# Adaptive Pushout: A Buffer Management Scheme to Improve TCP Fairness in Wireless LANs

Kazushige Hayashi*, Shigeo Shioda†, Nobuyoshi Komuro‡, Shiro Sakata‡ and Tutomu Murase§
*Faculty of Engineering, Chiba University, 1-33, Yayoi, Inage, Chiba 263-8522, Japan
†Graduate School of Engineering, Chiba University
1-33, Yayoi, Inage, Chiba 263-8522, Japan, E-mail: shioda@faculty.chiba-u.jp
‡Graduate School of Advanced Integration Science, Chiba University, 1-33, Yayoi, Inage, Chiba 263-8522, Japan
§NEC System Platforms Research Laboratories, 1753, Nakahara, Kawasaki 211-8666, Japan

***Abstract*—We propose a buffer management scheme, called *adaptive pushout*, in order to solve unfairness problems between competing TCP flows over wireless LANs. The adaptive pushout, applied to the transmission buffer of the access point (AP), relies on the fact that multiple ACK segments of a TCP connection are redundant because of the cumulative acknowledgment mechanism of the TCP. Under the adaptive pushout, an arriving packet encountering the full buffer removes one of redundant ACKs from the transmission buffer of the AP to enter there. When the buffer is full and has no redundant ACK, the arriving packet adaptively chooses one of packets in the buffer for pushing it out from the buffer. The choice of the packet to be pushed out depends on the number of uplink and downlink TCP flows, which are continuously monitored in the proposal. We have conducted simulation experiments to show that the adaptive pushout greatly improves the fairness compared with existing proposals.**

## I. Introduction

Wireless local area networks (WLANs) based on the IEEE 802.11 standard have been increasingly used as high speed and convenient access technologies to the Internet. The fundamental building block of the 802.11 WLANs is the basic service set (BSS), in which an access point (AP) serves as a gateway between wired and wireless domains. In the Distributed Coordination Function (DCF), which is a contention based Medium Access Control (MAC) mechanism defined by the IEEE 802.11 standard, the AP has the same access priority as each individual wireless station. However, the AP needs to serve all downlink flows, while each wireless station serves at most a few flows. This yields the bandwidth asymmetry of the WLAN; each individual downlink flow gets lower bandwidth than each individual uplink flow gets.

It is well known that the bandwidth asymmetry causes serious problems on feedback-based transport protocols such as TCP [1], [2], [3], [4], [5]. In this paper, we focus on two types of unfairness problems caused by the bandwidth asymmetry. One is the unfairness between data-bottlenecked and ACK-bottlenecked flows; here, *data (ACK) bottlenecked* means the situation, in which the stream of data (ACK) segments obtain much smaller bandwidth than that of ACK (data) segments. Note that uplink TCP flows on a WLAN are ACK bottlenecked, while downlink TCP flows on a WLAN are data bottlenecked. To see the unfairness between data- and ACK-bottlenecked flows, observe that any received ACK segments can cumulatively acknowledge all the data segments that have been sent before the data, the receipt of which the ACK segment is intended to acknowledge. Thus, the loss of a data segment readily reduces the congestion window while the loss of an ACK segment may not. As a result, ACK-bottlenecked flows are likely to get higher throughput than data-bottlenecked flows.

The other is the unfairness among ACK-bottlenecked flows. Since TCP ACKs are cumulative, flows with large congestion windows will not experience timeouts even when ACK segments are frequently lost. On the other hand, flows with small congestion window are likely to experience frequent timeouts and thereby decrease their congestion window even more. This fact brings about the unfairness among ACK-bottlenecked flows.

Although a large number of studies have been made on the fairness issues between competing TCP flows [3], [5], [6], [7], [8], [9], [10], [11], [12], only few attempts have been made at solving both types of unfairness problems simultaneously. We previously proposed a buffer management scheme [13], called the *ACK pushout*, for use in the transmission buffer of the AP to improve the two types of unfairness problems simultaneously. The ACK pushout comes from the understanding that multiple ACK segments in a TCP connection are redundant because of the cumulative acknowledgement mechanism of the TCP. Under the operation of the ACK pushout, when an arriving packet finds that the transmission buffer of the AP is fully occupied, one of redundant ACK segments is removed from the buffer and the arriving packet enters there.

