WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
*Wirel. Commun. Mob. Comput.* 2015; **15**:1376–1390
Published online 4 September 2013 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.2414

RESEARCH ARTICLE

# Fair and efficient Transmission Control Protocol access in the IEEE 802.11 infrastructure basic service set

Feyza Keceli[1], Inanc Inan[1] and Ender Ayanoglu[2]*

[1] Starix Technology, 9120 Irvine Center Drive, Irvine, CA 92618 U.S.A.
[2] Center for Pervasive Communications and Computing, Department of Electrical Engineering and Computer Science, The Henry Samueli School of Engineering, University of California, Irvine, CA 92697-3975, U.S.A.

## ABSTRACT

When the stations in an IEEE 802.11 infrastructure basic service set employ Transmission Control Protocol (TCP), this exacerbates per-flow unfair access problem. We propose a novel analytical model to approximately calculate the maximum per-flow TCP congestion window limit that prevents packet losses at the access point buffer and therefore provides fair TCP access both in the downlink and uplink. The proposed analysis is unique in considering the effects of varying number of uplink and downlink TCP flows, differing round trip times among TCP connections and the use of delayed TCP acknowledgment (ACK) mechanism. Motivated by the findings of this theoretical analysis and simulations, we design a link layer access control block to be employed only at the access point in order to resolve the unfair access problem. The proposed link layer access control block uses congestion control and ACK filtering approach by prioritizing the access of TCP data packets of downlink flows over TCP ACK packets of uplink flows. Via simulations, we show that the proposed algorithm can provide both short-term and long-term fair accesses while improving channel utilization and access delay. 

**KEYWORDS**
TCP fairness; IEEE 802.11 wireless LAN; infrastructure basic service set

***Correspondence**
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, CA 92697-3975, U.S.A.
E-mail: ayanoglu@uci.edu

## 1. INTRODUCTION

In the IEEE 802.11 wireless local area networks (WLANs), the medium access control (MAC) layer employs the distributed coordination function (DCF), which is a contention-based channel access scheme [1]. The DCF adopts a carrier sense multiple access with collision avoidance scheme using binary exponential backoff procedure. In the DCF, the wireless stations, using all equal contention parameters, have equal opportunity to access the channel. Over a sufficiently long interval, this results in station-based fair access, which can also be referred as MAC layer fair access. On the other hand, per-station MAC layer fair access does not simply translate into achieving per-flow transport layer fair access in the commonly deployed infrastructure basic service set (BSS), where an access point (AP) serves as a gateway between the wired and wireless domains. Because the AP has the same access priority with the wireless stations, an approximately equal bandwidth that an uplink 802.11 station may obtain is shared among all downlink traffic. This results in a considerable asymmetry between per-flow uplink and downlink bandwidth[†].

The network traffic is currently dominated by data traffic mainly using Transmission Control Protocol (TCP) in

[†]The term *bandwidth asymmetry* is typically used when the communication channel presents different physical layer transmission rates, therefore unequal bandwidth, in different directions. Although physical layer capabilities are equal both in the downlink and the uplink in the WLAN, the contention-based MAC layer protocol leads to unfair bandwidth allocation between uplink and downlink. As also used in the literature on this research subject, we label this phenomenon as bandwidth asymmetry in the WLAN.

the transport layer. TCP employs a reliable bidirectional communication scheme. The TCP receiver returns TCP acknowledgment (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 ACK packets of upstream TCP data are queued at the AP together with the downstream TCP data packets. 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 [2].

As will be described in more detail in Section 2.1, unfair bandwidth allocation is observed not only between uplink and downlink TCP flows but also individual uplink TCP flows.

A sufficient condition for resolving the unfair access problem in the 802.11 BSS for TCP is limiting the TCP packet source rate for all flows such that no packet drops occur at the AP [3]. This simply translates into limiting the maximum congestion window size of each TCP connection. In this paper, we propose a simple analytical method to *approximately* calculate the maximum TCP congestion window limit that prevents packet drops from the AP queue (fair congestion window assignment—FCWA). The proposed analysis shows that this window limit can be approximated by a simple linear function of the bandwidth of the 802.11 WLAN, the number of uplink and downlink flows, the wired link delay (LD) of the TCP connection, the MAC buffer size of the AP, and the number of TCP data packets each TCP ACK packet acknowledges. The proposed analysis is generic so that it considers varying number of uplink and downlink TCP flows, the use of delayed TCP ACK algorithm, and varying round trip times (RTTs) among TCP connections. Via simulations, we show that FCWA not only provides fair access but also higher channel utilization when compared with [3]. As we will also describe, the proposed analysis framework can also be used for buffer sizing at the AP in order to provide fair TCP access.

The underlying FCWA idea, that is, no packet drops at the AP buffer, is not a requirement but a sufficient condition for providing fair access in the WLAN. In the second part of the paper, we focus on the design of a link layer access control block that would conversely allow packet drops at the AP buffer but would do that intelligently such that the TCP connections still enjoy fair access. This link layer access control block employs a cross-layer algorithm by prioritizing the access of the TCP data packets of downlink flows over the TCP ACK packets of uplink flows at the AP buffer. This is achieved by applying TCP ACK filtering and compression [2] for uplink TCP flows in the WLAN. Similar to FCWA, the design for the proposed ACK congestion control and filtering (ACCF) algorithm is motivated by keeping the congestion window sizes of each connection at a fair ratio (which directly translates into fair bandwidth allocation). The specific algorithm parameters are quantified on the basis of 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.

The rest of this paper is organized as follows. We illustrate the TCP unfairness problem and provide a brief literature review on the subject in Section 2. Section 3 describes the proposed analytical method to calculate the TCP congestion window limit that prevents packet drops from the AP queue and provides per-flow fair access in the WLAN.

Section 4 describes the proposed link layer access control block, which uses ACK congestion control and filtering for fair access provisioning and evaluates its performance. We provide our concluding remarks in Section 5.

## 2. BACKGROUND

### 2.1. TCP unfairness in the 802.11 WLAN

In the 802.11 WLAN, a bandwidth asymmetry exists between contending upload and download flows[‡]. This is because the MAC layer contention parameters are all equal to the AP and the stations. If $N$ stations and an AP are always contending for the access to the wireless channel (i.e., in saturation[§]), each host ends up having approximately $1/(N+1)$ share of the total transmit opportunities over a long time interval. This results in $N/(N+1)$ of the transmissions being in the uplink, whereas only $1/(N+1)$ of the transmissions belong to the downlink flows.

This bandwidth asymmetry in the forward and reverse path may build up the AP queue resulting in packet drops. As previously stated, upstream TCP ACKs and downstream TCP data are queued at the AP together. 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, that is, a consequent TCP ACK can compensate for the loss of the previous 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, because of the cumulative property of the TCP ACK mechanism, upstream flows with large congestion windows will not experience such frequent timeouts. In

[‡]The terms uplink and upload or downlink and download are used interchangeably in this paper.

[§]Saturation is the limit reached by the system when each station *always* has a packet to transmit. Conversely, in nonsaturation, the (nonsaturated) stations experience idle times because they sometimes have no packet to send.

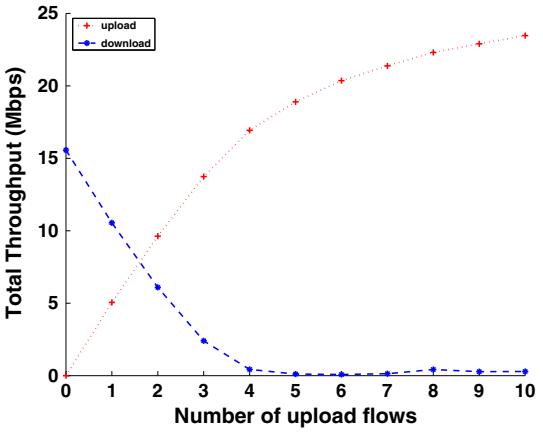



**Figure 1.** The total TCP throughput in the downlink and the uplink when there are 10 download TCP connections and the number of upload TCP connections varies from 1 to 10.

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, whereas 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.

