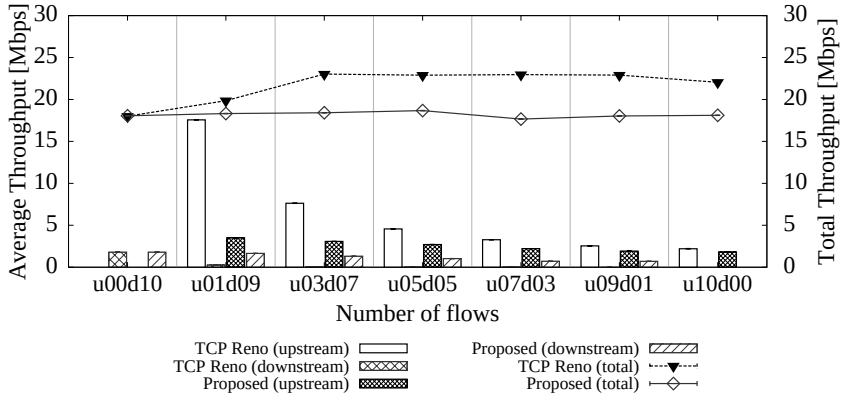
where w is a time-window size for evaluating the fairness, and $x_i(w)$ is an average throughput of flow i in a time-window w. In the present paper, parameter C in Eq. (12) is set to 29.60 Mbps according to the theoretical maximum throughput of IEEE 802.11a WLAN with 1460 bytes MTU [24].

6.3 Evaluation Results and Discussion

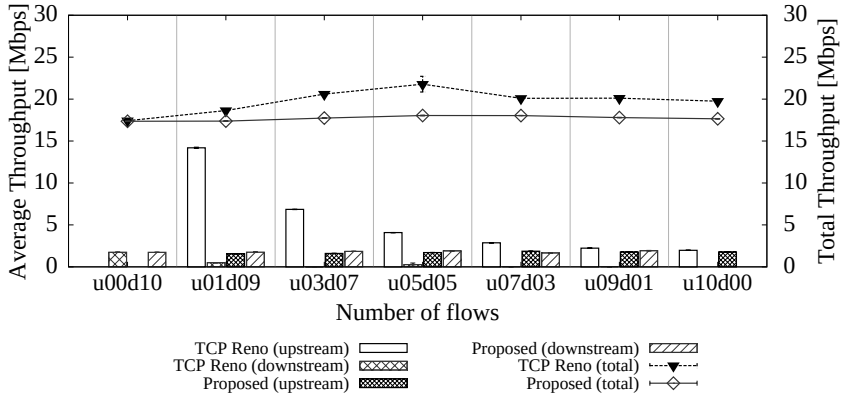
Figure 9 presents the average throughput of upstream and downstream flows and the total throughput for various ratios of upstream and downstream flows and three types of APs. In the figure, uxdy in x-axis denotes that the number of upstream and downstream TCP flows are x and y, respectively. The lines in these figures represent total throughput, and the bars represent average throughput of upstream and downstream flows, respectively. When the bars are the same height, it means the perfect fairness between upstream and downstream flows. The average throughput of each flow is calculated using the amount of data transmitted in 50-180 seconds in the experiment. Figure 9 shows that, when at least one upstream flow with TCP Reno 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 no flow is starved. Figure 9 reveals that the degree of fairness improvement of the proposed method is small when using NEC-AP. The reason for this is as follows. Flows with the proposed method do not



(a) Buffalo-AP



(b) NEC-AP



(c) Corega-AP

Fig. 9 Effect of the number of upstream and downstream flows

experience RTO when using Buffalo-AP or Corega-AP. In contrast, flows with the proposed method experience RTO when using NEC-AP because the buffer size of NEC-AP is too small for ten flows to flow. However, even when using NEC-AP, the proposed method can avoid starvation of flows.

In terms of total throughput, the total throughput of ten downstream flows with TCP Reno and with the proposed method are equivalent, regardless of type of AP. 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, comparing the total throughput of the proposed method and TCP Reno when there exist one or more upstream flows, the former is smaller than the latter. The reason for this is as follows. When the proposed method is not used, some 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 the 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.