The ACK pushout drops an arriving data segment if the transmission buffer is full and has no redundant ACK. An arriving ACK encountering the same situation is, however, able to enter the buffer if there are any other ACKs of the same connection in the buffer. Thus, data segments are more likely to be dropped than ACK segments. Because of this asymmetry (we call it *packet-drop asymmetry* in this work), under the ACK pushout, uplink TCP flows still get larger throughput than downlink TCP flows. To have better fairness, in this work, we make a simple but essential improvement on the ACK pushout. Our new algorithm, which we call the *adaptive pushout* in this work, chooses one of packets in the buffer for pushing it out from the buffer when the buffer is fully occupied

978-1-4673-0990-5/12/$31.00 ©2012 IEEE

and there is no redundant ACK there. The choice of the packet to be pushed out depends on how many uplink and downlink flows respectively use the transmission buffer of the AP. For example, if more uplink flows use the buffer than downlink flows when the numbers of active uplink and down flows are the same, a packet of uplink flows (that is, an ACK segment) in the buffer is randomly chosen to be pushed out. We show that the adaptive pushout yields better fairness than the ACK pushout as well as other existing proposals for a wide range of traffic mix conditions.

The rest of the article is organized as follows. In Section II, we present some previous works on the unfairness problems in WLANs. In Section III, we describe the detailed algorithm of our proposal. The performance evaluation via the simulation experiments is described in Section IV. Finally we provide our concluding remarks in Section V.

## II. Related Studies

A simple and intuitive solution for the unfairness problems caused by the bandwidth asymmetry is to make bandwidth-asymmetric links symmetric ones. Such a direct solution is possible for WLANs by adjusting several MAC parameters including the contention windows or AIFS. Hirantha *et al.* [14] proposed to decrease $CW_{min}$ of the access point (AP) so that the AP can get more chances to access wireless channel. Cali *et al.* [15] also proposed the dynamical tuning of the contention windows of stations to achieve the theoretically achievable throughput. These approaches, however, require online adaptation of $CW_{min}$, depending on the traffic mix of uplink and downlink flows.

Since the unfairness between competing TCP flows over a WLAN is caused by the congestion of the transmission buffer of the AP, several buffer management schemes have been proposed to apply there. Ha *et al.* [7] proposed to employ two queues, one for the data packets of downlink TCP flows and another for the ACK packets of uplink TCP flows, and to give higher priority to the data queue over the ACK queue by taking advantage of the fact that TCP ACKs are cumulative. Wu *et al.* [12] and Lin *et al.* [9] proposed per flow queueing to address fairness problem. Pilosof *et al.* [10] proposed a different approach; in their method, the AP manipulates the TCP advertised window field of ACK packets to decrease the window sizes of ACK-bottlenecked flows so that the buffer at the AP will not be fully occupied. Note that all of these proposals focus only on the unfairness between uplink and downlink flows. Keceli *et al.* [8] proposed an ACK congestion control and filtering algorithm that reduces the number of ACK segments sent over the wireless channel to decrease the collision with data packets. Their proposal is addressed only to the unfairness among uplink flows.

SPM-AF proposed by Wu *et al.* [11] is a combined buffer management scheme of ACK Filtering (AF) [2] and Selective Packet Marking (SPM) [16]. In the AF, the buffer is scanned to check if there are any other ACKs for the same connection when an ACK segment arrives at the transmission buffer. If so, all of these previous ACKs are removed from the buffer, relying on the cumulative acknowledgement nature of the newly stored ACK to supersede the information in the previously stored (but now removed) ACKs. The SPM gives priority to data segments of TCP connections whose congestion windows are small. When an arriving packet assigned priority encounters a full queue, it pushes one of non-priority packets out from the buffer to enter there. The SPM-AF can improve the two types of unfairness between competing flows simultaneously, but it has one side effect; it slows down the growth of sender's congestion window, so the change of the TCP congestion control mechanism is desirable.

