A Queueing Model for HTTP Traffic over IEEE 802.11 WLANs

Daniele Miorandi [*] [†], Arzad A. Kherani [*] and Eitan Altman [*]
[*] INRIA, 2004, Route des Lucioles – Sophia Antipolis (France)
`email:{daniele.miorandi,alam,eitan.altman}@sophia.inria.fr`
[†] Department of Information Engineering – Padova (Italy)
`email:daniele.miorandi@dei.unipd.it`

Abstract

We consider an IEEE 802.11 based wireless LAN where an access point is used to connect a fixed number of users to the web or to a shared file system. Users alternate between activity periods (corresponding to the download of a file) and idle periods (corresponding to think times). We first consider the interaction of TCP with the IEEE 802.11 MAC protocol, and get approximate expressions for the TCP throughput in presence of n_c competing persistent connections. This result is then used to develop a queueing model, which is used to determine the mean session delay in presence of short–lived flows. The analysis also accounts for the TCP delayed ACK option. Comparison with simulation outcomes is provided, validating the model and providing guidelines for network designers. A particular emphasis is devoted to the impact of the advertised window size; in particular, it is proved that setting it to a small value leads to insensitivity of mean file transfer times to the file size distribution.

1 Introduction

IEEE 802.11 [1] is the de facto standard for wireless local area networks (WLANs). WLANs aim at providing wireless connectivity to the Internet, representing a valid alternative to classical Ethernet LANs. Even if performance cannot be directly compared to those obtainable by means of a wired connection, it seems that future developments of the standard (namely, IEEE 802.11a), capable of transferring data at rates up to 54 Mb/s, will make it an even more attractive solution for ensuring nomadic connectivity. Since wireless access to the Internet is the most common application of this kind of networks, it is reasonable to assume that the traffic will be carried over the well–known TCP/IP protocol suite [2]. Hence, an important issue to address is the interaction between the 802.11 MAC and the closed–loop nature of TCP. This is accomplished in the first part of the paper, where we estimate the throughput achieved by TCP in a WLAN scenario with many concurrent persistent connections. In particular, it turns out that, due to the extremely high overhead encompassed by the protocol and the absence of an explicit duplexing mechanism, the feedback traffic (namely, that constituted by the flow of TCP acknowledgment packets) has an extremely negative effect on TCP throughput. This motivated us to study the impact of ACK thinning techniques, such as the delayed ACK option for TCP [3, 4] and some modifications, such as those presented in [5]. The results obtained in the first part, in terms of attained service rate, are then used to evaluate the network performance in presence of short–lived flows. We model the network as a processor sharing queue with state–dependent service rate, and compute the mean session delay for HTTP traffic. Results are first derived for exponentially distributed file size, and are then generalized applying insensitivity results, along the lines of [6]. In particular, we analyse the impact of the TCP maximum congestion window size on the mean session delay. It turns out that, by setting this parameter to a large value, unfairness problems do arise, leading to an overall performance worsening in presence of heavy tailed file size distribution. All results are compared with the outcomes of numerical simulations. The analysis is also extended to a generalized delayed ACK technique. Evidence is provided that, under a careful tuning of some system parameters, delayed ACK techniques can help in reducing the mean session delay.

The paper is organized as follows. Sec. 2 reports a broad description of the network scenario, of the 802.11 MAC protocol and of the delayed ACK mechanism. The throughput analysis for persistent TCP connections is presented in Sec. 3. The analysis of session delay for short–lived flows is presented in Sec. 4; Sec. 5 concludes the paper.

2 The Network Scenario

We consider a WLAN where an access point (AP) provides access to web or a shared file system to $(n-1)$ hosts (also referred to as stations or nodes in the following). At any instant of time, a host is downloading (or, receiving) at most one file via the AP. This file transfer is controlled using TCP. After the completion of a file transfer, a host waits for a random amount of time before initiating another file transfer request. The size of the requested files in successive requests are assumed to be independent and identically distributed. The TCP controlled Data traffic flows in the downlink direction (from the AP to the hosts) while the TCP Acknowledgement traffic flows from the hosts to the AP. The AP and various nodes use the IEEE 802.11 MAC protocol for transmission of their data (TCP DATA packets in case of AP and TCP ACK packets in case of nodes). Relevant details of the IEEE 802.11 protocol are summarized in Sec. 2.1.