Figures 10 and 11 show the evaluation results obtained using the proposed metric with SWM when ten upstream flows exist and when five upstream flows and five downstream flows coexist, respectively, under the same conditions as Fig. 9. 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. On the other hand, 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 as shown in Table 3. More specifically, as more the buffer size of an AP becomes large, the more packets of each flow are maintained at the AP. Because this leads the number of RTO occurred to reduce, the degree of unfairness is slightly alleviated.

In order to investigate the effect of using the delayed ACK option and using the equation in Eq. (4), we conducted experiments with and without the delayed ACK option under the environment shown in Fig. 8(b). Figure 12 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. 8(b) using Buffalo-AP with a 50 ms delay. In Fig. 12, the results obtained using and without using the delayed ACK option are labeled as *delack* and *nodelack*, respectively. From the viewpoint of fairness, 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 with or without the delayed ACK option. Furthermore, both the total throughput of TCP Reno and the proposed method increase when using the delayed ACK option. This is because the delayed ACK option decreases the number of ACK packets in the WLAN and that consequently increases the number of data packets injected into the WLAN. Therefore, when using the delayed ACK option, the proposed method can enhance the total throughput without degrading the improvement in fairness.

Figure 13 shows the evaluation results obtained using the proposed index under the same conditions as Fig. 12. Comparing the index values of the proposed method with and without using the delayed ACK option, the index value with the delayed ACK option is better than that without the delayed ACK option. This means that, in the proposed method, the improvements in bandwidth utilization leads to the improvement in the index values.

7. Conclusion

In the present paper, we first proposed a transport-layer solution for alleviating TCP unfairness in a WLAN environment. We then proposed a novel performance metric for evaluating the trade-off relationship between fairness and bandwidth utilization at a network bottleneck. The proposed index is based on the variations in throughput of concurrent flows and the ideal throughput distribution, in which all flows achieve the same throughput and the network bandwidth is fully utilized.

In order to confirm the basic characteristics of the proposed method, we conducted simulation experiments. Based on the results of the simulation experiments, we confirmed that the proposed method can alleviate TCP unfairness among upstream flows and between upstream and downstream flows while maintaining low RTT. Through extensive experiments using real WLAN environments with the products from several vendors, we then confirmed that the proposed method alleviates TCP unfairness regardless of the vendor of the APs and wireless interface cards. Moreover, the proposed method with 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, June 2007.
[2] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, "Understanding TCP fairness over wireless LAN," Proc. INFOCOM 2003, pp.863–872, March 2003.
[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," Proc. ICC 2007, pp.4512–4517, June 2007.
[4] J. Ha, E.C. Park, K.J. Park, and C.H. Choi, "A cross-layer dual queue approach for improving TCP fairness in infrastructure WLANs," Wireless Personal Commun., vol.51, no.3, pp.499–516, June 2009.
[5] S. Floyd, "The NewReno modification to TCP's fast recovery algorithm," RFC 3782, April 2004.
[6] Y. Fukuda and Y. Oie, "Unfair and inefficient share of wireless LAN resource among uplink and downlink data traffic and its solution," IEICE Trans. Commun., vol.E88-B, no.4, pp.1577–1585, April 2005.
[7] S.W. Kim, B.S. Kim, and Y. Fang, "Downlink and uplink resource allocation in IEEE 802.11 wireless LANs," IEEE Trans. Vehicular Tech., vol.54, no.1, pp.320–327, Jan. 2005.
[8] S. Gopal and D. Raychaudhuri, "Experimental evaluation of the TCP simultaneous-send problem in 802.11 wireless local area networks," Proc. the 2005 ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis, pp.23–28, Aug. 2005.
[9] N. Blefari-Melazzi, A. Detti, I. Habib, A. Ordine, and S. Salsano,

