Computer Networks 53 (2009) 2855–2869

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Modeling finite buffer effects on TCP traffic over an IEEE 802.11 infrastructure WLAN ☆

Onkar Bhardwaj [a], G.V.V. Sharma [b], Manoj Panda [a], Anurag Kumar [a,*]

[a] Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India
[b] Department of Electrical Engineering, Indian Institute of Technology, Bombay, Powai, Mumbai 400076, India

ARTICLE INFO

Article history:
Received 9 April 2009
Accepted 14 July 2009
Available online 26 July 2009
Responsible Editor: N.B. Shroff

Keywords:
TCP over IEEE 802.11 WLANs
TCP unfairness in WLANs
TCP modeling

ABSTRACT

The network scenario is that of an infrastructure IEEE 802.11 WLAN with a single AP with which several stations (STAs) are associated. The AP has a finite size buffer for storing packets. In this scenario, we consider TCP-controlled upload and download file transfers between the STAs and a server on the wireline LAN (e.g., 100 Mbps Ethernet) to which the AP is connected. In such a situation, it is well known that because of packet losses due to finite buffers at the AP, upload file transfers obtain larger throughputs than download transfers. We provide an analytical model for estimating the upload and download throughputs as a function of the buffer size at the AP. We provide models for the undelayed and delayed ACK cases for a TCP that performs loss recovery only by timeout, and also for TCP Reno. The models are validated in comparison with NS2 simulations.

1. Introduction

We consider a scenario in which several clients or stations (STAs) are associated with a single Access Point (AP). The AP has a finite amount of FIFO buffer. For simplicity, we consider associations only at a single Physical (PHY) rate.[1] We are concerned with TCP-controlled file transfer throughputs when each STA is either downloading or uploading a *single* large file via the AP. The other endpoint of the transfers, or the "server," is located on the high-speed Ethernet connected to the AP.[2] For such a situation, it has been reported that, with finite buffers at the AP, there is unfairness between the upload and download transfers with the upload transfers obtaining larger throughputs [1,2]. Our objective is to provide analytical models that explain this unfairness, thus providing quantitative insights into the unfairness, and also predictive models for network engineering.

Relation to the Literature: Bruno et al. [3] analyzed the scenario of upload and download TCP-controlled file transfers in a single cell infrastructure WLAN when there is no packet loss at the AP. They assumed equal TCP windows for all connections, that the TCP receivers use undelayed ACKs, and showed that the total TCP throughput is independent of the number of STAs in the system; further the upload and download transfers each obtain an equal share of the aggregate throughput. A variation of this approach for modeling TCP transfers, along with a fixed-point analysis of 802.11e EDCA, was employed by Sri Harsha et al. [4] to provide a combined analytical model for TCP transfers, CBR packet voice, and streaming video over an infrastructure WLAN. The delayed ACK case was analyzed by Kuriakose et al. [5].

It is known that, if there is packet loss at the AP due to finite buffers, then in a situation of simultaneous upload

☆ This is an expanded version of a paper that appeared in COMSNETS 2009 (The First International Conference on COMmunication Systems and NETworkS), Bangalore, India, January 5–10, 2009. The research on which this paper is based was supported by the Department of Information Technology, Government of India, and Airtight Networks, Pune, India.

* Corresponding author. Tel.: +91 80 2360 0855; fax: +91 80 2360 0991.
E-mail address: anurag@ece.iisc.ernet.in (A. Kumar).

[1] In IEEE 802.11b, for example, the PHY rates 11 Mbps, 5.5 Mbps and 2 Mbps are available. In general, STAs can associate with the AP at different PHY rates depending on the channel conditions. However, we assume that all STAs are associated with the AP at the same PHY rate.

[2] The situation in which the server is located across a wide-area network will be addressed in the future.

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.07.009

and download transfers, the upload transfers each obtain a larger throughput than any of the download transfers. Gong et al. [1] provide simulation results validating this fact. They also show that as the AP's buffer size increases, thus reducing packet loss at the AP, the throughput unfairness reduces. They also propose queue management strategies to alleviate the throughput unfairness. Pilosof et al. [2] analyzed the same problem of unfairness by assuming an M/M/1/K model for the finite buffer at the AP.

In this paper, we do not assume any conventional queueing model for the AP buffer, but we combine the earlier models for TCP-controlled file transfers in WLANs (i.e., [3,5]) with a detailed model of TCP window evolution under tail-drop loss at the AP. A detailed modeling of TCP window evolution quantifying the unfair division of throughputs among the upload and download transfers providing valuable insights is the main contribution of this paper.

Outline of the Paper: The analytical model comprises two steps. In the first step (Sections 2 and 3), we use a simple extension of the analytical model of [3] to obtain the upload and download throughputs for a given value of h, the fraction of contention cycles in which the AP contends with a download packet (i.e., a TCP data packet) at the head-of-the-line (HOL) of its FIFO buffer. In the second step (Section 4), we obtain the value of h as a function of AP buffer size, using a detailed study of TCP window evolution when the upload connections have a maximum window limit but the download connections have no such window limit. We do this for both the undelayed and delayed ACK cases for the TCP version in which all loss recovery is by timeouts. We also provide a bound on h for the case when all the TCP connections have a maximum window limit. Simulation results that validate our analyses are provided in Section 6.

2. Throughputs: undelayed ACK

Consider an infrastructure mode WLAN with $N(= N_d + N_u)$ STAs associated with the AP at the same PHY rate. Among these STAs, N_d STAs each have a single download TCP connection while each of the remaining N_u STAs have a single upload connection (see Fig. 1). All file transfers are to or from a "server" on the high-speed LAN to which the AP is connected. We analyze the case where TCP ACK transmissions on the WLAN use the "Basic Access" mode whereas TCP data transmissions use the "RTS–CTS" mode. This is reasonable since TCP ACKs are small (40 Bytes), but TCP data packets are much larger (typically, 1 KByte). We remark, however, that other alternatives can also be easily analyzed within our framework.

2.1. Observations and modeling approximations

(1) We assume that there are no packet losses because of wireless channel errors; such packet losses can be accounted for by extending our analysis. We also do not model *packet capture*, i.e., simultaneous attempts on the medium are assumed to result in a collision. Also, with the standard DCF parameters, packet drops due to the retransmission threshold being exceeded in the DCF CSMA/CA MAC are known to be rare, and hence, are ignored in our model.

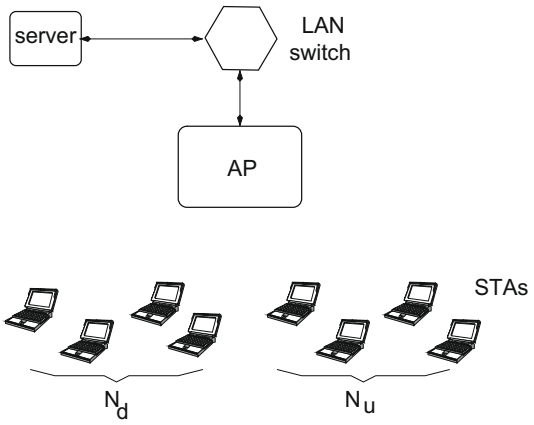

Fig. 1. The network scenario, comprising several STAs associated with an AP, each uploading (or downloading) a large file to (or from) a server attached to the high-speed wired LAN.

(2) It is now well known (see [3] or [5]) that when carrying TCP-controlled transfers the AP is the bottleneck and always contends for the channel. This is understood as follows. Considering the undelayed ACK case, for one packet sent by each of the STAs, N packets will need to be transmitted by the AP. Since DCF is packet fair, this situation is sustainable only if a very small number of the STAs contends at any time so that, on the average, half the packets transmitted are from the AP and the other half from the STAs. Recalling that we are dealing with the local area network case (so that the number of packets "in flight" outside the WLAN can be ignored) it follows that most of the packets in the TCP windows of the connections reside in the AP's queue.

(3) Furthermore, as in [5] (for the undelayed ACK case) we use the approximation that for large N, an STA can have at most one packet in its queue, with every successful transmission from the AP resulting in the generation of a packet at a previously empty STA, thus activating a new STA. A TCP data packet (resp., TCP ACK) transmitted by the AP results in a TCP ACK (resp., TCP data packet) being generated at a previously empty download (resp. upload) STA.

Fig. 2 depicts the model described above. Note that, since the success or failure of a transmission due to CSMA/CA contention does not depend on the length of the packet to be transmitted, Assumptions 2 and 3 above do not depend on the packet lengths and the PHY rates. Also, Assumptions 2 and 3 do not depend on the specific values of N_u and N_d as long as N is large.

2.2. The process (D_k, U_k) and its analysis

We now develop the stochastic analysis of the number of contending STAs along lines similar to [3] or [5]. Referring to Fig. 3, let $G_k, k = 0, 1, 2, \ldots$, denote the instants

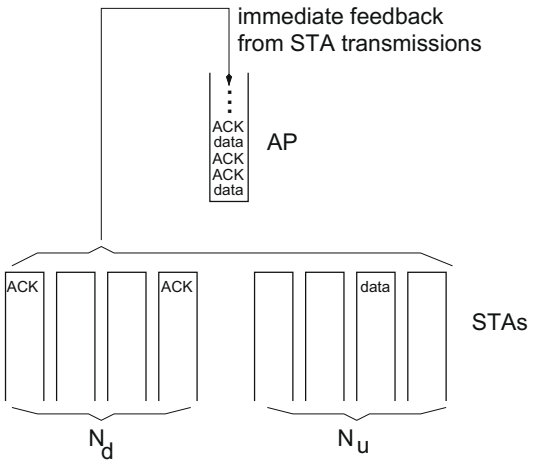

Fig. 2. Schematic based on the modeling assumptions discussed in the text. Since the server is attached to the same high speed LAN as the AP, we assume immediate "feedback" from the server, due to packets transmitted by the STAs.

when a successful transmission ends. According to our assumptions above, the AP always have packets in its queue, and hence, always contends. Consider the instant G_k. If the just completed successful transmission is from the AP then at G_k the number of contending STAs increases by one (by Assumption 3 in Section 2.1). If the HOL packet at the AP is a TCP data packet (resp., TCP ACK) then one more download (resp., upload) STA begins to contend with a TCP ACK (resp., a TCP data packet). Between G_k and the next success instant G_{k+1} there is no change in the number of contending STAs. At G_k, let D_k denote the number of downloading STAs that are nonempty (i.e., have a TCP ACK to send), and let U_k denote the number of uploading STAs that are nonempty (i.e., have a TCP data packet to send). Since there are no external arrivals, the process $\{(D_k, U_k), k \geqslant 0\}$ can only change state at the instants $G_k, k \geqslant 0$.