ACK pushout [13] is also a buffer management scheme that improves the two types of unfairness problems. In the ACK pushout, an arriving packet encountering the fully-occupied buffer removes one of redundant ACK segments from the buffer and joins the queue in the buffer. The ACK pushout is similar to the AF, but the former drops fewer ACKs than the latter, which mitigates the side effect of the SPM-AF (slowdown of the growth of sender's congestion window).

## III. Adaptive Pushout

### A. Algorithm

As we mentioned in Section I, because of the *packet-drop asymmetry* of the ACK pushout, data segments are more likely to be dropped than ACK segments, and thus uplink TCP flows obtain larger throughout than downlink TCP flows still under the ACK pushout.

To remove the packet-drop asymmetry, we propose a new algorithm, *adaptive pushout*, which *fairly* choose a packet in the buffer for pushing it out when an arriving packet encounters the fully-occupied buffer with no redundant ACK. The chosen packet is dropped from the buffer to store the arriving packet. Note that the pushout operation is conducted upon arrivals of ACK segments as well as data segments when they encounter the fully-occupied buffer with no redundant ACK. We also note that, if there is one or more redundant ACKs in the buffer, the adaptive pushout removes one of redundant ACKs from the buffer as the ACK pushout does.

The choice of the packet to be pushed out should depend on the traffic condition. In the adaptive pushout, the number of uplink TCP flows $n_{up}$ and the number of downlink TCP flows $n_{down}$ are continuously monitored. When an arriving packet finds the fully-occupied buffer with no redundant ACK, the adaptive pushout randomly chooses a data segment and remove it from the buffer to store the arriving packet if $\frac{nb_{up}}{n_{up}} \leq \frac{nb_{down}}{n_{down}}$; otherwise it randomly chooses an ACK segment and remove it from the buffer. Here $nb_{down}$ ($nb_{up}$) is the number of downlink (uplink) flows whose packets are stored in the buffer.

In Fig. 1, we show examples of typical operation of the adaptive pushout. In the figure, the number of uplink flows and that of downlink flows are the same ($n_{up} = n_{down} = 3$). In Fig. 1(a), $nb_{up} = 3$ and $nb_{down} = 2$, and thus an ACK segment is randomly selected to be pushed out from the buffer. In Fig. 1(b), $nb_{up} = 2$ and $nb_{down} = 3$, and thus an data segment is randomly selected to be pushed out from the buffer.

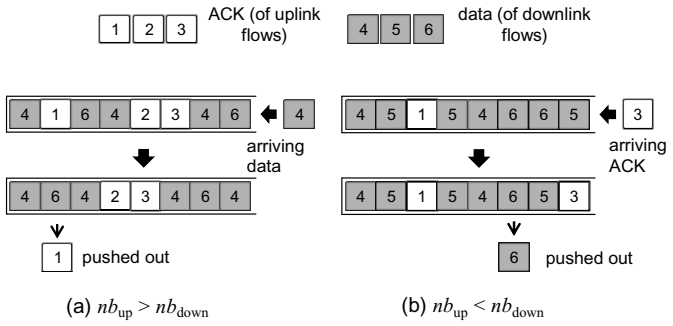


Fig. 1. Adaptive pushout

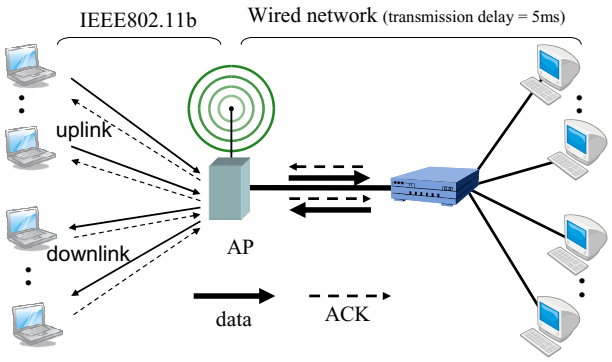


Fig. 2. Simulation condition (WLAN)

To know $n_{up}$, $nb_{up}$, $n_{down}$, and $nb_{up}$, the adaptive pushout needs to keep the states of all active TCP flows, which are identified based on the IP 5-tuple (source and destination IP addresses, and source and destination port numbers, and protocol number). Although keeping the states of flows incur some overhead to the AP, the overhead is not significant because the number of flows in a single WLAN would be less than a hundred and the physical-layer bandwidth is less than 100 Mbps.