The number of active file transfers is now time–varying. The variation in the number of concurrent file transfers is in turn determined by the effective service rate seen by the active file transfers. The effective rate at which a file is served at any instant of time is determined by the interaction between MAC and the closed–loop behaviour of TCP controlling the ongoing file transfers.

The problem that we address in this work is that of obtaining the average performance seen by the hosts. The approach is to first study the detailed interaction between the IEEE 802.11 MAC and TCP with a fixed number of file transfers to obtain the effective service rate of the files as a function of the number of concurrent file transfers (Sec. 3). This is then used to obtain a queueing model that captures the time varying behaviour of the number of concurrent file transfers, thus providing us with a node's average perceived performance (Sec. 4).

2.1 Protocol Description

In the IEEE 802.11 standard, the medium access control is based on a distributed CSMA/CA mechanism [1]. A node listens to the channel for a time equal to the distributed inter–frame spacing (DIFS). If the medium is sensed idle, then a random backoff is generated. During the backoff the node continues sensing the channel. If, at the end of the backoff, the medium is still idle, the node starts transmitting. Since no channel load sensing mechanism is provided, an explicit acknowledgment is necessary to inform the node of the success/failure of its transmission. To accomplish that, when a node receives a packet, after a short interframe spacing (SIFS), it sends a short ACK packet to inform the source of the outcome of the previous trasnmission. The length of the backoff interval, expressed in slots, is uniformly chosen in the set $\{0, 1, \ldots, CW-1\}$, where CW denotes the actual contention window size. At the beginning, CW is set to a predefined value CW_{min}. If a collision or loss occurs, the contention window of the nodes involved in the collision is doubled and another trasmission attempt is made. The contention window cannot grow indefinitely, but may reach a maximum value of $2^{\gamma}CW_{min} = CW_{max}$; moreover, if a packet incurs m collisions, where $m \geq \gamma$, it is dropped. If a transmission is detected while the backoff counter has not reached zero, its value is frozen and reloaded as soon as the channel is sensed idle for a DIFS.
An optional RTS/CTS mechanism is encompassed by the standard to avoid the well–known hidden terminal problem of CSMA–based MAC protocols. In this case, after a DIFS and a random backoff, a RTS packet is sent to the intended destination. After a SIFS, the destination replies with a CTS, signalling to all the stations in range the foreseen duration of the packet exchange. After a SIFS, the sender starts the transmission of the data packet. In this way, a virtual channel sensing mechanism is provided, since all the stations which received the CTS may update their network allocation vector (NAV) and go in stand–by for the whole duration of the packet exchange (the channel is "virtually" sensed busy). As shown in [7], the use of the RTS/CTS mechanism let the network performance be less sensitive to the mutual interference generated by a high number of devices. In the following we will assume that the RTS/CTS exchange precedes any packet transmission.
At the physical layer, the standard encompasses three different options: infrared, frequency hopping and direct sequence CDMA. We focus on the last option (the one currently implemented), and, furthermore, use the parameters described for operations in the 2.4 GHz ISM band, known as 802.11b [8].

2.2 Delaying ACK Techniques

When running over bandlimited channels, such as wireless ones, one of the basic techniques to improve TCP throughput is to thin the feedback traffic, by reducing the flow of TCP acknowledgment packets. In this way, more bandwidth is ensured to the forward TCP link, achieving thus a higher throughput. This mechanism presents some drawbacks. The main one is that, by thinning the TCP ACK flow, the congestion window grows slower, and this results, in general, in an in-