We call the time interval $[G_k, G_{k+1})$ between two consecutive success end instants a *contention cycle*. Each contention cycle consists of several *channel slots*. A channel slot is the time interval between two consecutive attempt opportunities on the channel. Thus, transmission attempts can occur only at channel slot boundaries. A channel slot is an idle back-off slot, or a successful transmission, or a collision if 0, or 1, or more than one node(s) transmit(s) in the beginning of the channel slot. Clearly, every contention cycle consists of several idle and collision channel slots and terminates with a success channel slot.

To model the way the DCF CSMA/CA serves packets from the queues, we assume that when m nodes are contending, either with TCP data packets or with TCP ACKs, each node contends with a probability β_m, where β_m is the steady state attempt probability when there are m *saturated* nodes contending (see [5]). Note that, β_m can be easily obtained from the analysis provided in [6] or [7]. Also note that, β_m includes the effect of all DCF parameters, such as the back-off windows, and the retransmission threshold. Thus, according to this model, if the state of the process (D_k, U_k) is (d, u), then, until the next success, each of the $1 + d + u$ contending nodes attempts with probability $\beta_{(1+d+u)}$. The 1 arises from the assumption that the AP always contends.

Define the process $Z_k \in \{0, 1\}$, embedded at the instants $G_k, k \geqslant 0$, by $Z_k = 1$ if the HOL packet at the AP at time G_{k+} is a TCP data packet (i.e., in the interval $[G_k, G_{k+1})$ the AP contends to transmit a TCP data packet), and by $Z_k = 0$ otherwise. Now define

$$h = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} Z_k, \tag{1}$$

i.e., h denotes the fraction of contention cycles in which the HOL packet at the AP is a TCP data packet.

Some observations about h: Let $\mathscr{A}_n$ (resp., $\mathscr{D}_n$) denote the number of download data packets that arrive at (resp., depart from) the AP's HOL position in the time interval $[0, G_n)$. Let V_j denote the *number* of contention cycles for which the j^{th} download data packet occupies the HOL position at the AP buffer. Then we can write

$$\frac{1}{n} \sum_{j=1}^{\mathscr{D}_n} V_j \leqslant \frac{1}{n} \sum_{k=0}^{n-1} Z_k \leqslant \frac{1}{n} \sum_{j=1}^{\mathscr{A}_n} V_j \tag{2}$$

from which it can easily be shown that

$$h = \lambda^{(d)} \mathrm{E}(V), \tag{3}$$

where

$$\lambda^{(d)} = \text{rate of download packets transmitted by the AP per contention cycle} = \lim_{n\to\infty} \frac{\mathscr{A}_n}{n} = \lim_{n\to\infty} \frac{\mathscr{D}_n}{n}, \tag{4}$$

and

$$\mathrm{E}(V) = \text{Average number of contention cycles taken to successfully transmit an HOL packet at the AP} = \lim_{n\to\infty} \frac{1}{\mathscr{D}_n} \sum_{j=1}^{\mathscr{D}_n} V_j = \lim_{n\to\infty} \frac{1}{\mathscr{A}_n} \sum_{j=1}^{\mathscr{A}_n} V_j. \tag{5}$$

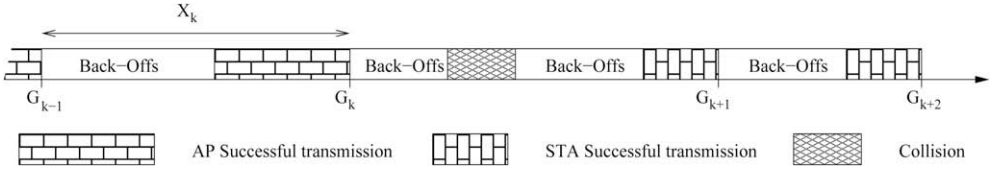

Fig. 3. Evolution of channel activity: Shown are the random time instants G_k at which successful transmissions end. X_k denotes the (random) duration of the kth contention cycle $[G_{k-1}, G_k)$. Each contention cycle consists of one or more back-off period(s) and collision(s). Each back-off period consists of one or more back-off slot(s). Each contention cycle terminates with a successful transmission.

Similarly, as the expected number of contention cycles required to successfully transmit an HOL packet from the AP is independent of it being a TCP data packet or a TCP ACK packet, we can also write

$$1 - h = \lambda^{(u)}\mathsf{E}(V), \tag{6}$$

where $\lambda^{(u)}$ is the rate of TCP ACK departures from the AP per channel slot. From Eqs. (3) and (6) we get

$$h = \frac{\lambda^{(d)}}{\lambda^{(d)} + \lambda^{(u)}}. \tag{7}$$

Thus, as expected, h is also the ratio of the download throughput from the AP to the total throughput from the AP.

Remark 2.1. In the following analysis we will use both of *the two meanings of h:* (i) as defined by (1), i.e., the fraction of contention cycles in which the AP contends with a data packet at its HOL position, and (ii) the fraction of AP services that are data packets (a meaning provided by (7)).

Analyzing the process (D_k, U_k)*:* Based on (7), we assume that after the HOL packet at the AP is transmitted, the next HOL packet is a data packet with probability h (an ACK packet with probability $1 - h$), independent of anything else. We note that this is an approximation; a more detailed model will require us to keep track of the entire queue of packets in the AP. With this modeling assumption about the HOL packet at the AP, and from the contention model introduced earlier in this section, it can be seen that (D_k, U_k) is a discrete time Markov chain (DTMC), embedded at the instants G_k, taking values in $\{(d,u) : d = 0,1,2,\ldots;\ u = 0,1,2,\ldots\}$. Note that, in reality, we have, $0 \leqslant d \leqslant N_d$ and $0 \leqslant u \leqslant N_u$. However, our modeling assumption that "every successful transmission by the AP generates a packet at a previously empty STA" yields the state space $\{(d,u) : d = 0,1,2,\ldots;\ u = 0,1,2,\ldots\}$.

The DTMC (D_k, U_k) has the following transition probabilities:

$$\begin{aligned}&\Pr((d,u-1)/(d,u))\\ &= \Pr(\text{An upload STA wins the contention}) = \frac{u}{u+d+1}\end{aligned} \tag{8}$$

$$\begin{aligned}&\Pr((d-1,u)/(d,u))\\ &= \Pr(\text{A download STA wins the contention}) = \frac{d}{u+d+1}\end{aligned} \tag{9}$$

$$\begin{aligned}&\Pr((d,u+1)/(d,u))\\ &= \Pr(\text{The AP succeeds and transmits to an upload STA})\\ &= \frac{1-h}{u+d+1}\end{aligned} \tag{10}$$

$$\begin{aligned}&\Pr((d+1,u)/(d,u))\\ &= \Pr(\text{The AP succeeds and transmits to a download STA})\\ &= \frac{h}{u+d+1}\end{aligned} \tag{11}$$

The transition probability structure of the DTMC (D_k, U_k) is shown in Fig. 4. Let $\pi(d,u),\ d = 0,1,2,\ldots,\ u = 0,1,2,\ldots$ denote its stationary distribution. It can be shown that the DTMC (D_k, U_k) satisfies the Kolmogorov's criterion for *reversibility* [8]. Using reversibility, the explicit closed-form expression for $\pi(d,u)$ can be shown to be (see Appendix A.1)

$$\pi(d,u) = \frac{u+d+1}{2e} \times \frac{h^d(1-h)^u}{d!u!}. \tag{12}$$

Write the stationary marginal of the random process (D_k, U_k) by (D, U). Since the number of download STAs that are contending changes only at the instants $G_k, k \geqslant 0$, we observe that the time average number of contending download and upload STAs is given by (see Appendix A.2)

$$\mathsf{E}(D) = \frac{3h}{2}, \tag{13}$$

$$\mathsf{E}(U) = \frac{3(1-h)}{2}. \tag{14}$$

Thus, the mean number of contending STAs is $\frac{3}{2}$.

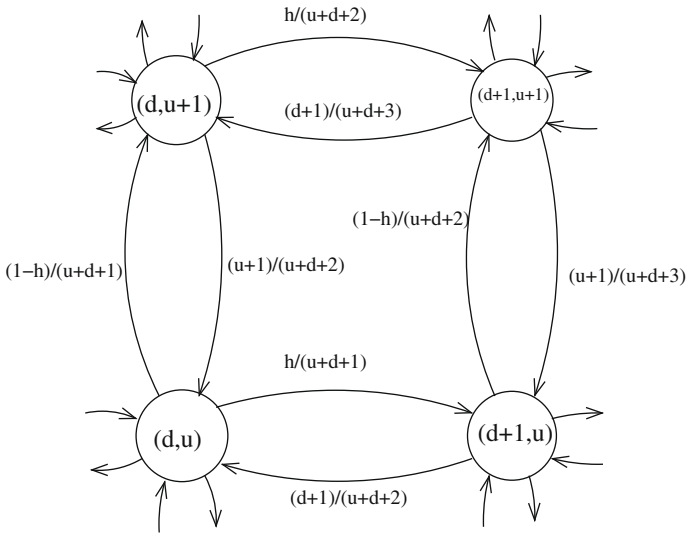

Fig. 4. Transition probability diagram of the process (D_k, U_k) for the case of undelayed ACKs.