### B. Comparison with the existing techniques

Since the adaptive pushout improves the *packet drop asymmetry*, which is a drawback of the ACK pushout, the adaptive yields better fairness than the ACK pushout. In addition to this, through simulation experiments (Section IV), we have confirmed that the adaptive pushout achieves better fairness than the SPM-AF. We also note that the adaptive pushout drops fewer ACKs than the SPM-AF does, which would curb the side effect of the SPM-AF (slowdown of the growth of sender's congestion window).

Another advantage over the existing proposals (ACK pushout and SPM-AF) is the applicability to asymmetric cases, where the number of uplink TCP flows is different from the number of downlink TCP flows. The adaptive pushout monitors the numbers of uplink and downlink flows to decide the choice of packets to be pushed out, which makes the adaptive pushout applicable to asymmetric cases. The ACK pushout and the SPM-AF do not have such traffic monitoring functions.

## IV. Numerical Examples

### A. Simulation Conditions

To see the performance of the adaptive pushout, we conducted simulation experiments using the network simulator ns-2 (version 2.34) under the scenario where multiple uplink and downlink TCP flows were multiplexed over the WLAN (Fig. 2). In the simulation, each wireless station sent (received) a large bulk data over an uplink (downlink) TCP flow to (from) a peer station located in the wired network. A TCP receiver acknowledged each data packet with a TCP ACK. The size of TCP data packet was 1500 bytes and the TCP congestion window advertized by receiver was set to 42 packets. The version of TCP used in the simulations was TCP Reno. The buffer size of the AP was 100 packets. All the stations had IEEE 802.11b physical layer with data rate equal to 11 Mbps. The transmission delay between the AP to each station in the wired network was 5 ms and the bandwidth of the wired line was 100 Mbps. The wired link had much larger bandwidth than wireless link so that no congestion would occur in the wired link. The parameters of the IEEE 802.11b (DCF) were set at default values (Table I). The run time of each simulation was 600 s besides the initial bias (150 s).

TABLE I
System parameters used in simulations.

| | |
|---|---|
| MAC header | 240 bits |
| LLC header | 64 bits |
| FCS | 32 bits |
| PHY header | 192 bits |
| ACK | 112 bits + PHY header |
| Basic rate | 1.0 Mbps |
| Data rate | 11.0 Mbps |
| Slot time | 20 $\mu$s |
| SIFS | 10 $\mu$s |
| DIFS | 50 $\mu$s |

In the simulation, the following four different buffer management schemes were applied to the transmission buffer of the AP: AF (ACK filtering), SPM-AF, ACK pushout, and the adaptive pushout. The difference of AF and SPM-AF was in the differentiation of the data and ACK segments; data and ACK segments had the same priority in the AF, while data segments had higher priority than ACK segments in the SPM-AF. Thus, in the SPM-AF, an arriving data segment encountering the full buffer removed an ACK segment and joined the queue in the buffer.

*Remark 1:* The original SPM-AF [11] assigns priority to only data segments in TCP connections whose congestion windows are smaller than 4 packets. In actual situations, however, it is difficult for the AP to know the congestion window of each connection. So, in this work, we assumed that all data segments have priority over ACK segments under the SPM-AF.