Figure 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 physical layer (PHY) layer with PHY data rate set to 54 Mbps. Other simulation parameters are specified in Section 4.2. The unfairness problem between upstream and downstream TCP flows is evident from the results. For example, in the case of two upload connections, 10 download TCP connections share a total bandwidth of 6.09 Mbps, whereas two upload TCP connections enjoy a larger total bandwidth of 9.62 Mbps. The download TCP connections starve in terms of throughput when the number of upload TCP connections is increased. Note also that these results do not explicitly present the unfairness problem between individual TCP uplink flows. The interested reader is referred to [4] for a specific illustration.

## 2.2. Literature overview

In this section, we classify the studies in the literature on the unfair access problem in the 802.11 WLAN into two categories.

The first group mainly proposes access parameter differentiation between the AP and the stations to combat the problem. Distributed algorithms for achieving MAC layer fairness in 802.11 WLANs are proposed in [5,6]. Several studies propose using the traffic category-based MAC prioritization schemes of the IEEE 802.11e standard [7] mainly designed for QoS provisioning for uplink/downlink direction-based differentiation in order to improve fairness and channel utilization [8–12]. Algorithms that study enhancements on the backoff procedure for fairness provisioning are proposed in [13,14]. 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 techniques that do not require any changes in the 802.11 standard. As we will discuss in the sequel, the proposed methods also do not require any changes in the non-AP 802.11 stations. The proposed solutions can directly be implemented via cross-layer software modules in the AP protocol stack. We leave the performance comparison with MAC layer solutions as future work.

The second group focuses on designing higher layer solutions such as employing queue management and packet filtering schemes, especially for TCP. The TCP uplink and downlink asymmetry problem in the IEEE 802.11 infrastructure BSS is first studied in [3].

The proposed solution of [3] is to manipulate advertised receiver windows of the TCP packets at the AP. In this paper, we propose a simple analytical model to calculate the congestion window limit of TCP flows for the generic case of delayed TCP ACK schemes and varying RTTs among TCP connections [15]. The results of the proposed analysis can be used in the same way as proposed in [3] for fair and efficient access provisioning. Per-flow queueing [16] and per-direction queueing [17] algorithms where distinct queues access the medium with different probabilities are designed for fair access provisioning. A rate-limiter block, which filters data packets both in the uplink and the downlink using instantaneous WLAN bandwidth estimations is proposed in [18]. The use of size-based scheduling policies to enforce fairness among TCP connections is proposed in [19]. 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 [20]. The work presented in this paper proposes a unique congestion control and filtering technique, which also considers the TCP downlink traffic [21]. Note that because TCP downlink traffic load is expected to be larger than the uplink traffic load, this enhancement is vital for a practical implementation. Similar to ACCF algorithm proposed in this paper, virtual queue management (VQ-RED) approach proposed in [22], selective packet marking-ACK filtering (SPM-AF) and least attained service scheduling techniques presented (with limited technical detail on the algorithms) in [23] employ heuristic queue management methodologies to solve the fairness problem in the WLAN. VQ-RED does not consider ACK compression unlike ACCF. SPM-AF uses packet marking at the TCP source to enable prioritization and differentiation among TCP packets at the

AP buffer, where ACCF does not require any changes at mobile stations and/or TCP sources.

An extensive body of work exists relating to the impact of asymmetric paths on TCP performance [24,25] in the wired link context. The effects of ACK congestion control on the performance of an asymmetric network are analyzed in [26] for wired scenarios consisting of only one or two simultaneous flows.

The effects of forward and backward link bandwidth asymmetry have been analyzed in [27] for a wired scenario consisting of only one flow. Similar effects are also observed in practical broadband satellite networks [28]. The effects of delayed acknowledgements and byte counting on TCP performance are studied in [29]. Several schemes are analyzed in [30,31] 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 because of the dynamics of two-way traffic using the same buffer is presented in [32]. Acknowledgement based on congestion window estimation technique is proposed in [33]. The features of TCP control block sharing from ensemble TCP (E-TCP) [34] with the ACK filtering and ACK congestion control mechanisms [2] are combined in [35] to improve E-TCP performance over asymmetric networks. The interactions between two-way TCP connections sharing bottleneck wired links are thoroughly analyzed in [36], and a core phenomenon named as data pendulum is observed. As [36] shows, the data pendulum effect, that is, data and ACK segments alternately fill only one of the link buffers (on the upload or download side), arises when the upload bandwidth is smaller than or equal to the download bandwidth for a wired network scenario. Conversely, as we discussed in Section 2.1, WLAN traffic may enjoy larger total bandwidth in the uplink, which is a scenario not considered in the analysis in [36].

In this paper, we design an 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 are unique in that the parameters of the algorithm are quantified according to the TCP access characteristics in an 802.11 infrastructure BSS.

## 3. TCP FAIRNESS ANALYSIS

The TCP unfairness problem originating from the uplink/downlink access asymmetry can be resolved if packet drops at the AP buffer are prevented such as in the unrealistic case of an infinitely long AP queue. In this case, congestion windows of all TCP flows whether in the downlink or uplink reach up to the receiver advertised congestion window limit and stay at this value. This results in fair access as opposed to the fact that the access is asymmetric in the 802.11 infrastructure BSS as described in Section 2.1. As the infinitely long queue assumption is unrealistic, the exact same result of no packet drops can be achieved if TCP senders are throttled by limiting the number of packets in flight, that is, the TCP congestion windows are assigned regarding the available AP bandwidth in the downlink. In this section, we propose a simple and novel analytical model to approximately calculate the maximum congestion window limit of each TCP flow that prevents packet losses at the AP buffer, therefore provides fair and efficient TCP access in the BSS.

Each random access system exhibits cyclic behavior. The cycle time is defined as the average duration in which an arbitrary tagged station successfully transmits one packet on average. Our analytical method for calculating the TCP congestion window limit that achieves fair and efficient access is based on the cycle time analysis previously proposed for 802.11 MAC performance modeling [37,38]. The simple cycle time analysis assesses the asymptotic performance of the DCF accurately (when each contending AC always has a packet in service). We use the approach in [37] to derive the explicit mathematical expression for the average DCF cycle time when necessary. In Section 3.2, we will describe the necessary extensions to employ the cycle time analysis in the proposed analysis. Because of space limitations, the reader is referred to [37,38] for details on the derivation of the DCF cycle time.

We consider a typical network topology where a TCP connection is initiated between a wireless station and a wired station either in the downlink or uplink of the WLAN. The WLAN traffic is relayed to the wired network through the AP and vice versa. Let RTT denote the average length of the interval from the time a TCP data packet is generated until the corresponding TCP ACK packet arrives. RTT is composed of three main components as follows[¶].

- **Wired LD.** The flow-specific average propagation delay of the packet between the AP and the wired node in the uplink ($LD_u$ from the AP to the wired node) and in the downlink ($LD_d$ from the wired node to the AP).
- **Queueing delay ($QD$).** The average delay experienced by a packet at the wireless station buffer until it reaches the head of the queue. Note that due to the unequal traffic load at the AP and the stations, $QD_{AP}$ and $QD_{STA}$ may highly differ.
- **Wireless medium access delay ($AD$).** The average AD experienced by a packet from the time it reaches the head of the MAC queue until the transmission is completed successfully.

[¶]In this work, we do not consider the time varying aspect of RTT in order to keep the analysis simple. The extension of the analysis for this case is left as future work. The motivation behind the analysis is to calculate the *approximate* congestion window that provides fair access. This also shows that the throttling of downlink and uplink TCP sources results in fair TCP access in the WLAN. Moreover, as it will be described in the sequel, this approximate analysis can effectively be employed in applications such as buffer sizing.

Then, RTT is calculated as follows[‖].

$$RTT = LD_u + LD_d + QD_{AP} + QD_{STA} + AD_{AP} + AD_{STA} \quad (1)$$

For the first part of the analysis, each TCP data packet is assumed to be acknowledged by a TCP ACK packet where this assumption is later released and the delayed TCP ACK algorithm is considered.

We claim that if the system is to be stabilized at a point such that no packet drops occur at the AP queue, then the following conditions should hold.

