This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2008 proceedings.

Achieving Fair TCP Access in the IEEE 802.11 Infrastructure Basic Service Set[†]

Feyza Keceli, Inanc Inan, and Ender Ayanoglu

Center for Pervasive Communications and Computing
Department of Electrical Engineering and Computer Science
The Henry Samueli School of Engineering
University of California, Irvine
Email: {fkeceli, iinan, ayanoglu}@uci.edu

***Abstract*—We illustrate the transport layer unfairness problem in the IEEE 802.11 Wireless Local Area Networks (WLANs). We design a link layer access control block for the Access Point (AP) in order to provide fair Transmission Control Protocol (TCP) access in an 802.11 infrastructure Basic Service Set (BSS). The novel and simple idea of the proposed control block is employing a congestion control and filtering algorithm on TCP Acknowledgment (ACK) packets of uplink flows, thereby prioritizing the access of TCP data packets of downlink flows at the AP. We quantify the parameters of the proposed ACK congestion control and filtering algorithm based on the measured average downlink TCP data access rate. Via simulations, we show that the introduction of the proposed link layer access control block to the protocol stack enhances short- and long-term fairness in a wide range of scenarios.**

I. Introduction

An IEEE 802.11 Wireless Local Area Network (WLAN) is built around a Basic Service Set (BSS) [1]. In the common deployment, the wireless stations communicate with an Access Point (AP) which provides the connection of the WLAN to the wired network. This constitutes the so-called infrastructure BSS.

In the Medium Access Control (MAC) layer, the IEEE 802.11 WLAN employs a mandatory contention-based channel access function called the Distributed Coordination Function (DCF). The DCF adopts Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and binary exponential backoff. In DCF, the wireless stations, having all equal contention parameters, have equal opportunity to access the transmission medium. Over a sufficiently long interval, this results in station-based fair access which is also referred as MAC layer fair access.

On the other hand, per-station MAC layer fairness does not translate into achieving per-flow transport layer fairness. In DCF, the AP has the same access priority with the wireless stations. Therefore, an approximately equal bandwidth that an uplink 802.11 station may get is shared among all downlink traffic in the AP. This results in a considerable asymmetry between per-flow uplink and downlink bandwidth in a typical 802.11 WLAN scenario (where the downlink traffic load is usually much larger than the uplink traffic load). The effects of this asymmetry can be detrimental in the case it couples with the bi-directional and reliable data delivery method of Transmission Control Protocol (TCP). As will be described in more detail in Section II, unfair bandwidth allocation is observed between not only uplink and downlink TCP flows but also individual uplink TCP flows.

In this paper, we focus on fair access provisioning for uplink and downlink TCP flows in an IEEE 802.11 infrastructure BSS. We propose a novel link layer access control block for the AP. The control block manages the limited AP bandwidth intelligently by prioritizing the access of the TCP data packets of downlink flows over the TCP ACK packets of uplink flows. This is achieved by employing a congestion control and filtering algorithm on the TCP ACK packets of uplink flows. The specific algorithm parameters are quantified based on the measured average downlink data transmission rate. We test the performance of the protocol stack enhanced with the proposed access control block in terms of transport layer fairness and throughput via simulations. The simulation results show that fairness and high channel utilization can be maintained in a wide range of scenarios.

As for the organization of this paper, we illustrate the TCP unfairness problem in Section II. We provide a brief literature overview in Section III. Section IV describes the proposed link layer access control block. The performance evaluation is the topic of Section V. We provide our concluding remarks in Section VI.

II. TCP Unfairness in the 802.11 WLAN

In the 802.11 WLAN, a bandwidth asymmetry exists between contending upload and download flows. This is due to the fact that the MAC layer contention parameters are all equal for the AP and the stations. If K stations and an AP are always contending for the access to the wireless channel, each host ends up having approximately $1/(K+1)$ share of the total transmit opportunities over a long time interval. This results in $K/(K+1)$ of the transmissions being in the uplink, while

[†] This work is supported by the Center for Pervasive Communications and Computing, and by National Science Foundation under Grant No. 0434928. Any opinions, findings, and conclusions or recommendations expressed in this material are those of authors and do not necessarily reflect the view of the National Science Foundation.

only $1/(K+1)$ of the transmissions belong to the downlink flows.

The TCP receiver returns TCP ACK packets to the TCP transmitter in order to confirm the successful reception of the data packets. In the case of multiple uplink and downlink flows in the WLAN, returning TCP ACKs of upstream TCP data are queued at the AP together with the downstream TCP. When the bandwidth asymmetry in the forward and reverse path builds up the queue in the AP, the dropped packets impair the TCP flow and congestion control mechanisms which assume equal transmission rate both in the forward and the reverse path [2].