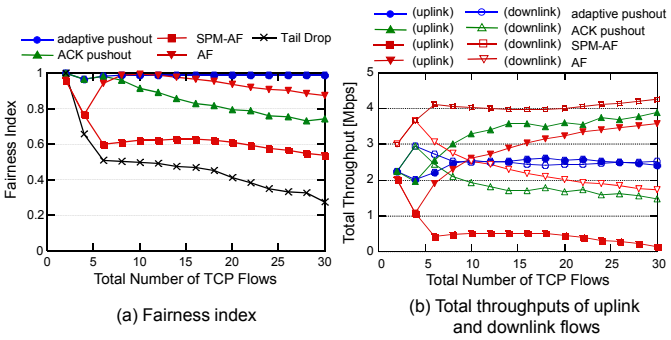


Fig. 3. Fairness Index and throughput (symmetric case)

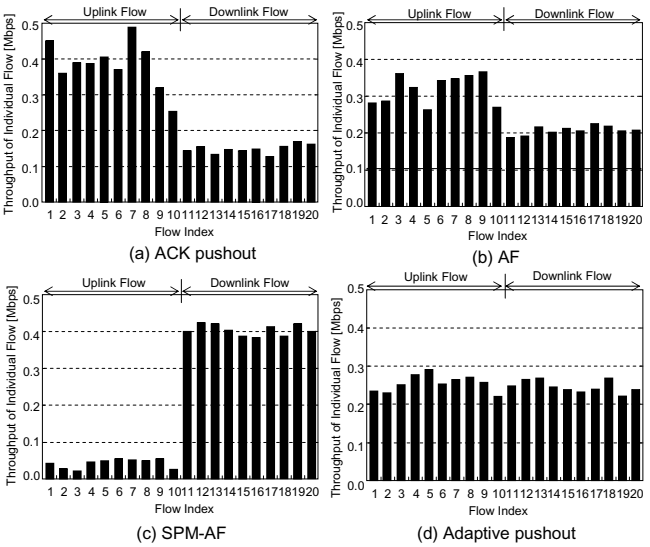


Fig. 4. Throughput of each flow in symmetric case

### B. Symmetric traffic cases

First we conducted simulation experiments in symmetric traffic cases where the number of uplink flows ($n_{up}$) was equal to that of down link flows ($n_{down}$). Figure 3(a) shows the fairness index defined below [17]

$$\text{Fairness Index} = \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)^2 \Big/ \left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right),$$

where $N$ is the total number of TCP flows, and $x_i$ denotes the throughput of the $i$th connection. Note that the fairness index, ranging from 0 to 1, approaches 1 as the resource is more fairly shared. All of the buffer management schemes showed better fairness than the tail drop, and the adaptive pushout showed the best performance. Among the four schemes, SPM-AF was the worst because SPM-AF assigned priority to data segments over ACK segments in the simulation, which largely lowered the throughputs of all uplink flows. The AF outperformed the ACK pushout; the former drops all redundant ACKs, which sometimes triggers the timeouts of uplink flows. It would mitigate the *packet-drop asymmetry*, which is a drawback of the ACK pushout (see Section III-A).

Figure 3(b) compares the sum of throughputs of uplink flows with that of downlink flows. Under the ACK pushout or the AF, uplink flows obtained larger throughput than downlink flows, while under the SPM-AF downlink flows obtained larger throughput than uplink flows. Under the operation of the adaptive pushout, the uplink and downlink flows obtained almost the same throughputs.

In Fig. 4, we show the throughput for each of 20 flows in a simulation run when the total number of TCP flows was 20. Flows 1 to 10 were uplink flows and flows 11 to 20 were downlink flows. These figures also confirm that the adaptive pushout solves the two types of unfairness problems (one among uplink flows and one between uplink and downlink flows) in the most satisfactory way.

### C. Asymmetric traffic cases

Next, we investigated the performance of the four buffer management schemes in asymmetric traffic cases, where the numbers of uplink and downlink flows are different. For this purpose, we conducted simulations with nine different combinations of $n_{down}$ and $n_{up}$ as depicted in Table II ($n_{down} + n_{up} = 20$).

TABLE II
COMBINATIONS OF $n_{down}$ AND $n_{up}$ USED IN THE SIMULATIONS.

| $n_{down}$ | 18 | 16 | 14 | 12 | 10 | 8 | 6 | 4 | 2 |
|---|---|---|---|---|---|---|---|---|---|
| $n_{up}$ | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |

Figure 5(a) plots the fairness indexes as a function of the number of downlink TCP flows. The adaptive pushout showed the best performance for all combinations of $n_{down}$ and $n_{up}$ except when $(n_{down}, n_{up}) = (2, 18)$, at which the ACK pushout shows the best performance.