creased session delay for short TCP sessions [4], which are known to constitute most of the traffic carried over the Internet. However, this drawback can be, if not eliminated, seriously mitigated, by using adaptive techniques such as the ones described in [5]. Further, problems arise at both the beginning and the end of a file transfer. Indeed, at the beginning of a TCP connections, if the TCP initial window size is set to a value less than d, a timeout necessarily takes place. Such problem may be solved by enlarging the initial window size to a value greater than 1 [9]. On the other hand, nothing can be done for the end of a file transfer, where, if the number of packets is not a multiple of d, time has to be "wasted" waiting for a timeout expiration.
The standard delayed ACK technique for TCP [3] suggests that an ACK packet should be sent every $d = 2$ received packets. If no packet is received for more than a timeout period τ_{out}, an acknowledgment packet has to be generated. It is clear that by further reducing the frequency with which ACKs are sent, we may, in principle, gain bandwidth, at the expense of a much slower growth of the congestion window at the beginning of the connection or after a packet loss. In reality, the value of d cannot be indefinitely increased, since this would cause harmful expirations of the TCP sender timeout, which result in an overall performance degradation. Further, in the case of short sessions, increasing d increases the probability of incurring a timeout at the end of a file transfer, with a negative impact on the mean session delay.

In Tab.1 we report the parameters we use for evaluation purposes. T_P and T_{PHY} represent the time necessary to transmit the long PLCP preamble and the physical–layer header, respectively. (We treat the case of long PLCP header; the analysis is the same for short PLCP preamble.) As far as the transmission rates are concerned, we assume ideal channel conditions; hence all hosts use a data rate of $R_{data} = 11$ Mb/s. Further, we take the control rate to be equal to $R_{control} = 2$ Mb/s and assume that TCP packets are $L_{TCP} = 8000$ bits long. The length of TCP/IP header is $L_{IPH} = 320$ bits, whereas the length of the RTS and CTS packets are given by $L_{RTS} = 180$ bits and $L_{CTS} = 112$ bits, respectively. A MAC–layer ACK consists of $L_{ACK} = 112$ bits. The overhead added by the MAC layer (MAC header plus FCS field) amounts to $L_{MAC} = 272$ bits.

The raw trasmission times of a TCP data packet and a TCP ACK, disregarding the backoff, are given, respectively, by:

Parameter	Value
T_{slot}	$20\mu s$
T_{SIFS}	$10\mu s$
T_{DIFS}	$50\mu s$
T_P	$144\mu s$
T_{PHY}	$48\mu s$
CW_{min}	32
CW_{max}	1024
m	7
γ	5

Table 1: Parameters of 802.11b (long PLCP preamble)

$$\begin{aligned}T_{TCP\_data} &= T_{DIFS} + T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{SIFS} + T_P + \\ &+T_{PHY} + \frac{L_{CTS}}{R_{control}} + T_{SIFS} + T_P + T_{PHY} + \\ &+\frac{L_{MAC} + L_{IPH} + L_{TCP}}{R_{data}} + T_{SIFS} + T_P + T_{PHY} + \frac{L_{ACK}}{R_{control}},\\ T_{TCP\_ack} &= T_{DIFS} + T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{SIFS} + T_P + \\ &+T_{PHY} + \frac{L_{CTS}}{R_{control}} + +T_{SIFS} + T_P + T_{PHY} + \frac{L_{MAC} + L_{IPH}}{R_{data}} + \\ &+T_{SIFS} + T_P + T_{PHY} + \frac{L_{ACK}}{R_{control}}.\end{aligned}$$

Note that a TCP ACK packet consists just of the TCP/IP header, of length L_{IPH}.

3 Persistent TCP Connections: Throughput Analysis

In this section we find the effective service rate that the TCP controlled file transfers get when the number of file transfers is fixed and each of these files are of infinite length. In order to do this, we find the probability of collision owing to multiple simultaneous transmission attempts when there are n_c concurrent TCP connections. Let $n_b(\leq n_c + 1)$ be the expected number of backlogged nodes (i.e., nodes having packets to be transmitted). We then find the expected time required for a successful transmission of a packet. The relation between n_c and n_b will be made explicit later.