2.3. Throughput analysis

With reference to Fig. 3, let $X_k := G_k - G_{k-1}$, i.e., X_k is the random duration of the kth contention cycle. The process $\{((D_k, U_k), G_k), k \geqslant 0\}$, is a Markov renewal process with cycle lengths $\{X_k, k \geqslant 1\}$. We can use Markov regenerative analysis [9] to obtain performance measures such as the throughput Θ of the AP. For $t \geqslant 0$, let $H(t)$ denote the total number of AP successes in $[0, t]$. Let the number of successful transmissions made by the AP in the interval $(G_{k-1}, G_k]$ (i.e., the kth contention cycle) be denoted by $H_k \in \{0, 1\}$. Recall that, each contention cycle contains a successful transmission. A successful transmission may belong either to the AP (i.e., $H_k = 1$) or to one of the STAs (i.e., $H_k = 0$). We view H_k as a "reward" associated with the kth cycle. Then by Markov regenerative analysis [9] we conclude that

$$\Theta := \lim_{t\to\infty} \frac{H(t)}{t} \overset{w.p.\ 1}{=} \frac{\mathsf{E}H}{\mathsf{E}X} = \frac{\sum_{(d,u)} \pi(d,u) \frac{1}{u+d+1}}{\sum_{(d,u)} \pi(d,u) \mathsf{E}_{(d,u)} X}, \tag{15}$$

where $\mathsf{E}H$ denotes the expected reward per cycle, $\mathsf{E}X$ denotes the expected duration of a cycle, $\sum_{(d,u)}$ represents a double sum over all $d \geqslant 0$, $u \geqslant 0$, and $\mathsf{E}_{(d,u)}X$ denotes the expected duration of a cycle starting in the state (d, u). The numerator of (15) is also equal to the probability that the AP succeeds in a randomly chosen cycle. Using the expression for $\pi(d, u)$ given by (12), we obtain

$$\begin{aligned} \mathsf{E}H &= \sum_{(d,u)} \pi(d,u) \left(\frac{1}{u+d+1}\right) \\ &= \sum_{(d,u)} \frac{(u+d+1)}{2e} \times \frac{h^d (1-h)^u}{u!d!} \left(\frac{1}{u+d+1}\right) \\ &= \frac{1}{2e} \sum_{(d,u)} \frac{h^d (1-h)^u}{u!d!} = \frac{1}{2e} e^h e^{(1-h)} = \frac{1}{2}. \end{aligned} \tag{16}$$

This is expected for TCP transfers with undelayed ACKs, as the AP must transmit half the number of total packet transmissions. Derivation of $\mathsf{E}_{(d,u)}X$ in terms of packet lengths, PHY rates and the attempt probabilities can be found in Appendix C. Substituting the value of $\mathsf{E}_{(d,u)}X$ and the expression for $\pi(d, u)$ (from (12)) into (15), we obtain the throughput Θ of the AP in packets/second. Recalling (7), the total throughput Θ_d (resp. Θ_u) for downloading (resp. uploading) STAs can then be obtained as

$$\Theta_d = h\Theta; \quad \Theta_u = (1-h)\Theta, \tag{17}$$

both in packets per second. Multiplication by the user payload in each packet yields the throughput in bytes per second.

3. Throughputs: delayed ACK

In the case of upload traffic with delayed ACKs, in steady state, every TCP ACK from the AP will generate two data packets at the STA. Thus, our earlier approximation that there can be at most one packet in the STA queue is no longer valid. However, assuming validity of the assumption that the transmission from the AP is always to an empty STA, we provide a simple upper bound on the throughput by assuming that whenever an STA wins the contention for the channel, it transmits both the TCP data packets in its queue. Thus, a successful STA does not have to contend again for the second packet, thus reducing the contention time and increasing the throughput to provide an upper bound.

For downloading STAs, ACKs are generated by the STAs for alternate packets that they receive. We model this probabilistically, as in [5]; when the AP transmits a data packet to a downloading STA, an ACK is generated at that STA with probability $\frac{1}{2}$. With h defined as before, the process (D_k, U_k) is a DTMC with the transition probability structure shown in Fig. 5, and detailed below.

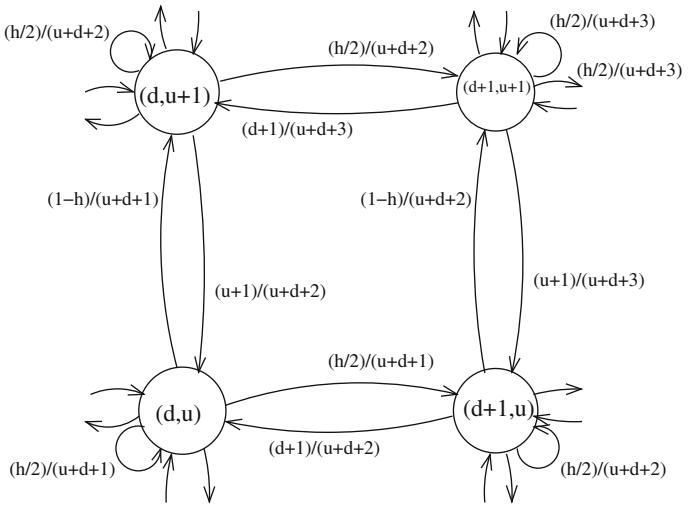

Fig. 5. Transition probability diagram for the process (D_k, U_k) for the case of delayed ACKs. Notice that the horizontal transitions from left to right are halved as compared to the undelayed ACK case. Also notice the self-loops.

$$\Pr((d,u-1)/(d,u)) = \Pr(\text{An upload STA wins the contention}) = \frac{u}{u+d+1} \tag{18}$$

$$\Pr((d-1,u)/(d,u)) = \Pr(\text{A download STA wins the contention}) = \frac{d}{u+d+1} \tag{19}$$

$$\Pr((d+1,u)/(d,u)) = \Pr(\text{AP succeeds and transmits an even numbered TCP data packet to a download STA}) = \frac{h/2}{u+d+1} \tag{20}$$

$$\Pr((d,u)/(d,u)) = \Pr(\text{AP succeeds and transmits an odd numbered TCP data packet to a download STA}) = \frac{h/2}{u+d+1} \tag{21}$$

$$\Pr((d,u+1)/(d,u)) = \Pr(\text{AP succeeds and transmits to an upload STA}) = \frac{1-h}{u+d+1} \tag{22}$$

The difference from Eqs. (8)–(11) is in the numerator for the expression $P((d,u+1)/(d,u))$. This captures the above stated probabilistic model of the fact that a TCP ACK is generated at a downloading STA only on the receipt of two data packets from the AP. The stationary distribution of this DTMC is given by (see Appendix B)

$$\pi(d,u) = \frac{u+d+1}{e^{1-\left(\frac{h}{2}\right)}\left(2-\frac{h}{2}\right)} \times \frac{\left(\frac{h}{2}\right)^d (1-h)^u}{d!u!}. \tag{23}$$

The rest of the analysis follows along the same lines as in Section 2. The aggregate AP throughput in packet per second is again given by (15). The total throughput for the downloading and uploading STAs is given by

$$\Theta_d = h\Theta \quad \text{and} \quad \Theta_u = 2(1-h)\Theta. \tag{24}$$

The factor 2 arises because each ACK transmitted from the AP acknowledges 2 data packets. The value of $\mathsf{E}_{(d,u)}X$ (recall the meaning of $\mathsf{E}_{(d,u)}X$ provided just after (15)) is obtained in a manner similar to the undelayed ACK case.

4. Determining *h*: TCP window analysis

In this section we obtain expressions for h for both delayed and undelayed ACK cases when the AP has a finite buffer, making suitable approximations in the process. Note that, as already stated earlier, our analysis is valid for the scenario when the STAs have TCP connections with a server located on the Ethernet to which the AP is connected, i.e., the delay between the server and the AP is negligible. In this section, the version of TCP analyzed does not support fast retransmit and fast recovery; loss recovery is by timeout only, and the window is reset to 1 after a timeout. We call this version "OldTahoe." TCP Reno is analyzed in the next section.

4.1. The case of undelayed ACKs

4.1.1. Modeling assumptions and approximations

(a) From (13) and (14) it can be seen that the average number of active STAs is 3/2. Hence, for a large number of STAs, and for sufficiently large upload and download connection windows, it can be assumed that most of the TCP packets (data or ACKs) reside in the AP (as will be seen later, even 5 STAs, with a maximum TCP window of 20 packets suffices to make this approximation accurate). Here we are also using the fact that the remote end-point is on the LAN.

(b) Assume that the maximum congestion windows for upload connections is W_{max}. Since TCP ACKs are just 40 bytes, their loss probability is small; also, due to the *cumulative acknowledgement* property of TCP ACKs, infrequent ACK losses do not result in the TCP window being reduced. Hence, we assume that the TCP congestion windows of the upload connections grow and stay at W_{max}. Define $\mu = N_u W_{max}$, i.e., the total number of packets belonging to the uploading STAs in the system. Since most of these packets reside in the AP, we assume that the AP buffer always contains $\mu L_{TCP-ACK}$ bytes of TCP ACKs for the upload connections.

(c) Thus, if the AP buffer size is $\mathscr{B}$ bytes then the buffer available for the download connections can be assumed to be $\mathscr{B} - \mu L_{TCP-ACK}$. In terms of packets, the number of download data packets that can be accommodated in the AP is given by

$$b = \frac{\mathscr{B} - \mu L_{TCP-ACK}}{L_{TCP-DATA}}. \tag{25}$$

We denote the capacity of the buffer in terms of TCP data packets (for downloading STAs) by B which is given by

$$B = \mu + b. \tag{26}$$

Also, for simplicity in some calculations to be shown later, we assume that, for $i \in \{2,4,6,\ldots\}$,

$$b = i \times N_d. \tag{27}$$

(d) In order to analyze the evolution of the TCP window while accounting for tail-drop loss in the AP buffer, we now propose a simple model for the AP buffer and the service applied to it; see Fig. 6. Since the

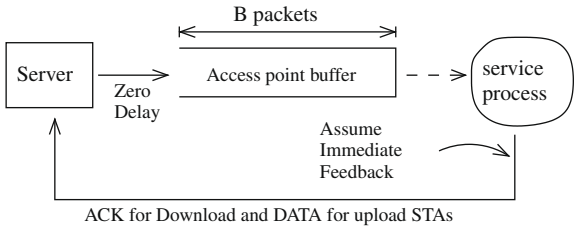

Fig. 6. Model for analyzing the AP buffer in order to obtain h.

number of active STAs is small, we ignore the round trip time between a packet being served at the AP and the corresponding packet (e.g., data packet for an ACK, and an ACK for a data packet) being received back at the AP. Thus we assume *immediate feedback*.