- *All non-AP stations are in nonsaturated condition.* Let us assume that a station has $X$ packets (TCP data or ACK) in its queue. A new packet is generated only if the station receives packets (TCP ACK or data) from the AP (as a result of ACK-oriented rate control of TCP). Let $Y > 1$ users be active. Every station (including the AP) sends one packet successfully on average every cycle time [37]. In the stable case, while the tagged station sends $Y$ packets every $Y$ cycle time, it receives only one packet. Note that the AP also sends $Y$ packets during $Y$ cycle times, but on the average, $Y - 1$ of these packets is destined for the stations other than the tagged one. Therefore, after $Y$ cycle times, the tagged station's queue size will drop down to $X - Y + 1$. Because $Y > 1$, the tagged station's queue will be empty eventually. A new packet will only be created when the AP sends a TCP packet to the tagged station, which will be served before it receives another packet (on average). This proves that all the non-AP stations are in nonsaturated condition if no packet losses occur at the AP.
- *The AP contends with at most one station at a time on average.* Following the previous claim, a non-AP station (which is nonsaturated) can have a packet ready for transmission if the AP has previously sent a packet to the station. There may be transient cases where the instantaneous number of active stations may become larger than 1. On the other hand, as we have previously shown, when $Y > 1$, the queue at any non-AP station eventually empties. If we assume the transient duration being very short, the number of actively contending stations on average is 1. Therefore, at each DCF cycle time, the AP and distinct station will transmit a packet successfully.

We define $CT_{AP}$ as the duration of the average cycle time during which the AP sends an arbitrary packet (TCP data or ACK) successfully. We will derive $CT_{AP}$ in Section 3.2. Let the average duration between two successful packet transmissions of an arbitrary flow at the AP (or at the non-AP station) be $CT_{flow}$. Assuming there are $n_{up}$ and $n_{down}$ upload and download TCP connections, respectively, we make the following approximation based on our claims that the AP contends with one station on average, and the TCP access will be fair if no packet drops are observed at the AP buffer

$$CT_{flow} \cong (n_{up} + n_{down}) \cdot CT_{AP} \quad (2)$$

As it will be shown by comparing with simulation results in Section 3.5, the approximation in (2) leads to analytically correct results.

Then, the throughput of each station (whether it is running an uplink or a downlink TCP connection) is limited by $1/CT_{flow}$ (in terms of packets per second). We can also write the TCP throughput using $W_{lim}/RTT$, where we define $W_{lim}$ as the TCP congestion window limit for a TCP connection.

Following our previous claims, $QD_{STA} = 0$ (the stations are nonsaturated), $QD_{AP} = (BS_{AP} - 1) \cdot CT_{AP}$ (we consider the limiting case when the AP buffer is full, but no packet drop is observed), and $AD_{AP} + AD_{STA} = CT_{AP}$ (the AP contends with one station on average), where $BS_{AP}$ is the buffer size of the AP MAC queue. Using $1/CT_{flow} = W_{lim}/RTT$, we find

$$W_{lim} = \frac{LD_u + LD_d}{CT_{flow}} + \frac{BS_{AP}}{n_{up} + n_{down}} \quad (3)$$

Note that $CT_{flow}$ is an indication of the bandwidth at the bottleneck (at the AP). If the data rate exceeds this bandwidth, the excess data will be queued at the AP, eventually overflowing the AP buffer. We calculate $W_{lim}$ considering a full AP buffer, therefore, $W_{lim}$ is approximately the *maximum* congestion window limit for a TCP connection that prevents the packet drops at the AP queue of size $BS_{AP}$.

We can make following observations from (3).

- $W_{lim}$ is a function of $LD_u$ and $LD_d$. Therefore, $W_{lim}$ is flow-specific and varies among connections with various link delays.
- The first term is the effective number of packets that are in flight in the wired link for any flow, whereas the second term is the number of packets that are in the AP buffer for the same flow.

### 3.1. Delayed TCP acknowledgements

In the delayed TCP ACK mechanism, the TCP receiver acknowledges every $b$ TCP data packets ($b > 1$). A typical value (widely used in practice) is $b = 2$.

The use of delayed TCP ACK mechanism changes the system dynamics. On the other hand, we still employ our assumption that the AP contends one station at a time on the average to calculate $CT_{AP}$. As will be shown by comparison with simulation results in Section 3.5, this assumption still leads to analytically accurate results.

[‖]Round trip time is calculated as in (1) irrespective of the direction of the TCP connection. On the other hand, specific values of $AD$ and $QD$ depend on the packet size, the number of contending stations, and so on. Therefore, RTT of an uplink connection may differ from RTT of a downlink connection.

We update (2) and (3) accordingly for delayed TCP ACKs. Let the average duration between two successful packet transmissions of the flow at the non-AP station be $CT_{flow,del}$ when delayed TCP ACK mechanism is used. Each uplink flow completes the successful transmission of $b$ packets in an interval of average length $b \cdot CT_{flow,del}$. When the access is fair, the AP transmits $b$ data packets for each downlink flow (i.e., a total of $b \cdot n_{down}$) and one ACK packet for each uplink flow (i.e., a total of $n_{up}$ ACK packets) during the same interval. Then,

$$CT_{flow,del} \cong \left(\frac{n_{up}}{b} + n_{down}\right) \cdot CT_{AP} \tag{4}$$

$$W_{lim} = \frac{LD_u + LD_d}{CT_{flow,del}} + \frac{BS_{AP}}{n_{up}/b + n_{down}} \tag{5}$$

### 3.2. Calculating $CT_{AP}$

We are interested in the case when there are two active (saturated) stations (as the AP contends with one station at a time). The average cycle time in this scenario can easily be calculated using the model in [37]. In our case, the AP sends the TCP ACK packets of the uplink TCP connections and the TCP data packets of the downlink TCP connections which contend with the TCP ACK packets of the downlink TCP connections and the TCP data packets of the uplink connections that are generated at the stations. Note that the cycle time varies according to the packet size of contending stations. Then,

$$CT_{AP} = \sum_{p_1 \in S} \Pr(p_{AP} = p_1) \sum_{p_2 \in S} \Pr(p_{STA} = p_2)\, CT_{p_1,p_2} \tag{6}$$

where $S = \{ACK, DATA\}$ is the set of different types of packets, $\Pr(p_{AP} = p_1)$ is the probability that the AP is sending a packet of type $p_1$, $\Pr(p_{STA} = p_2)$ is the probability that the non-AP station is sending a packet of type $p_2$, and $CT_{p_1,p_2}$ is the average cycle time when one station is using a packet of type $p_1$ and the other is using a packet type of $p_2$. We differentiate between the data and the ACK packets because the size of the packets thus the cycle time duration depends on the packet type.

Using simple probability theory, we can calculate $\Pr(p_{AP})$ and $\Pr(p_{STA})$ as follows

$$\Pr(p_{AP} = p_1) = \begin{cases} \dfrac{n_{down}}{n_{up}/b + n_{down}}, & \text{if } p_1 = DATA \\ \dfrac{n_{up}/b}{n_{up}/b + n_{down}}, & \text{if } p_1 = ACK, \end{cases} \tag{7}$$

$$\Pr(p_{STA} = p_2) = \begin{cases} \dfrac{n_{up}}{n_{down}/b + n_{up}}, & \text{if } p_2 = DATA \\ \dfrac{n_{down}/b}{n_{down}/b + n_{up}}, & \text{if } p_2 = ACK. \end{cases} \tag{8}$$

### 3.3. Fair congestion window assignment

A control block located at the AP can modify the advertised receiver window field of the ACK packets that are all relayed through with the $W_{lim}$ value calculated using the proposed model. Therefore, we call this procedure FCWA.

The analysis requires accurate estimations on $LD_u + LD_d$ and $b$. The control block may distinguish among TCP connections via the IP addresses and the ports they use. An averaging algorithm can be used to calculate the average time that passes between sending a data (ACK) packet into the wired link and receiving the ACK (data) packet, which is generated by the reception of the former packet (which is $LD_u + LD_d$). The TCP header of consecutive ACK packets may be parsed to figure out the value of $b$.

