Modeling Finite Buffer Effects on TCP Traffic over an IEEE 802.11 Infrastructure WLAN

Onkar Bhardwaj*, G. V. V. Sharma†, Manoj K. Panda*, Anurag Kumar*
*Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India.
Email: onkar.bhardwaj@gmail.com, manoj@ece.iisc.ernet.in, anurag@ece.iisc.ernet.in
†Department of Electrical Engineering, Indian Institute of Technology, Bombay, India.
Email: gadepall@ee.iitb.ac.in

***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 contending for transmission over the wireless medium. 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 known (see, for example, [3], [9]) that because of packet loss 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 for TCP Reno.**

***Index Terms*—TCP over IEEE 802.11 WLANs, TCP unfairness in WLANs, TCP modeling**

I. Introduction

We consider a scenario in which several stations (STAs) are associated with a single Access Point (AP). The AP has a finite amount of FIFO buffer to store packets. In this paper, for simplicity, we consider associations only at a single Physical (PHY) rate; e.g., in IEEE 802.11b the PHY rates 11 Mbps, 5.5 Mbps and 2 Mbps are available. 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[1]. 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 [3], [9]. Our objective in this paper 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. [2] 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 that the TCP windows of all the connections are equal, 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 EDCA, was employed by Sri Harsha et al. [10] 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. [7]. It is known that, if there is packet loss at the AP due to finite buffers, then in a situation of simultaneous upload and download transfers, the upload transfers each obtain a larger throughput than any of the download transfers. Gong et al. [3] 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. Gong et al. also propose queue management strategies in order to alleviate the throughput unfairness. Among the other literature, Pilosof et al. [9] analyzed the same problem of unfairness by assuming an M/M/1/K model for the finite buffer at the AP. In the present paper, we do not assume any such conventional queueing model, but develop a model that combines the earlier models for TCP controlled file transfers (i.e., [2] and [7]) along with a detailed model of TCP window evolution under tail-drop loss at the AP.

Outline of the Paper: The analytical model comprises two steps. In the first step (Sections II and III), we use a simple extension of the analytical model of [2] 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 IV), we obtain the value of h 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 analysis are provided in Section VI.

II. 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

This work was supported by Airtight Networks, Pune, India.

[1]The situation in which the server is located across a wide-area network will be considered in our future research.

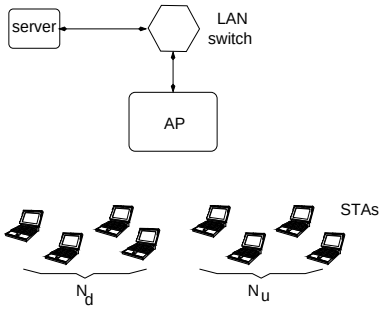

Fig. 1. The network scenario, comprising several STAs associated with an AP, each uploading (or downloading) a large file to (or from) a server attached to the high-speed wired LAN.

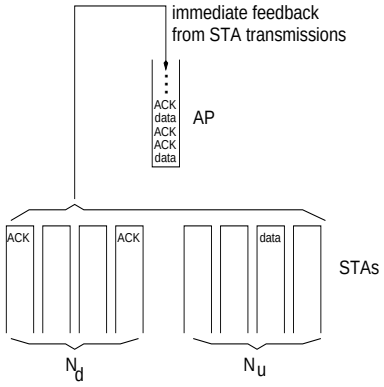

Fig. 2. Schematic based on the modeling assumptions discussed in the text. There is immediate "feedback" due to packets transmitted by the STAs.

a single upload connection (see Figure 1). All file transfers are to or from a "server" on the high-speed LAN to which the AP is connected. TCP ACK transmissions on the WLAN use the "Basic Access" mode whereas TCP data transmissions use the "RTS-CTS" mode[2].

A. Modeling Observations and 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. 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.

2) It is now well known (see [2] or [7]) 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.

[2]Other alternatives can also easily be analyzed in the same framework.

3) Furthermore, as in [7] (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 an empty STA. A TCP data packet (resp., TCP ACK) transmitted by the AP results in a TCP ACK (resp., TCP data packet) being generated at an STA.

Figure 2 depicts the model described above. Note that, since the success or failure of a 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 specific values of N_u and N_d (as long as N is large), nor on the packet lengths and PHY rates.

B. The Process (D_k, U_k) and its Analysis

We now develop the stochastic analysis along lines similar to [2] or [7].

With reference to Figure 3, let $G_k, k \in \{0, 1, 2, \cdots\}$, denote the instants when a successful transmission ends. According to our assumptions above, the AP 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. 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 STAs contending. 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 $(D_k, U_k), k \geq 0$ can only change state at the instants $G_k, k \geq 0$.

