This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

TCP ACK Congestion Control and Filtering for Fairness Provision in the Uplink of 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*—Most of the deployed IEEE 802.11 Wireless Local Area Networks (WLANs) use infrastructure Basic Service Set (BSS) in which an Access Point (AP) serves as a gateway between wired and wireless domains. We present the transport layer uplink unfairness problem that occurs due to uneven bandwidth sharing between upstream Transmission Control Protocol (TCP) data and downstream TCP acknowledgment (ACK) packets. We propose ACK congestion control and filtering in the WLAN downlink to improve fairness between uplink flows. Through extensive simulations, we show that the proposed heuristic algorithms not only enhance both short- and long-term fairness but also increase channel utilization in the 802.11 WLAN. Another attractive feature for the proposed algorithms is the independence of the performance from variable TCP Round Trip Times (RTTs).**

I. Introduction

IEEE 802.11 Wireless Local Area Network (WLAN) architecture is built around a Basic Service Set (BSS) [1]. While a number of stations may gather to form an independent BSS with no connectivity to the wired network, the common deployment is the infrastructure BSS which includes an Access Point (AP). The AP provides a connection to the wired network. The IEEE 802.11 standard defines Distributed Coordination Function (DCF) as a contention based Medium Access Control (MAC) mechanism. The DCF uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique. The DCF operates only during the Contention Period (CP) where each station contends for the access to the medium. If each station in a BSS uses the same contention parameters, fair access can be achieved in the MAC layer for all contending stations in terms of the average number of granted transmission opportunities, over a sufficiently long interval.

However, providing fairness in the MAC layer access does not directly translate into achieving transport layer fairness among individual flows for an 802.11 infrastructure BSS scenario. The AP which serves all downlink flows has the same access priority with the stations that serve uplink flows. Therefore, an approximately equal bandwidth that an uplink node may get is shared among all downlink flows in the AP. This leads to the so-called uplink/downlink unfairness problem in the WLAN as described in Section II where each individual downlink flow gets comparably lower bandwidth than each individual uplink flow gets.

The Transport Control Protocol (TCP) defines reliable bi-directional communication. The forward link is used for the transmission of data segments and the backward link carries the acknowledgment (ACK) packets for the successfully received TCP frames. In the case of multiple TCP connections in the uplink, the AP has to transmit TCP ACK packets in the downlink. As stated earlier and as will be described in detail later, there exists an inherent uplink/downlink unfairness in a WLAN. This can result in a bottleneck for the TCP ACKs in the AP which may lead to severe packet loss in the MAC queue. As a result of the cumulative nature of TCP ACKs, a few connections may be able to survive and the others may starve.

First, we will provide a review of the literature to differentiate our work from similar approaches. The problem of WLAN fairness has been analyzed at the MAC layer in [2], [3]. However, as also pointed out in [4], MAC layer fairness in the 802.11 WLAN does not translate into TCP layer fairness. On the other hand, the proposed solution of [4] depends on connection Round Trip Times (RTT) and may result in poor channel utilization. Uplink/downlink fairness problem is studied in [5] using per-flow queueing, but the TCP unfairness problem between the uplink flows is not addressed. A similar approach is proposed in [6] where two separate queues for TCP data and ACKs are used. The use of IEEE 802.11e Enhanced Distributed Channel Access (EDCA) function [7] for providing fairness in the 802.11 WLAN is investigated in [8]. While achieving fairness, the MAC access parameters of TCP traffic in [7] may impair the Quality-of-

† This work is supported by the Center for Pervasive Communications and Computing, and by Natural 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 Natural Science Foundation.

1-4244-0353-7/07/$25.00 ©2007 IEEE

Service (QoS) of higher priority traffic such as voice and video flows. No guidelines are stated on how to decide on good contention parameters for an arbitrary scenario. A rate-limiter approach is used in [9] which requires available instantaneous WLAN bandwidth estimation in both directions. The rate-limiter approach can be bandwidth inefficient.

An extensive body of work exists relating to the impact of asymmetric paths on TCP performance [10], [11] in the wired link context. The effects of ACK congestion control on the performance of an asymmetric network in analyzed in [12] for wired scenarios consisting of only one or two simultaneous flows. The effects of forward and backward link bandwidth asymmetry has been analyzed in [13] for a wired scenario consisting only one flow. Similar effects are also observed in practical broadband satellite networks [14]. The effects of delayed acknowledgements and byte counting on TCP performance is studied in [15]. Several schemes are analyzed in [16] 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 same buffers is presented in [17].