It is also worth noting that although the analytical calculation uses a simple cycle time method in calculating $CT_{AP}$ and $CT_{flow}$, the AP may use a measurement-based technique rather than the model-based technique used in this paper.

### 3.4. Buffer sizing

The proposed analysis can effectively be used for buffer sizing purposes. The 802.11 vendors may use the proposed method with statistics of TCP connections and WLAN traffic to decide on a *good* size of AP buffer that would provide fair TCP access**.

$$BS_{AP} = \left(W_{lim} - \frac{LD_u + LD_d}{CT_{flow}}\right) \cdot (n_{up}/b + n_{down}) \tag{9}$$

### 3.5. Performance evaluation

We validate the analytical results obtained from the proposed model via comparing them with the simulation results obtained from ns-2 [39]. We obtained $W_{lim}$ via simulations in such a way that increasing the TCP congestion window limit of TCP connections by one packet results in a packet loss ratio larger than 1% at the AP buffer.

As previously stated, the network topology is such that 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 an FTP agent that models bulk data transfer. TCP NewReno with its default parameters in ns-2 is used. All the stations have 802.11g PHY [40] with 54 and 6 Mbps as the data and the basic rate, respectively. The wired link data rate is

**Note that (9) is just a different representation of (3). $W_{lim}$ and $LD_u + LD_d$ are flow specific. When the system designer treats each flow separately, $BS_{AP}$ may differ depending on target $W_{lim}$ and estimated link delays for each flow. The system designer may adjust target $W_{lim}$ for each flow as needed and set the appropriate buffer size to meet the fairness requirement.

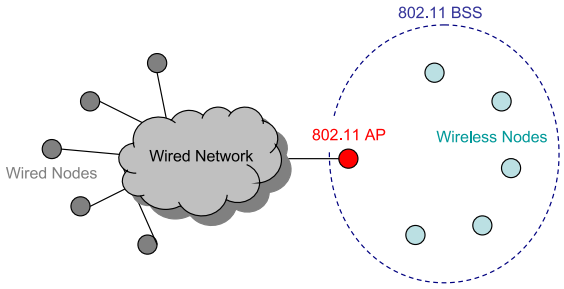



**Figure 2.** The network topology for simulations.

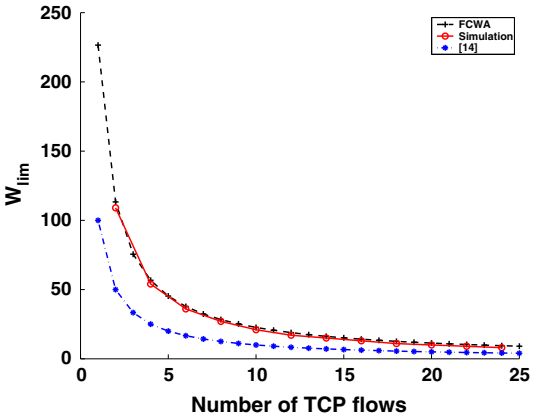



**Figure 3.** Congestion window limits calculated by FCWA, simulation, and [3].

100 Mbps. The default DCF MAC parameters are used [1]. The packet size is 1500 bytes for all flows. The MAC buffer size at the stations and the AP is set to 100 packets. Figure 2 shows the network topology for simulations.

In the first set of experiments, we set the wired LD of each connection to 50 ms. Each TCP data packet is acknowledged by an ACK packet ($b = 1$). In Figure 3, we compare the estimation of (3) on the congestion window limit with the values obtained from the simulation results and the proposed method of [3] for the increasing number of TCP connections. The number of upload flows is equal to the number of download flows.

As Figure 3 implies, the analytical results for FCWA and the simulation results are well in accordance. The analysis in [3] calculates the congestion window limit by $BS_{AP}/(n_{up} + n_{down})$ and underestimates the actual fair TCP congestion window limit.

Although the corresponding results are not displayed, both FCWA and [3] achieve perfect fairness in terms of per-connection FTP throughput for any number of TCP connections (Jain's fairness index [41], $f > 0.9999$ where 1 shows perfect fairness). We will evaluate FCWA in terms of fairness index in Section 4.2 for another scenario.

The total throughput of the system when the TCP connections employ analytically calculated congestion window limits in simulation for increasing number of TCP

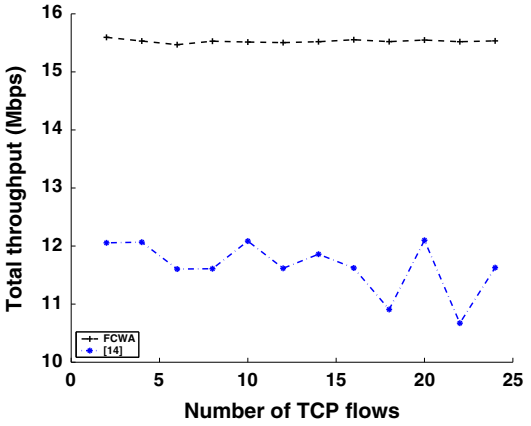



**Figure 4.** Total throughput of the system when the TCP connections employ analytically calculated congestion window limits by FCWA and [3].

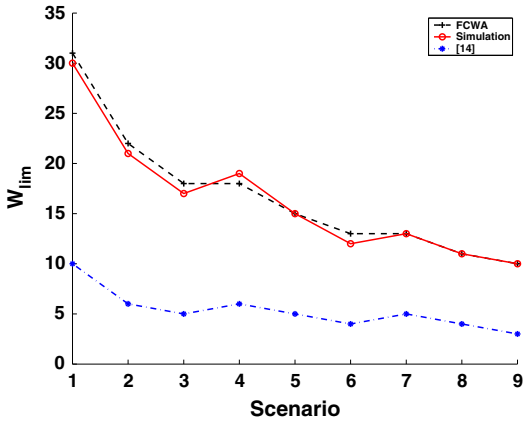



**Figure 5.** Congestion window limits calculated by FCWA, simulation, and [3], when TCP connections employ delayed ACK mechanism with $b = 2$.

connections is shown in Figure 4. As the comparison with [3] reveals, the congestion window limits calculated via FCWA result in approximately 35–50% higher channel utilization for the specific scenario. We note that in order to provide a close comparison, in Figure 4 and a number of upcoming figures, the vertical scale has been expanded and does not start from 0.

In the second set of experiments, we consider a scenario where the TCP connections use the delayed ACK mechanism with $b = 2$. We consider nine different scenarios where in each scenario, the number of uplink and downlink TCP connections varies. In the first three scenarios, the number of downlink flows is set to 5, and the number of uplink flows is varied among 5, 10, and 15, respectively. Varying the number of uplink flows in the same range, the next three scenarios use 10, and the following three scenarios use 15 downlink flows. In Figure 5,

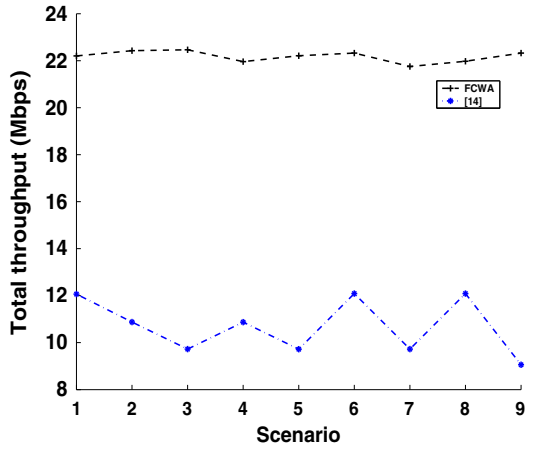



**Figure 6.** Total throughput of the system when the TCP connections employ analytically calculated congestion window limits by FCWA and [3] in the case delayed ACK mechanism is used ($b = 2$).

we compare the estimation of (5) on the congestion window limit with the values obtained from the simulation results and the proposed method of [3]. The analytical results for the proposed model and the simulation results are well in accordance. The total throughput of the system when the TCP connections employ the analytically calculated congestion window limits is shown in Figure 6. As the comparison with [3] reveals, the congestion window limits calculated via our method result in approximately 90–105% higher channel utilization. Although the corresponding results are not presented, the congestion window limits calculated by both FCWA and the method of [3] achieve perfect fair resource allocation in terms of throughput in all cases (Jain's fairness index, $f > 0.998$). On the other hand, the proposed FCWA method results in a significantly higher channel utilization.