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. Hence, a channel slot can be an idle back-off slot, or a collision, or a successful transmission. 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, which is the steady-state attempt probability when m saturated nodes are contending (see [7]), and can be obtained from the approximate analysis provided in [1] or [5]. Note that β_m will include 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 $G_k, k \geq 0$, by $Z_k = 1$ if the HOL packet at the AP at time G_{k+} (i.e., in the interval $[G_k, G_{k+1})$) is data, and by $Z_k = 0$

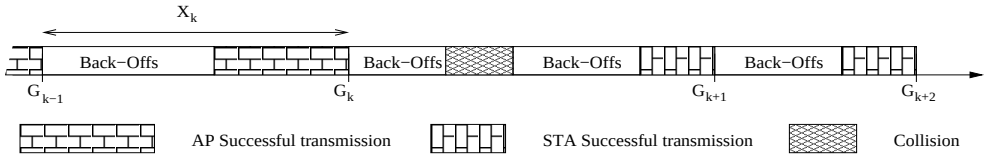

Fig. 3. Evolution of channel activity, showing the random times G_k at which successful transmissions end.

otherwise. Now define

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

i.e., the fraction of contention cycles $[G_k, G_{k+1})$ in which the HOL packet at the AP is a data packet.

Some observations about h: Let $\mathcal{A}_n$ (resp., $\mathcal{D}_n$) 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 packet occupies the HOL position at the AP buffer. Then we can write

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

from which it can easily be shown that

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

where

$$\begin{aligned} \lambda^{(d)} &= \text{rate of download packets transmitted by the AP per contention cycle} \\ &= \lim_{n\to\infty} \frac{\mathcal{A}_n}{n} = \lim_{n\to\infty} \frac{\mathcal{D}_n}{n} \qquad (4) \\ \mathsf{E}(V) &= \text{Average number of contention cycles taken to successfully transmit an HOL packet at the AP} \\ &= \lim_{n\to\infty} \frac{1}{\mathcal{D}_n}\sum_{j=1}^{\mathcal{D}_n} V_j = \lim_{n\to\infty} \frac{1}{\mathcal{A}_n}\sum_{j=1}^{\mathcal{A}_n} V_j \qquad (5) \end{aligned}$$

Similarly, as the expected number of channel slots 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 equations (3) and (6) we get

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

Thus h is also the ratio of the download throughput from the AP to the total throughput from the AP. 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)).

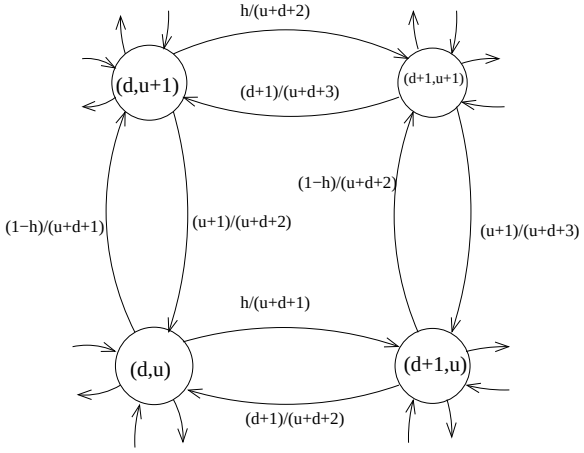

Fig. 4. Transition probability diagram of the process (D_k, U_k) for the case of undelayed ACKs.

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 $\{(d,u) : d \in \{0,1,\cdots,N_d\}, u \in \{0,1,\cdots,N_u\}\}$.

The transition probability structure of the DTMC (D_k, U_k) is shown in Figure 4. This is a finite state irreducible DTMC. Let us denote the stationary distribution by $\pi = \pi(d,u), d \in \{0,1,\cdots,N_d\}, u \in \{0,1,\cdots,N_u\}$. It can easily be seen (from Kolmogorov's criterion) that this a reversible DTMC. Using this observation the explicit expression for $\pi(d,u)$ can be shown to be the following

$$\pi(d,u) = \frac{u+d+1}{2e} \times \frac{h^d (1-h)^u}{d!u!} \tag{8}$$

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 $G_k, k \ge 0$, we observe that the time average number of active download STAs will be given by (we skip the derivation due to space constraints)

$$\mathsf{E}(D) = \sum_{u=0}^{\infty}\sum_{d=0}^{\infty} d \times \frac{u+d+1}{2e} \times \frac{h^d(1-h)^u}{d!u!} = \frac{3h}{2} \tag{9}$$

Observing symmetry of equation 8 in u and d, we see that the time average number of active upload STAs is given by

TABLE I
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

$$\mathsf{E}(U) = \frac{3(1-h)}{2} \tag{10}$$

Thus, the mean number of STAs with packets in them is $\frac{3}{2}$.