Any TCP data packet that is dropped from the AP buffer is retransmitted by the TCP sender following a timeout or the reception of duplicate ACKs. Conversely, any received TCP ACK can *cumulatively* acknowledge all the data packets sent before the data packet for which the ACK is intended for, i.e., a consequent TCP ACK can compensate for a dropped TCP ACK. When the packet loss is severe in the AP buffer, the downstream flows will experience frequent timeouts thus congestion window size decreases, resulting in significantly low throughput. On the other hand, due to the cumulative property of the TCP ACK mechanism, upstream flows with large congestion windows will not experience such frequent timeouts. In the latter case, it is a low probability that many consecutive TCP ACK losses occur for the same flow. Conversely, the upstream flows with small congestion windows (fewer packets currently on flight) may also experience timeouts and decrease their congestion windows even more. Therefore, a number of upstream flows may starve in terms of throughput while some other upstream flows enjoy a high throughput. In summary, the uplink/downlink bandwidth asymmetry creates a congestion at the AP buffer which results in unfair TCP access.

Fig. 1 shows the total TCP throughput in the downlink and the uplink when there are 10 download TCP connections and the number of upload TCP connections is varied from 0 to 10. Each station runs a File Transfer Protocol (FTP) session over TCP. Each station uses 802.11g PHY layer with physical layer (PHY) data rate set to 54 Mbps. Other simulation parameters are as specified in Section V. The unfairness problem between upstream and downstream TCP flows is evident from the results. For example, in the case of 2 upload connections, 10 download TCP connections share a total bandwidth of 6.09 Mbps, while 2 upload TCP connections enjoy a larger total bandwidth of 9.62 Mbps. As the number of upload connections increases, the download TCP connections are almost shut down. Note that these results do not explicitly present the unfairness problem between individual TCP uplink flows. The interested reader is referred to [3] for a specific illustration.

III. Literature Overview

In this section, we provide a brief overview of the literature related to this subject.

Distributed algorithms for achieving MAC layer fairness in 802.11 WLANs are proposed in [4], [5]. A number of studies consider MAC parameter differentiation (or use of 802.11e [6]) for improved fairness and channel utilization

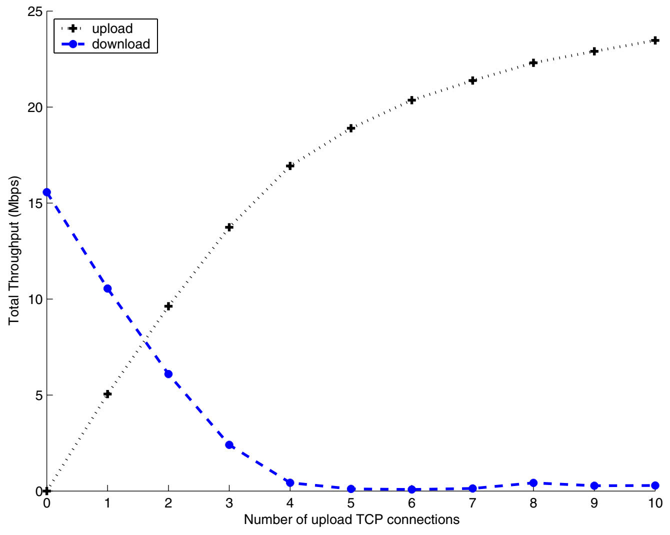

Fig. 1. The total TCP throughput in the downlink and the uplink when there are 10 download TCP connections and the number of upload TCP connections varies from 1 to 10.

[7]–[10]. Algorithms that study enhancements on the backoff procedure for fairness provisioning are proposed in [11], [12]. In a companion paper, we also analyze the use of 802.11e in fairness provisioning [13]. Although MAC parameter differentiation, adaptation, and backoff procedure enhancements can be effective in fair access provisioning, the 802.11 hardware (Network Interface Cards (NICs), APs, etc.) without these capabilities is still widely deployed. Therefore, in this paper, we focus on a technique that does not require any changes in the 802.11 standard or in the non-AP stations.

The TCP uplink and downlink asymmetry problem in the IEEE 802.11 infrastructure BSS is first studied in [14]. The proposed solution of [14] is to manipulate advertised receiver windows of the TCP packets at the AP. Similar algorithms using per-flow or per-direction queueing where distinct queues access the medium with different probabilities are proposed in [15], [16]. A rate-limiter block which filters data packets in both uplink and downlink using instantaneous WLAN bandwidth estimations is proposed in [17]. Differing from all of these techniques, in our previous work, we proposed using congestion control and filtering techniques on top of the MAC queue to solve the TCP *uplink* unfairness problem [3]. The work presented in this paper proposes a novel congestion control and filtering technique which also considers TCP downlink traffic. Note that since TCP downlink traffic load is expected to be larger than the uplink traffic load, this enhancement is vital for a practical implementation.

