TCP Dynamics over IEEE 802.11e WLANs: Modeling and Throughput Enhancement

Jeonggyun Yu, Sunghyun Choi
School of Electrical Engineering and INMC
Seoul National University, Seoul, 151-744, Korea
jgyu@mwnl.snu.ac.kr, schoi@snu.ac.kr

Daji Qiao
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011
daji@iastate.edu

***Abstract*— Today, IEEE 802.11 Wireless LAN (WLAN) has become a prevailing solution for broadband wireless Internet access while Transport Control Protocol (TCP) is the dominant transport protocol in the Internet. It is known that, in an infrastructure-based WLAN with multiple stations carrying *long-lived* TCP flows, the number of stations that are actively contending to access the channel is very small. Therefore, the aggregate TCP throughput is basically independent of the total number of stations. This phenomenon is due to the closed-loop nature of TCP flow control and the bottleneck downlink (i.e., AP-to-station) transmissions in infrastructure-based WLANs.**

In the emerging Enhanced Distributed Channel Access (EDCA)-based IEEE 802.11e WLANs, with a proper configuration, packet congestion at the bottleneck downlink could be alleviated since the AP and stations are allowed to use different channel access parameters. In this paper, we first conduct a rigorous, comprehensive analysis of the TCP dynamics over the 802.11e EDCA. Then, the effects of minimum contention window sizes (of both AP and stations) on the aggregate TCP throughput are evaluated via mathematical analysis and simulation. We also show that the best TCP aggregate throughput performance can be achieved via AP's contention-free access for downlink packet transmissions. Finally, some of the simplifying assumptions used in our mathematical model are evaluated via simulation, and results show that our model is reasonably accurate when the wireline delay is small and the packet loss rate is low.

I. Introduction

In recent years, IEEE 802.11 WLAN has become the dominant technology for (indoor) broadband wireless networking. The IEEE 802.11 standard [1] specifies the protocols for the Medium Access Control (MAC) layer and the Physical (PHY) layer. The mandatory protocol of the 802.11 MAC is the Distributed Coordination Function (DCF), which is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).

As pointed out in [6], more than 90% of the Internet traffic is carried by the Transport Control Protocol (TCP). Consequently, most of data traffic over the 802.11 WLAN is TCP traffic. Therefore, it is critical to have a good understanding of the TCP dynamics over WLAN. It has been shown in [10], [11] that, with multiple stations in an infrastructure-based 802.11 WLAN, each carrying *long-lived* TCP flows, the number of stations that are actively contending is very small. Hence, the aggregate TCP throughput is almost independent of the total number of stations in the network. This interesting phenomenon is due to (i) the closed-loop nature of TCP flow control and (ii) the bottleneck downlink (i.e., AP-to-station) transmissions in infrastructure-based WLANs. More specifically, since a long-lived upload TCP flow typically operates in the congestion avoidance phase, the source station can only transmit a TCP Data after it receives a TCP Ack from the AP. In the meanwhile, each client station of a download TCP flow only replies with a TCP Ack upon reception of a TCP Data packet from the AP. On the other hand, the 802.11 DCF guarantees long-term equal channel access probabilities among contending stations in the network, regardless whether it is an AP or a station. As a result, most of the outstanding TCP packets (in the form of either TCP Ack for upload or TCP Data for download) are accumulated at the AP, while most of the stations remain *inactive* waiting for their turns to transmit.

The above observations do not always hold in the emerging IEEE 802.11e WLANs, because the 802.11e MAC supports service differentiation. The 802.11e MAC defines an enhanced contention-based channel access mechanism, called EDCA (Enhanced Distributed Channel Access). With EDCA, the AP can perform a service differentiation between itself and stations as well as among stations with different classes of traffic, thanks to its capability of setting flexible channel access parameters. For example, if the AP uses a smaller minimum contention window size than stations, the aforementioned TCP packet accumulation at the AP may be alleviated.

There have been efforts to understand the TCP dynamics over legacy 802.11 WLANs [11]–[15]. However, none of them analyzed the TCP dynamics over 802.11e WLANs. In [11], Choi $et\ al.$ analyzed the number of active stations using a Markov chain, but simply computed the state transition probabilities based on empirical results. In [12], Kherani $et\ al.$ conducted a theoretical analysis on the TCP performance over multi-hop 802.11 WLANs, while our study focuses on infrastructure-based WLANs. Burmeister $et\ al.$ analyzed the TCP over multi-rate WLANs [13]. However, they approximated the AP behavior as an M/G/1/B queuing system, and did not study the effects of contention-based MAC on TCP dynamics in detail. Bruno $et\ al.$ addressed this problem using

The research reported in this paper was supported in part by the Korea Research Foundation Grant funded by Korean Government (MOEHRD) (R08-2004-000-10384-0), Seoul R&BD Program (10544), the Information Infrastructure Institute (I-Cube) of Iowa State University, and the U.S. National Science Foundation under Grant No. CNS 0520102.

a p-persistent DCF model [14] under a critical assumption that no more than one TCP packet is enqueued in the buffer of an active station. This implies that a station becomes inactive after transmitting its packet successfully while each successful transmission from the AP always activates a different station, which is not true in practice. In a real network, the number of outstanding packets in a long-lived TCP flow is usually larger than one. Moreover, each successful transmission from the AP may not always activate a different station, because multiple outstanding TCP packets belonging to the same flow could be accumulated at the AP at the same time.

In this paper, we conduct a rigorous and comprehensive analysis of the TCP dynamics over EDCA-based IEEE 802.11e WLANs. Our model is based on the p-persistent model developed by Cali $et\ al.$ [8]. We study the effects of minimum contention window (CWmin) sizes of both AP and stations on the aggregate TCP throughput, and verify it using numerical and simulation results. We show that the aggregate TCP throughput can be enhanced via a proper combination of CWmin values for the AP and stations. Moreover, we observe that the best aggregate throughput performance can be achieved via AP's contention-free access for downlink packet transmissions.

The remainder of the paper is organized as follows. In Section II, IEEE 802.11e EDCA, the p-persistent CSMA/CA model, and the dynamics of long-lived TCP flows are briefly discussed. Section III describes our analytical model for TCP over the 802.11e EDCA. AP's contention-free access for downlink TCP packet transmissions is presented in Section IV. In Section V, our analytical model is evaluated via simulation, and the effects of AP/station access parameters on the aggregate TCP throughput are studied. We also discuss briefly the effects of various assumptions on the accuracy of our model. Finally, we conclude the paper in Section VI.