In this work, we focus on the unfairness problem between the individual uplink TCP flows in the WLAN which occurs in the case of a congested downlink AP buffer. Our approach is based on our conjecture that if the AP manages bandwidth carefully and schedules TCP ACKs wisely in the downlink, the uplink TCP unfairness issue can be resolved. We propose an ACK congestion control and filtering algorithm that improves the uplink TCP fairness considerably. While providing fairness, the algorithm does not sacrifice high channel utilization and low delay. We also show that the proposed algorithm improves unfairness that results from uneven TCP RTTs.

The rest of the paper is organized as follows. We define the uplink TCP unfairness problem in Section II. In the description, it is assumed that the reader has sufficient background on IEEE 802.11 WLAN and TCP. In Section III, the proposed ACK congestion control and filtering algorithm is described. The performance evaluation is the topic of Section IV. Finally, we provide our concluding remarks in Section V.

II. TCP Uplink Unfairness over 802.11 WLANs

The 802.11 MAC layer contention mechanism provides MAC level fair access if the contention parameters of all nodes are equal. If n stations and an AP are contending for the access to the wireless channel, each host ends up having approximately $1/(n+1)$ share of the total transmissions over a long time interval. This results in $n/(n+1)$ of the transmissions to be in the uplink, while only $1/(n+1)$ of the transmissions belong to the downlink flows. Under the assumption of equal packet sizes and physical layer data rates, these ratios directly reflect the 802.11 uplink and downlink bandwidth share. This is the inherent WLAN uplink/downlink unfairness stated previously.

TCP flow control and congestion control mechanisms run on bi-directional communication to ensure reliable transfer of TCP data. The transmission data rate is adjusted according to network capacity. The TCP receiver returns TCP ACK packets to the TCP transmitter in order to confirm the successful reception of data packets. In the case of multiple uplink flows in the WLAN which are destined to some point in the wired network, returning TCP ACKs are queued at the AP to be transmitted at the wireless hop. However, as explained above the AP can enjoy a fairly small downlink bandwidth for individual ACK flows when compared to the bandwidth of each uplink data flow. This asymmetry in the forward and reverse path builds up the queue in the AP. The dropped TCP ACKs impair the TCP flow and congestion control mechanisms which assume equal transmission rate both in the forward and reverse path.

TCP's timeout mechanism initiates a retransmission of a data packet if it has not been acknowledged during *timeout* duration. However, any received TCP ACK can cumulatively acknowledge all the data packets sent before the data packet for which the ACK is intended to. Therefore, when ACK packet loss is severe in the AP buffer, flows with high congestion windows will not experience frequent timeouts. In this case, it is a low probability that many consecutive TCP ACK losses occur for the same flow. On the other hand, flows with low congestion window (fewer packets currently on flight) may experience frequent timeouts, and decrease their congestion windows even more. This results in a number of flows starving in terms of throughput while others enjoy a high throughput.

Fig. 1 shows the average throughput observed by each TCP flow for a scenario of 15 stations and an AP in an ns-2 simulation [18]. Each station runs an FTP session over TCP with a peer station located in the wired network. Each station has 802.11g physical layer with physical data rate set to 24 Mbps [20]. Other simulation parameters are as specified in Section IV. The results illustrate the unfairness in the throughput achieved by the uplink FTP flows. 9 of the TCP flows starve in terms of throughput.

Our main performance metric is the widely used fairness index [19]. The fairness index, f, is defined as follows: if there are n_{act} concurrent connections in the network and the throughput achieved by connection i is equal to x_i, $1 \leq i \leq n_{act}$, then

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

The fairness index lies between 0 and 1, 1 being the most fair situation where every flow gets equal throughput. For the specific case illustrated in Fig. 1, f is 0.4.

III. TCP ACK Congestion Control and Filtering for Fairness Provision

ACK congestion control and filtering is a gateway-based technique that decreases the number of TCP ACKs sent over the constrained channel by taking advantage of the fact that TCP ACKs are cumulative. Reducing the number of ACK packets generated for a given amount of data decreases the collisions with data packets in the wireless channel.