We evaluate the accuracy of analytical estimations and the performance of FCWA for varying wired LD among connections in a technical report due to space limitations [4].

## 4. LINK LAYER ACCESS CONTROL BLOCK

As illustrated in Section 2.1, unfair access problem originates from the uplink/downlink bandwidth asymmetry in the 802.11 BSS. As our analysis in Section 3 shows, fair access can be achieved if the congestion window limits of the downlink and the uplink TCP sources are set regarding the network bandwidth so that no packet drops occur at the AP buffer. On the other hand, FCWA's philosophy, that is, *no packet drop at the AP buffer*, is not a requirement but a sufficient condition for achieving fair TCP access in the WLAN. In this section, we focus on the design of a cross-layer algorithm to be employed at a link layer access control block at the AP. The research mainly focuses on the design of an algorithm that would keep instantaneous TCP connection window sizes at fair levels (aligns with FCWA motivation) but would allow *intelligently made* packet drops at the AP buffer (challenges FCWA idea).

We can further make the following observations for TCP behavior in 802.11 WLAN

- For the default DCF scenario, when only downlink flows are present, the data packet drops at the AP buffer implicitly and effectively throttles the downlink TCP sources (i.e., TCP access is fair among downlink flows in this case as we also present in [4] via simulations).
- Conversely, when uplink and downlink flows coexist, downlink flows may starve in terms of throughput as some uplink flows are fortunate enough to reach a high congestion window by making use of the cumulative property of TCP ACKs. The TCP ACKs of uplink flows occupy most of the AP buffer, which results in data packet drops for downlink flows.

These observations motivate the main idea the proposed link layer access control block employs, prioritizing TCP data packets of downlink flows over TCP ACK packets of uplink flows at the AP MAC buffer. This is achieved through TCP ACK compression and filtering for uplink TCP flows.

The proposed ACCF 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 are 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 ACCF 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 (because 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 flow $i$ in the access control block buffer. Let $t_{buf,i}$ denote the total time that has passed because the last

TCP ACK for flow $i$ has been sent to the MAC queue. Let $\beta$ be a constant weighting factor and $\gamma$ be a variable weighting factor, which is a function of $num_{cum,i}$. Let $AvgInt_i$ be the measured average packet interarrival time for flow $i$[††]. Let $AvgDataInt$ be the average downlink data interarrival duration, which we use to decide how frequent the ACKs of uplink flows should be sent down to the MAC queue for transmission. We calculate $AvgDataInt$ by taking the mean of $AvgInt$ of the downlink flows with $AvgInt < \alpha \cdot \min(AvgInt_j : \forall j \; in \; downlink)$ where $1 < \alpha$ is a constant. Note that this averaging calculation excludes the TCP sources with packet interarrivals higher than a threshold (as quantified by $\alpha$) in order to prevent slow downlink flows limiting the frequency of uplink ACKs, therefore the uplink bandwidth unnecessarily.

According to the proposed ACCF 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 the control queue buffering time for each ACK packet of uplink flow $i$ as $D_i = \gamma \cdot num_{cum,i} \cdot AvgDataInt - t_{buf,i}$. The rationale behind this equation is as follows.

- We consider the cumulative number of ACK packets that the currently buffered ACK packet represents. 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 weighting 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 $\gamma$ 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 subtract $t_{buf,i}$ 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 $num_{cum,i} > num_{thresh}$).

As we have also observed via simulations, the ACK filtering scheme makes the ACK arrivals to the AP queue bursty [2]. For an arbitrary uplink flow, this behavior corresponds to alternating idle times with no packet arrivals and active times consisting of a bunch of highly frequent ACK arrivals to the AP queue. This bursty behavior may result in $t_{buf,i} > \gamma \cdot num_{cum,i} \cdot AvgDataInt$ (probably when the corresponding idle duration is long), therefore $D_i < 0$, especially for the first few ACK arrivals at the AP queue following an idle time for the corresponding flow. Note that the case of $D_i < 0$ actually translates into the case of the ACK being already due for transmission. In this case, our design takes one of the two alternative actions regarding the value of $D_i$ as follows.

- $D_i + \beta \cdot AvgInt_i < 0$. This serves as an indication of the last ACK passed to the MAC queue having been performed probably within the previous burst. Although the ACK transmission is due ($D_i < 0$), an immediate pass to the MAC queue punishes uplink throughput unnecessarily as $t_{buf,i}/num_{cum,i}$ (which is an approximation for the average data transmission interval of uplink flow $i$) is much larger than $AvgDataInt$. In this case, the ACK packet of flow $i$ is delayed for the duration equal to $D_i' = \beta \cdot AvgInt_i$ in the control block queue (counting on the high probability of further ACK arrivals in the current burst). Our intuition behind the calculation of $D_i'$ is the possibility of the next ACK of the same flow arriving possibly in an average interarrival time $AvgInt_i$. We also introduce the constant weighting factor $\beta > 1$ in order to compensate for the potential variance of the *instantaneous* ACK interarrival time. A new ACK arrival will probably decrease $t_{buf,i}/num_{cum,i}$ taking it closer to $AvgDataInt$.
- If $D_i + \beta \cdot AvgInt_i > 0$, the relaying ACK packet is sent down to the MAC queue as the ACK is already due for transmission and $t_{buf,i}/num_{cum,i}$ is close to $AvgDataInt$.

As previously stated, if a new TCP ACK packet arrives before the timer that is initially set to $D_i$ or ($D_i'$) 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 $D_i$ or ($D_i'$) 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.

Acknowledgement filtering may slow down the congestion window growth rate, negatively impacts the performance during loss recovery and slow start, and increases the RTT [26]. On the other hand, because our idea is trying to slow down uplink TCP flows in order to prioritize

[††] $AvgInt_i$ denotes 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. $AvgInt_i$ can be calculated by employing simple averaging methods (such as exponential moving weighted average (EWMA) that we have employed for uplink measurements in [20]) on periodic measurements results.

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 or duplicate TCP ACKs.

The proposed ACCF algorithm introduces a number of configurable variables. As pointed out in Section 4.2, we decided the values for these variables through extensive simulations.

### 4.1. Fairness measure

Most of the studies in the literature quantify the fairness by employing Jain's fairness index [41] or providing the ratio of the throughput achieved by individual or all flows in the specific directions. On the other hand, such measures have the implicit assumption of each flow or station demanding asymptotically high bandwidth (i.e., in saturation). As these measures quantify, a perfectly fair access translates into each flow or station receiving an equal bandwidth. On the other hand, in a practical scenario of flows with finite and different bandwidth requirements (i.e., some stations in nonsaturation), these measures cannot directly be used to quantify the fairness of the system.

We define the fair access in a scenario where flows with different bandwidth requirements coexist as follows.

- The flows with the total bandwidth requirement lower than the fair per-flow channel capacity in the specific direction receive the necessary bandwidth. Such flows are called nonsaturated in the sequel.
- The flows with total bandwidth requirement higher than the fair per-flows channel capacity receive an equal bandwidth. Such flows are called saturated in the sequel.

In order to quantify fair access, we propose to use the MAC queue packet loss rate (a PLR of 0 for all flows corresponds to fair access) for the latter together with the comparison on channel access rate (equal channel access rate corresponds to fair access) for the former. Note that the latter can employ Jain's fairness index, $f$, which is defined in [41] as follows: if there are $n$ concurrent connections in the network and the throughput achieved by connection $i$ is equal to $x_i$, $1 \leq i \leq n$, then

$$f = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \tag{10}$$

### 4.2. Performance evaluation

We implemented the proposed link layer control access block employing ACCF in ns-2 [39].