C. Throughput Analysis

With reference to Figure 3, let $X_k := G_k - G_{k-1}$. The process $\{((D(k), U(k)), G_k), k \geq 0\}$, is a Markov renewal process with cycle lengths $\{X_k, k \geq 1\}$. We can use Markov regenerative analysis to obtain performance measures such as the throughput Θ of the AP ([4]). For $t \geq 0$, let $H(t)$ denote the total number of AP successes in $[0, t]$. Let the number of successful attempts made by the AP in the interval $(G_{k-1}, G_k]$ (the k^{th} "cycle") be denoted by $H_k \in \{0, 1\}$. We view H_k as a "reward" associated with the k^{th} cycle. Then by Markov regenerative analysis ([4]) we conclude that

$$\Theta := \lim_{t\to\infty} \frac{H(t)}{t} \stackrel{w.p.\ 1}{=} \frac{\sum_{d,u} \pi(d,u) \frac{1}{u+d+1}}{\sum_{d,u} \pi(d,u) \mathsf{E}_{d,u} X} \tag{11}$$

which can be interpreted as the mean reward in a cycle divided by the expected cycle length. We now compute these two terms. The numerator of (11) gives the probability that the AP succeeds in a cycle. Using the expression for $\pi(d, u)$ in (8), we obtain (omitting the algebra)

$$\sum_{d,u} \pi(d,u) \left(\frac{1}{u+d+1}\right) = \frac{1}{2e} e^h e^{(1-h)} = \frac{1}{2} \tag{12}$$

This is as expected for TCP transfers with undelayed ACKs, since the AP must transmit half the number of total packet transmissions.

The denominator of (11) requires $\mathsf{E}_{(d,u)} X$, the mean cycle time starting in the state (d, u). The "back-off" periods shown in Figure 3 comprise several idle slots in which none of the nodes attempts. If one or more attempts occur, there is a collision or success accordingly. 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. If a channel slot starts in state (d, u) then in the channel slot the following events can happen.

- The AP succeeds with probability $P_{sAP} = \beta_{u+d+1}(1 - \beta_{u+d+1})^{u+d}$
- A download STA succeeds with probability $P_{sSTAd} = d\beta_{u+d+1}(1 - \beta_{u+d+1})^{u+d}$
- An upload STA succeeds with probability $P_{sSTAu} = u\beta_{u+d+1}(1 - \beta_{u+d+1})^{u+d}$
- The slot goes idle with probability $P_{idle} = (1 - \beta_{u+d+1})^{u+d+1}$
- There is a collision with the remaining probability

The time spent in collision will be dominated by either RTS duration or TCP ACK duration. These time intervals are given by (see Table I)

$$\begin{aligned} T_{coll1} &= T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{EIFS} \\ T_{coll2} &= T_P + T_{PHY} + \frac{L_{MAC} + L_{IPH} + L_{TCP-ACK}}{R_{data}} + T_{EIFS} \end{aligned}$$

where T_{coll1} is the time spent in collision when RTS is the longest packet involved in the collision and T_{coll2} is the time spent in collision when a TCP ACK is the longest packet in the collision.

1) An AP transmission collides with a transmission by the download STA with probability $P_1 = \beta_{u+d+1}(1 - \beta_{u+d+1})^u[1 - (1 - \beta_{u+d+1})^d]$
2) The AP transmission collides with the transmission by the upload STA with probability: event is the cause of collision is $P_2 = \beta_{u+d+1}(1 - \beta_{u+d+1})^d[1 - (1 - \beta_{u+d+1})^u]$
3) Two or more download STAs collide with probability $P_3 = (1 - \beta_{u+d+1})^{u+1} \times [1 - (1 - \beta_{u+d+1})^d - d\beta_{u+d+1}(1 - \beta_{u+d+1})^{d-1}]$
4) Two or more upload STAs collide with probability $P_4 = (1 - \beta_{u+d+1})^{d+1} \times [1 - (1 - \beta_{u+d+1})^u - u\beta_{u+d+1}(1 - \beta_{u+d+1})^{u-1}]$
5) The collision is between upload and download STAs with probability $P_5 = [1 - (1 - \beta_{u+d+1})^u][1 - (1 - \beta_{u+d+1})^d]$