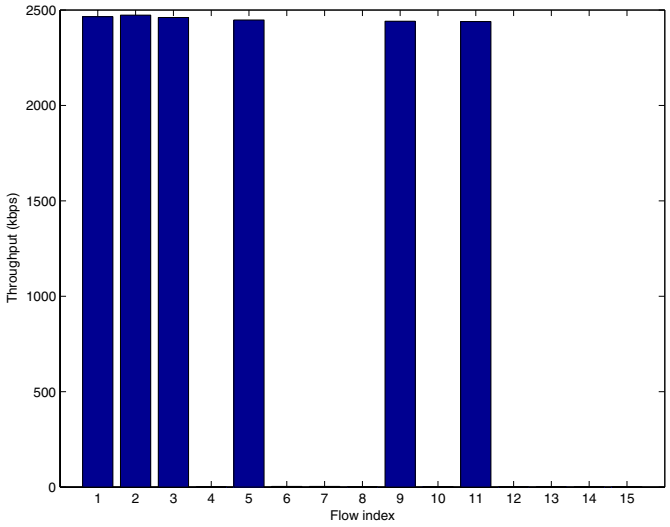

Fig. 1. Average throughput of 15 TCP uplink flows with default 802.11 parameters.

Uplink TCP unfairness problem in the 802.11 WLAN basically arises because of limited downlink bandwidth and buffer space for TCP ACKs. We propose an ACK congestion control and filtering algorithm that effectively schedules TCP ACK access in the downlink in order to provide fair number of average transmission grants to each uplink flow. Via simulations, we show that this algorithm not only provides fair uplink access, but also results in high throughput and low delay when compared with other proposed techniques.

The MAC queue is known as the drop-tail Interface Queue (IFQ) in 802.11. Our algorithm enhances the drop-tail IFQ implementation in an 802.11 AP. Rather than queueing every relayed TCP ACK and dropping the overflown packets, the algorithm controls the downstream ACK traffic according to estimated average bandwidth each flow gets. The algorithm adaptively determines at which time which ACK shall be transmitted.

As a direct result of IEEE 802.11 DCF using equal access parameters for both the AP and the stations, the AP may get the channel grant n times while every other station in the WLAN accesses the channel n times on average over a time interval. Since smooth progress of a TCP connection is highly dependent on the reception timing of TCP ACKs, the AP has to equally divide downlink bandwidth among ACK streams for fair uplink access. Simple math shows if the AP were associated with n TCP upstream flows, it would be the fair approach to send an ACK per flow per n uplink TCP data packet transmissions. This way every associated flow gets equal number of ACKs over a sufficiently long time interval while the AP accesses the medium in an equal amount of times a station accesses satisfying DCF definition.

Unfortunately, such a simple ACK congestion control and filtering approach at the gateway is not expected to work well in practice when time-varying TCP congestion window sizes are considered. At a point in time, some TCP flows may have considerably smaller congestion windows due to recent TCP connection start, reception of duplicate ACKs, or a timeout. Sticking with the congestion control and filtering with a fixed constant does not prioritize slow TCP connections over faster ones. This may result in TCP connections with smaller congestion windows not being able to speed up again and starving in terms of throughput in the case of a highly congested AP buffer.

We present our approach which adaptively determines the ACK to be transmitted at the AP according to the estimates of the average interarrival time of ACK packets that are calculated by applying widely used exponential averaging algorithm on actual interarrival time. If we let $AvgACKint_i$ be the average ACK interarrival time, $LastACKarr_i$ be the last ACK reception time for flow i, now be current ACK reception time at the AP, and α be the averaging constant ($0 \leq \alpha \leq 1$), then

$$AvgACKint_i = \alpha \cdot AvgACKint_i + (1-\alpha) \cdot (now - LastACKarr_i). \quad (2)$$

According to our proposed ACK congestion control and filtering algorithm, each ACK is buffered for the duration of an adaptive ACK waiting time before it is scheduled for transmission. Therefore, the proposed algorithm requires the AP to maintain two buffers. The first one is assigned as the conventional drop-tail IFQ buffer which the 802.11 MAC layer uses. The second buffer is used by the proposed algorithm to control downlink ACK schedule and filtering.

The relaying ACK packet waits for the duration equal to $\beta \cdot AvgACKint_i$ in the control queue before it is queued to IFQ for transmission (β is a constant weighing factor). If another ACK packet of flow i is received while there is an ACK packet of flow i in the control queue, the previous ACK in the buffer is replaced with the new cumulative ACK. The current number of accumulated ACKs for the corresponding flow ($cum_{cur,i}$) is incremented. The waiting timer is restarted again with the updated duration.