(e) With the model in Fig. 6, let us consider the process of services to the AP buffer when it just becomes full with $\mu + b$ packets, without any of the connections having suffered a tail-drop loss. Now all the download TCP connections will lose packets in the process of serving these $\mu + b$ packets from the AP. To see this, suppose that the TCP window of download connection i is W_i when the AP buffer just becomes full; by our assumption, these W_i packets will all be in the AP. Hence we can write

$$b = \sum_{i=1}^{N_d} W_i. \tag{28}$$

Among these packets, there will be (at least) one packet that will cause the window to grow. As soon as this packet is served from the AP, two packets will arrive at the tail of the AP buffer; one will be accommodated and one will be lost. Hence, after the buffer becomes full, all the download TCP connections will lose packets in the process of serving $\mu + b$ packets from the AP. We call this the *loss phase*. In the loss phase, the TCP connections are assumed to be in congestion avoidance; this assumption is based on our extensive observation of download window evolution in simulations, a snapshot of which is given in Section 6.1.

(f) Suppose one of the download connections reach its maximum TCP window then that connection should not suffer loss. In simulations it was observed that such connections stay at the fixed window and do not suffer loss for large time intervals. The reason they ultimately do suffer a loss can be explained along the lines of the previous

(e) (Also see Section 6.1). Since modeling this phenomenon is complicated, we have assumed that the download connections do not have a maximum window limit. The upload connections do have a maximum window limit of W_{max}. Having no maximum window limit for a connection is possible in modern TCP implementations by means of enabling the window scaling option [10]. At the end of this subsection we also give a simple upper bound on h when download connections do have a maximum window limit.

(g) Note that if upload connections do not have a maximum window limit, then, as the loss of a few ACKs does not affect their window evolution, these windows will grow forever and the space for download connections will go on reducing. Hence, it is important to assume a maximum window limit on the upload connections.

4.1.2. The TCP window evolution process

If the AP buffer occupancy at an instant is x packets, then the interval during which these x packets are served will be called a *round*.

The loss phase: There is one round in this phase. We recall that the loss phase starts after the AP buffer just becomes full (with μ ACKs and b data packets). While serving all these packets, each download connection loses a packet (assuming the congestion avoidance phase). Thus, after this round, there will still be $\mu + b$ packets in the AP buffer. If download connection i had the window W_i at the time of losing a packet, it still has W_i packets in the AP buffer at the end of this round. Let us assume that among the W_i packets that began the loss phase, it was the last packet that caused the window to grow and, hence, caused the loss of a packet. This assumption will be supported later in this section, after explaining the synchronization process.

The reset phase: Round 1: In the beginning of this round there are $\mu + b$ packets in the AP buffer. Consider the first packet served from the AP buffer pertaining to connection i. When the TCP ACK corresponding to this packet goes to the server and returns as a TCP data packet to the AP, a *gap* is created in the sequence numbers of the packets corresponding to that connection. This gap rests between the remaining $W_i - 1$ data packets in the buffer and the newly arrived data packet. These remaining $W_i - 1$ packets will be served during the services of $\mu + b$ packets from the AP and will return as new data packets to the AP. Thus after this round, there will be again b packets in the AP buffer among which W_i packets will correspond to the ith download connection. Also, there is a gap in the sequence numbers before the very first packet of each download connection.

The reset phase: Round 2: The service of the very first packet from the ith download connection informs the receiver about the packet loss. The receiver returns a duplicate TCP ACK. In the TCP version we are analyzing, a timeout is needed for resetting the connection window. We assume that in reset phase round 2, during the services of the $\mu + b$ packets from the AP there will be timeout for all the download connections. This is based on the fact that after the lost packet was sent by the server, $2(\mu + b)$ are to be sent until the end of the reset phase round 2, the time taken for which suffices to cause a TCP sender timeout. Thus, during this round the windows of all the download connections will be reset to 1, with the slow start threshold for download connection i being set to $W_{th}^{(i)} = \frac{W_i}{2}$. Hence, after this round, there will be $\mu + N_d$ packets in the AP buffer with one packet out of the N_d packets belonging to each download connection.

The window evolution is synchronized: After the round 2 in the reset phase, all connection windows are reset to 1 and the ith download connection has a slow start threshold of $W_{th}^{i} = \frac{W_i}{2}$. Refering to (28), it can be shown that

$$\sum_{i=1}^{N_d} W_{th}^{(i)} = b/2. \tag{29}$$

For moderate values of b, it has been observed (see Section 6.1) that after some number of occurrences of the loss phase and the reset phase, there are very small differences between the slow start thresholds of download connections. Hence, all the connection windows become synchronized to the same values, the synchronization instants

being the ends of the *rounds* corresponding to the loss phase and the reset phase, and also the phases that will be discussed next. Henceforth, the analysis is carried out assuming all the window evolutions are synchronized.

The slow start phase: Denote by $T_{k,1}$ the instant when the kth reset phase ends. The windows of the download transfers are modeled as evolving in cycles that start at instants $\{T_{k,1}, k \geqslant 0\}$. At these instants the download windows are synchronized and have been set to 1. Thus, the AP buffer occupancy is $\mu + N_d$. All the download windows have the same slow start threshold and all are in the slow start phase. The first round in a cycle consists of serving all the packets in the AP; this results in there being $\mu + 2N_d$ packets in the AP buffer. Call this instant $T_{k,2}$. In the slow start phase, after each round all the download connection windows will be doubled. Thus, during the jth round in the slow start phase (corresponding to the time interval $[T_{k,j}, T_{k,j+1})$) the AP buffer occupancy increases from $\mu + 2^{j-1}N_d$ to $\mu + 2^j N_d$; $\mu + 2^{j-1}N_d$ packets are served from the AP during this interval. Since the download connection windows are assumed to be synchronized, and have the same slow start thresholds, they all enter the congestion avoidance phase at the instant $T_{k,r+1}$, where r is defined by

$$\mu + 2^r N_d = \mu + N_d W_{th} = \mu + b/2. \tag{30}$$

yielding

$$r = \log_2\left(\frac{b}{2N_d}\right). \tag{31}$$

By assuming window synchronization, all the download connection windows leave slow start phase at the end of the same round, and when the buffer is not yet full. This is consistent with the previously made assumption that the connections are in congestion avoidance phase at the time of buffer overflow.

The congestion avoidance phase: Following the previous discussion, the buffer occupancy at the beginning of this phase is $\mu + b/2$. The free space for download connections in the AP buffer is now $b/2$ packets. Due to the linear increase in the congestion avoidance phase, after the jth round in this phase, the buffer occupancy will increase from $\mu + b/2 + (j-1)N_d$ to $\mu + b/2 + jN_d$. At the end of the xth round, the buffer becomes full, where x is defined by $xN_d = b/2$, yielding $x = \frac{b}{2N_d}$, where x is an integer due to (27). After this round, the loss phase of this cycle begins. This AP buffer evolution has been summarized compactly in Table 1. Note that the number of upload packets transmitted by the AP is just μ in all the rounds.

Thus, in each cycle the window evolution and the number of packets served are deterministic. Hence, the ratio of download packets transmitted by the AP to the total packets transmitted by the AP is constant and can be calculated using Table 1. From (7), h is the same as the value of this fraction. Thus, h is given by

$$h = \frac{\left[(2^r - 1) + \frac{x(x-1)}{2} + 3x\right]N_d + (x+3)\frac{b}{2}}{(r+x+3)\mu + \left[(2^r - 1) + \frac{x(x-1)}{2} + 3x\right]N_d + (x+3)\frac{b}{2}}. \tag{32}$$

The value of h calculated using the above procedure can be substituted into the analysis in Section 2 to obtain the value of the AP throughput, the download throughput and the upload throughput for the undelayed ACK case.

A simple upper bound when all the connections have a maximum window limit: We so far assumed that the upload connections have a maximum window limit but the download connections have no such limit. When the download connections also have a maximum window limit, a simple upper bound can be obtained as follows. Let the download connections have a maximum window limit of W_{max}. The space available for download packets in the AP buffer is b packets. Assume that $\left\lfloor \frac{b}{W_{max}} \right\rfloor$ connections reach their maximum window limit and stay there indefinitely. Then assume that remaining $N_d - \left\lfloor \frac{b}{W_{max}} \right\rfloor$ download connections follow the window evolution process described above with $b - W_{max}\left\lfloor \frac{b}{W_{max}} \right\rfloor$ space available for their packets in the AP buffer. Calculating h with this model provides an upper bound on h.

4.2. The case of delayed ACKs

The above analysis for calculating h can be easily extended to the delayed ACK case; also recall the discussion in Section 3. We assume that every alternate TCP DATA packet gets acknowledged. Now it can be assumed that $B = \mu/2 + b$ as the AP stores one ACK corresponding to two DATA packets for the upload connections. There is another minor modification in the model from Section 4.1.1.

Table 1
AP buffer evolution for TCP OldTahoe with Undelayed ACKs.

Phase	Buffer at $T_{k,i}$	Buffer at $T_{k,i+1}$	Buffer services in $[T_{k,i}, T_{k,i+1})$
Slow start	$\mu + N_d$	$\mu + 2N_d$	$\mu + N_d$
	$\mu + 2N_d$	$\mu + 4N_d$	$\mu + 2N_d$
	$\mu + 4N_d$	$\mu + 8N_d$	$\mu + 4N_d$
	...	...	...
	$\mu + 2^{r-1}N_d$	$\mu + 2^r N_d = \mu + \mathbf{b/2}$	$\mu + 2^{r-1}N_d$
Congestion avoidance	$\mu + b/2$	$\mu + b/2 + N_d$	$\mu + b/2$
	$\mu + b/2 + N_d$	$\mu + b/2 + 2N_d$	$\mu + b/2 + N_d$
	...	...	...
	$\mu + b/2 + (x-1)N_d$	$\mu + b/2 + xN_d = \mathbf{B}$	$\mu + b/2 + (x-1)N_d$
Losses	$\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$
Reset	$\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$
	$\mu + b/2 + xN_d$	$\mu + N_d$	$\mu + b/2 + xN_d$

In the delayed ACK case, when the AP sends an ACK to an STA that is performing an upload, the STA generates two DATA packets, as opposed to one in the undelayed ACK case. These two DATA packets are assumed to be "served" together (recall the first paragraph of Section 3); they then travel the round trip to the server and return to the AP as a single ACK, with immediate feedback. Similarly, an ACK packet sent by a downloading STA returns to the AP as two DATA packets, again with immediate feedback from the server.