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

$$\mathsf{E}_{(d,u)} = h\, \mathsf{E}^{DATA}_{(d,u)} X + (1-h)\, \mathsf{E}^{ACK}_{(d,u)} X \tag{13}$$

Here $\mathsf{E}^{DATA}_{(d,u)}$ and $\mathsf{E}^{ACK}_{(d,u)}$ denote the expected cycle lengths given the the packet at HOL at the AP was Data or ACK respectively. We can see that

$$\begin{aligned} \mathsf{E}^{DATA}_{(d,u)} X &= P_{idle}(\delta + \mathsf{E}^{DATA}_{(d,u)} X) \\ &+ P^{DATA}_{coll1}(T_{coll1} + \mathsf{E}^{DATA}_{(d,u)} X) \\ &+ P^{DATA}_{coll2}(T_{coll2} + E^{DATA}_{(d,u)} X) + P_{sAP} T_{TCP-DATA} \\ &+ P_{sSTAu} T_{TCP-DATA} + P_{sSTAd} T_{TCP-ACK} \quad (14) \\ \mathsf{E}^{ACK}_{(d,u)} X &= P_{idle}(\delta + \mathsf{E}^{ACK}_{(d,u)} X) \\ &+ P^{ACK}_{coll1}(T_{coll1} + \mathsf{E}^{ACK}_{(d,u)} X) \\ &+ P^{ACK}_{coll2}(T_{coll2} + E^{ACK}_{(d,u)} X) + P_{sAP} T_{TCP-ACK} \\ &+ P_{sSTAu} T_{TCP-DATA} + P_{sSTAd} T_{TCP-ACK} \quad (15) \end{aligned}$$

TABLE II
PARAMETER VALUES FOR EQUATIONS (13), (14), (15)

Parameter	Value at 11Mbps	Value at $2, 5.5$Mbps
P_{coll1}^{DATA}	$P_1 + P_2 + P_4 + P_5$	$P_2 + P_4$
P_{coll2}^{DATA}	P_3	$P_1 + P_3 + P_5$
P_{coll1}^{ACK}	$P_2 + P_4 + P_5$	P_4
P_{coll2}^{ACK}	$P_1 + P_3$	$P_1 + P_2 + P_3 + P_5$

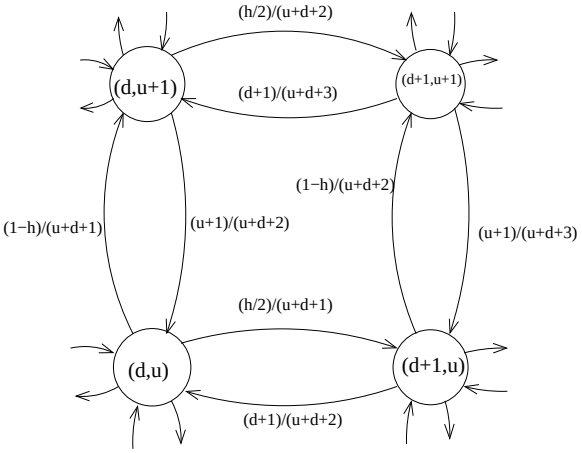

Fig. 5. Transition probability diagram for the process (D_k, U_k) for the case of delayed ACKs.

where $T_{TCP-DATA}$ and $T_{TCP-ACK}$ are the times taken for transmission of TCP data packet and TCP ACK respectively. P_{coll1}^{DATA} and P_{coll2}^{DATA} are the probabilities that the time spent in collision is T_{coll1}, and P_{coll1}^{ACK} and P_{coll2}^{ACK} are the probabilities that the time spent in collision is T_{coll2}. These values are summarized in Table II. Note that $T_{coll2} < T_{coll1}$ at 11Mbps and $T_{coll2} > T_{coll1}$ at 2Mbps and 5.5Mbps. Hence the time spent in collision when RTS and TCP ACK collide will be different depending on the PHY rate, consequently the time spent in collision state will be different for different PHY rates. Solving equations (13), (14), (15) we obtain $\mathsf{E}_{(d,u)}X$.