The algorithm specifies a threshold on the limit of the number of ACKs that can be accumulated at the AP for each flow i ($cum_{thresh,i}$). As soon as $cum_{thresh,i} \leq cum_{cur,i}$ is satisfied, the cumulative ACK is enqueued to IFQ for transmission and $cum_{cur,i}$ is reset to zero.

The decision on the $cum_{thresh,i}$ is made adaptively regarding the instantaneous number of active TCP connections, n_{act}, and $AvgACKint_i$ where $0 \leq i \leq n_{act}$. A TCP connection is labelled as active if the AP has received an ACK belonging to that flow in the time interval $now - ActiveThresh$.

The idea is to set $cum_{thresh,i}$ higher for flows with lower $AvgACKint_i$ which are expected to get a larger bandwidth share in the uplink (or vice versa). Therefore, the AP can use the larger portion of the downlink bandwidth share in ACK transmissions for slow flows. This approach will help slow TCP flows get ACKs more frequently and increase their congestion windows faster, thus prevent them from starving in terms of throughput.

In this paper, we report the performance results of two different methods for the selection of $cum_{thresh,i}$. These heuristic methods are very simple, and are shown to provide satisfactory performance in simulations.

- **Sorted assignment:** The AP stores a sorted list of active flows according to $AvgACKint_i$ from highest to lowest. This method assigns $cum_{thresh,i}$ the list index l of flow i times an integer constant c_{srt}. Therefore, $cum_{thresh,i}$ is set to c_{srt} if the flow has the highest $AvgACKint_i$. Conversely, $cum_{thresh,i}$ of the flow with the lowest $AvgACKint_i$ is assigned to $c_{srt} \cdot n_{act}$. As the order in the sorted list changes through time, $cum_{thresh,i}$ is adaptively updated with the new list index.
- **Grouped assignment:** To select $cum_{thresh,i}$, the AP groups all flows satisfying (3) in the sorted list in one entry.

$$\{j,\ 0 \leq j \leq n_{act} :\ (1-\gamma) \cdot AvgACKint_i \leq AvgACKint_j \leq (1+\gamma) \cdot AvgACKint_i\} \quad (3)$$

  According to the new sorted list, $cum_{thresh,i}$ is assigned an integer from set $C = \{c_k : 1 < c_k < c_{max},\ c_k < c_{k+1}\}$. Assume C has size $N \leq n_{act}$ and c_{max} is an integer constant. If flow i is in top $\frac{k \cdot 100}{N}$ percent in the list, then $cum_{thresh,i}$ is assigned to c_k. As the order in the sorted list changes through time, $cum_{thresh,i}$ is updated with the appropriate element of set C.

Both methods ensure slow flows get acknowledged more frequently than faster flows. Grouped assignment provides much coarser $cum_{thresh,i}$ assignment than sorted assignment. This enables accumulating equal number of ACKs for flows that use approximately the same bandwidth on the average.

If special care is not taken, the ACK congestion control and filtering approach may incur timeouts at the TCP sender. Therefore, we define another timer for each flow which is started at the time when the first ACK is received with no cumulative ones are buffered in the control queue. When this timer expires, the cumulative ACK is directly enqueued to IFQ independent of $cum_{cur,i}$ value. In the meantime, $cum_{cur,i}$ is reset to zero.

We also address a number of known issues about ACK filtering [12]. The TCP ACKs with flags set such as duplicate ACKs are not filtered. If a flow is detected to be newly started or recently received three duplicate ACKs, the first couple of TCP ACKs are directly enqueued to the IFQ and scheduled for transmission. This approach partially combats the slow congestion window growth problem during the slow start phase.

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

IV. Simulations

The focus of the simulation study is to analyze the performance of the proposed ACK congestion control and filtering algorithm in terms of fairness, average throughput, and channel utilization.

We have implemented the proposed ACK congestion control and filtering algorithm in ns-2.28 [18]. This distribution of ns-2 supports DCF in the 802.11 MAC layer. We use the FTP traffic generator which is modelled as bulk data transfer. Due to space limitations, we only present the results for TCP-NewReno. On the other hand, our simulations show that similar results hold for TCP-Tahoe and TCP-Reno. The TCP receiver acknowledges each data packet with a TCP ACK. The TCP parameters are left as default, except the receiver advertised TCP congestion window of all flows is set to 42 packets. The TCP packet size is set to 1500 bytes. Unless otherwise stated, all the stations have IEEE 802.11g physical layer with physical layer data rate set to 54Mbps [20]. DCF access parameters are set to default values except minimum and maximum contention window values are set to 31 and 1023 respectively [1]. The received power levels are adjusted to be sufficiently over the minimum threshold value. Specifically, issues that occur as a direct result of limited buffer size, downlink bandwidth of the AP, and varying TCP RTT are addressed.