The network topology in Figure 2 and the stated parameters in Section 3.5 are used. The TCP traffic uses either a FTP agent, which models bulk data transfer, or a Telnet agent, which simulates the behavior of a user with a terminal emulator or a web browser[‡‡]. Unless otherwise stated, flows are considered to be lasting through the simulation duration and are 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 are transferred. 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 are inherited from [3]. 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 $\alpha = 1.5$, $\beta = 2$, $\gamma_{min} = 0.5$, and $num_{thresh} = 10$ to be appropriate through extensive simulations. The simulation duration is 350 s.

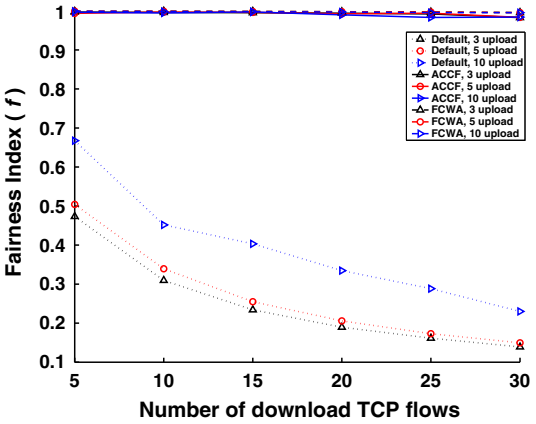



**Figure 7.** Fairness index among all TCP flows when 3, 5, or 10 upload TCP connections are generated, and the number of download TCP connections are varied from 5 to 30.

We investigate the system performance when wired ($LD$) differ among TCP connections. For simulation simplicity, we assume that $LD_u$ and $LD_d$ are equal for a specific connection. The wired LD 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 LD of 2 ms larger than the previous one in the same direction.

#### 4.2.1. The basic scenario

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

Figure 7 shows the fairness index among all connections. We compare the default DCF results with the results

[‡‡]File transfer protocol flows are also labeled as saturated flows in this paper as they intend to consume all the available bandwidth (bulk data transfer model). Conversely, Telnet flows are rate-controlled and experience idle times and therefore are labeled as nonsaturated.

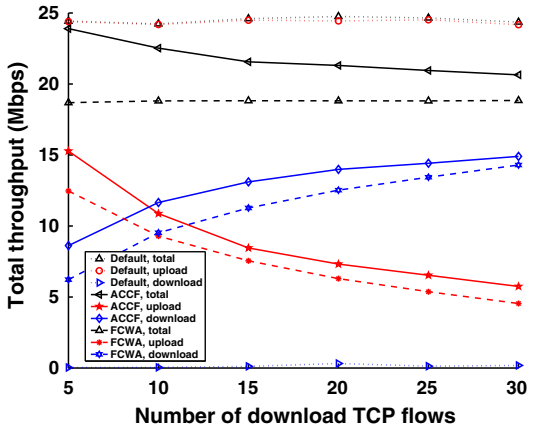



**Figure 8.** Total throughput of upload, download, and both directions, when 10 upload TCP connections are generated, and the number of download TCP connections are varied from 5 to 30.

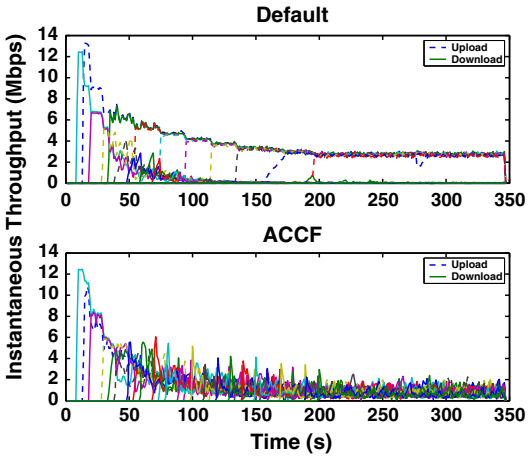



**Figure 9.** Individual instantaneous throughput for upload and download TCP connections when TCP receivers employ the delayed ACK mechanism ($b = 2$).

obtained when the AP employs the proposed *i)* FCWA or *ii)* ACCF. As the results imply, with the introduction of any of the proposed control blocks at the AP, an almost perfect fair resource allocation can be achieved in both cases.

In Figure 8, we plot the uplink, downlink, and total TCP throughput in the infrastructure BSS. We only present the scenario when there are 10 upload TCP connections. Similar results are observed for other cases with 3 and 5 upload TCP connections. As the results show, using the proposed ACCF scheme, the downlink flows (which starve in the default DCF case) can achieve reasonable throughput. If we employ FCWA instead, the total throughput observed is slightly lower. In this case, the proposed ACCF scheme makes use of the ACK filtering scheme to achieve a higher channel utilization. The comparison with the performance of the default DCF algorithm implies that the proposed methods do not sacrifice channel utilization while providing fair access.

#### 4.2.2. Delayed TCP ACKs

In the second set of experiments, we use a scenario when TCP connections use the delayed TCP ACK mechanism ($b = 2$).

We start the download and upload FTP connections in 10 and 20 s intervals, respectively. Figure 9 shows the instantaneous throughput for individual TCP flows over the 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 ACCF scheme, all uplink and downlink TCP flows enjoy fair access. The results are important in showing the proposed algorithm's effectiveness even when the delayed TCP ACK mechanism is used.

#### 4.2.3. Wireless channel errors

In the third set of experiments, we assume the wireless channel to be an additive white Gaussian noise 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 [42] using the way of realization in [43]. Our model considers the channel noise power in signal-to-noise ratio.

We set wireless channel noise levels such that each station experiences a finite data packet error rate (PER). We repeat the tests for additive white Gaussian noise channel signal-to-noise ratio 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, because 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. Figure 10 shows that the proposed ACCF scheme provides fair access. The performance of ACCF is resilient to wireless channel errors, that is, fair access is preserved even when there are errors in the wireless channel. Although not presented here, the throughput drops slightly when compared with the errorless wireless channel case due to the MAC retransmissions.

#### 4.2.4. Varying source packet rates among TCP connections

In the fourth set of experiments, we test the performance when half of the stations use the FTP agent, whereas the other half use the Telnet agent with packet rates ranging from 150 Kbps to 550 Kbps.

Figures 11 and 12 compare the performance in terms of fair access and total throughput for default DCF and ACCF for increasing the number of TCP stations in each direction, respectively. In, Figure 11, the right $y$-axis denotes the fairness index, $f$, among the FTP (saturated) flows,

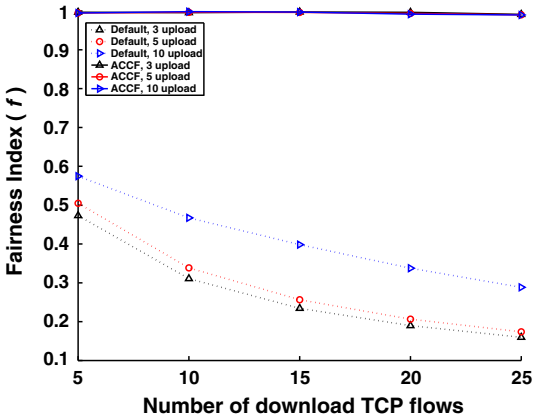



**Figure 10.** Fairness index among all TCP flows over an AWGN channel when 3, 5, or 10 upload TCP connections are generated, and the number of download TCP connections are varied from 5 to 30.

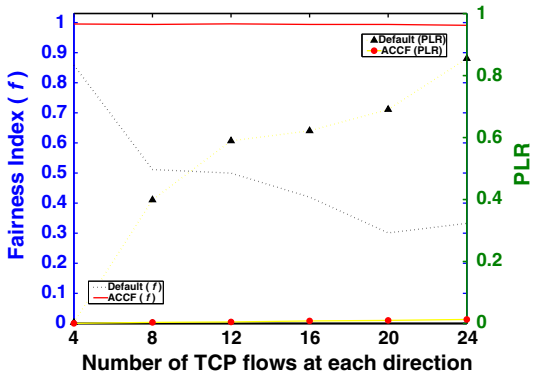



**Figure 11.** Fairness index for saturated and packet loss rate (PLR) for nonsaturated TCP flows when the default DCF or ACCF is employed.

whereas the left $y$-axis denotes the average PLR for Telnet (nonsaturated) flows. As the results present, the proposed ACCF scheme can provide fair access (i.e., $f = 1$ and $PLR = 0$) irrespective of the number of stations.