The analysis steps will be exactly like the undelayed ACK case except that we replace μ by $\mu/2$, which is the number of TCP ACKs corresponding to upload STAs resting in the AP. Thus replacing μ by $\mu/2$ in (32), we obtain

$$h = \frac{\left[(2^r-1)+\frac{x(x-1)}{2}+3x\right]N_d+(x+3)\frac{b}{2}}{(r+x+3)\frac{\mu}{2}+\left[(2^r-1)+\frac{x(x-1)}{2}+3x\right]N_d+(x+3)\frac{b}{2}}. \quad (33)$$

5. Extension to TCP Reno

The analysis easily extends to the Reno version of TCP. We first consider the undelayed ACK case, with no limit on the congestion window for download connections. The analysis is similar to that in Section 4.1. For the Reno case we assume that there are no timeouts and the recovery uses only Fast Retransmit and Fast Recovery mechanisms [11]. We assume that there are sufficient number of packets buffered for every download connection to trigger the Fast Retransmit mechanism. This will lead to absence of the slow start phase in the cumulative window evolution. Also, we note that for the Reno case it is not necessary to assume that all the download windows have the same value at the instants $T_{k,i}$. All that matters is that the cumulative download window increases linearly from $b/2$ to b in the congestion avoidance phase, with increments of N_d in $[T_{k,i}, T_{k,i+1})$. After the buffer occupancy reaches b, the loss phase and reset phase occur similar to the TCP OldTahoe case, the only difference being that at the end of the reset phase, the buffer occupancy is $b/2$ instead of N_d as in the OldTahoe. The AP buffer evolution is summarized in Table 2. The formula for h for Reno becomes:

$$h = \frac{\left[\frac{x(x-1)}{2}+3x\right]N_d+(x+3)\frac{b}{2}}{(x+3)\mu+\left[\frac{x(x-1)}{2}+3x\right]N_d+(x+3)\frac{b}{2}}. \quad (34)$$

Table 2
AP buffer evolution for TCP Reno with undelayed ACKs.

Phase	Buffer at $T_{k,i}$	Buffer at $T_{k,i+1}$	Buffer services in $[T_{k,i}, T_{k,i+1})$
Congestion avoidance	$\mu+b/2$	$\mu+b/2+N_d$	$\mu+b/2$
	$\mu+b/2+N_d$	$\mu+b/2+2N_d$	$\mu+b/2+N_d$
	...	...	...
	$\mu+b/2+(x-1)N_d$	$\mu+b/2+xN_d=\mathbf{B}$	$\mu+b/2+(x-1)N_d$
Losses	$\mu+b/2+xN_d$	$\mu+b/2+xN_d$	$\mu+b/2+xN_d$
Reset	$\mu+b/2+xN_d$	$\mu+b/2+xN_d$	$\mu+b/2+xN_d$
	$\mu+b/2+xN_d$	$\mu+b/2$	$\mu+b/2+xN_d$

Following similar lines as in Section 4.2, h for TCP Reno for the delayed ACK case is given by

$$h = \frac{\left[\frac{x(x-1)}{2}+3x\right]N_d+(x+3)\frac{b}{2}}{(x+3)\frac{\mu}{2}+\left[\frac{x(x-1)}{2}+3x\right]N_d+(x+3)\frac{b}{2}}. \quad (35)$$

6. Analytical and simulation results

All the simulation results are obtained using ns-2.31 using parameters summarized in Table 3.

6.1. Synchronized window evolution

Fig. 7 shows a window evolution snapshot for $N_u = 5$, $N_d = 5, b = 50$ in support of the assumptions and approximations made in Section 4.1. We have assumed a synchronized window evolution in Section 4.1.1. From Fig. 7 we

Table 3
IEEE 802.11b and TCP/IP parameters.

Parameter	Symbol	Value
Data rate 1	R_1	2 Mbps
Data rate 2	R_2	5.5 Mbps
Data rate 3	R_3	11 Mbps
Control rate	C_c	2 Mbps
PHY preamble time	T_P	144 μs
PHY header	T_{PHY}	48 μs
MAC header size	L_{MAC}	34 bytes
RTS packet size	L_{RTS}	20 bytes
CTS packet size	L_{CTS}	14 bytes
MAC ACK packet size	L_{ACK}	14 bytes
System slot time	δ	20 μs
DIFS time	T_{DIFS}	50 μs
SIFS time	T_{SIFS}	10 μs
EIFS time	T_{EIFS}	364 μs
Min. contention window	CW_{min}	31
Max. contention window	CW_{max}	1023
IP header	L_{IPH}	20 bytes
TCP header	L_{TCPH}	20 bytes
TCP ACK packet size	$L_{TCP-ACK}$	20 bytes
TCP data packet size	$L_{TCP-DATA}$	1500 bytes

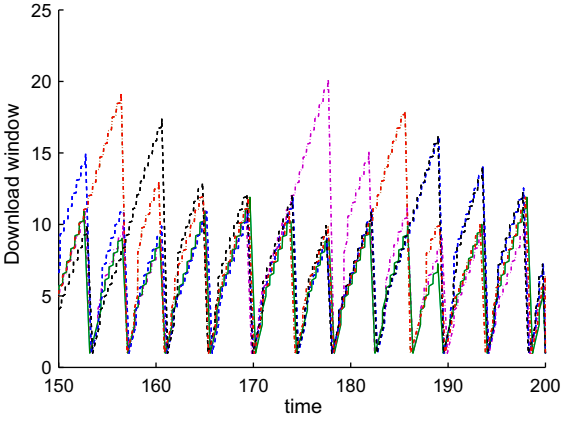

Fig. 7. Sample path of window evolutions of several connections, in support of our assumption of synchronization of the window evolution processes.

see that this assumption holds good most of the time but not always. One reason why synchronization might fail to occur is the following. Suppose that two download STAs are active at some moment, and serving one of these can potentially cause a window increase in the connection corresponding to that STA. If such an STA is served before the other STA then two packets will return to the AP because of the window increase. Now as the AP has space for two packets, both these packets will be accommodated in the AP and thus the connection will not lose packets, even though there are $\mu + b$ packets in the AP buffer. But we have found that the assumption of a synchronized window evolution model provides results that are close to the actual performance. The simulation results in support of this claim are provided in the following sections.

6.2. TCP OldTahoe

Undelayed ACK Case: Figs. 8–10 provide a validation of the analysis performed in Section 4.1 for $N_d = N_u = 5, 8, 10$. Here h is plotted vs. the buffer size expressed as $\frac{b}{2N_d}$. For uploads the maximum TCP window is $W_{\max} = 20$. Thus, for example, $\frac{b}{2N_d} = 10$, with $N_u = N_d = 5$, means that the AP buffer can accommodate 100 TCP data packets and 100 TCP ACKs. It can be seen that the analysis provides a very accurate estimate of h in spite of our several simplifying modeling assumptions. We see that for a small AP buffer, the download transfers can obtain as little as just 10% of the total packet throughput from the AP, and $\frac{b}{2N_d}$ needs to be 10 for the download throughput to be 40% of the total packet throughput from the AP.

The upload and download throughputs are obtained by multiplying the aggregate packet throughput from the AP by h; see (17). For PHY rates of 2 Mbps, 5.5 Mbps and 11 Mbps, the throughput Θ in packets/s provided by the simulations was consistently found to be 116, 230, 318, respectively, regardless of h and the number of STAs, whereas the corresponding analytically obtained values were 117, 231 and 320 in terms of packets/s. Thus, the analysis also provides a very accurate estimate of the upload and download throughputs.

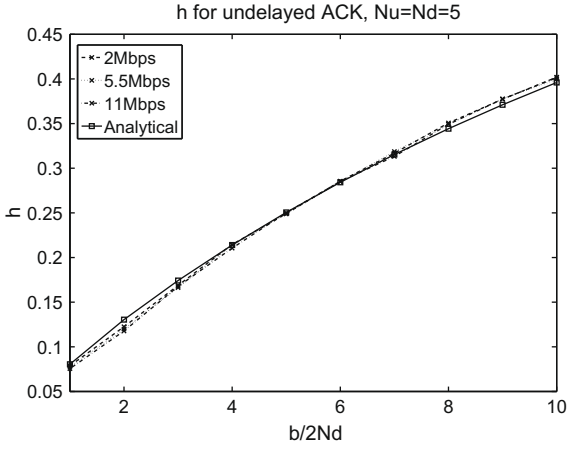

Fig. 8. TCP OldTahoe, undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$.

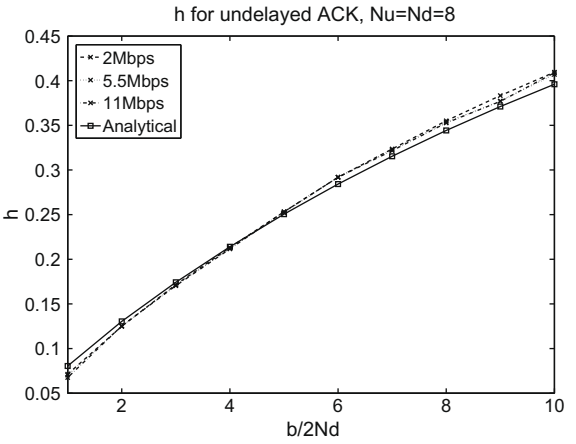

Fig. 9. TCP OldTahoe, undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 8$.