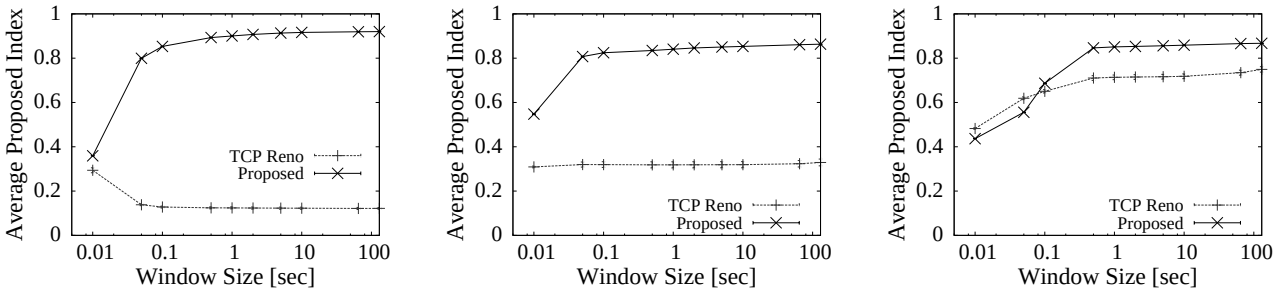


Fig. 10 Proposed index with SWM when ten upstream flows exist

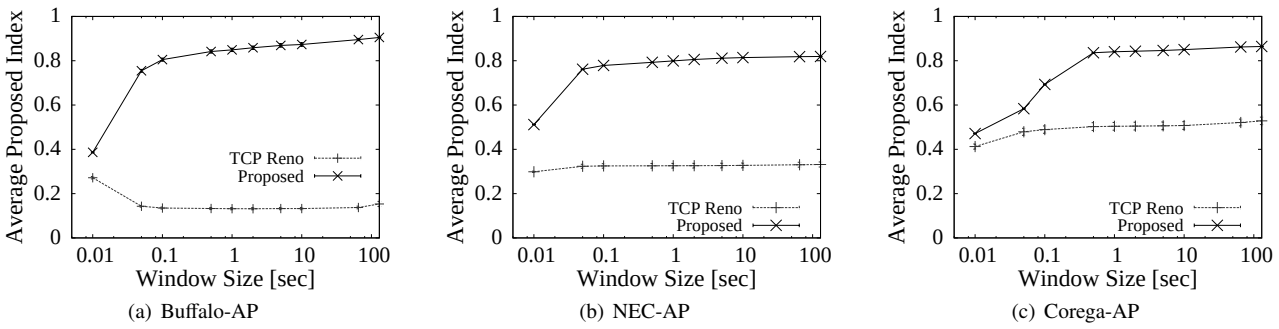


Fig. 11 Proposed index with SWM when five upstream flows and five downstream flows coexist

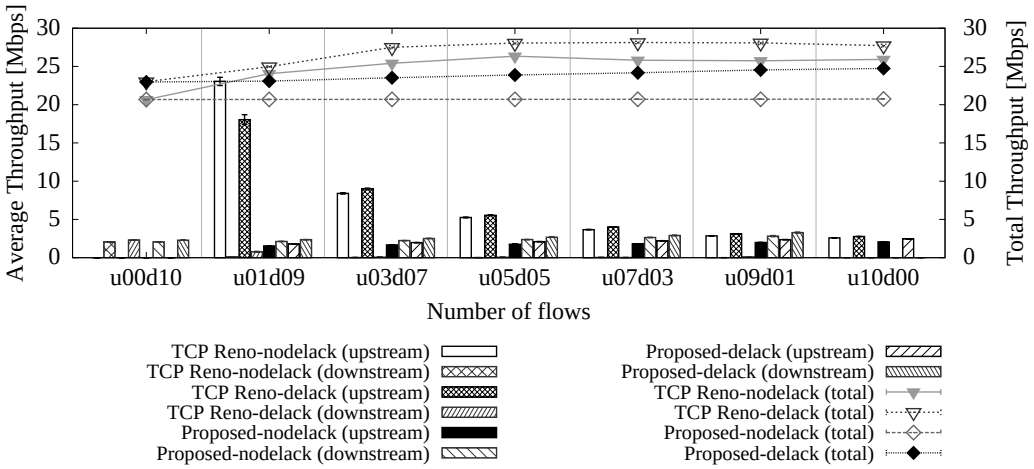


Fig. 12 Effect of the delayed ACK option when using Buffalo-AP with a 50 ms one-way delay

"TCP fairness issues in IEEE 802.11 networks: Problem analysis and solutions based on rate control," IEEE Trans. on Wireless Commun., vol.6, pp.1346–1355, April 2007.