An extensive body of work exists relating to the impact of asymmetric paths on TCP performance [18], [19] in the wired link context. The effects of ACK congestion control on the performance of an asymmetric network are analyzed in [20] for wired scenarios consisting of only one or two simultaneous flows. The effects of forward and backward link bandwidth asymmetry have been analyzed in [21] for a wired scenario consisting only one flow. Similar effects are also observed in practical broadband satellite networks [22]. The effects of delayed acknowledgements and byte counting on TCP performance are studied in [23]. Several schemes are analyzed in [24] for improving the performance of two-way TCP traffic over asymmetric links where the bandwidths in two dimensions differ substantially. The ACK compression phenomenon that occurs due to the dynamics of two-way traffic using the same buffer is presented in [25]. In this paper, we design a novel ACK congestion control and filtering algorithm to be implemented as a link layer access control block in the protocol stack at an 802.11 AP. The congestion control and filtering algorithm is unique in that the parameters of the algorithm are quantified according to the TCP access characteristics in an 802.11 infrastructure BSS.

IV. Link Layer Access Control Block

As illustrated in Section II, downlink TCP flows may starve in terms of throughput if a few uplink TCP flows coexist in an 802.11 infrastructure BSS. This is a direct result of the effect of a TCP ACK packet loss being different from the effect of a TCP data packet loss. Using the cumulative nature of TCP ACK packets, most uplink flows can sustain high throughput. On the other hand, downlink flows experience frequent timeouts and are shut out.

This observation motivates the approach of our solution which is simply prioritizing TCP data packets of downlink flows over TCP ACK packets of uplink flows at the AP MAC buffer. The proposed link layer access control block achieves this by effectively filtering and scheduling TCP ACK packets of the uplink flows. The TCP data packets of downlink flows are never delayed nor filtered by the proposed control block.

The proposed link layer access control block lies above the MAC layer at the AP. The link layer access control block adaptively decides the time an ACK packet to be sent down to the AP MAC buffer. Any packet that enters the AP MAC queue uses DCF rules to access the channel. Therefore, the proposed solution does not require any changes in the 802.11 standard, nor any enhancement at the stations.

The congestion control and filtering scheme delays the TCP ACK packets of uplink flows (using a separate control block buffer) regarding the measured average packet interarrival time of the downlink TCP data packets. In other words, the downlink data to uplink ACK prioritization ratio is quantified by means of estimating what the uplink ACK transmission rate should be for the given average downlink TCP data transmission rate. The rationale behind the proposed method is sending the TCP ACKs of uplink connections only as often as the TCP data of downlink connections are sent.

The proposed algorithm uses the cumulative property of TCP ACKs by employing ACK filtering. If another ACK packet of flow i is received while there is an ACK packet of flow i in the control block buffer, the previous ACK in the buffer is replaced with the new one. Our rationale behind introducing ACK filtering is to reduce the number of ACK packets transmitted by the AP. This creates more room in the AP buffer for TCP data packets of downlink flows (which in turn decreases TCP data packet loss ratio). Moreover, filtering ACK packets also slows the growth rate of TCP congestion windows of uplink flows (since the TCP senders receive less frequent ACK packets) which further limits the share of the uplink bandwidth.

We define the following notation for the description of the algorithm provided in the sequel. Let $num_{cum,i}$ be the current number of accumulated ACKs for the flow i in the access control block buffer. Let $AvgDataInt$ be the average data packet interarrival time measured at the AP for all downlink TCP flows. Let $t_{buf,i}$ denote the total time that has passed since the last TCP ACK for flow i has been sent to the MAC queue. Let β be a constant weighing factor. Let γ be a variable weighing factor which is a function of $num_{cum,i}$.

We calculate $AvgDataInt$ by averaging the mean packet interarrival time of all individual downlink flows. In the calculation of the average packet interarrival time, the proposed control block applies widely used exponential averaging algorithm. If we let $AvgInt_i$ be the average packet interarrival time for flow i at the AP, $LastArr_i$ be the last packet reception time for flow i at the AP, now be the current packet reception time at the AP, and α be the averaging constant ($0 \leq \alpha \leq 1$), then

$$AvgInt_i = \alpha \cdot AvgInt_i + (1-\alpha) \cdot (now - LastArr_i). \tag{1}$$

Note that $AvgInt_i$ shows the average TCP data packet interarrival time if flow i is a downlink TCP flow and the average TCP ACK packet interarrival time if flow i is an uplink TCP flow.