II. Preliminaries

A. IEEE 802.11e EDCA

The 802.11 DCF [1] does not support service differentiation. Basically, the DCF provides long-term fair channel access to contending stations with equal opportunities. In contrast, the EDCA [3] provides four access categories (ACs) and the channel access function of each AC is an enhanced variant of the DCF. More specifically, each EDCA channel access function uses AIFS[AC], CWmin[AC], and CWmax[AC] instead of DIFS, CWmin, and CWmax in the DCF, respectively, for the contention to transmit a packet belonging to the corresponding AC. AIFS[AC] is determined by

$$AIFS[AC] = SIFS + AIFSN[AC] \cdot SlotTime, \quad (1)$$

where AIFSN[AC] is an integer greater than 1 for stations and an integer greater than 0 for APs. The backoff counter is selected randomly from [0, CW[AC]]. The values of AIFSN[AC], CWmin[AC], and CWmax[AC], which are referred to as the EDCA parameter set, are advertised by the AP via Beacons and Probe Response frames. Basically, if an EDCA channel access function uses smaller AIFSN[AC] and CWmin[AC], it contends for the channel more aggressively. Hence, it may use more bandwidth than other ACs with larger AIFSN[AC] and CWmin[AC]. In this work, we study how the aggregate TCP throughput performance may be enhanced by using different EDCA parameter sets for the AP and stations. Service differentiation among different traffic priorities is not the focus of this paper.

In an 802.11e EDCA WLAN, after a station transmits a packet successfully, it resets its CW value to CWmin, and performs AIFS deference and random backoff. This is often referred to as *post backoff* since this backoff is done after, not before, a transmission. Post backoff ensures that there is at least one backoff interval between two consecutive frame transmissions. On the other hand, when there is no on-going backoff and when the channel has been idle for longer than AIFS time, a frame can be transmitted immediately without additional backoff. This is referred to as *immediate access*.

B. p-persistent CSMA/CA Model

Cali $et\ al.$ [8] studied the DCF behavior with a p-persistent model. Instead of using the binary exponential backoff, the p-persistent model determines the backoff interval by sampling from a geometric distribution with parameter p. Thanks to the memoryless property of the geometric distribution, it is more tractable to analyze the p-persistent model. Based on the geometric backoff assumption, the processes that define the occupancy behavior of the channel (i.e., idle slots, collisions, and successful transmissions) are regenerative, where the regenerative points correspond to completions of successful transmissions. As shown in Fig. 1, the v-th virtual slot, denoted by μ_v, consists of the v-th successful transmission, as well as the preceding idle periods and collision periods. An idle period is a time interval during which all the backlogged stations are performing backoff without transmission attempts. A collision period is the interval during which two or more stations transmit simultaneously and collide with each other.

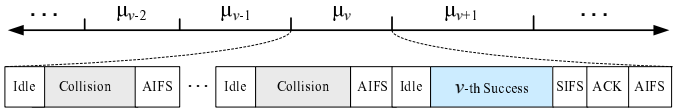


Fig. 1. Illustration of virtual slots (μ) in the p-persistent model.

Statistically, the standard DCF/EDCA operations and their p-persistent models are equivalent when the stations are always active (i.e., when they always have a frame to transmit). However, when a station carries long-lived TCP flows, it may not always have a frame to transmit. For example, during the congestion avoidance phase, such a station is allowed to transmit an additional TCP Data only when it receives a TCP Ack for one of its outstanding TCP Data packets. Therefore, when analyzing the TCP dynamics over WLAN, there are slight differences between the standard DCF/EDCA operations and their corresponding p-persistent models. Nevertheless, our analysis is based on p-persistent models for simplicity, and the accuracy of the analysis will be evaluated in Section V.

C. Dynamics of Long-lived TCP Flows

A TCP flow operates in either slow-start or congestion avoidance phase irrespective of the TCP version. The slow-start phase occurs during startup as well as when a timeout occurs at the sender due to packet loss. In this paper, we consider long-lived TCP flows so that we may ignore the effects of slow start since its fraction in the flow lifetime is very small.

In an earlier version of TCP, known as *TCP Tahoe*, the sender resets its congestion window to 1 Maximum Segment Size (MSS) and enters the slow start phase after each packet loss. This deficiency was corrected in later TCP versions such as *TCP NewReno*, which has been implemented in Microsoft Windows XP. The key difference between TCP NewReno and TCP Tahoe is the addition of the *Fast Retransmit* and *Fast Recovery* mechanisms [4]. Fast Retransmit prevents the timeout by retransmitting a packet when three duplicate ACKs are received at the sender. Fast Recovery cancels the slow start phase by setting the congestion window to approximately half its current value and thus keeping the connection in the congestion avoidance phase. Although bursts of packet losses may still cause timeouts at the sender, and subsequently forcing slow start to be invoked, TCP NewReno recovers from slow start much faster than earlier versions of TCP [5].

Because the current 802.11 WLAN has a smaller bandwidth compared to the wireline network, WLAN is very likely to be the bottleneck in the round trip path of TCP traffic. Therefore, most of the outstanding TCP packets stay in WLAN for most of their life time. Moreover, in the absence of frequent or bursty packet losses in the wireline network, the TCP send window may grow up to the maximum window size allowed by the receiver.

III. Analytical Model for TCP Dynamics over 802.11e EDCA

In this section, we analyze the average number of active stations and the aggregate TCP throughput in an 802.11e EDCA WLAN.

A. Network Model and Assumptions

We consider an infrastructure-based 802.11e WLAN with a single AP and N_{STA} stations, each carrying long-lived TCP flows. Stations either upload data to or download from FTP servers. We conduct our analysis with the following assumptions for simplicity and the effects of some of the assumptions will be studied in Section V.

- [A1] The queue size of the AP and stations is large enough not to incur buffer overflow.
- [A2] A successful transmission of each TCP Data is notified by an immediate TCP Ack instead of the delayed TCP Ack.
- [A3] As mentioned in Section II-C, long-lived TCP flows often operate in the congestion avoidance phase during most of their lifetime as long as packet loss does not occur in bursts frequently. In our model, we assume ideal channel conditions (i.e., no hidden terminals, no channel error, and no capture effect) but will study the effects of packet loss on the aggregate TCP throughput in Section V-D.
- [A4] In practice, the TCP receive window size is a small number, typically smaller than the TCP congestion window size during the congestion avoidance phase. We also know that the number of outstanding packets is determined by the minimum of the TCP congestion window size and the TCP receive window size. Based on this observation and [A3], we assume that the number of outstanding packets (including TCP Data and TCP Ack) for a TCP flow is equal to the maximum TCP receive window size (in packets).[1]
- [A5] There is no TCP Ack processing delay between a TCP Data reception and the corresponding TCP Ack transmission. We will study the effect of the TCP Ack processing delay on the aggregate TCP throughput in Section V-C.

TABLE I
List of notations and symbols

Term	Definition
W	Maximum TCP receive window size (in packets)
M	Total number of states
N_{STA}	Total number of TCP stations in an infrastructure WLAN
$\overrightarrow{N}_k$	*State k* vector
N_k^i	i-th element of the *State k* vector
η_k	Number of active TCP stations in *State k*
$P_{m,k}$	State transition probability from *State m* to *State k*
Q_k^{AP}	Length of the AP queue in *State k*
$P_{STA,k}^{succ}$	Probability that a non-AP station succeeds in a transmission attempt in *State k*
$P_{AP,k}^{succ}$	Probability that the AP succeeds in a transmission attempt in *State k*
$P_{AP,k}^{i}$	Probability that the AP succeeds in transmitting a packet to a class-i station in *State k*.
$P_{STA,k}^{i}$	Probability that a class-i station succeeds in transmitting a packet in *State k*

B. State Space

Table I lists the notations and symbols used in the following analysis. We use a discrete-time Markov chain to model the distribution of the numbers of stations carrying different numbers of TCP packets in their queues at the end of the *v*-th virtual slot.