We are considering a simulation scenario where the associated wireless stations upload a file of infinite size to a server located in the wired domain. Each connection starts with 1 second difference, and the simulations last 100 seconds. Unless otherwise stated, each uplink TCP packet traverses a wired link with a 15ms delay after the wireless hop to reach the destination server. This means that RTT for individual uplink TCP flows are approximately equal.

We list the values of configuration parameters for the proposed ACK congestion control algorithm. These values are decided through extensive simulations for the specified scenario. We have tested them in other scenarios, and similar results hold. All results cannot be included due to space limitations. In simulations, $\alpha = 0.8$, $\beta = 2$, $ActiveThresh = 500ms$, $c_{srt} = 1$, $\gamma = 0.5$, $C = \{2, 6, 12, 16, 20\}$. The AP IFQ size is set to 50 packets for the default case. For the proposed algorithm, we divide the buffer into two separate buffers of length 25 packets. One of them is used as the conventional IFQ and the other one is assigned as the control queue.

We also implemented the method in [4]. The solution in [4] suggests changing the advertised window field in the ACK packets to $\lfloor B/n_{act} \rfloor$ through the AP (B is the buffer size).

Fig. 2 shows instantaneous throughput (averaged over each 1 second interval) for individual uplink TCP flows in the scenario comprising 15 uplink TCP flows. For this experiment, we set the 802.11g physical layer data rate to 24 Mbps (to have a fair comparison with Fig. 1). As it can be observed from Fig. 2, each TCP stream can grab the wireless channel at the start, speed up as a result of controlled ACK scheduling at the AP and achieve a fair average throughput value without experiencing starving at no point in time. On the other hand, this is not the case for the default 802.11 scenario where most of the flows starve in terms of throughput as can be seen in Fig. 1.

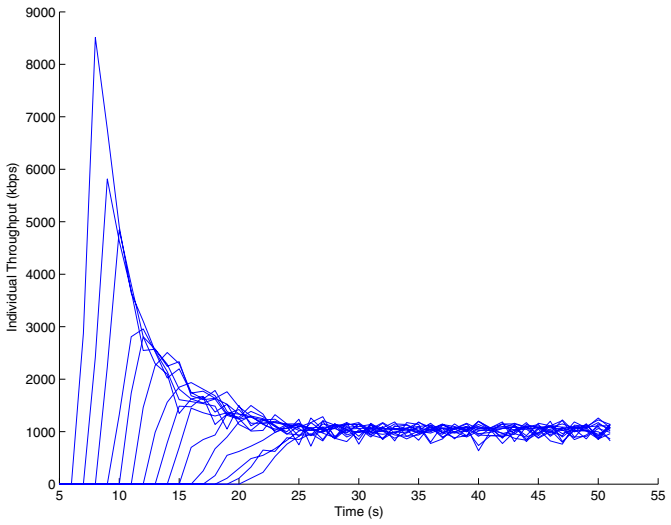

Fig. 2. Instantaneous throughput for individual TCP uplink flows over time for a scenario with 15 uplink streams with the proposed algorithm (grouped assignment method) running at the AP.

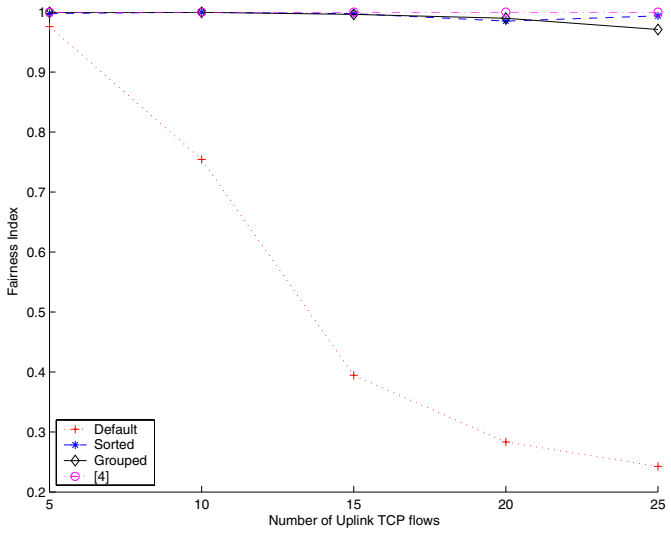

Fig. 3. Fairness index for the proposed and default algorithms and [4] with respect to increasing number of uplink flows with equal RTTs.