[10] B.A.H.S. Abeysekera, T. Matsuda, and T. Takine, "Dynamic contention window control mechanism to achieve fairness between uplink and downlink flows in IEEE 802.11 wireless LANs," IEEE Trans. Wireless Commun., vol.7, no.9, pp.3517–3525, Sept. 2008.

[11] 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.

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

[13] R. Braden, "Requirements for internet hosts – communication layers," RFC 1122, Oct. 1989.

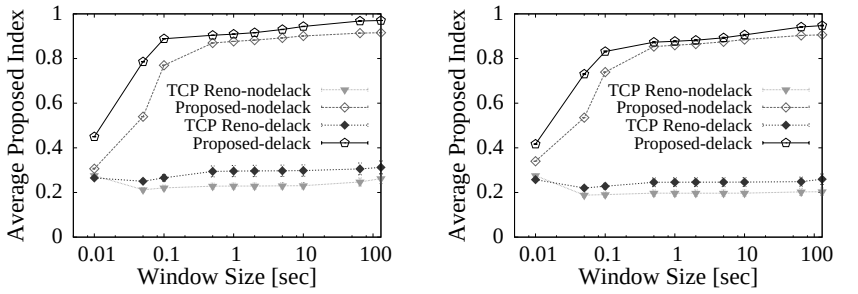
[14] Microsoft Corporation, Microsoft Windows Server 2003 TCP/IP Implementation Details, June 2003.

[15] P. Sarolahti and A. Kuznetsov, "Congestion control in linux TCP," Proc. USENIX Annu. Tech. Conf., pp.49–62, June 2002.

[16] J. Mo, R.J. La, V. Anantharm, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," Proc. INFOCOM 1999, pp.1556–1563, March 1999.

[17] G. Hasegawa, K. Kurata, and M. Murata, "Analysis and improvement of fairness between TCP reno and vegas for deployment of TCP vegas to the Internet," Proc. IEEE ICNP 2000, Nov. 2000.

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



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

Fig. 13 Proposed index with SWM when using Buffalo-AP with a 50 ms one-way delay

[19] netem. available at `http://www.linuxfoundation.org/en/Net:Netem`.

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

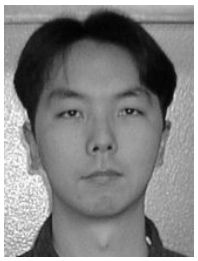
[21] F. Li, M. Li, R. Lu, H. Wu, M. Claypool, and R. Kinicki, "Measuring queue capacities of IEEE 802.11 wireless access points," Proc. BROADNETS 2007, pp.846–853, Sept. 2007.

[22] 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/`.

[23] C.E. Koksal, H. Kassab, and H. Balakrishnan, "An analysis of short-term fairness in wireless media access protocols," Proc. ACM Sigmetrics 2000, June 2000.

[24] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," Proc. NCA 2003, pp.249–256, April 2003.

Masafumi Hashimoto received the M.E. degree in Information Science and Technology from Osaka University, Japan, in 2010. He is now a D.E. candidate at Graduate School of Information Science and Technology, Osaka University. His research work is in the area of transport architecture for future high-speed networks.



Go Hasegawa received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1997 and 2000, respectively. From July 1997 to June 2000, he was a Research Assistant of Graduate School of Economics, Osaka University. He is now an Associate Professor of Cybermedia Center, Osaka University. His research work is in the area of transport architecture for future high-speed networks and overlay networks. He is a member of the IEEE.

Masayuki Murata received the M.E. and D.E. degrees in Information and Computer Science from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. In April 1999, he became a Professor of Cybermedia Center, Osaka University, and is now with Graduate School of Information Science and Technology, Osaka University since April 2004. He has more than five hundred papers of international and domestic journals and conferences. His research interests include computer communication network architecture, performance modeling and evaluation. He is a member of IEEE, ACM and IEICE. He is a chair of IEEE COMSOC Japan Chapter since 2009. Also, he is now partly working at NICT (National Institute of Information and Communications Technology) as Deputy of New-Generation Network R&D Strategic Headquarters.