$\overrightarrow{N}(v) = (N^0(v), N^1(v), N^2(v), \ldots, N^W(v))$ is the state vector, where $N^i(v)$ represents the number of stations, each carrying i TCP packets in its queue at the end of the v-th virtual slot. We call such stations *class-i* stations. Hence, $N^0(v)$ is the number of stations that have no TCP packets in their queues. In other words, $N^0(v)$ is the number of

[1]The TCP window sizes are actually maintained in bytes. In our model, for simplicity, we assume that the TCP Data packets have a fixed size and are transmitted at a fixed rate. Hence, the TCP window sizes can be measured in packets. Similar analysis could be done for variable packet sizes and multi-rate WLANs, which is omitted due to space limitation.

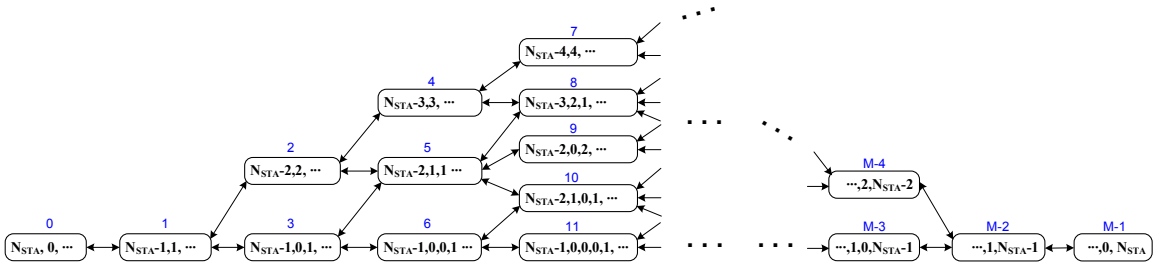


Fig. 2. Our markov chain model to study TCP dynamics

inactive stations. W is the maximum TCP receive window size, or equivalently, the maximum number of outstanding TCP packets under assumption [A4].

Our Markov chain is shown in Fig. 2 with the index of each state shown as the number above it. At *State k*, the following equation always holds:

$$N_{STA} = \sum_{i=0}^{W} N_k^i,$$

where N_k^i is the i-th element of the *State k* vector. N_k^0 is hence the number of inactive stations at *State k*. Consequently, the length of the AP queue Q_k^{AP} and the number of active TCP stations η_k at *State k* can be calculated by

$$\begin{cases} Q_k^{AP} \triangleq \sum_{j=0}^{W} N_k^j \cdot (W-j), \\ \\ \eta_k \triangleq N_{STA} - N_k^0. \end{cases}$$

Let M be the total number of states and the steady-state probability of the Markov chain is

$$\pi_k = \lim_{v\to\infty} \mathrm{P}\left\{\overrightarrow{N}(v) = \overrightarrow{N}_k\right\}, k \in [0,\, M-1].$$

In this Markov chain, the state transition occurs only at the end of each virtual slot, which ends with a successful transmission. A transition from a right state to a left state in Fig. 2 is caused by a station's successful transmission, and a transition from a left state to a right state is caused by a successful transmission by the AP. For example, at *State 2*, the number of active stations (including the AP) is three. Each of the two non-AP stations has exactly one TCP packet in its queue. If one of them ends the current virtual slot with a successful transmission, the transition from *State 2* to *State 1* occurs. On the other hand, if the AP ends the current virtual slot with its successful transmission, there are two possible cases. The first case is that the AP transmits a TCP packet to one of the $(N_{STA}-2)$ inactive stations. In such a case, the number of active stations becomes 2 + 1 = 3, hence the transition from *State 2* to *State 4* occurs. The second case is that the AP transmits a TCP packet to one of the two active stations, each already has one TCP packet in its queue. As a result, the number of active stations remains the same, while the station that just received the TCP packet from the AP is allowed to add one more packet to its queue. This means that the transition from *State 2* to *State 5* has occurred.

C. State Transition Probabilities

Now, we derive the state transition probabilities of all possible transitions. As discussed above, during each transition, two and only two elements of the state vector change their values. Each of them changes by one. This is because state transitions are triggered by a single successful transmission from either the AP or a station. Therefore, the transition probability from *State k* to *State m* is given by

$$P_{k,m} = \begin{cases} P_{AP,k}^i, & \text{if } \exists \text{ a unique } i \in [0,\, W-1] \text{ such that} \\ & N_m^i = N_k^i - 1,\, N_m^{i+1} = N_k^{i+1} + 1, \\ \\ P_{STA,k}^i, & \text{if } \exists \text{ a unique } i \in [0,\, W-1] \text{ such that} \\ & N_m^i = N_k^i + 1,\, N_m^{i+1} = N_k^{i+1} - 1, \\ \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The first equation in Eq. (2) accounts for the fact that the AP ends the current virtual slot with its successful transmission. $P_{AP,k}^i$ is the probability that the AP successfully transmits its TCP packet to one of the class-i stations. The second equation accounts for the fact that a station ends the current virtual slot with its successful transmission. $P_{STA,k}^i$ is the probability that a class-$(i+1)$ station succeeds in transmitting its TCP packet.

We assume that the AP and a station access the medium with attempt probabilities τ_{AP} and τ_{STA}, respectively, where these values are derived in Section III-F. Then, the probability $P_{AP,k}^i$ is given by

$$P_{AP,k}^i \;=\; P_{AP,k}^{succ} \cdot P_k^{AP(i)}, \tag{3}$$

where

$$\begin{aligned} P_{AP,k}^{succ} &= P\left(\text{AP success} \,|\, \text{a tx success at } State\ k\right) \\ &= \begin{cases} \frac{\tau_{AP}\cdot(1-\tau_{STA})^{\eta_k}}{\tau_{AP}\cdot(1-\tau_{STA})^{\eta_k}+\eta_k\cdot\tau_{STA}(1-\tau_{STA})^{\eta_k-1}\cdot(1-\tau_{AP})}, \\ \qquad\qquad\qquad\qquad N_k^W < N_{STA}, \\ \\ 0, \qquad\qquad\qquad\qquad N_k^W = N_{STA}, \end{cases} \end{aligned}$$

and

$$\begin{aligned} P_k^{AP(i)} &= P\left(\text{AP xmit to a class}-i \text{ STA} \,|\, \text{AP success}\right) \\ &= \begin{cases} N_k^i\,(W-i)\,/Q_k^{AP}, & Q_k^{AP} > 0, \\ \\ 0, & Q_k^{AP} = 0. \end{cases} \end{aligned}$$