Note that in Fig. 2, the initial slopes are steep. Therefore, Fig. 2 implies that the proposed algorithms provide short-term fairness in a congested scenario. This is important because the majority of today's World Wide Web traffic consists of short-lived TCP flows. Our algorithms protect short-lived TCP flows that are sensitive to losses during the early stages of TCP window growth.

Fig. 3 depicts the fairness index for varying number of uplink TCP flows. As can be seen from Fig. 3, the proposed methods considerably improve fairness when compared to the conventional default 802.11 scheme. Actually, grouped assignment method preserves perfect fairness even in the case of 15 uplink TCP flows. The method in [4] can preserve perfect fairness at all loads. On the other hand, as will be shown, this only holds for a scenario with equal RTTs.

Fig. 4 shows the total uplink throughput for varying number of uplink TCP flows. The results also represent the efficiency of different methods in channel utilization. Although providing fairness, Fig. 4 shows [4] achieves %38-%45 lower total throughput than proposed ACK congestion control and filtering methods proposed in this paper. The proposed methods are also shown to achieve higher total throughput than the conventional scheme.

Our methods can finish the upload much quicker on average when compared with the conventional scheme. Although not displayed here, it is worthwhile to note that the mean packet delay is larger for the proposed algorithms. In spite of this, still our proposed algorithms finish the upload faster. Cumulative ACK packets result in new TCP packets to be generated in bursts. Therefore, the data packets in the bursts experience queueing delay before they are transmitted, which increases the mean packet delay time. On the other hand, the bursty nature does not negatively affect the average upload duration.

Our algorithms also provide fair access in the case of

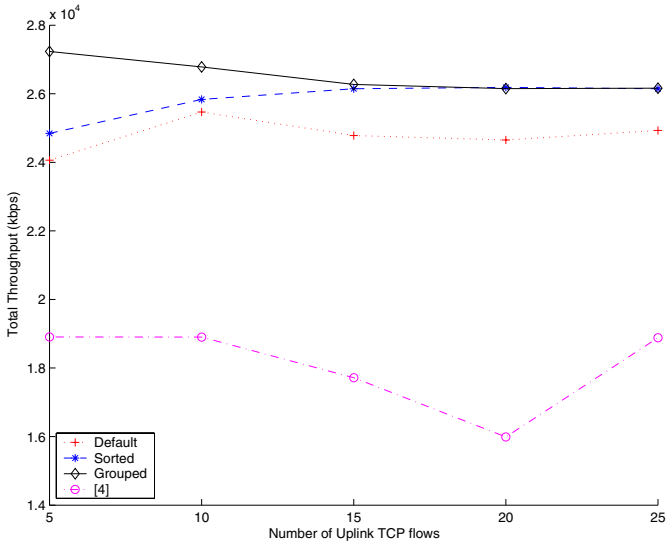

Fig. 4. Total uplink throughput for the proposed and default algorithms and [4] with respect to increasing number of uplink flows with equal RTTs.

varying RTTs. For this scenario, each flow i assumed to have a different wired link delay which is equal to $((i+1)\cdot 10)$ms on the wired route behind the AP. Fig. 5 depicts the fairness index for varying number of uplink TCP flows. As can be seen from Fig. 5, the proposed methods considerably improve fairness when compared to the conventional default 802.11 scheme and [4]. As shown in Fig. 6, ACK congestion control and filtering provides higher total throughput in the case of varying RTTs for individual flows.