Under the AF, the fairness index attains its maximum when $(n_{down}, n_{up}) = (6, 14)$, and under the ACK pushout it attains its maximum when $(n_{down}, n_{up}) = (4, 16)$. To understand this, observe that redundant ACKs in the buffer decreases as $n_{down}$ increases when $n_{down} + n_{up}$ is kept constant. Thus, packets (data and ACK segments) more frequently encounter the full buffer with no redundant ACK as $n_{down}$ increases. When data segments are less frequently dropped, the drop of a data segment triggers the three duplicate ACKs and the congestion window of a downlink flow becomes half, while the drop of ACKs triggers the time out, making the congestion window of an uplink flow one. Thus, the downlink flows are likely to obtain larger throughput than the uplink flows. However, when data segments are frequently dropped, the drop of a data segment would also trigger the time out as well as the ACK-segment drop, and the downlink flows would obtain larger throughput than the uplink flow because of the *packet-drop asymmetry*. This turning point is $(n_{down}, n_{up}) = (6, 14)$ under the AF and $(n_{down}, n_{up}) = (4, 16)$ under the ACK pushout. The adaptive pushout monitors the throughput balance between uplink and downlink flows through the observation of $nb_{down}$

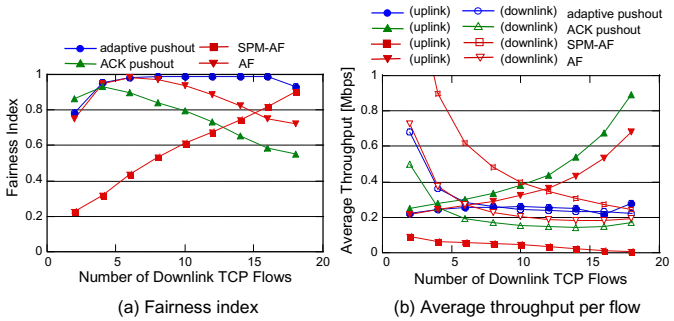


Fig. 5. Fairness index and average throughput per flow in asymmetric case (the numbers of uplink and downlink flows are different).

and $nb_{up}$, which allows the adaptive pushout to achieve the best fairness.

Under the operation of the SPM-AF, each downlink flow always obtains larger throughput than each uplink flow, but the difference in the throughput between uplink and downlink flows constantly decreases as $n_{down}$ increases (see Fig. 5(b)). Thus the fairness is improved as $n_{down}$ increases.

Figure 5(b) compares the average throughput of an uplink flow with that of a downlink flow. Under the ACK pushout and AF, an uplink (downlink) flow got larger (smaller) throughput as $n_{down}$ increased. As we have explained, each downlink flow more frequently experiences the timeout and thus the average throughput of a downlink flow decreases as $n_{down}$ increases. In contrast, uplink flows could obtain the bandwidth released by downlink flows and thus their throughputs increase as $n_{down}$ increases. Under the SPM-AF, the average throughput per flow constantly decreases as $n_{down}$ increases. Under the adaptive pushout, the average throughput per flow is fairly constant independently of $n_{down}$ when the total number of flows is fixed.

In Fig. 6, we show the throughput for each of 20 flows in a simulation run when $(n_{down}, n_{up}) = (6, 14)$. In these figures, flows 1 to 6 were uplink flows and flows 7 to 20 were downlink flows. These figures also confirm that the adaptive pushout is the best buffer management scheme in terms of the fairness between competing TCP flows.