According to the proposed congestion control and filtering algorithm, the relaying ACK packet of flow i is delayed *at least* for the duration equal to $\beta \cdot AvgInt_i$ in the control block queue. Introducing such a delay paves the way for the ACK filtering algorithm. Our intuition behind the proposed duration of delay is that any ACK is delayed owing to the possibility of any further ACK of the same flow arriving some time later (possibly in an average interarrival time $AvgInt_i$). We also introduce the constant weighing factor $\beta > 1$ in order to compensate for the potential variance of the *instantaneous* ACK interarrival time.

Moreover, according to the proposed congestion control and filtering algorithm, the TCP ACKs are scheduled for transmission (sent down to the MAC queue) such that the average per-flow ACK rate does not exceed the average per-flow TCP downlink packet rate. Using this idea, we quantify a *minimum* ACK packet buffering time of $\gamma \cdot num_{cum,i} \cdot AvgDataInt - t_{buf,i}$. The rationale behind this equation is as follows.

- We consider the cumulative number of ACK packets that the currently buffered ACK packet represents. Note that the transmission of an accumulated ACK packet is expected to trigger the generation of $num_{cum,i}$ data packets in the uplink. Therefore, any accumulated TCP ACK packet is delayed until that many TCP downlink data transmissions can be made on average ($num_{cum,i} \cdot AvgDataInt$).

- If a few consecutive timeouts are experienced when the TCP congestion window is small, the uplink TCP flow may hardly recover, and consequently may suffer from low throughput (as we also observed via simulations). Therefore, we introduce an adaptive weighing factor $\gamma_{min} \leq \gamma \leq 1$ in the minimum buffering duration. We use the value of $num_{cum,i}$ as an indication of the current size of the TCP congestion window of the corresponding flow. The value of γ is set smaller than 1 when $num_{cum,i}$ is smaller than a threshold, num_{thresh}. The idea is to prevent longer delays at the control block buffer thus possible timeouts at the TCP agent at the station if the uplink TCP connection is expected to have a small instantaneous congestion window (e.g., a recently initiated TCP connection).
- We substract $t_{buf,i}$ from $\gamma \cdot num_{cum,i} \cdot AvgDataInt$ in order to make the duration of the interval between two consecutive ACKs sent down to the MAC buffer approximately equal to $num_{cum,i} \cdot AvgDataInt$ (in the case $num_{cum,i} > num_{thresh}$). Each ACK cumulatively acknowledges $num_{cum,i}$ data packets. This approximately results in an average uplink TCP data transmission interval equal to an average downlink data transmission interval $AvgDataInt$ (which is required for achieving uplink/downlink fair TCP access).

Combining both limits we proposed on the duration of TCP ACK delay, we can define the simple rule on how long the relaying ACK packet waits in the control buffer before it is enqueued to the MAC queue for transmission

$$D_i = \max(\beta \cdot AvgInt_i, \gamma \cdot num_{cum,i} \cdot AvgDataInt - t_{buf,i}). \quad (2)$$

If a new TCP ACK packet arrives before the timer that is initially set to D_i at the arrival of previous ACK expires, the new ACK replaces the previously buffered ACK. The link layer access control block parses the TCP header to calculate $num_{cum,i}$, and restarts the timer with the new D_i for the accumulated ACK. When the timer expires, the TCP ACK is sent down to the MAC queue and both $t_{buf,i}$ and $num_{cum,i}$ are reset to 0. To sum up, the ideas behind (2) are very simple: *i)* delay the ACK for an average interarrival time to find out if another ACK is coming and *ii)* do not send the ACK down to the MAC buffer for transmission if this can create an uplink data transmission rate larger than measured data downlink transmission rate.

ACK filtering may slow down the congestion window growth rate, negatively impact the performance during loss recovery and slow start, and increase the round trip time [20]. On the other hand, since our idea is trying to slow down uplink TCP flows in order to prioritize downlink TCP flows, most of these issues do not negatively affect fairness and overall channel utilization. Still, the proposed algorithm does not filter the TCP ACKs with flags set such as duplicate ACKs which are directly enqueued to the MAC queue.

The proposed ACK congestion control and filtering algorithm introduces a number of configurable variables. As pointed out in Section V, we decided the values for these variables through extensive simulations.