We now thread the footprints of [10], enhancing the model by considering that, at backoff stage k, the average backoff time is $\frac{2^k CW_{min}-1}{2}$ if $k \leq \gamma$ and $\frac{2^\gamma CW_{min}-1}{2}$ for $\gamma \leq k \leq m$. Furthermore, we neglect the packet drops which take place due to attained maximum number of retransmissions. Hence, denoting with P_C the collision probability, and considering it to be independent of the backoff stage (this has been observed in [7]

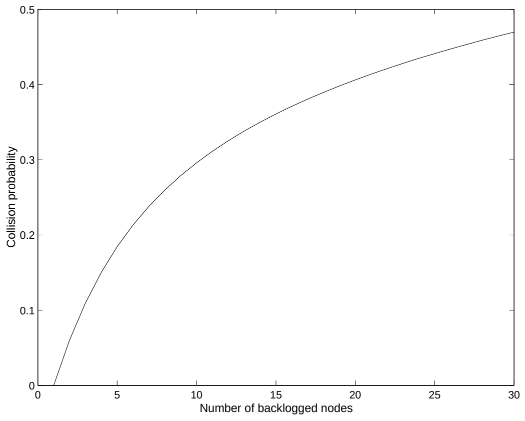

Figure 1: Collision probablity as a function of the number of backlogged nodes.

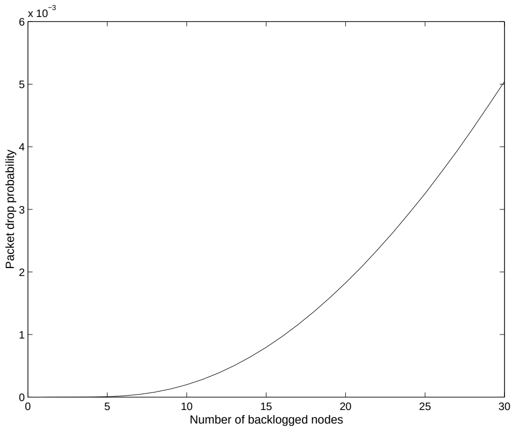

Figure 2: Packet drop probability as a function of the number of backlogged nodes.

to provide good approximation), we get for the average backoff time:

$$T_b = \frac{(1-P_C)}{2} CW_{min} \frac{1-(2P_C)^\gamma}{1-2P_C} - \\ - \frac{1-P_C^\gamma}{2} + \frac{2^\gamma CW_{min}-1}{2}(P_C^\gamma - P_C^m). \quad (1)$$

The complete derivation of this relation is reported in [11]. Thus, since n_b nodes are backlogged, we may, as in [10], find the collision probability by numerically solving (1) together with:

$$P_C = 1 - \left(1 - \frac{1}{T_b}\right)^{n_b-1}. \quad (2)$$

An alternative fixed point formalization of such procedure for the computation of the collision probability is reported in [12].
The probability of dropping a packet can be estimated as $P_{drop} = P_C^m$. Numerical results for P_C and P_{drop} as a function of n_b are presented in Fig. 1 and Fig. 2.

Observe that, when using the RTS/CTS mechanism, collisions may occur only on RTS packets. By optimistically considering that retransmissions are triggered after a SIFS (i.e. when no CTS is detected by the sender), we get that the average time "lost" in each collision is:

$$T_{coll} = T_{DIFS} + T_b + T_P + T_{PHY} + \frac{L_{RTS}}{R_{control}} + T_{SIFS}. \quad (3)$$

Since the collision probability P_C is assumed to be independent of the backoff stage of a node, the number of collisions that a node sees before succeding in transmission is a geometric random variable. Since the node loses, on an average, T_{coll} amount of time for each collision, the average total time "wasted" in collisions for any succesful packet transmission may be computed as:

$$T_W = T_{coll} \cdot \sum_{k=0}^{m-1} k(1-P_C)P_C^k \approx \\ \approx T_{coll} \cdot \sum_{k=0}^{+\infty} k(1-P_C)P_C^k = T_{coll} \cdot \frac{P_C}{1-P_C}, \quad (4)$$

where the approximation is based on the assumption that no packet drop takes place due to reached maximum number of retransmissions. Similarly, the mean number of backoff periods which precede a succesful transmission is given by $\sum_{k=1}^{m} k(1-P_C)P_C^{k-1} \approx \sum_{k=1}^{+\infty} k(1-P_C)P_C^{k-1} = \frac{1}{1-P_C}$. Since each backlogged node attempts transmitting at a rate $\frac{1}{T_b}$, the mean total time spent in backoff can be estimated as:

$$T_{tbo} = \frac{T_b}{n_b \cdot (1-P_C)} \quad (5)$$

Note that, to keep the notation clear, the dependence of T_b, P_C, T_W and T_{tbo} on n_b is suppressed.