Similarly, the probability $P^{i}_{STA,k}$ is given by

$$P^{i}_{STA,k} = P^{succ}_{STA,k} \cdot P^{STA(i+1)}_{k}, \tag{4}$$

where

$$\begin{aligned} P^{succ}_{STA,k} &= P\left(\text{STA success} \,|\, \text{a tx success at } State\ k\right) \\ &= 1 - P^{succ}_{AP,k}, \end{aligned}$$

and

$$\begin{aligned} P^{STA(i+1)}_{k} &= P\left(\text{A class}-(i+1)\ \text{STA success} \mid \text{STA success}\right) \\ &= \begin{cases} \frac{N^{i+1}_{k}}{N_{STA}-N^{0}_{k}}, & N^{0}_{k} < N_{STA}, \\ 0, & N^{0}_{k} = N_{STA}. \end{cases} \end{aligned}$$

D. Average Number of Active Stations

Since the length of the virtual slot (i.e., the sojourn time) in each state varies, the average number of active stations (excluding the AP) is

$$E\left[N_{active}\right] = \sum\nolimits_{k=0}^{M-1} \eta_k p_k, \tag{5}$$

where p_k is given by

$$p_k = \pi_k \mu_k \Big/ \left(\sum\nolimits_{j=0}^{M-1} \pi_j \mu_j\right),$$

where μ_k is the sojourn time at *State k*. In the case of download TCP flows, μ_k is given by

$$\begin{aligned} \mu_k \ &= E\left[N^{col}_{STA,k}\right] T^{ack}_{col} + E\left[N^{col}_{AP,k}\right] T^{data}_{col} \\ &\quad + \left(E\left[N^{col}_{k}\right] + 1\right)\left(AIFS + E\left[T^{idle}_{k}\right]\right) \\ &\quad + E\left[T^{data}_{k}\right] + SIFS + ACK. \end{aligned} \tag{6}$$

Here, T^{ack}_{col} and $E\left[N^{col}_{STA,k}\right]$ are, respectively, the collision time and the average number of collisions due to contention among uplink TCP Ack packets only. T^{data}_{col} and $E\left[N^{col}_{AP,k}\right]$ are, respectively, the collision time and the average number of collisions due to contention between a downlink TCP Data and uplink TCP Acks. Moreover, $E\left[T^{data}_{k}\right]$ and ACK are the average TCP packet transmission time and MAC-layer ACK frame transmission time, respectively. $E\left[N^{col}_{k}\right]$ and $E\left[T^{idle}_{k}\right]$ are the average number of collisions and the average idle time preceding a collision or a successful transmission, respectively. On the other hand, in the case of upload TCP, μ_k can be calculated by replacing T^{ack}_{col} by T^{data}_{col} in Eq. (6), because at least one of the colliding packets is a TCP Data, hence the collision time is always T^{data}_{col}.

In the following, we derive individual elements used in Eq. (6). $E\left[N^{col}_{k}\right]$, $E\left[N^{col}_{AP,k}\right]$ and $E\left[N^{col}_{STA,k}\right]$ can be expressed by

$$\begin{cases} E\left[N^{col}_{k}\right] = \sum_{i=0}^{\infty} i \cdot P\left(N^{col}_{k} = i\right), \\ E\left[N^{col}_{AP,k}\right] = \sum_{i=0}^{\infty} j \cdot P\left(N^{col}_{AP,k} = j\right), \\ E\left[N^{col}_{STA,k}\right] = E\left[N^{col}_{k}\right] - E\left[N^{col}_{AP,k}\right], \end{cases} \tag{7}$$

where $P\left(N^{col}_{k} = i\right)$ and $P\left(N^{col}_{AP,k} = j\right)$ are the probability that the total number of collisions is i and the probability that the number of collisions due to the AP's transmissions is j, respectively, which are given by

$$\begin{cases} P\left(N^{col}_{k} = i\right) = \left(P^{col}_{k}\right)^{i} P^{succ}_{k}, \\ P\left(N^{col}_{AP,k} = j\right) = \left(P^{col}_{AP,k}\right)^{j} P^{succ}_{k}. \end{cases} \tag{8}$$

P^{succ}_{k} is the conditional successful transmission probability at *State k*, assuming that at least one station (including the AP) transmits, which can be calculated by

$$\begin{aligned} P^{succ}_{k} &= \mathrm{P}\left(N^{tx}_{k} = 1 \,\middle|\, N^{tx}_{k} \geqslant 1\right) \\ &= \begin{cases} \frac{\tau_{AP}(1-\tau_{STA})^{\eta_k} + \eta_k \tau_{STA}(1-\tau_{STA})^{\eta_k - 1}(1-\tau_{AP})}{1-(1-\tau_{AP})(1-\tau_{STA})^{\eta_k}}, \\ \qquad\qquad N^{W}_{k} < N_{STA}, \\ \frac{\eta_k \cdot \tau_{STA}(1-\tau_{STA})^{\eta_k - 1}}{1-(1-\tau_{STA})^{\eta_k}}, \qquad N^{W}_{k} = N_{STA}. \end{cases} \end{aligned}$$

Here, N^{tx}_{k} denotes the number of transmitting stations at *State k*. P^{col}_{k} is the conditional collision probability, assuming that at least one station transmits, which is $P^{col}_{k} = 1 - P^{succ}_{k}$. Moreover, $P^{col}_{AP,k}$ is the conditional probability that AP's transmission fails, assuming that at least one station (including the AP) transmits, which is given by

$$P^{col}_{AP,k} = \begin{cases} \frac{\tau_{AP}(1-(1-\tau_{STA})^{\eta_k})}{1-(1-\tau_{AP})(1-\tau_{STA})^{\eta_k}}, & N^{W}_{k} < N_{STA}, \\ 0, & N^{W}_{k} = N_{STA}. \end{cases}$$

The average idle time preceding a collision or a successful transmission in *State k* can be calculated as

$$\begin{aligned} E[T^{idle}_{k}] &= SlotTime \cdot \sum_{i=0}^{\infty} i \cdot P\left(N^{tx}_{k} > 0\right) \cdot \left(P\left(N^{tx}_{k} = 0\right)\right)^{i} \\ &= SlotTime \cdot \frac{(1-\tau_{AP})(1-\tau_{STA})^{\eta_k}}{1-(1-\tau_{AP})(1-\tau_{STA})^{\eta_k}}. \end{aligned}$$

In the case of download TCP, the average TCP packet transmission time can be expressed as

$$E\left[T^{data}_{k}\right] = T_{TCP\,DATA} \cdot P^{succ}_{AP,k} + T_{TCP\,ACK} \cdot P^{succ}_{STA,k}, \tag{9}$$

where $T_{TCP\,DATA}$ and $T_{TCP\,ACK}$ are TCP Data and TCP Ack transmission durations, respectively. In the case of upload TCP, the positions of $P^{succ}_{AP,k}$ and $P^{succ}_{STA,k}$ are exchanged in Eq. (9).