Substituting the value of $\mathsf{E}_{(d,u)}X$ and the expression for $\pi(d,u)$ (from (8)) into (11), we obtain the throughput Θ of the AP in packets/second. Recalling (6), the total throughput Θ_d (resp. Θ_u) for downloading (resp. uploading) STAs can then be obtained as

$$\Theta_d = h\Theta \ ; \ \Theta_u = (1-h)\Theta \tag{16}$$

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

III. 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 [7]; 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 Figure 5. The stationary distribution of this DTMC is

$$\pi(d,u) = \frac{u+d+1}{e^{1-\left(\frac{h}{2}\right)}\left(2-\frac{h}{2}\right)} \times \frac{\left(\frac{h}{2}\right)^d (1-h)^u}{d!u!} \tag{17}$$

for $0 \le d \le N_d$ and $0 \le u \le N_u$. The rest of the analysis follows along the same lines as in Section II. The aggregate AP throughput in packet per second is again given by (11). The total throughput for the downloading and uploading STAs is given by

$$\Theta_d = h\Theta \ \text{ and } \ \Theta_u = 2 * (1-h)\Theta \tag{18}$$

The factor 2 arises because each ACK transmitted from the AP acknowledged 2 data packets. The value of $E_{(d,u)}X$ is obtained in a manner similar to the undelayed ACK case.

IV. 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 this analyis is valid for the scenario when the STAs have TCP connections with a system located on the Ethernet to which the AP is connected, i.e., the delay between that system 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, and the window is reset to 1 after a timeout; we call this version "OldTahoe." TCP Reno is analyzed in the next section.

A. The Case of Undelayed ACKs

1) Modeling Assumptions and Approximations: a) From (9) and (10) 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. 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.

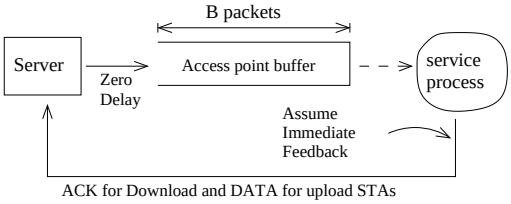

Fig. 6. Model for analyzing the AP buffer in order to obtain h.

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

$$b = \frac{\mathcal{B} - \mu L_{TCP-ACK}}{L_{TCP-DATA}} \tag{19}$$

We denote the capacity of the buffer in terms of packets by B which is given by

$$B = \mu + b \tag{20}$$

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

$$b = i \times N_d \tag{21}$$

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 for model for the AP buffer and the services applied to it; see Figure 6. Since the 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.

e) With the model in Figure 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

$$b = \sum_{i=1}^{N_d} W_i \tag{22}$$

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 VI-A.

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 large time intervals. The reason they ultimately do suffer a loss can be explained along the lines of the previous remark. 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 [11]. Later we will 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.

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 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 i^{th} 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 i^{th} download connection informs the receiver about the packet loss. The receiver returns a duplicate TCP

ACK. In the TCP version we are analyzing, we need a timeout 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 i^{th} download connection has a slow start threshold of $W_{th}^{i} = \frac{W_i}{2}$. Refering to (22), it can be that

$$\sum_{i=1}^{N_d} W_{th}^{(i)} = b/2 \tag{23}$$

For moderate values of b (the maximum number of download packets in the AP buffer) it has been observed (see Section VI-A) 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 has been done assuming all the window evolutions are synchronized.

The slow start phase: Denote by $T_{k,1}$ the instant when the k^{th} reset phase ends. The windows of the download transfers are modeled as evolving in cycles that start at instants $\{T_{k,1}, k \geq 0\}$. 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 j^{th} 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

$$\mu + 2^r N_d = \mu + N_d W_{th} = \mu + b/2 \tag{24}$$

yielding

$$r = \log_2 \left(\frac{b}{2N_d} \right) \tag{25}$$

TABLE III
CUMULATIVE WINDOW EVOLUTION FOR UNDELAYED ACK.

Phase	Buffer at $T_{k,i}$	Buffer at $T_{k,i+1}$	Buffer services in $[T_{k,i}, T_{k,i+1})$
slow start	$\mu + N_d$ $\mu + 2N_d$ $\mu + 4N_d$ .. $\mu + 2^{r-1}N_d$	$\mu + 2N_d$ $\mu + 4N_d$ $\mu + 8N_d$ .. $\mu + 2^{r}N_d$ $= \mu +$ **b/2**	$\mu + N_d$ $\mu + 2N_d$ $\mu + 4N_d$ .. $\mu + 2^{r-1}N_d$
cong. avoidance	$\mu + b/2$ $\mu + b/2 + N_d$ .. $\mu + b/2$ $+(x-1)N_d$	$\mu + b/2 + N_d$ $\mu + b/2 + 2N_d$ .. $\mu + b/2 + xN_d$ $=$ **B**	$\mu + b/2$ $\mu + b/2 + N_d$ .. $\mu + b/2$ $+(x-1)N_d$
losses	$\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$
reset	$\mu + b/2 + xN_d$ $\mu + b/2 + xN_d$	$\mu + b/2 + xN_d$ $\mu + N_d$	$\mu + b/2 + xN_d$ $\mu + b/2 + xN_d$

By the assumption of 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 will be 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 j^{th} 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 x^{th} 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 (21). Following this round, the loss phase of this cycle begins. This window evolution has been summarised in a compact form in Table III. Note that the number of upload packets transmitted by the AP is just μ in all the rounds.