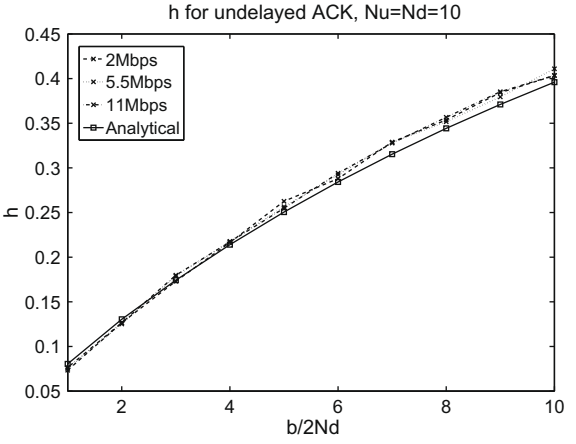

Fig. 10. TCP OldTahoe, undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 10$.

Discussion: As an illustration, we see that if $\frac{b}{2N_d} = 10$ with $N_u = N_d = 5$ (see Fig. 8), and, keeping the same buffer, we make $N_d = N_u = 10$ (see Fig. 10), then the download throughput will drop from about 40% of aggregate throughput to about 25%. Our observation of constant aggregate throughput with increasing number of nodes is consistent with the earlier work on TCP reported in [3,5].

Delayed ACK Case: Fig. 11 shows simulation results for h for the delayed ACK case in Section 4.2 for $N_u = N_d = 5$. Similar results were obtained for $N_u = N_d = 8, 10$. As for the case of undelayed ACKs, the values of total throughput were again found to be almost insensitive to h and the total number of STAs. From our simulations, these values were found to be, approximately, 123, 254, 360 packets/s for the PHY rates of 2, 5.5, 11 Mbps, respectively, very close to the analytically obtained values of 125, 257, 365, respectively.

We find that the analysis underestimates the value of h in this case. This can be explained as follows: We have assumed in Section 4.2 that when an upload ACK is transmit-

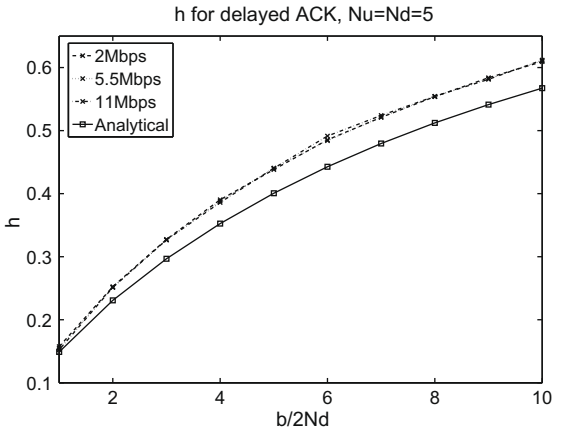

Fig. 11. TCP OldTahoe, delayed ACKs: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$.

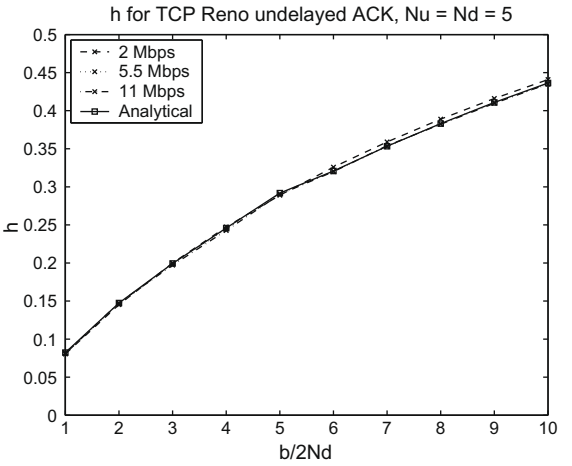

Fig. 12. TCP Reno, undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$.

ted by the AP, the resulting two DATA packets at the corresponding STA are served together, back-to-back, after one contention by the STA (see also the first paragraph of Section 3). These two DATA packets instantly reach the server and return as an ACK into the AP buffer. In practice, however, one of the upload DATA packets will stay in the STA, waiting for another contention, so that the number of upload ACKs in the AP is less than what our analysis assumes. Thus, the number of upload ACKs in the AP is overestimated by our model, which results in an underestimate of the value of h.

Discussion: It can be seen from (24) that the upload and download throughputs are equal when $h = \frac{2}{3}$. We see from Fig. 11 that for $N_d = N_u = 5$ this situation is approached for $\frac{b}{2N_d} = 10$. Another insight we obtain is that the aggregate throughput in packets per second is almost constant with buffer size and the number of nodes. Further, the simplification we made for carrying out the analysis with delayed ACKs in the first paragraph of Section 3 is seen to yield a very good approximation for throughput.

6.3. TCP Reno

Fig. 12 shows the simulation results obtained for h for TCP Reno with undelayed ACKs for $N_u = N_d = 5$. Similar results were obtained for $N_u = N_d = 8, 10$. We notice that the values of h for the same value of buffer are a little greater than with OldTahoe. This can be explained as follows: With TCP Reno, the cumulative TCP window of downloading STAs oscillates between $b/2$ and b. For TCP OldTahoe, because of repeated resetting of the window 1 and entry into the slow start phase, the cumulative window has to climb to $b/2$ and then it enters the congestion avoidance phase. Thus, the slow start phase of TCP OldTahoe results in reducing the throughput obtained by downloading STAs and hence h. Again similar to OldTahoe, for PHY rates of 2 Mbps, 5.5 Mbps and 11 Mbps, the simulated throughput Θ in packets/s was consistently found to be 116, 230, 318, respectively, regardless of h and the number of STAs, whereas the corresponding analytically obtained values were 117, 231 and 320 in terms of packets/s.

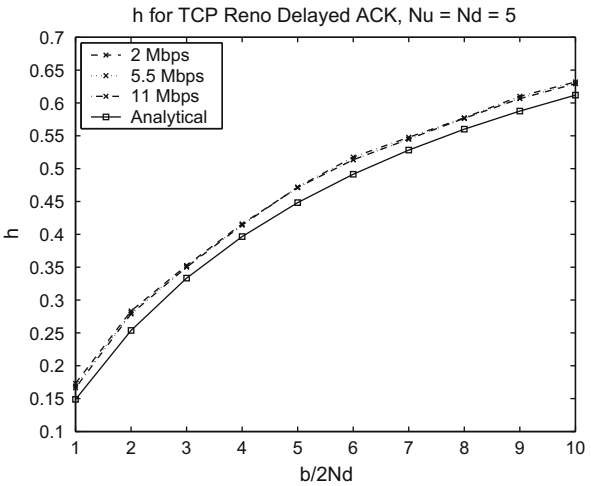

Fig. 13. TCP Reno, delayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$.

Fig. 13 shows the simulation and analysis results obtained for h with $N_u = N_d = 5$ for TCP Reno with delayed ACKs. Similar results were obtained for $N_u = N_d = 8, 10$. Again the values of total throughput were found to be almost insensitive to h and the total number of STAs. These values were approximately 125, 257, 365 packets/s for the PHY rate of 2, 5.5, 11 Mbps, respectively and very close to the analytical results.

6.4. Bounds on h with finite $W_{\max}$

The results in Section 6.2 were provided assuming no maximum window limit on download connections but the upload connections had a maximum window limit of 20. At the end of Section 4.1.2 we provided a simple upper bound on h for the case of undelayed ACK and when all connections have a maximum window limit. Intuitively,

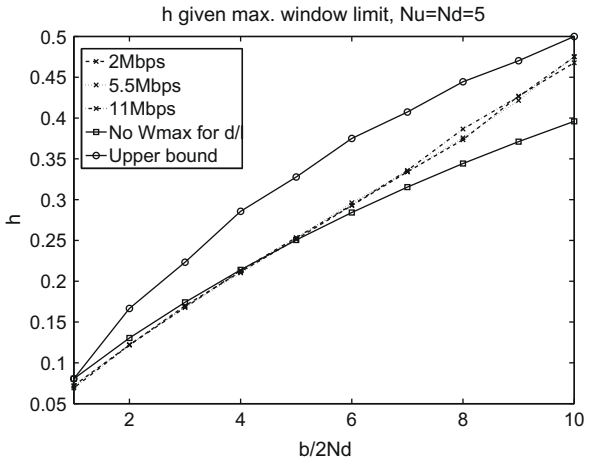

Fig. 14. TCP OldTahoe, undelayed ACK: h for $N_u = N_d = 5$, $W_{max} = 20$.

we anticipate that for small values of the AP buffer, the analytical h would be close to the h obtained with no maximum window limit only on download connections. Fig. 14 shows the simulated value of h given the maximum window limit of 20 on all the connections, and analytical values of h for the case when there is maximum window limit only on upload connections for $N_u = N_d = 5$. Similar results were obtained for $N_u = N_d = 8, 10$. We see that for small buffer sizes the analytical model with no maximum limit on the download TCP windows provides an accurate estimate of h. For other cases, the two analyses provide bounds on h.

7. Conclusion

The analysis for calculating h is essentially *rateless*, i.e., the value of h does not change with PHY rate as long as the number of uploading STAs, the number downloading STAs, AP buffer size and maximum window limit for upload connections remain same. Thus, we can expect to obtain the same value of h even in the scenario where STAs associate with the AP with different PHY rates. The analysis for calculating h made use of only the fact that the average number of active STAs is small, as stated in Section 4.1.1, and has no other dependence on the underlying MAC layer analysis.

Since, for a given the PHY rate, there is no variation in aggregate throughput with the number of STAs (see Sections 6.2 and 6.3), it motivates the processor sharing model for the case of randomly arriving short file transfers. The model will consist of $N = N_u + N_d$ connections and a server serving packets at $\Theta = \Theta_u + \Theta_d$ packets/second. Each download STA will get service at the rate $\frac{h\Theta}{N_d}$ packets/s and each upload STA will get service with $\frac{(1-h)\Theta}{N_u}$ packets/s. With this model, the mean time taken to complete the file transfers can be predicted [11]; we leave the study of the accuracy of such a model as future work.

We have thus provided a fairly general analytical model that (i) explains the observations made by several prior experimental and simulation studies (e.g., [1]), (ii) yields several new insights into the interaction of the TCP protocol and the IEEE 802.11 MAC (e.g., beyond those in [3,5]), and (iii) provides an accurate model that could be used to predict performance, perhaps for the purpose of network engineering.