As Figure 12 shows, high channel utilization is also maintained. Although not explicitly shown in Figure 12, all TCP downlink flows are shut down when the default DCF is employed especially when the number of uplink TCP connections is large (as described in Section 2.1). In this case, the shared channel can mainly be utilized by data packets of the uplink TCP connections and the corresponding ACKs. In a fair scenario, as for ACCF, the TCP ACKs of the downlink connections sharing the medium are considerably higher in number than it is the case for the default DCF. As the MAC efficiency decreases when packets of shorter length access the channel, ACCF channel utilization efficiency is slightly lower than DCF when the number of uplink flows is large.

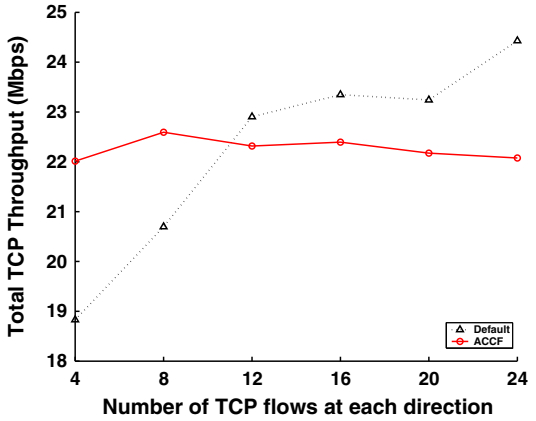



**Figure 12.** Total TCP throughput when the default DCF or ACCF is employed.

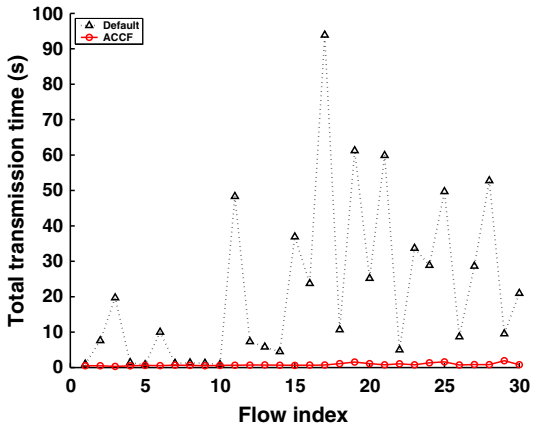



**Figure 13.** The total transmission duration for individual short-lived TCP flows.

### 4.2.5. Short-lived TCP flows

In the fifth set of experiments, we test the performance in terms of short-term fairness. First, we generate five uplink and 10 downlink long-lived FTP flows. Then, 15 short-lived uplink and downlink FTP flows are generated with 5 s interval, consecutively. Figure 13 shows the total transmission duration for individual short-lived FTP flows for the proposed ACCF algorithm and the default DCF. Note that the flow indices from 1 to 15 represent uplink FTP flows, whereas flow indices from 16 to 30 represent downlink FTP 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 ACCF algorithm is short-term fair. Although not explicitly presented, most of the downlink connections experience connection timeouts and even cannot complete the whole transaction within the simulation duration for the default case.

The reader is referred to [4] for additional simulation results.

## 5. CONCLUSION

In this paper, we focused on unfair TCP access problem in an IEEE 802.11 infrastructure BSS. We have presented a novel and simple analytical model to calculate the TCP congestion window limit that provides fair TCP access in a wired/wireless scenario. The key contribution of this study is that the proposed analytical model considers varying wired LDs among connections, varying number of uplink and downlink connections, and the use of the delayed ACK mechanism. Via simulations, we have shown that the congestion window limits calculated via the proposed analysis (FCWA) provide fair TCP access and high channel utilization. The same model can also be used to decide on the required AP buffer size for fair TCP access given the TCP congestion window limits used by the connections. The cycle time analysis can be extended to the IEEE 802.11e WLANs [7] as in [38], therefore, the analysis in this paper can also be extended to the case when MAC parameter differentiation is used.

We have also designed a novel link layer access control block for the AP that provides fair TCP access in an 802.11 infrastructure BSS. Our simple idea for resolving the unfairness problem in the WLAN is prioritizing TCP data packets of downlink flows over TCP ACK packets of uplink flows at the AP. This idea originates from the main finding of the proposed analytical model, which shows that fair access can be achieved by throttling TCP traffic (i.e., limiting congestion windows). By design, the link layer access control block employs a proposed ACCF algorithm. The proposed ACCF algorithm is unique in that the specific algorithm parameters are based on the measured average data transmission rate at the AP. Via simulations, we showed that fair resource allocation for uplink and downlink TCP flows can be provided in a wide range of practical scenarios when the proposed ACCF 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, ACCF does not require any changes in the 802.11 standard nor any enhancement at the stations.

Both FCWA and ACCF schemes employ a cross-layer design principle, accessing the transport layer headers and parameters at the link layer. The analysis and the simulation results present such cross-layer schemes can resolve the unfair TCP access problem effectively. The comparison of this approach with sole transport and MAC layer approaches is an immediate future research subject.

## ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers whose comments improved the quality of the paper.

This work was supported by the Center for Pervasive Communications and Computing and by the 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.

## REFERENCES

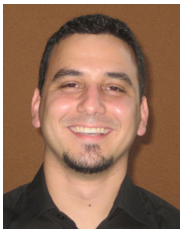
1. IEEE Standard 802.11: wireless LAN medium access control (MAC) and physical layer (PHY) Specifications, 1999.
2. Balakrishnan H, Padmanabhan V, Katz RH. The effects of asymmetry on TCP performance. *ACM Baltzer Mobile Networks and Applications (MONET)* 1999; **4**(3): 219–241.
3. Pilosof S, Ramjee R, Raz D, Shavitt Y, Sinha P. Understanding TCP fairness over wireless LAN, In *Proc. IEEE Infocom '03*, San Francisco, California, USA, 2003; 863–872.
4. Keceli F, Inan I, Ayanoglu E. Fair and efficient TCP access in the IEEE 802.11 infrastructure basic service set. *ArXiV cs.IT/0806.1089*, June 2008. arxiv.org.
5. Vaidya NH, Bahl P, Gupta S. Distributed fair scheduling in a wireless LAN, In *Proc. ACM Mobicom '00*, Boston, Massachusetts, USA, 2000; 167–178.
6. Nandagopal T, Kim T, Gao X, Bharghavan V. Achieving MAC layer fairness in wireless packet networks, In *Proc. ACM Mobicom '00*, Boston, Massachusetts, USA, 2000; 87–98.
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, 2005.
8. Casetti C, Chiasserini CF. Improving fairness and throughput for voice traffic in 802.11e EDCA, In *Proc. IEEE PIMRC '04*, Barcelona, Spain, 2004; 525–530.
9. Leith DJ, Clifford P, Malone D, Ng A. TCP fairness in 802.11e WLANs. *IEEE Communications Letters* 2005; **9**(11): 964–966.
10. Freitag J, da Fonseca NLS, de Rezende JF. Tuning of 802.11e network parameters. *IEEE Communications Letters* 2006; **10**(8): 611–613.
11. Tinnirello I, Choi S. Efficiency analysis of burst transmissions with block ACK in contention-based 802.11e WLANs, In *Proc. IEEE ICC '05*, Seoul, Korea, 2005; 3455–3460.
12. Keceli F, Inan I, Ayanoglu E. Weighted fair uplink/downlink access provisioning in IEEE 802.11e WLANs, In *IEEE ICC '08*, Beijing, China, 2008; 2473–2479.
13. Kim SW, Kim B-S, Fang Y. Downlink and uplink resource allocation in IEEE 802.11 wireless LANs. *IEEE Transactions on Vehicular Technology* 2005; **54**(1): 320–327.