Thus, in each cycle the evolution of the windows and the number of packets served is 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 III. From (7), h is the same as the value of this fraction. Thus, h is given by

$$h = \frac{\left[(2^r - 1) + \frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}}{(r+x+3)\mu + \left[(2^r - 1) + \frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}} \tag{26}$$

The value of h calculated using above procedure can be substituted into the analysis in section II 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 have thus 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

TABLE IV
CUMULATIVE WINDOW EVOLUTION FOR UNDELAYED ACK, TCP RENO

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

W_{max}. The space available for download packets in the AP buffer is b packets. Assume that $\lfloor \frac{b}{W_{max}} \rfloor$ connections reach their maximum window limit and stay there indefinitely. Then assume that remaining $N_d - \lfloor \frac{b}{W_{max}} \rfloor$ download connections follow the window evolution process described above with $b - W_{max} \lfloor \frac{b}{W_{max}} \rfloor$ space available for their packets in the AP buffer. Calculating h with this model will provide an upper bound on h.

B. The Case of Delayed ACKs

The above analysis for calculating h can be easily extended to the delayed ACK case. Now it can be assumed that $B = \mu/2 + b$ as every alternate data packet is acknowledged for the upload connections. Replacing μ by $\mu/2$ in the analysis in the previous section, we obtain

$$h = \frac{\left[(2^r - 1) + \frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}}{(r+x+3)\frac{\mu}{2} + \left[(2^r - 1) + \frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}} \quad (27)$$

V. 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 IV-A. For the Reno case we assume that there are no timeouts and the recovery uses only Fast Retransmit and Fast Recovery mechanisms ([6]). We assume that there are sufficient number of packets buffered for every download connection so as to trigger the Fast Retransmit mechanism. Note that 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 akin 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 cumulative download window evolution is summarized in the Table IV. The formula for h in the case of Reno becomes:

$$h = \frac{\left[\frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}}{(x+3)\mu + \left[\frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}} \quad (28)$$

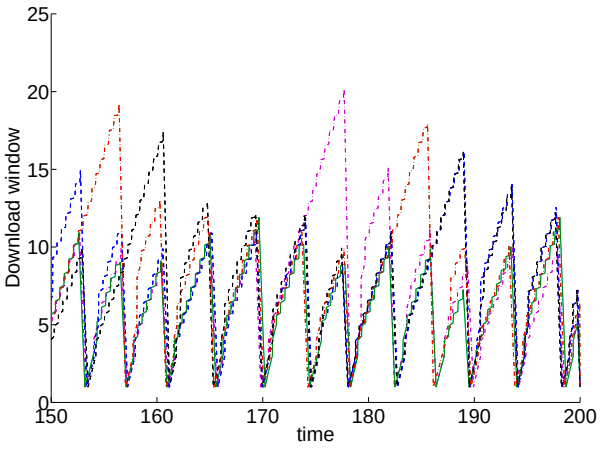

Fig. 7. Sample path of window evolutions of several connections, in support of our assumption of synchronization of the window evolution processes.

Following along similar lines as in Section IV-B, h for TCP Reno for the delayed ACK case is given by

$$h = \frac{\left[\frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}}{(x+3)\frac{\mu}{2} + \left[\frac{x(x-1)}{2} + 3x\right] N_d + (x+3)\frac{b}{2}} \quad (29)$$

VI. SIMULATION RESULTS

All the simulation results are obtained using ns-2.31.

A. Synchronized Window Evolution

Figure 7 shows a window evolution snapshot for $N_u = 5, N_d = 5, b = 50$ in support of the assumptions and approximations made for the finite buffer analysis in Section IV-A. We have assumed in Section IV-A1 that the window evolution is synchronized. From Figure 7 we 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 STAs 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.