E. Aggregate TCP Throughput

In the p-persistent CSMA/CA model [8], the aggregate throughput (in bits/second) is given by

$$\rho = \frac{\text{Average length of a TCP Data (bits)}}{\text{Average length of a virtual slot (seconds)}}.$$

In our analysis, with the assumption that there exist a fixed number of active stations in a given state, the download TCP throughput at *State k* can be derived by

$$\rho_k = \frac{L_{TCP\,DATA} \cdot P^{succ}_{AP,k}}{\mu_k}, \tag{10}$$

where $L_{TCP\,DATA}$ is the TCP Data size and μ_k is the sojourn time at *State k*. Finally, the average aggregate TCP throughput can be calculated by

$$\rho_{TCP} = \sum_{i=0}^{M-1} \rho_i \cdot p_i. \quad (11)$$

For the upload TCP throughput, $P_{AP,k}^{succ}$ is replaced by $P_{STA,k}^{succ}$ in Eq. (10). For the mixed case of upload and download, we can calculate the throughput by modifying Eqs. (6) and (10), and the calculation details are omitted due to space limitation.

F. Estimation of τ from CW_{min} at Each State

In order to obtain the medium access probabilities τ_{AP} and τ_{STA}, we extend the Average Contention Window Estimation algorithm in [8] by considering different CWmin sizes for the AP and stations. We assume that τ is $1/(\overline{B}+1)$ where $\overline{B}$ is the average number of backoff slots, and is equal to $\frac{1}{2}\left(\overline{CW}+1\right)$ [8]. Because the average contention window size $\overline{CW}$ is determined by CWmin and the number of active stations in a given state, we can estimate τ from CWmin in each state using an iterative algorithm. The algorithm used in this work is based on and an improvement upon the one we proposed in [15] by considering different CWmin sizes for the AP and stations.

IV. AP PIFS Access

As mentioned in Section II-A, AIFSN[AC] is an integer greater than 1 for STAs and an integer greater than 0 for APs. Moreover, the values of CWmin[AC] and CWmax[AC] can be set to zero. Therefore, for the AP, we can use access parameter values of AIFSN[AC] = 1, CWmin[AC] = 0, and CWmax[AC] = 0, which are the smallest access parameter values, for downlink TCP packet transmissions from the AP. This allows the AP to transmit its pending TCP packets after a PIFS channel idle time without backoff. This scheme is referred to as *PIFS Access* in the rest of this paper. In [16], we have shown that, PIFS Access by the AP can improve the VoWLAN (Voice over WLAN) performance significantly.

When the AP does *PIFS Access* for TCP packet transmissions, all the outstanding TCP packets are located in the queues of stations. As a result, the number of active stations is equal to the number of stations in a WLAN, and hence the behavior of stations is identical to that of saturated UDP stations. The AP sends a TCP Data (or TCP Ack) at PIFS time after it receives a TCP Ack (or TCP Data) from one of the stations. That is, in Fig. 2, the state transitions only occur between *State (M-1)* and *State (M-2)*. When there exist download TCP flows only, the aggregate TCP throughput with the AP conducting *PIFS Access* can be calculated by

$$\rho_{\text{pifs}} = \frac{L_{TCP\,DATA}}{\mu_{M-1} + PIFS + T_{TCP\ DATA} + SIFS + ACK}, \quad (12)$$

where μ_{M-1} is the sojourn time at *State (M-1)* in Fig. 2. In the case of upload TCP, $T_{TCP\ DATA}$ in denominator of Eq. (12) is replaced by $T_{TCP\ ACK}$.

V. Analytical and Simulation Studies

In this section, we validate our model via ns-2 simulation [20]. We consider an IEEE 802.11b WLAN with a single AP and seven stations that either download or upload large data files to remote FTP servers. Fig. 3 shows the network topology. As shown in the figure, the Round Trip Time (RTT) of TCP packet in the wireline network is referred to as the *wireline delay*. TCP NewReno is used for our simulation. We assume that the maximum TCP receive window size is W = 4 packets.[2] Parameters used to produce analytical and simulation results are summarized in Table II. The considered values for CWmin are $\{2^n - 1 \mid 2 \le n \le 8\}$.

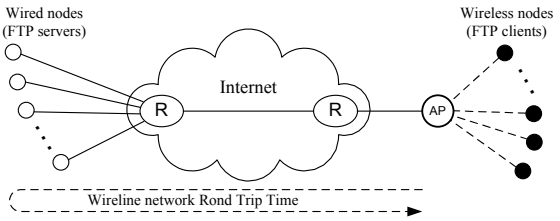


Fig. 3. Network topology

TABLE II
Parameters Used to Produce Analysis and Simulation Results

Parameters	Values
SlotTime	20 μs
SIFS, PIFS, AIFS	10 μs, 30 μs, 50 μs
CWmax	1023
Data, ACK Rates	11 Mb/s, 2 Mb/s
PHY Overhead	192 μs
ACK Length	14 bytes
MAC Overhead	30 bytes
TCP Data frame	Data (1460 bytes) + TCP/IP headers (40 bytes) + SNAP*header (8 bytes) + MAC/PHY overheads
TCP Ack frame	TCP/IP headers (40 bytes) + SNAP header (8 bytes) + MAC/PHY overheads

* When an IP datagram is transferred over the 802.11 WLAN, it is typically encapsulated in an IEEE 802.2 Sub-Network Access Protocol (SNAP) packet.

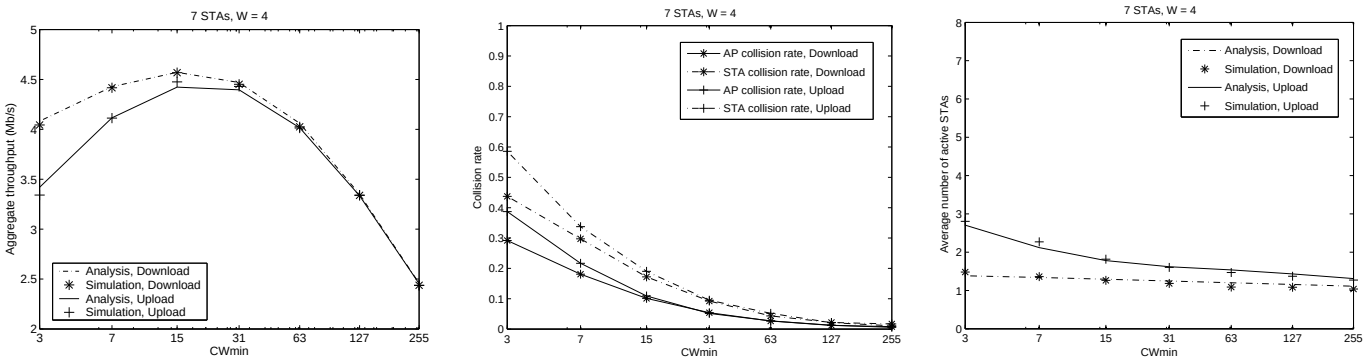
We first study how the TCP throughput performance is affected by the CWmin values, assuming no TCP Ack processing delay[3], no packet loss, and no wireline delay. Then, we show the effects of these parameters on the aggregate TCP throughput in Sections V-C and V-D.

A. Effects of CWmin Sizes

1) When the AP and stations use the same access parameters (including CWmin): Fig. 4 shows (i) the aggregate

[2]Actually, typical TCP configurations of commercial operating systems use a larger receive window size, e.g., the 12-packet (or 17520-byte) window used in MS Windows XP. We consider the 4-packet receive window size in order to reduce the calculation overhead; similar trends can be observed for other receive window sizes.