V. Performance Evaluation

We implemented the proposed link layer control access block in ns-2 [26]. In this section, we present the simulation results on the performance of the network in terms of fairness and throughput. Our main performance metric is the widely used fairness index [27]. The fairness index, f, is defined as follows: if there are K concurrent connections in the network and the throughput achieved by connection i is equal to x_i, $1 \leq i \leq K$, then

$$f = \frac{\left(\sum_{i=1}^{K} x_i\right)^2}{K \sum_{i=1}^{K} x_i^2}. \quad (3)$$

The fairness index lies between 0 and 1, 1 being the most fair situation where every flow gets equal throughput.

We consider a network topology where each wireless station initiates a connection with a wired station and where the traffic is relayed to/from the wired network through the AP. The TCP traffic uses a File Transfer Protocol (FTP) agent which models bulk data transfer. TCP NewReno with its default parameters in ns-2 is used. Unless otherwise stated, TCP flows are considered to be lasting through the simulation duration and called long-lived in the sequel. On the other hand, in some experiments, we also use short-lived TCP flows which consist of 31 packets and leave the system after all the data is transferred. All the stations have 802.11g PHY [28] with 54 Mbps and 6 Mbps as the data and basic rate respectively. The wired link data rate is 100 Mbps. The default DCF MAC parameters are used [1]. The packet size is 1500 bytes for all flows. The MAC queue size at the stations and the AP is set to 100 packets. The receiver advertised congestion window limits are set to 42 packets for each flow. Note that the scale on the buffer size and TCP congestion window limit is inherited from [14]. Although the practical limits may be larger, the unfairness problem exists as long as the ratio of the buffer size to the congestion window limit is not arbitrarily large (which is not the case in practice). We found $\alpha = 0.9$, $\beta = 2$, $\gamma_{min} = 0.5$, and $num_{thresh} = 10$ to be appropriate through extensive simulations. The simulation duration is 350 ms.

We investigate the system performance when wired link delays differ among TCP connections. We define wired link delay as the propagation delay between the TCP sender/receiver in the wired network and the AP. The wired link delay of the first upload or download TCP connection is always set to 10 ms. Then, any newly generated upload or download TCP connection has a wired link delay of 2 ms larger than the previous one in the same direction.

a) The Basic Scenario: In the first set of experiments, we generate 3, 5, or 10 upload TCP connections and vary the number of download TCP connections from 5 to 30. The wireless channel is assumed to be errorless.

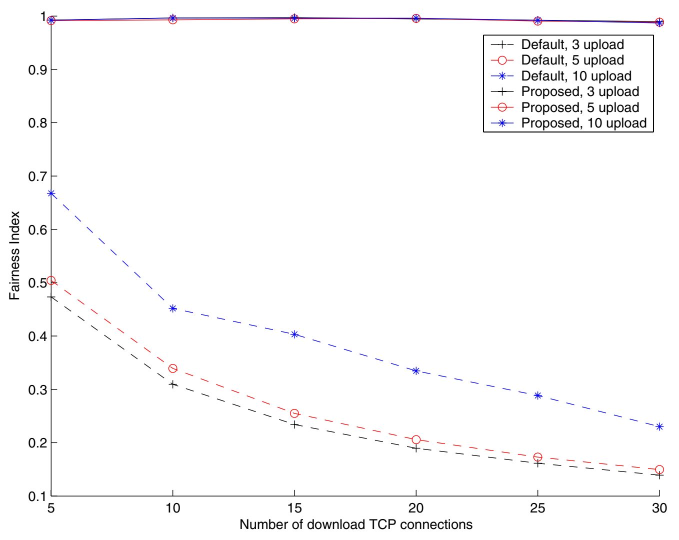

Fig. 2. Fairness index among all TCP flows when 3, 5, or 10 upload TCP connections are generated and the number of download TCP connections are varied from 5 to 30.

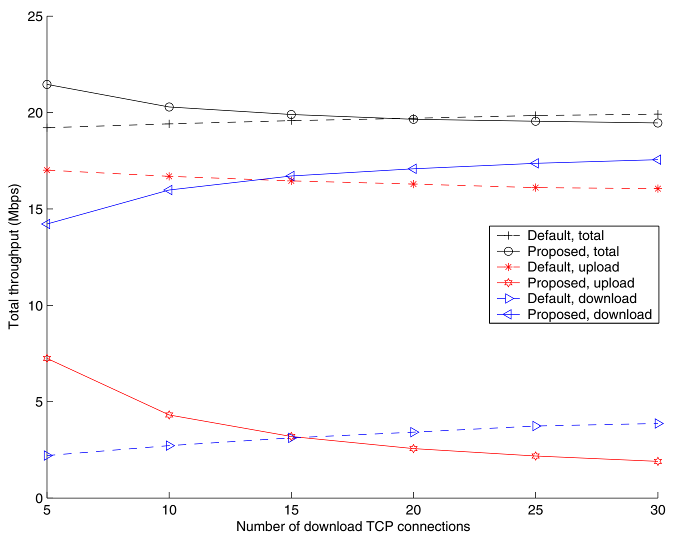

Fig. 3. Throughput of upload and download TCP connections when 3 upload TCP connections are generated and the number of download TCP connections are varied from 5 to 30.