B. TCP OldTahoe

Figures 8 and 9 provide a validation of the analysis performed in Section IV-A. 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 50 TCP ACKs. It can be seen that the analysis

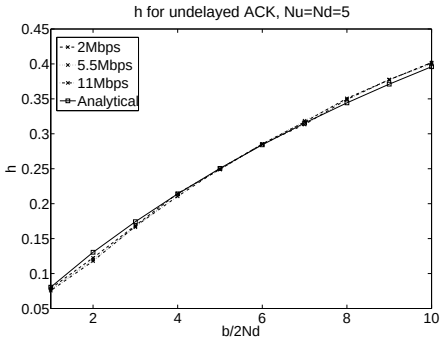

Fig. 8. TCP OldTahoe, Undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$

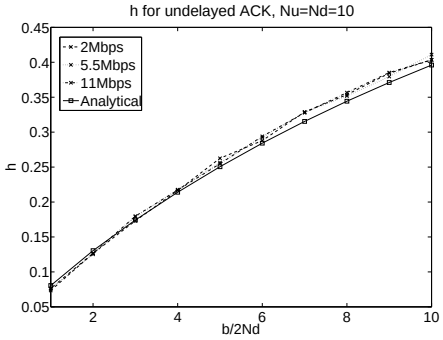

Fig. 9. TCP OldTahoe, Undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 10$

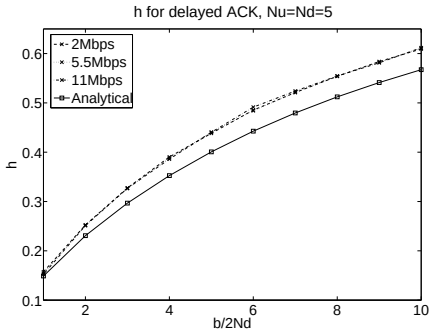

Fig. 10. TCP OldTahoe, Delayed ACKs: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$

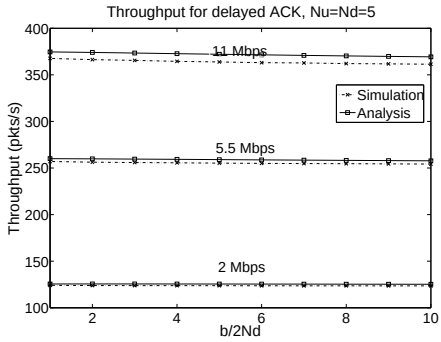

Fig. 11. TCP OldTahoe, Delayed ACK: $\Theta_d + \Theta_u$ for $N_u = N_d = 5$

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.

The upload and download throughputs are obtained by multiplying the aggregate packet throughput from the AP by h; see (16). For PHY rates of 2Mbps, 5.5Mbps 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.

Discussion: As an illustration, we see that if $\frac{b}{2N_d} = 10$ with $N_u = N_d = 5$ (see Figure 8), and, keeping the same buffer, we make $N_d = N_u = 10$ (see Figure 9), then the download throughput will drop from about 40% of aggregate throughput to about 25%.

Figure 10 shows simulation results for the analysis for the delayed ACK case in Section IV-B. The analytical values of h were tested against the simulated values for $N_u = N_d = 5$. Nearly identical results are obtained for $N_u = N_d = 10$, for example. The deviation of the analytical value from the simulation is more in this case as while calculating h we have considered an upper bound on upload throughput thus overestimating the upload throughput. Hence, we underestimate the download throughput, consequently reducing the value of h.

Figure 11 shows the results obtained for the value of $\Theta_d + \Theta_u$ for $N_u = N_d = 5$; almost identical results are obtained for $N_u = N_d = 10$.

Discussion: It can be seen from (18) that the upload and download throughputs are equal when $h = \frac{2}{3}$. We see from Figure 10 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 (Figure 11) is almost constant with buffer size, and the bounding approach we took in our analysis in Section IV-B is seen to yield a very good approximation.

C. TCP Reno

Figure 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 larger than with OldTahoe, since the download connection windows do not drop drastically due to timeouts. Again similar to OldTahoe, for PHY rates of 2Mbps, 5.5Mbps 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.

Figures 13 shows the simulation and analysis results obtained for h with $N_u = N_d = 5$ for TCP Reno with delayed ACKs. The plots of $\Theta_d + \Theta_u$ for $N_u = N_d = 5$ are almost identical to Figure 11. Similar results were obtained for $N_u = N_d = 8, 10$.