Appendix A. DTMC for the undelayed ACK case

Consider the DTMC (D_k, U_k) characterized by the following transition probabilities:

$$\Pr((d-1,u)/(d,u)) = \frac{d}{u+d+1} \tag{36}$$

$$\Pr((d,u-1)/(d,u)) = \frac{u}{u+d+1} \tag{37}$$

$$\Pr((d+1,u)/(d,u)) = \frac{h}{u+d+1} \tag{38}$$

$$\Pr((d,u+1)/(d,u)) = \frac{1-h}{u+d+1} \tag{39}$$

The Markov chain is depicted in Fig. 4. The following lemmas will be used in the analysis [8].

Lemma 7.1. *A Markov chain is reversible if and only if there exists a probability distribution π on S, where S is the set of all the states of the Markov chain, such that, for all i, and $j \in S$,*

$$\pi_i p_{ij} = \pi_j p_{ji} \tag{40}$$

where p_{ij} denotes the transition probability from state i to state j.

Lemma 7.2. *A Markov chain is reversible if, in every closed (circular) path formed by its states, the product of the transition probabilities in the clockwise direction is equal to the product of the transition probabilities in the anti-clockwise direction.*

A.1. Stationary state distribution

Using Lemma 7.2, the DTMC in Fig. 4 can be easily seen to be reversible. Recall that, $\pi(d,u)$ denotes the stationary state probability of the state (d,u). Using Lemma 7.1, we can write

$$\pi(d,u-1)\frac{1-h}{u+d} = \pi(d,u)\frac{u}{u+d+1}. \tag{41}$$

Hence,

$$\frac{\pi(d,u)}{\pi(d,u-1)} = \frac{(1-h)(u+d+1)}{u(u+d)}. \tag{42}$$

Applying (42) repetitively, we get

$$\frac{\pi(d,u)}{\pi(d,u-1)} \times \cdots \times \frac{\pi(d,1)}{\pi(d,0)} = \frac{(1-h)^u}{u!} \times \frac{u+d+1}{d+1} \tag{43}$$

or

$$\frac{\pi(d,u)}{\pi(d,0)} = \frac{(1-h)^u}{u!} \times \frac{u+d+1}{d+1}. \tag{44}$$

Similarly, we can obtain

$$\frac{\pi(d,u)}{\pi(0,u)} = \frac{h^d}{d!} \times \frac{u+d+1}{u+1}. \tag{45}$$

Substituting $u = 0$ in the above equation gives

$$\frac{\pi(d,0)}{\pi(0,0)} = \frac{h^d}{d!} \times (d+1). \tag{46}$$

Multiplying Eqs. (44) and (46) we get

$$\frac{\pi(d,u)}{\pi(0,0)} = (u+d+1) \times \frac{h^d(1-h)^u}{d!u!}. \tag{47}$$

Now using the normalization equation

$$\sum_{d=0}^{\infty}\sum_{u=0}^{\infty} \pi(d,u) = 1, \tag{48}$$

we get

$$\sum_{d=0}^{\infty}\sum_{u=0}^{\infty} (u+d+1)\frac{h^d(1-h)^u}{d!u!} \times \pi(0,0) = 1. \tag{49}$$

Thus, we can write

$$\begin{aligned}\frac{1}{\pi(0,0)} &= \sum_{d=0}^{\infty}\frac{h^d}{d!}\sum_{u=0}^{\infty}(u+d+1)\frac{(1-h)^u}{u!} \\ &= \sum_{d=1}^{\infty}\frac{h^d}{(d-1)!}\left\{\sum_{u=0}^{\infty}\frac{(1-h)^u}{u!} + \sum_{d=0}^{\infty}\frac{h^d}{d!}\sum_{u=1}^{\infty}\frac{(1-h)^u}{(u-1)!} + \sum_{d=0}^{\infty}\frac{h^d}{d!}\sum_{u=0}^{\infty}\frac{(1-h)^u}{u!}\right\} \\ &= e^{(1-h)}e^h\{h+(1-h)+1\} = 2e. \end{aligned} \tag{50}$$

$$\therefore \pi(0,0) = \frac{1}{2e}. \tag{51}$$

Substituting $\pi(0,0) = \frac{1}{2e}$ in Eq. (47), we get

$$\pi(d,u) = \frac{u+d+1}{2e} \times \frac{h^d(1-h)^u}{d!u!}. \tag{52}$$

A.2. Mean number of active download and upload STAs

The mean number of active download STAs is given by

$$\begin{aligned}\mathrm{E}(D) &= \sum_{u=0}^{\infty}\sum_{d=0}^{\infty} d \times \frac{u+d+1}{2e} \times \frac{h^d(1-h)^u}{d!u!} \\ &= \frac{1}{2e}\sum_{u=0}^{\infty}\frac{(1-h)^u}{u!}\sum_{d=0}^{\infty}(d^2+d(u+1)) \times \frac{h^d}{d!} \\ &= \frac{1}{2e}\sum_{u=0}^{\infty}\frac{(1-h)^u e^h}{u!}\{(h^2+h)+(u+1)h\} \\ &= \frac{1}{2e}e^h e^{(1-h)}\{h^2+h+h(1-h)+h\} = \frac{3h}{2}. \end{aligned} \tag{53}$$

Observing symmetry of Eq. (52) in u and d, it can be easily seen that the mean number of active upload STAs is given by

$$\mathrm{E}(U) = \frac{3(1-h)}{2}. \tag{54}$$

Appendix B. DTMC for the delayed ACK case

The DTMC (D_k, U_k) for this case is characterized by following equations:

$$\Pr((d-1,u)/(d,u)) = \frac{d}{u+d+1} \tag{55}$$

$$\Pr((d,u-1)/(d,u)) = \frac{u}{u+d+1} \tag{56}$$

$$\Pr((d+1,u)/(d,u)) = \frac{\frac{h}{2}}{u+d+1} \tag{57}$$

$$\Pr((d,u)/(d,u)) = \frac{\frac{h}{2}}{u+d+1} \tag{58}$$

$$\Pr((d,u+1)/(d,u)) = \frac{1-h}{u+d+1} \tag{59}$$

This DTMC is depicted in Fig. 5. Using Lemma 7.2, it can be verified to be reversible. Eq. (47) now changes to

$$\frac{\pi(d,u)}{\pi(0,0)} = (u+d+1) \times \frac{\left(\frac{h}{2}\right)^d(1-h)^u}{d!u!}. \tag{60}$$

Following the same analysis steps as in the undelayed ACK case, we can write

$$\begin{aligned}\frac{1}{\pi(0,0)} &= \sum_{d=0}^{\infty}\frac{\left(\frac{h}{2}\right)^d}{d!}\sum_{u=0}^{\infty}(u+d+1)\frac{(1-h)^u}{u!} \\ &= e^{(1-h)}\sum_{d=1}^{\infty}\left\{\frac{\left(\frac{h}{2}\right)^d}{(d-1)!} + (1-h)\sum_{d=0}^{\infty}\frac{\left(\frac{h}{2}\right)^d}{d!} + \sum_{d=0}^{\infty}\frac{\left(\frac{h}{2}\right)^d}{d!}\right\} \\ &= e^{(1-h)}e^{\left(\frac{h}{2}\right)}\left\{\frac{h}{2}+(1-h)+1\right\} = e^{1-\left(\frac{h}{2}\right)}\left(2-\frac{h}{2}\right). \end{aligned}$$

$$\therefore \pi(0,0) = \frac{1}{e^{1-\left(\frac{h}{2}\right)}\left(2-\frac{h}{2}\right)}. \tag{61}$$

Hence,

$$\pi(d,u) = \frac{u+d+1}{e^{1-\left(\frac{h}{2}\right)}\left(2-\frac{h}{2}\right)} \times \frac{\left(\frac{h}{2}\right)^d(1-h)^u}{d!u!}. \tag{62}$$

Appendix C. Derivation of $E_{(d,u)}X$

The denominator of (15) requires $\mathrm{E}_{(d,u)}X$, i.e., the mean cycle time starting in the state (d,u). As explained earlier, a contention cycle comprises several channel slots, and we obtain the mean cycle time by writing down simple recursive expressions by embedding at channel slot boundaries. The "back-off" periods shown in Fig. 3 comprise several idle slots in which none of the nodes attempts. If one or more attempts occur at a channel slot boundary, there is a success or collision accordingly. Starting in the state (d,u), the state remains (d,u) until a successful transmission ends. The following events can happen at the channel slot boundaries:

- The slot goes idle with probability $P_{idle} = (1-\beta_{u+d+1})^{u+d+1}$.
- The AP succeeds with probability $P^{AP}_{succ} = \beta_{u+d+1}(1-\beta_{u+d+1})^{u+d}$.
- A download STA succeeds with probability $P^{d}_{succ} = d\beta_{u+d+1}(1-\beta_{u+d+1})^{u+d}$.

- An upload STA succeeds with probability $P^{u}_{succ} = u\beta_{u+d+1}(1-\beta_{u+d+1})^{u+d}$.
- There is a collision with the remaining probability.