14. Jeong J, Choi S, Kim CK. Achieving weighted fairness between uplink and downlink in IEEE 802.11 DCF-based WLANs, In *Proc. IEEE QSHINE '05*, Greater Noida, India, 2005; 1–10.
15. Keceli F, Inan I, Ayanoglu E. Fair and efficient TCP access in IEEE 802.11 WLANs, In *IEEE WCNC '08*, Las Vegas, Nevada, USA, 2008; 1745–1750.
16. Wu Y, Niu Z, Zheng J. Study of the TCP upstream/downstream unfairness issue with per-flow queueing over infrastructure-mode WLANs. *Wireless Communications and Mobile Computing* 2005; **5**(4): 459–471.
17. Ha J, Choi CH. TCP fairness for uplink and downlink flows in WLANs, In *Proc. IEEE Globecom '06*, San Francisco, California, USA, 2006; 1–5.
18. Blefari-Melazzi N, Detti A, Habib I, Ordine A, Salsano S. TCP Fairness issues in IEEE 802.11 networks: problem analysis and solutions based on rate control. *IEEE Transactions on Wireless Communications* 2007; **6**(4): 1346–1355.
19. Urvoy-Keller G, Beylot AL. Improving flow-level fairness and interactivity in WLANs using size-based scheduling policies. *Technical Report*. *RR-07-206*, Institut Eurocom, Sophia-Antipolis, 2007.
20. Keceli F, Inan I, Ayanoglu E. 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*, Glasgow, Scotland, 2007; 4512–4517.
21. Keceli F, Inan I, Ayanoglu E. Achieving fair TCP access in the IEEE 802.11 infrastructure basic service set, In *IEEE ICC '08*, Beijing, China, 2008; 2637–2643.
22. Lin X, Chang X, Muppala J. VQ-RED: an efficient virtual queue management approach to improve fairness in infrastructure WLANs, In *Int. IEEE Workshop on WLNs*, Sydney, Australia, 2005; 632–638.
23. Wu Q, Gong M, Williamson C. TCP fairness issues in IEEE 802.11 wireless LANs. *Computer Communications* 2008; **31**(10): 2150–2161.
24. RFC3449—TCP performance implications of network path asymmetry, 2002.
25. RFC2760—ongoing TCP research related to satellites, 2000.
26. Balakrishnan H, Padmanabhan VN, Katz RH. The effects of asymmetry on TCP performance, In *Proc. ACM/IEEE MobiCom '97*, Budapest, Hungary, 1997; 77–89.
27. Lakshman TV, Madhow U, Suter B. Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance, In *Proc. IEEE Infocom '97*, Kobe, Japan, 1997; 1199–1209.
28. Fairhurst G, Samaraweera N, Sooriyabandara M, Harun H, Hodson K, Donadio R. Performance issues in asymmetric TCP service provision using broadband satellite. *IEE Proceedings - Communications* April 2001; **148**: 95–99.
29. Allman M. On the generation and use of TCP acknowledgements. *ACM Computer Communications Review* 1998; **28**: 4–21.
30. Kalampouskas L, Varma A, Ramakrishnan KK. Improving TCP throughput over two-way asymmetric links: analysis and solutions, In *Proc. ACM Sigmetrics '98*, Madison, Wisconsin, USA, 1998; 78–89.
31. Barakat C, Altman E. On ACK filtering on a slow reverse channel, In *Proc. Int. Workshop on Quality of Future Internet Services*, Berlin, Germany, 2000; 80–92.
32. Zhang L, Shenker S, Clark DD. Observations on the dyanmics of a congestion control algorithm: the effects of two-way traffic. *ACM Computer Communications Review* 1991; **21**: 133–147.
33. Ming-Chit I, Jinsong D, Wang W. Improving TCP performance over asymmetric networks. *ACM Computer Communications Review* 2000; **30**(3): 45–54.
34. Eggert L, Heidemann J, Touch J. Effects of ensemble-TCP. *ACM Computer Communications Review* 2000; **30**(1): 15–29.
35. Wu Q, Williamson C. Improving ensemble-TCP performance on asymmetric networks, In *Proc. Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Cincinnati, Ohio, USA, 2001; 205–214.
36. Heusse M, Merritt SA, Brown TX, Duda A. Two-way TCP connections: old problem, new insight. *ACM Computer Communications Review* 2011; **41**(2): 5–15.
37. Medepalli K, Tobagi FA. Throughput analysis of IEEE 802.11 wireless LANs using an average cycle time approach, In *Proc. IEEE Globecom '05*, St. Louis, Missouri, USA, 2005; 3007–3011.
38. Inan I, Keceli F, Ayanoglu E. Performance analysis of the IEEE 802.11e enhanced distributed coordination function using cycle time approach, In *Proc. IEEE Globecom '07*, Washington, DC, USA, 2007; 2552–2557.
39. The Network Simulator, ns-2. http://www.isi.edu/nsnam/ns.
40. 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, 2003.
41. Jain R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons: New York, New York, USA, 1991.

42. Qiao D, Choi S. Goodput enhancement of IEEE 802.11a wireless LAN via link adaptation, In *Proc. IEEE ICC '01*, Helsinki, Finland, 2001; 1995–2000.
43. Lacage M. Ns-2 802.11 support, INRIA Sophia Antipolis, France, 2006. http://spoutnik.inria.fr/code/ns-2.

## AUTHORS' BIOGRAPHIES

**Feyza Keceli** received her BS degree from the Middle East Technical University, Ankara, Turkey, in June 2001, and her MS degree from Bilkent University, Ankara, Turkey, in September 2003, and her PhD degree from the University of California, Irvine, U.S.A. in September 2008, all in Electrical Engineering. She was with Conexant Systems Inc. from June 2006 to December 2006, where she worked on the design of MAC algorithms for enhancing Bluetooth/WLAN coexistence performance. She was with Telecommunications Systems Inc. (formerly Networks in Motion) from September 2008 to May 2012, where she worked on the design and development of Location-Based Services solutions for GPS-enabled mobile systems. Her current research interests include analytical network modeling and simulation, MAC and transport layer protocol design, and fair access and QoS provisioning in wireless networks.



**Inanc Inan** received his BS degree from the Middle East Technical University, Ankara, Turkey, in June 2001, his MS degree from Bilkent University, Ankara, Turkey, in September 2003, and his PhD degree from the University of California, Irvine, U.S.A. in September 2007, all in Electrical Engineering. From July 2007 to September 2010, he worked as a systems engineer at Wionics Research—Realtek Group, concentrating mainly on MAC, link, and transport layer protocol research and development for UWB wireless networks. Since September 2010, he has been the engineering team leader at Starix Technology, providing single- and multi-hop/mesh wireless networking solutions in the medical and defense markets. His current research interests include analytical network modeling and simulation, QoS provisioning, fair and efficient resource allocation, and protocol and algorithm design for wireless networks.

**Ender Ayanoglu** (S'82-M'85-SM'90-F'98) received his BS degree from the Middle East Technical University, Ankara, Turkey, in 1980, and his MS and PhD degrees from Stanford University, Stanford, CA, U.S.A. in 1982 and 1986, respectively, all in Electrical Engineering. He was with the Communications Systems Research Laboratory of AT&T Bell Laboratories (Bell Labs, Lucent Technologies after 1996) until 1999 and was with Cisco Systems until 2002. Since 2002, he has been a professor at the Department of Electrical Engineering and Computer Science, the Henry Samueli School of Engineering, University of California, Irvine, U.S.A. From 2002 to 2010, he served as the Director of the Center for Pervasive Communications and Computing and held the Conexant–Broadcom Endowed Chair. Ayanoglu is the recipient of the IEEE Communications Society Stephen O. Rice Prize Paper Award in 1995 and the IEEE Communications Society Best Tutorial Paper Award in 1997. From 1993 to 2008, he was as an editor of the IEEE Transactions on Communications, and from 2004 to 2008 served as its Editor-in-Chief. He served on the Executive Committee of the IEEE Communications Society Communication Theory Committee from 1990 until 2002 and from 1999 to 2002 was its Chair.