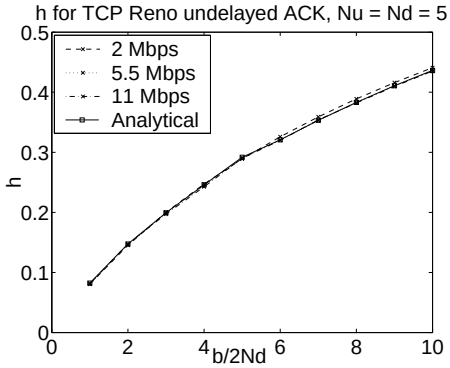

Fig. 12. TCP Reno, Undelayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$

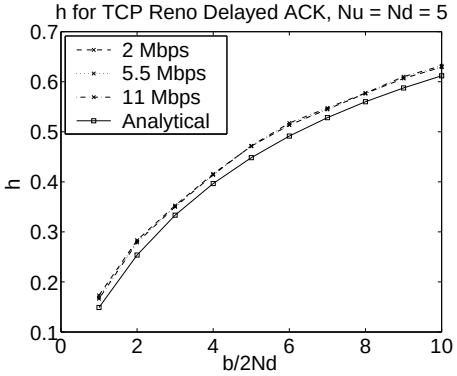

Fig. 13. TCP Reno, Delayed ACK case: h vs. buffer size (expressed as $\frac{b}{2N_d}$) for $N_u = N_d = 5$

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

All the simulation results in Section VI-B 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 IV-A2 we provided a simple upper bound on h for the case of undelayed ACK and when all connections have a maximum window limit. Intuitively, we anticipate that for small values of the AP buffer, the h for such case would be close to the h obtained with no maximum window limit only on download connections. In Figure 14, for the case $N_u = N_d = 5$, we have plotted the upper bound on h, 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. These figures supports the claims we have made. Nearly identical numerical results are obtained for other values of N_u and N_d, such as 8 and 10.

VII. 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 same value of h even in the scenario where STAs associate with

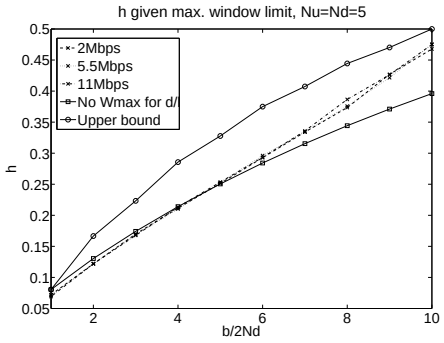

Fig. 14. TCP OldTahoe, Undelayed ACK: h for $N_u = N_d = 5$,$W_{max} = 20$

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 IV-A1, and has no other dependence on the underlying MAC layer analysis.

We have thus provided a fairly general analytical model that (i) explains the observations made by several prior experimental and simulation studies (e.g., [3]), (ii) yields several new insights into the interaction of the TCP protocol and the IEEE 802.11 MAC (e.g., beyond those in [2] and [7]), and (iii) provides an accurate model that could be used to predict performance, perhaps for the purpose of network engineering.

References

[1] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, pp. 535–547, March 2000.

[2] R. Bruno, M. Conti, E. Gregori, "Modeling TCP Throughput Over Wireless LANs," *Proc. 17th IMACS World Congress Scientific Computation,Applied Mathematics and Simulation*, Paris, France, July 11 - 15, 2005

[3] Mingwei Gong, Qian Wu, Carey Williamson, "Queue management strategies to improve TCP fairness in IEEE 802.11 wireless LANs,"*The 2nd Workshop on Resource Allocation in Wireless NETworks, RAWNET*, Boston, MA, April 3, 2006.

[4] V. G. Kulkarni, *Modeling and Analysis of Stochastic Systems*, Chapman and Hall, London, UK, 1995.

[5] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New Insights from a Fixed Point Analysis of Single Cell IEEE 802.11 WLANs," in *Proceedings of IEEE INFOCOM '05*, 13 - 17 March 2005, pp. 1550 – 1561.

[6] Anurag Kumar, D. Manjunath, Joy Kuri, *Communication Networks: An Analytical Approach*, Morgan Kaufman Networking Series, Elsevier 2004.

[7] George Kuriakose, Sri Harsha, Anurag Kumar, Vinod Sharma, "Analytical models for capacity estimation of IEEE 802.11 WLANs using DCF for Internet applications ," *Wireless Networks*, to appear (online version appeared August 2007).

[8] Juho Ha, Chong-Ho Choi, "TCP Fairness in Uplink and Downlink Flows in WLANs," in *IEEE GLOBECOM 2006.*

[9] Saar Pilosof, Ramachandran Ramjee, Danny Raz, Yuval Shavit, Prasun Sinha, "Understanding TCP Fairness Over Wireless LAN", in *Proceedings of IEEE INFOCOM'03.*

[10] Sri Harsha, Anurag Kumar, Vinod Sharma, "An Analytical Model for the Capacity Estimation of Combined VoIP and TCP File Transfers over EDCA in an IEEE 802.11e WLAN," *14th IEEE International Workshop on Quality of Service (IWQoS)*, Yale University, New Haven, June 2006.

[11] RFC 1323, http://www.ietf.org/rfc/rfc1323.txt