[3]In this paper, TCP Ack processing delay is defined as time duration between a TCP Data reception and the corresponding TCP Ack arrival at the MAC Service Access Point (SAP).



(a) Aggregate throughput

(b) Collision rate

(c) Average number of active stations (excluding AP)

Fig. 4. TCP performance when the AP and stations use the same access parameters (including CWmin)

TCP throughput, (ii) the collision rates experienced by the AP and a station, and (iii) the average number of active stations (excluding the AP), for both download and upload cases. We observe that analytical and simulation results match very well for all simulated scenarios. In Fig. 4(a), in the case of download, the aggregate TCP throughput of the 802.11 DCF (i.e., CWmin = 31) is about 4.46 Mb/s while the maximum download throughput of about 4.56 Mb/s is achieved when CWmin = 15. From this result, one may quickly conclude that TCP over DCF is operating near the optimal throughput point. Although this is true when the AP and stations are required to use the same access parameters, we will show in the subsequent subsection that the aggregate TCP throughput can be enhanced further by using different access parameters for the AP and stations.

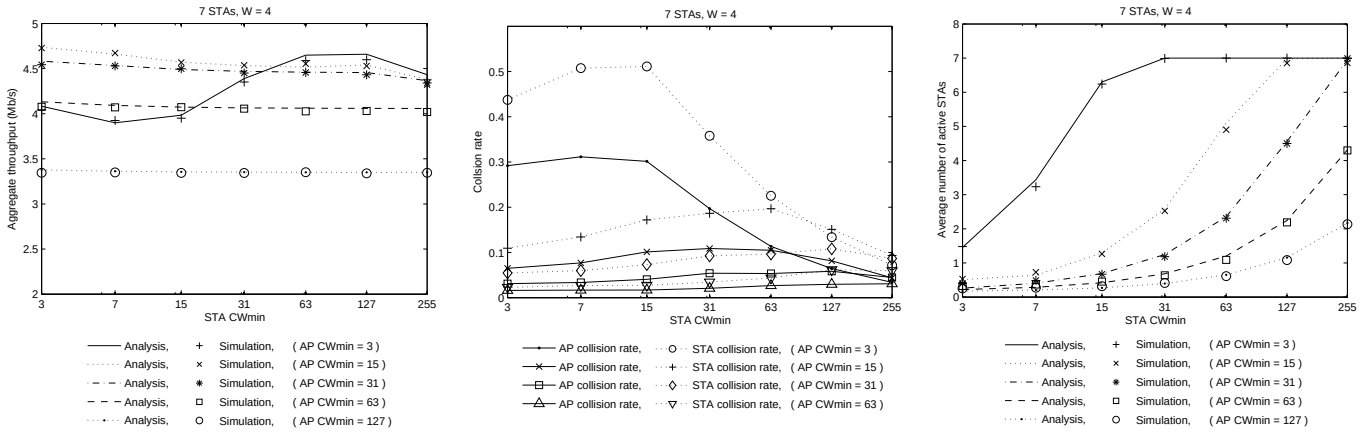
The collision rate of a station is higher than that of the AP when CWmin is small. This is due to the fact that an active station contends with at least one station, i.e., the AP, because the AP is always active in this scenario. Moreover, note that a station transmits long TCP Data and short TCP Ack for upload and download cases, respectively. Therefore, as CWmin decreases, the throughput in the case of upload decreases more quickly than the download case because the wasted time due to collisions is more severe.

As shown in Fig. 4(c), the average number of active stations remains small regardless of the CWmin size. This is due to (i) the closed-loop nature of TCP flow control and (ii) the bottleneck downlink (i.e., AP-to-station transmissions) in the infrastructure-based WLAN, as we have discussed in Section I. Under such scenario, the AP is the bottleneck because it uses the same CWmin as stations but needs to serve multiple TCP flows.

2) When the AP and stations use different access parameters: We study the aggregate TCP throughput performance with various combinations of the CWmin sizes used by the AP and stations. Fig. 5 plots (i) the aggregate TCP throughput, (ii) the collision rates experienced by the AP and a station, and (iii) the average number of active stations (excluding the AP), for the download case. Similar results have also been observed for the upload case, which are omitted due to space limitation.

We have two sets of observations. Firstly, when $\text{CWmin}_{\text{AP}} \geqslant \text{CWmin}_{\text{STA}}$, the AP's downlink becomes a more severe bottleneck. The network is in a non-saturated state. As shown in Fig. 5(c), the average number of active stations (excluding the AP) is less than or equal to one. As the gap between CWmin_{AP} and $\text{CWmin}_{\text{STA}}$ increases, the average number of active stations decreases. This is due to the fact that most stations spend more time idle, waiting for a TCP Data from the AP. On the other hand, it is interesting to see that the aggregate throughput does not increase monotonically as the gap decreases. For example, the combination of $\text{CWmin}_{\text{AP}} = 31$ and $\text{CWmin}_{\text{STA}} = 7$ yields a higher throughput than the combination of $\text{CWmin}_{\text{AP}} = 3$ and $\text{CWmin}_{\text{STA}} = 3$. This is because, the collision rates (especially, the AP collision rate) increase as the gap decreases, which may offset the effects of smaller number of active stations under certain circumstances.

Secondly, when $\text{CWmin}_{\text{AP}} < \text{CWmin}_{\text{STA}}$, packet congestion at the AP bottleneck is alleviated. As the gap between CWmin_{AP} and $\text{CWmin}_{\text{STA}}$ increases, the average number of active stations increases, and is eventually saturated to 7 (i.e., all the stations are actively contending). On the other hand, as shown in Fig. 5(b), the collision rates initially increase and then decrease eventually. The reason is as follows. For a given CWmin_{AP}, the number of active stations keeps increasing until $\text{CWmin}_{\text{STA}}$ reaches a certain threshold; when $\text{CWmin}_{\text{STA}}$ becomes larger than this threshold, the number of active stations is saturated to 7 but the idle backoff time continues to increase. For example, when CWmin_{AP} is 3, such threshold for $\text{CWmin}_{\text{STA}}$ is 31, which is shown in Fig 5(c). In general, this threshold for $\text{CWmin}_{\text{STA}}$ varies with CWmin_{AP}, and a larger CWmin_{AP} corresponds to a larger threshold. Accordingly, we can clearly observe from the figure that, for small CWmin_{AP} values, the aggregate throughput decreases (due to increasing collision rate), then increases (due to decreasing collision rate),



(a) Aggregate throughput

(b) Collision rate

(c) Average number of active stations (excluding AP)

Fig. 5. Download TCP performance when the AP and stations use different CWmin

and eventually decreases (due to increasing idle backoff time). The reason that we don't observe similar behaviors in the figure for larger CWmin_{AP} values is because the corresponding thresholds for $\text{CWmin}_{\text{STA}}$ are very large, beyond the x-axis range of the figure.