Fig. 2 shows the fairness index among all TCP connections. As the results imply, with the introduction of the proposed link layer control block, an almost perfect fair resource allocation is achieved.

In Fig. 3, we plot the uplink, the downlink, and the total TCP throughput in the infrastructure BSS. We only present the scenario when there are 3 upload TCP connections. Similar results are observed for other cases with 5 and 10 upload TCP connections. As the results show, using the proposed link layer access control block, the downlink flows (which starve in the default DCF case) can achieve reasonable throughput. The comparison with the performance of the default DCF algorithm implies that the proposed algorithm does not sacrifice channel utilization while providing fair access. Note that the total uplink throughput is less than the total downlink throughput in Fig. 3 for the proposed method, since the number of TCP connections in the uplink is fewer than the number of TCP connections in the downlink. As Fig. 2 shows, per-flow throughput is fair.

b) Delayed TCP ACKs: In the second set of experiments, we use a scenario when TCP connections use the delayed TCP ACK mechanism. In the delayed TCP ACK mechanism, the TCP receiver acknowledges every b TCP data packets ($b > 1$). We use a typical value (widely used in practice), $b = 2$, for the specific experiment.

We start download and upload TCP connections in 10 s and 20 s intervals, respectively. Fig. 4 shows the instantaneous throughput for individual TCP flows over simulation duration. As the results imply, in the default case, TCP download connections starve in terms of throughput as the number of TCP upload connections increase. In the meantime, some upload flows experience long delays in starting and achieving high throughput while some do not. On the other hand, using the proposed algorithm, all uplink and downlink TCP flows

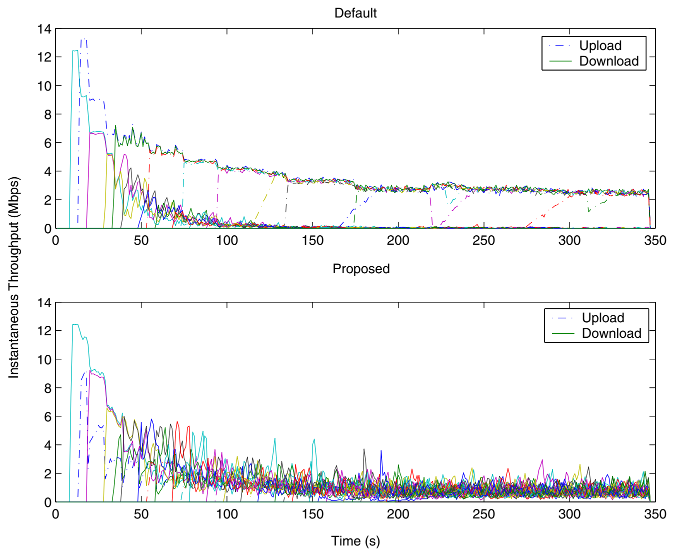

Fig. 4. Individual instantaneous throughput for upload and download TCP connections when TCP receivers employ the delayed ACK mechanism.

enjoy fair access. The results are important in showing the proposed algorithm's effectiveness even when the delayed TCP ACK mechanism is used.

c) Wireless Channel Errors: In the third set of experiments, we assume the wireless channel to be an Additive White Gaussian Noise (AWGN) channel. On top of the energy-based PHY model of ns-2, we implemented a BER-based PHY model according to the framework presented in [29] using the way of realization in [30]. Our model considers the channel noise power in Signal-to-Noise Ratio (SNR). A packet is detected at the receiver if the received power is over a specified threshold. Then, the packet error probability is calculated using the theoretical model presented in [29] regarding the channel

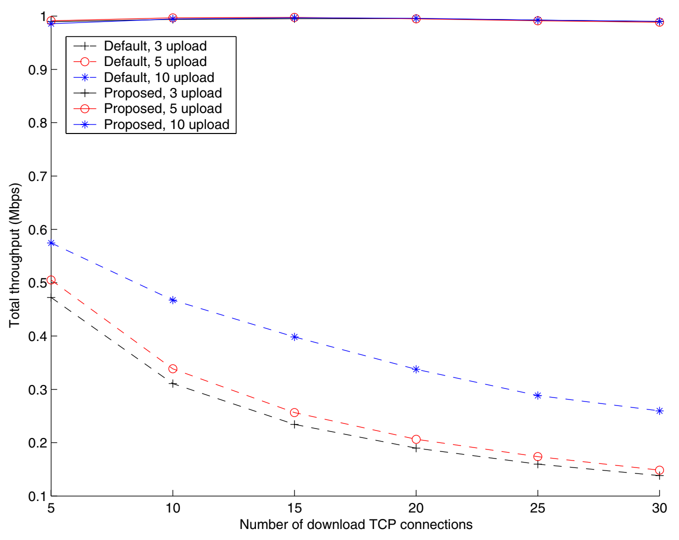

Fig. 5. Fairness index among all TCP flows over an AWGN channel when 3, 5, or 10 upload TCP connections are generated and the number of download TCP connections are varied from 5 to 30.