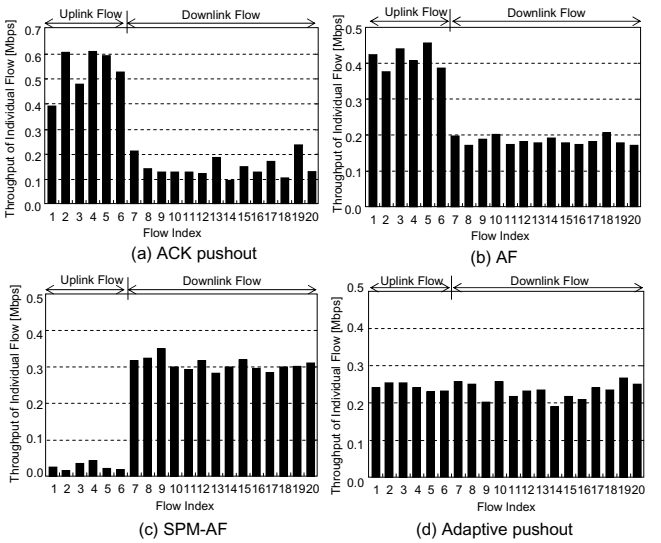


Fig. 6. Throughput of each flow in asymmetric case (the numbers of uplink and downlink flows are different).

## V. Conclusion

In this work, we propose a buffer management scheme, called *adaptive pushout*, as a solution for the unfairness problems between competing TCP flows over WLANs. The simulation experiments confirm that the adaptive pushout yields much better performance than the existing buffer management schemes (AF, APM-AF, ACK pushout) as well as the tail drop in terms of the fairness. A remaining subject is the performance evaluation in the cases where short-lived and long-lived TCP flows are multiplexed together.

## References


[1] H. Balakrishnan and V. Padmanabhan, "How network asymmetry affects TCP," *IEEE Communication Magazine*, vol. 39, no. 4, pp. 60–67, 2001.

[2] H. Balakrishnan1, V.N.Padmanabhan, and R. Katz, "The effects of asymmetry on TCP performance," *Mobile Networks and Applications*, vol. 4, no. 3, pp. 219–241, 1999.

[3] D. Leith and P. Clifford, "TCP dynamics in wireless networks," in *Proc. Wirelesscom*, 2005.

[4] ——, "Using the 802.11e EDCF to achieve TCP upload fairness over WLAN links," in *Proc. WiOpt 05*, 2005.

[5] D. Leith, P. Clifford, D. Malone, and A. Ng, "TCP fairness in 802.11e WLANs," *IEEE Communications Letters*, pp. 964–966, 2005.

[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. Communications*, pp. 1577–1585, 2005.

[7] J. Ha and C. Choi, "TCP fairness for uplink and downlink flows in WLANs," in *Proc. IEEE Globecom*, 2006.

[8] F. Keceli, I. Inan, and E. Ayanoglu, "TCP ACK congestion control and filtering for fairness provision in the uplink of IEEE802.11 infrastructure basic service set," in *Proc. IEEE ICC*, 2007.

[9] X. Lin, X. Chang, and J. Muppala, "VQ-RED: an efficient virtual queue management approach to improve fairness in infrastructure WLAN," in *Proc. IEEE Workshop on Wireless Local Networks*, 2005.

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

[11] Q. Wu, M. Gong, and W. Carey, "TCP fairness issues in IEEE 802.11 wireless LANs," *Computer Communications*, vol. 31, no. 10, pp. 2150–2161, 2008.

[12] Y. Wu, Z. Niu, and J. Zheng, "Study of the TCP upstream/downstream unfairness issue with per-flow queuing over infrastructure-mode WLANs," *Wireless Communications and Mobile Computing*, vol. 5, no. 4, pp. 459–471, 2005.

[13] S. Shioda, H. Iijima, T. Nakamura, S. Sakata, Y. Hirano, and T. Murase, "ACK pushout to achieve TCP fairness under the existence of bandwidth asymmetry," in *Proc. ACM PM2HW2N*, 2010.

[14] B. Hirantha, T. M. S. Abeysekera, 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 Communications*, vol. 7, no. 9, pp. 3517–3525, 2008.

[15] F. Cali, M. Conti, and E. Gregori, "Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit," *IEEE/ACM Trans. Networking*, vol. 8, no. 6, pp. 785–799, 2000.

[16] C. Williamson and Q. Wu, "A case for context-aware TCP/IP," *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 4, pp. 11–23, 2002.

[17] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," *ATM Forum Contribution*, no. 99-0045, 1999.