Comparing Fig. 5(a) with Fig. 4(a), we can see that the maximum throughput in Fig. 5(a) is larger than that in Fig. 4(a). It means that there exists at least one optimal combination of CWmin_{AP} and $\text{CWmin}_{\text{STA}}$ to maximize the aggregate throughput. This fact is not surprising since a similar fact has been well known for the saturation throughput maximization in 802.11 DCF WLANs. According to [7], [8], there exists an optimal CWmin value that maximizes the saturation throughput for a given number of stations. Unfortunately, it is very difficult, if not impossible, to derive the closed-form expression for the optimal CWmin combination in our study, because the CWmin values and the number of active stations are mutually dependent in the case of TCP.

B. Effects of AP PIFS Access

Fig. 6 plots the aggregate TCP throughput and the collision rate when the AP uses *PIFS Access* for its downlink transmissions. In the case of download, we observe that the maximum aggregate TCP throughput of *PIFS Access* is much higher than the previous two cases in Section V-A. Note that, with *PIFS Access*, the AP's queue is always empty and all the stations always have packets to transmit. That is, *PIFS Access* makes the stations behave like saturated UDP stations. As a result, the difficult problem of maximizing the TCP throughput is simplified and becomes equivalent to the easier problem of maximizing the UDP saturation throughput.

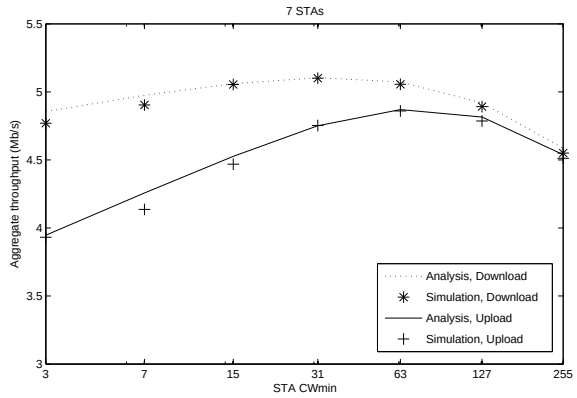
With *PIFS Access*, there is no collision between the AP and stations, hence less time is wasted due to collisions. For example, when only download TCP flows are present in the network, the AP transmits long TCP Data to stations and stations respond by transmitting short TCP Ack back to the AP. In this situation, if the AP's transmission collides with stations' transmissions, the wasted time is equal to the TCP Data transmission time, which is about 4 times the TCP Ack transmission time (i.e., the wasted time due to collisions among stations).

In [17], the authors showed that the maximum throughput of a saturated UDP network is independent of the number of stations and, with 1000-byte MAC frames, the maximum throughput is achieved when the collision rate experienced by a station is about 0.2. This is coincident with our simulation results shown in Fig. 6(b).

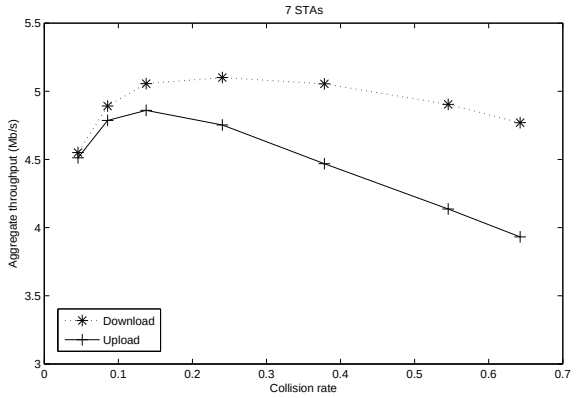
So far, we have evaluated the effects of CWmin values and AP PIFS Access on the aggregate TCP throughput without considering TCP Ack processing delay, wireline delay, and packet loss. Next, we study the effects of these parameters.

C. Effects of TCP Ack Processing Delay

Fig. 7 shows the effects of TCP Ack processing time on the aggregate TCP throughput. We assume no wireline delay in this scenario. In general, we observe that, with non-zero TCP Ack processing delay, the aggregate throughput deviates little from our analysis result (i.e. 4.46 Mb/s) by at most 5%. This means that our analysis is reasonably accurate even with the simplifying assumption of no TCP Ack processing delay. As shown in the figure, the throughput increases when the TCP Ack processing delay is around 308 μs. This is due to the *immediate access* behavior that was discussed in Section II-A. The reason can be explained as follows. Fig. 8 shows the timing diagram of the AP and a station that receives a TCP Data from the AP. If the TCP Ack is generated within duration **A** in Fig. 8, the MAC starts backoff at the end of duration **A** because it senses the channel busy. On the other hand, if the TCP Ack is generated within



(a) Aggregate TCP throughput



(b) Collision rate vs. Aggregate throughput

Fig. 6. Performance of PIFS Access when all stations either download or upload data.

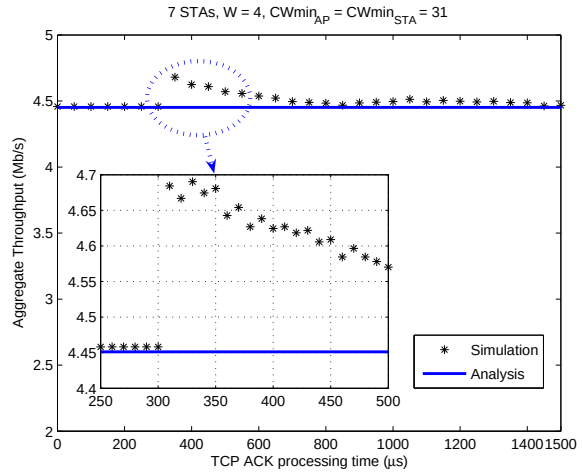


Fig. 7. The effects of TCP Ack processing delay (with no wireline delay) when all stations download data from FTP servers.

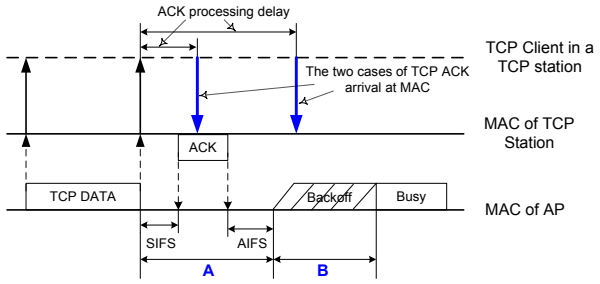


Fig. 8. Illustration of the TCP Ack processing delay in a station when it downloads data from a server via the AP.

duration **B**, the MAC can send the TCP Ack immediately without backoff. Note that the length of duration **A** is 308 μs = SIFS(10 μs) + ACK(248 μs) + AIFS(50 μs).