SNR, the modulation type used for the transmission of the packet (considering the different modulation of the headers and the payload), and the packet size.

We set wireless channel noise levels such that each station experience a finite data packet error rate (PER). We repeat the tests for AWGN channel SNR values when PER is 0.001 or 0.01. We only present the results on fairness index for the case when PER is 0.01, since the results slightly differ and a similar discussion holds for the case when PER is 0.001.

As in the first set of experiments, we generate 3, 5, or 10 upload TCP connections and vary the number of download TCP connections from 5 to 30. Fig. 5 shows that the proposed algorithm provides fair access. The performance of the proposed algorithm is resilient to wireless channel errors, i.e., fair access is preserved even when there are errors in the wireless channel. Although not presented here, the throughput drops slightly when compared to errorless wireless channel case due to the MAC retransmissions.

d) Short-lived TCP flows: In the fourth set of experiments, we test the performance in terms of short-term fairness. First, we generate 5 uplink and 10 downlink long-lived TCP flows. Then, 15 short-lived uplink and downlink TCP flows are generated with 5 s intervals consecutively. Fig.6 shows the total transmission duration for individual short-lived TCP flows for the proposed algorithm and the default DCF. Note that the flow indices from 1 to 15 represent uplink TCP flows while flow indices from 16 to 30 represent downlink TCP flows. As the results imply, the short-lived file transfer can be completed in a significantly shorter time when the proposed algorithm is used. We can conclude that the proposed algorithm is short-term fair.

VI. Conclusion

We have designed a novel link layer access control block for the AP in order to provide fair TCP access in an 802.11

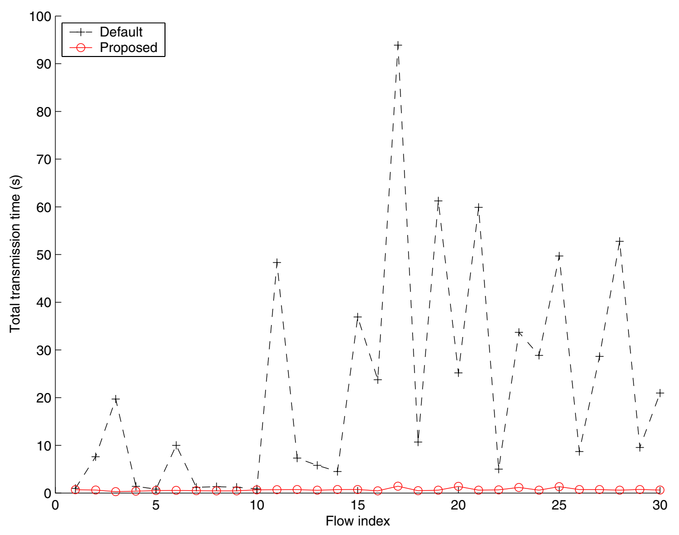

Fig. 6. The total transmission duration for individual short-lived TCP flows.

infrastructure BSS. Our simple idea in solving the unfairness problem in the WLAN is prioritizing TCP data packets of uplink flows over TCP ACK packets of uplink flows at the AP. We achieve this by designing an ACK congestion control and filtering algorithm which is employed in the proposed link layer access control block. The proposed algorithm is unique in that the specific algorithm parameters are based on the measured average data transmission rate at the AP. Via simulations, we show that fair resource allocation for uplink and downlink TCP flows can be provided in a wide range of practical scenarios when the proposed method is used. A key insight that can be obtained from this study is that fair and efficient TCP access in a WLAN can simply be achieved by intelligently scheduling TCP ACK transmissions at the AP. As an attractive feature, the proposed solution does not require any changes in the 802.11 standard, nor any enhancement at the stations.