When two or more transmissions collide, the duration of collision is given by the duration of the longest transmission. Hence, we distinguish among the various possibilities of collisions depending on whether a download STA or an upload STA is involved in the collisions, and also whether the AP, if involved in the collision, has a TCP data packet or a TCP ACK at the HOL position. This is due to the fact that, the time spent in collision can be dominated by either the duration of RTS or the duration of TCP ACK depending on the PHY rates. Let T_{coll1} (resp. T_{coll2}) denote the duration of a collision given that the RTS (resp. the TCP ACK) is the longest packet involved in the collision. Then, T_{coll1} and T_{coll2} are given by (see Table 3)

$$T_{coll1} = T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{EIFS}$$

$$T_{coll2} = T_P + T_{PHY} + \frac{L_{MAC} + L_{IPH} + L_{TCP-ACK}}{R_{data}} + T_{EIFS}$$

Note that $T_{coll2} < T_{coll1}$ at 11 Mbps and $T_{coll2} > T_{coll1}$ at 2 Mbps and 5.5 Mbps. The various possibilities of collisions can now be summarized as follows:

- An AP transmission collides with a transmission by the download STA (and upload STAs are not involved in the collision) with probability $P^{AP,d}_{coll} = \beta_{u+d+1}(1-\beta_{u+d+1})^{u}[1-(1-\beta_{u+d+1})^{d}]$.
- The AP transmission collides with a transmission by the upload STA (and download STAs are not involved in the collision) with probability $P^{AP,u}_{coll} = \beta_{u+d+1}(1-\beta_{u+d+1})^{d}[1-(1-\beta_{u+d+1})^{u}]$.
- Two or more download STAs collide (and neither the AP nor the upload STAs are involved in the collision) with probability $P^{d}_{coll} = (1-\beta_{u+d+1})^{u+1} \times [1-(1-\beta_{u+d+1})^{d} - \beta_{u+d+1}(1-\beta_{u+d+1})^{d-1}]$.
- Two or more upload STAs collide (and neither the AP nor the download STAs are involved in the collision) with probability $P^{u}_{coll} = (1-\beta_{u+d+1})^{d+1} \times [1-(1-\beta_{u+d+1})^{u} - u\beta_{u+d+1}(1-\beta_{u+d+1})^{u-1}]$.
- Both download and upload STAs are involved in the collision (AP may or may not be involved) with probability $P^{d,u}_{coll} = [1-(1-\beta_{u+d+1})^{u}][1-(1-\beta_{u+d+1})^{d}]$.

As the time interval $(G_{k-1}, G_k]$ depends on whether the packet at HOL at the AP was Data or ACK packet, the expected cycle length can be expressed as

$$\mathsf{E}_{(d,u)}X = h\ \mathsf{E}^{DATA}_{(d,u)}X + (1-h)\ \mathsf{E}^{ACK}_{(d,u)}X. \tag{63}$$

where $\mathsf{E}^{DATA}_{(d,u)}$ and $\mathsf{E}^{ACK}_{(d,u)}$ denote the expected cycle lengths starting in the state (d,u) given that the HOL packet at the AP is a TCP data packet or TCP ACK, respectively. Let P^{DATA}_{coll1} (resp. P^{DATA}_{coll2}) denote the probability that the time spent in collision is T_{coll1} (resp. T_{coll2}) given that the AP's HOL position contains a TCP data packet. Let P^{ACK}_{coll1} (resp. P^{ACK}_{coll2}) denote the probability that the time spent in collision is T_{coll1} (resp. T_{coll2}) given that the AP's HOL position contains a TCP ACK. As noted earlier, the above probabilities depend on the PHY rates and have been summarized in Table 4.

Table 4
Parameter values for Eqs. (64) and (65).

Parameter	Value at 11 Mbps	Value at 2 Mbps and 5.5 Mbps
P^{DATA}_{coll1}	$P^{AP,d}_{coll} + P^{AP,u}_{coll} + P^{u}_{coll} + P^{d,u}_{coll}$	$P^{AP,u}_{coll} + P^{u}_{coll}$
P^{DATA}_{coll2}	P^{d}_{coll}	$P^{AP,d}_{coll} + P^{d}_{coll} + P^{d,u}_{coll}$
P^{ACK}_{coll1}	$P^{AP,u}_{coll} + P^{u}_{coll} + P^{d,u}_{coll}$	P^{u}_{coll}
P^{ACK}_{coll2}	$P^{AP,d}_{coll} + P^{d}_{coll}$	$P^{AP,d}_{coll} + P^{AP,u}_{coll} + P^{d}_{coll} + P^{d,u}_{coll}$

Table 4 can be explained as follows. Consider the 11 Mbps case when the AP's HOL position contains a TCP data packet. Since $T_{coll2} < T_{coll1}$ at 11 Mbps, the time spent in collision is T_{coll2} iff neither the AP nor any upload STA is involved in the collision. If either the AP or any of the upload STAs is involved in the collision, then the time spent in collision is T_{coll1}. Thus, in this case, we have

$$P^{DATA}_{coll2} = P^{d}_{coll}$$

and

$$P^{DATA}_{coll1} = P^{AP,d}_{coll} + P^{AP,u}_{coll} + P^{u}_{coll} + P^{d,u}_{coll}.$$

Other entries in Table 4 can be similarly explained.

Applying a renewal argument, $\mathsf{E}^{DATA}_{(d,u)}$ and $\mathsf{E}^{ACK}_{(d,u)}$ can be recursively written as follows:

$$\begin{aligned}\mathsf{E}^{DATA}_{(d,u)}X = {} & P_{idle}\left(\delta + \mathsf{E}^{DATA}_{(d,u)}X\right) + P^{DATA}_{coll1}\left(T_{coll1} + \mathsf{E}^{DATA}_{(d,u)}X\right) \\ & + P^{DATA}_{coll2}\left(T_{coll2} + \mathsf{E}^{DATA}_{(d,u)}X\right) + P^{AP}_{succ}T_{TCP-DATA} \\ & + P^{u}_{succ}T_{TCP-DATA} + P^{d}_{succ}T_{TCP-ACK}\end{aligned} \tag{64}$$

$$\begin{aligned}\mathsf{E}^{ACK}_{(d,u)}X = {} & P_{idle}\left(\delta + \mathsf{E}^{ACK}_{(d,u)}X\right) + P^{ACK}_{coll1}\left(T_{coll1} + \mathsf{E}^{ACK}_{(d,u)}X\right) \\ & + P^{ACK}_{coll2}\left(T_{coll2} + \mathsf{E}^{ACK}_{(d,u)}X\right) + P^{AP}_{succ}T_{TCP-ACK} \\ & + P^{u}_{succ}T_{TCP-DATA} + P^{d}_{succ}T_{TCP-ACK}\end{aligned} \tag{65}$$

where $T_{TCP-DATA}$ and $T_{TCP-ACK}$ denote the times taken for transmission of TCP data packet and TCP ACK, respectively. Eqs. (63)–(65) provide $\mathsf{E}_{(d,u)}X$. The attempt probabilities needed to compute various probabilities can be obtained by a saturated analysis as in [6] or [7], and the various time durations can be obtained using the parameter values summarized in Table 3.

References

[1] M. Gong, Q. Wu, C. Williamson, Queue management strategies to improve TCP fairness in IEEE 802.11 wireless LANs, in: The Second Workshop on Resource Allocation in Wireless NETworks, RAWNET, Boston, MA, 2006.
[2] S. Pilosof, R. Ramjee, D. Raz, Y. Shavit, P. Sinha, Understanding TCP Fairness Over Wireless LAN, in: Proceedings of IEEE INFOCOM'03, 2003.
[3] R. Bruno, M. Conti, E. Gregori, Modeling TCP Throughput Over Wireless LANs, in: Proceedings of the 17th IMACS World Congress Scientific Computation, Applied Mathematics and Simulation, Paris, France, 2005.

[4] S. Harsha, A. Kumar, V. Sharma, An analytical model for the capacity estimation of combined VoIP and TCP file transfers over EDCA in an IEEE 802.11e WLAN, in: 14th IEEE International Workshop on Quality of Service (IWQoS), Yale University, New Haven, 2006.
[5] G. Kuriakose, S. Harsha, A. Kumar, V. Sharma, Analytical models for capacity estimation of IEEE 802.11 WLANs using DCF for internet applications, Wireless Networks, Springer 15 (2).
[6] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE Journal on Selected Areas in Communications 18 (3) (2000) 535–547.
[7] A. Kumar, E. Altman, D. Miorandi, M. Goyal, New insights from a fixed point analysis of single cell IEEE 802.11 WLANs, IEEE/ACM Transactions on Networking 15 (3) (2007) 588–601 (also appeared in INFOCOM, March 13–17, 2005, pp. 1550–1561).
[8] F. Kelly, Reversibility and Stochastic Networks, John Wiley, 1979.
[9] V.G. Kulkarni, Modeling and Analysis of Stochastic Systems, Chapman and Hall, London, UK, 1995.
[10] RFC 1323, <http://www.ietf.org/rfc/rfc1323.txt>, 1992.
[11] A. Kumar, D. Manjunath, J. Kuri, Communication Networking: An Analytical Approach, The Morgan Kaufman Series in Networking, Morgan Kaufmann, an Imprint of Elsevier Series in Networking, 2004.

Onkar Bhardwaj obtained his Masters in Telecommunication from the Indian Institute of Science, Bangalore, India in 2008 where he worked on IEEE 802.11 WLANs. His interests are Communication Networking and Algorithms. Currently he is with Computational Research Laboratories, Pune, India, where he is working on Numerical Linear Algebra algorithms.

G.V.V. Sharma was born in Visakhapatnam, India. He received the B.Tech. degree in Electronics and communication engineering from the Indian Institute of Technology, Guwahati, India, in 1999 and the M.Sc. (Eng.) degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 2004. Currently, he is pursuing the Ph.D. degree at the Department of Electrical Engineering, Indian Institute of Technology, Bombay, India. From August 2004 to July 2006, he was with the Applied Research Group of Satyam Computers, Bangalore. His research interests include communication theory and signal processing algorithms fro communication systems.

Manoj K. Panda obtained an M.Tech., degree in Electrical Engineering from Indian Institute of Technology (IIT) Kanpur, India, in March 2003. He was with the Applied Research Group (ARG), Satyam Computers Services Ltd., Bangalore, until January 2005 when he joined Indian Institute of Science (IISc) Bangalore as a Ph.D. student. He is currently working towards his Ph.D. His areas of interest include modeling, model based simulation and optimization of communication networks, in general, and of wireless local area networks, in particular.

Anurag Kumar (B.Tech., IIT Kanpur, Ph.D. Cornell University, both in EE) was with Bell Labs, Holmdel, for over 6 years. He is now a Professor in the ECE Department at the Indian Institute of Science (IISc), Bangalore, and Chair of the Division of Electrical Sciences. His area of research is communication networking, and he has recently focused primarily on wireless networking. He is a Fellow of the IEEE, of the Indian National Science Academy (INSA), and of the Indian National Academy of Engineering (INAE). He has been an associate editor of IEEE Transactions on Networking, and of IEEE Communications Surveys and Tutorials. He is a coauthor of the advanced text-books "Communication Networking: An Analytical Approach," and "Wireless Networking," by Kumar, Majunath and Kuri, published by Morgan-Kaufman/Elsevier.