We have also tested the performance of our algorithms in the case of wireless channel with errors. For AWGN channel with packet error rate (PER) set to 0.1, similar improvements over the default case are maintained in terms of fairness and channel utilization. Owing to space limitations, we do not include these

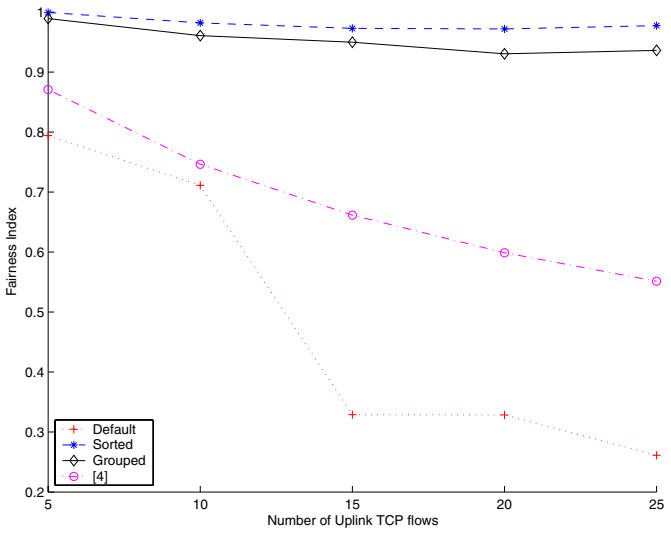

Fig. 5. Fairness index for the proposed and default algorithms and [4] with respect to increasing number of uplink flows and varying RTTs.

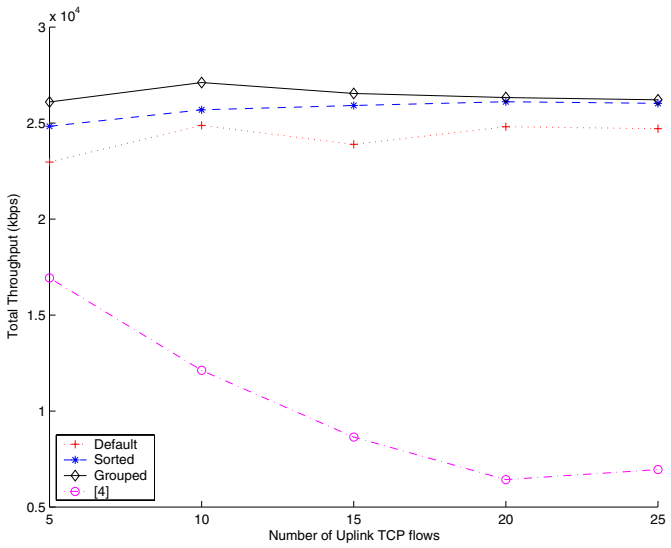

Fig. 6. Total uplink throughput for the proposed and default algorithms and [4] with respect to increasing number of uplink flows and varying RTTs.

results. The overall channel utilization decreases around %10 for all of the algorithms when compared to the case with zero PER due to MAC and TCP layer retransmissions. We should note that TCP layer retransmissions are very rare, although a large PER is employed. This is due to large MAC layer retransmission limit (which is set to 7 as suggested in [1]).

V. Conclusion

In this paper, we have proposed that ACK congestion control and filtering at the AP resolves the uplink TCP unfairness problem in the IEEE 802.11 WLAN. The key point is that the AP efficiently manages downlink ACK packet scheduling among the associated flows with ACK congestion control and filtering. This approach prevents the uplink flows with smaller instantaneous congestion windows starving in terms of throughput due to severe ACK loss in the AP buffer. ACK congestion control and filtering results in a fair bandwidth share for uplink flows.

Two heuristic methods proposed not only improve short- and long-term fairness substantially but also result in efficient channel utilization and high TCP throughput. Moreover, the performance of the proposed methods is shown to be independent of variable TCP RTT. With the proposed method implemented at the AP, the WLAN users can enjoy lower and fair average file upload delays.

References

[1] *IEEE Standard 802.11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE 802.11 Std., 1999.

[2] N. H. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling in a Wireless LAN," in *Proc. ACM Mobicom '00*, August 2000.

[3] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan, "Achieving MAC Layer Fairness in Wireless Packet Networks," in *Proc. ACM Mobicom '00*, August 2000.

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

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

[6] J. Ha and C.-H. Choi, "TCP Fairness for Uplink and Downlink Flows in WLANs," in *Proc. IEEE Globecom '06*, November 2006.

[7] *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.

[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] N. Blefari-Melazzi, A. Detti, A. Ordine, and S. Salsano, "Controlling TCP Fairness in WLAN access networks using a Rate Limiter approach," in *Proc. ISWCS '05*, September 2005.

[10] "RFC3449 - TCP Performance Implications of Network Path Asymmetry," 2002.

[11] "RFC2760 - Ongoing TCP Research Related to Satellites," 2000.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] (2006) The Network Simulator, ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns

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

[20] *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.