References

[1] *IEEE Standard 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE 802.11 Std., 1999.
[2] H. Balakrishnan, V. Padmanabhan, and R. H. Katz, “The Effects of Asymmetry on TCP Performance,” *ACM Baltzer Mobile Networks and Applications (MONET)*, 1999.
[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 *Proc. IEEE ICC ’07*, June 2007.
[4] N. H. Vaidya, P. Bahl, and S. Gupta, “Distributed Fair Scheduling in a Wireless LAN,” in *Proc. ACM Mobicom ’00*, August 2000.
[5] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, “Achieving MAC Layer Fairness in Wireless Packet Networks,” in *Proc. ACM Mobicom ’00*, August 2000.
[6] *IEEE Standard 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Medium access control (MAC) Quality of Service (QoS) Enhancements*, IEEE 802.11e Std., 2005.
[7] C. Casetti and C. F. Chiasserini, “Improving Fairness and Throughput for Voice Traffic in 802.11e EDCA,” in *Proc. IEEE PIMRC ’04*, September 2004.
[8] D. J. Leith, P. Clifford, D. Malone, and A. Ng, “TCP Fairness in 802.11e WLANs,” *IEEE Commun. Lett.*, pp. 964–966, November 2005.
[9] J. Freitag, N. L. S. da Fonseca, and J. F. de Rezende, “Tuning of 802.11e Network Parameters,” *IEEE Commun. Lett.*, pp. 611–613, August 2006.

[10] I. Tinnirello and S. Choi, “Efficiency Analysis of Burst Transmissions with Block ACK in Contention-Based 802.11e WLANs,” in *Proc. IEEE ICC ’05*, May 2005.

[11] S. W. Kim, B.-S. Kim, and Y. Fang, “Downlink and Uplink Resource Allocation in IEEE 802.11 Wireless LANs,” *IEEE Trans. Veh. Technol.*, pp. 320–327, January 2005.

[12] J. Jeong, S. Choi, and C.-K. Kim, “Achieving Weighted Fairness between Uplink and Downlink in IEEE 802.11 DCF-based WLANs,” in *Proc. IEEE QSHINE ’05*, August 2005.

[13] F. Keceli, I. Inan, and E. Ayanoglu, “Weighted Fair Uplink/Downlink Access Provisioning in IEEE 802.11e WLANs,” in *Proc. IEEE ICC ’08*.

[14] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, “Understanding TCP Fairness over Wireless LAN,” in *Proc. IEEE Infocom ’03*, April 2003.

[15] Y. Wu, Z. Niu, and J. Zheng, “Study of the TCP Upstream/Downstream Unfairness Issue with Per-flow Queueing over Infrastructure-mode WLANs,” *Wireless Commun. and Mobile Comp.*, pp. 459–471, June 2005.

[16] J. Ha and C.-H. Choi, “TCP Fairness for Uplink and Downlink Flows in WLANs,” in *Proc. IEEE Globecom ’06*, November 2006.

[17] 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 Trans. Wireless Commun.*, pp. 1346–1355, April 2007.

[18] “RFC3449 - TCP Performance Implications of Network Path Asymmetry,” 2002.

[19] “RFC2760 - Ongoing TCP Research Related to Satellites,” 2000.

[20] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, “The Effects of Asymmetry on TCP Performance,” in *Proc. ACM/IEEE MobiCom ’97*, November 1997.

[21] T. V. Lakshman, U. Madhow, and B. Suter, “Window-based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A Study of TCP/IP Performance,” in *Proc. IEEE Infocom ’97*, April 1997.

[22] G. Fairhurst, N. K. G. Samaraweera, M. Sooriyabandara, H. Harun, K. Hodson, and R. Donadio, “Performance Issues in Asymmetric TCP Service Provision using Broadband Satellite,” *IEE Proc. Commun.*, pp. 95–99, 2001.

[23] M. Allman, “On the Generation and Use of TCP Acknowledgements,” *ACM Comp. Comm. Review*, October 1998.

[24] L. Kalampouskas, A. Varma, and K. K. Ramakrishnan, “Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions,” in *Proc. ACM Sigmetrics ’98*, 1998.

[25] L. Zhang, S. Shenker, and D. D. Clark, “Observations on the Dyanmics of a Congestion Control Algorithm: The Effects of Two-Way Traffic,” *ACM Comp. Comm. Review*, pp. 133–147, 1991.

[26] (2006) The Network Simulator, ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns

[27] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons, 1991.

[28] *IEEE Standard 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band*, IEEE 802.11g Std., 2003.

[29] D. Qijao and S. Choi, “Goodput Enhancement of IEEE 802.11a Wireless LAN via Link Adaptation,” in *Proc. IEEE ICC ’01*, June 2001.

[30] M. Lacage. (2006) Ns-2 802.11 Support. INRIA Sophia Antipolis. France. [Online]. Available: http://spoutnik.inria.fr/code/ns-2