D. Effects of Wireline Delay and Packet Loss

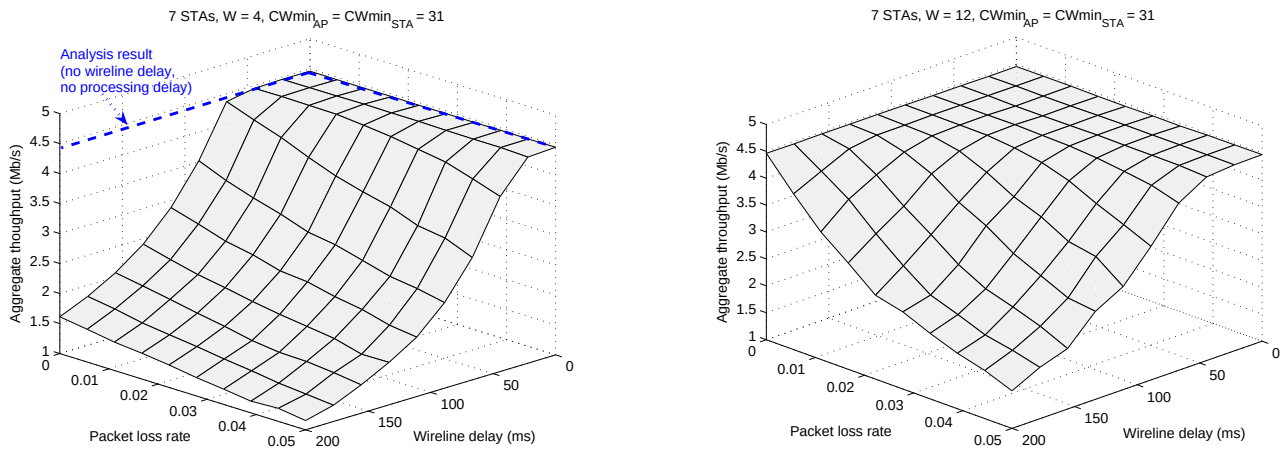
Fig. 9 shows the effect of wireline delay (from 0 to 200 ms) and the end-to-end packet loss rate (from 0.1% to 5% according to [19]) on the aggregate TCP throughput. Results with the maximum TCP receive window sizes of W = 4 and W = 12 are shown in Figs. 9(a) and 9(b), respectively. It is clear from the figure that our model produces accurate results when the wireline delay is small and the packet loss rate is low, i.e., when the WLAN is the bottleneck in the network.

However, when the wireline delay gets larger, the WLAN is not bottleneck anymore. That is, the aggregate TCP throughput is dominantly determined by the wireline delay. We observe that the aggregate TCP throughput decreases as the wireline delay increases. Moreover, packet loss accelerates the throughput degradation. We can see that, with a higher packet loss rate, the aggregate throughput decreases more quickly as the wireline delay increases. The reason is as follows. The high packet loss rate causes TCP timeout and in turn decreases the number of outstanding TCP packets, because the TCP send window shrinks to one MSS after timeout.

Fig. 9(b) shows that a larger maximum TCP receive window results in a bigger applicable region (i.e., the flat region) for our model. This is because, with a larger maximum TCP receive window, more outstanding TCP packets are allowed in the network, hence WLAN may remain as the bottleneck even in the presence of larger wireline delay and/or higher packet loss rate.

VI. Conclusion and Future Work

In this paper, we conduct a rigorous and comprehensive analysis of the TCP dynamics over EDCA-based IEEE 802.11e WLANs, and show that the aggregate TCP throughput can be enhanced via a proper selection of channel access parameters for the AP and stations. In particular, we show that the best aggregate throughput performance can be achieved via AP's contention-free access for downlink transmissions. Finally, the assumptions used in our model are evaluated via simulation, and the results show that our model is reasonably accurate when the wireline delay is small and the packet loss rate is low.



(a) W = 4

(b) W = 12

Fig. 9. The effects of wireline delay and packet loss on the aggregate TCP throughput when all stations download data from FTP servers. W is the maximum TCP receive window size (in packets).

IEEE 802.11e defines a concept of transmission opportunity (TXOP), during which the AP or a station may transmit multiple frames back-to-back without backoff [18]. We are currently extending our work to consider the TXOP mechanism. Moreover, we are also working on optimization of the 802.11e performance when considering other types of traffic such as VoWLAN.

REFERENCES

[1] IEEE, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, Reference number ISO/IEC 8802-11:1999(E), IEEE Std 802.11, 1999 edition, 1999. IEEE Std 802.11-1999, 1999.

[2] IEEE, Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed Physical Layer Extension in the 2.4 GHz Band, IEEE Std. 802.11b-1999, 1999.

[3] IEEE, Supplement to Part 11: Wireless LAN Medium Access Control (MAC) and Physical layer (PHY) specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS), IEEE Std 802.11e, Nov. 2005.

[4] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Reading, MA: Addison-wesley, 1994, vol. 1.

[5] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP," *Computer Communication Review*, Vol. 26, No. 3, pp. 5-21, July 1996.

[6] H. Jiang and C. Dovrolis, "Why is the Internet Traffic Bursty in Short Time Scales?", in *Proc. ACM SIGMETRICS'05*, June 2005.

[7] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, pp. 1774-1768, Mar. 2000.

[8] F. Cali, M. Conti, and E. Gregori, "Dynamic Tuning of the IEEE 802.11 Protocol," *IEEE/ACM Trans. Networking*, Vol. 8, No. 6, Dec. 2000.

[9] I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11," in *Proc. IEEE INFOCOM'01*, March 2001.

[10] J. Yu, S. Choi, and J. Lee, "Enhancement of VoIP over IEEE 802.11 WLAN via Dual Queue Strategy," in *Proc. IEEE ICC'04*, June 2004.

[11] S. Choi, K. Park and C. Kim, "Performance Impact of Interlayer Dependence in Infrastructure WLANs," *IEEE Trans. Mobile Computing*, Vol. 5, No. 7, July 2006.

[12] A. A. Kherani and R. Shorey, "Modelling TCP Performance in Multihop 802.11 Networks with Randomly Varying Channel," in *Proc. WILLOPAN'06*, Jan. 2006.

[13] C. Burmeister and U. Killat, "TCP over Rate-Adaptive WLAN-An Analytical Model and its Simulative Verification," in *Proc. IEEE WoWMoM'06*, June 2006.

[14] R. Bruno, M. Conti, and E. Gregori, "Analytical Modeling of TCP Clients in Wi-Fi Hot Spot Networks," in *Proc. IFIP Networking'04*, May 2004.

[15] J. Yu and S. Choi, "Modeling and Analysis of TCP Dynamics over 802.11 WLAN," in *Proc. IEEE/IFIP WONS'07*, Jan. 2007.

[16] J. Yu and S. Choi, "Comparison of Modified Dual Queue and EDCA for VoIP over IEEE 802.11 WLAN," *Euro. Trans. Telecomms.*, Vol. 17, No. 3, Jan. 2006.

[17] H. Zhai, X. Chen, and Y. Fang, "How Well Can the IEEE 802.11 Wireless LAN Support Quality of Service?," *IEEE Trans. Wireless Commun.*, Vol. 4, No. 6, Nov. 2005.

[18] I. Tinnirello and S. Choi, "Temporal Fairness Provisioning in Multi-Rate Contention-Based 802.11e WLANs," in *IEEE WoWMoM'05*, June 2005.

[19] Y. Yamasaki, H. Shimonishi, and T. Murase, "Statistical Estimation of TCP Packet Loss Rate from Sampled ACK Packets," in *Proc. IEEE GLOBECOM'05*, Dec. 2005.

[20] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/, online link.