We now use the above results to obtain $S_{TCP}(n_c, d)$, the throughput achieved by a single TCP connection in presence of n_c competing connections, where each receiver is employing delACK parameter d. As far as TCP is concerned, we work under the following assumptions:

- the retransmission timer at the TCP source is large enough so that no timeouts take place;
- the transmit buffers at the nodes are well–dimensioned, in the sense that no packet drop takes place due to buffer overflow;

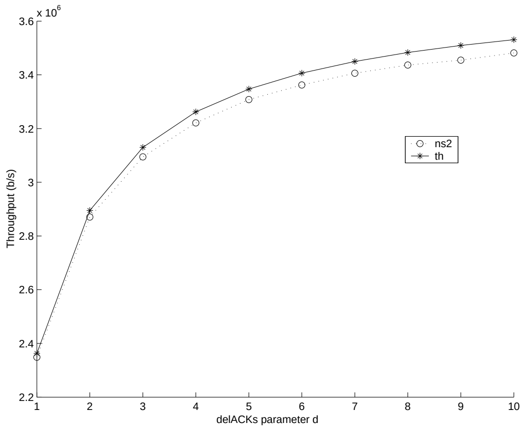

Figure 3: TCP throughput vs. delACK parameter d for the single station case.

Since packet drops due to contention are considered negligible, and the network operates over a lossless channel, the TCP congestion window would grow. However this growth can not continue indefinitely, since the TCP window is bounded by the receiver's advertised window W^*. Thus TCP window will stabilize, after a transient phase, at the maximum TCP window W^*.

3.1 Single station

We start with an analysis of the simplest case, namely a single TCP connection between the access point (AP) and the station, so that $n_c = 1$. Let us consider the case $W^* \gg 1$; a study of the impact on throughput of the value of W^* is reported in the appendix, complementing the results of [13] to the single–hop case. Thus, with high probability, both nodes will be backlogged and contend for the channel so that $n_b = 2$. The collision probability can be found by specializing (1) and (2) to the case $n_b = 2$, obtaining P_C as the solution of:

$$\frac{1}{P_C} = \frac{(1-P_C)}{2} CW_{min} \frac{1-(2P_C)^\gamma}{1-2P_C} - \\ - \frac{1-P_C^\gamma}{2} + \frac{2^\gamma CW_{min}-1}{2}(P_C^\gamma - P_C^m), \quad (6)$$

giving $P_C \approx 0.060$. Accordingly, the throughput for standard TCP can be estimated as:

$$S_{TCP}(1,1) = \frac{L_{TCP}}{T_{TCP\_data} + T_{TCP\_ack} + 2T_{tbo} + 2T_W}. \quad (7)$$

To consider the case where the host employs the delACK mechanism, we note that with probability $\frac{d-1}{d}$ no acknowledgment will be sent by the receiver, so that no contention for the channel takes place. Thus, for $W^* \gg d$, we get the following approximate expression for the TCP throughput:

$$S_{TCP}(1,d) = L_{TCP} \cdot \Bigg[T_{TCP\_data} + \frac{1}{d}(T_{TCP\_ack} + \\ +2T_{tbo} + 2T_W) + \frac{d-1}{d} \cdot \frac{CW_{min}-1}{2} \Bigg]^{-1}. \quad (8)$$

Results are plotted in Fig. 3, together with outcomes of ns–2 [14] simulations (obtained with $W^* = 1000$). It is worth noting that the gain achieved with delACK increases with the data rate of the network. For example, using the parameters of 802.11a, and assuming that the MAC–layer ACK is sent at 6 Mb/s and that the highest bit rate (54 Mb/s) is used on the forward link, we get for the case $d = 2$ a throughput gain of 26.7 % .

3.2 Multiple stations

In the case of multiple competing connections, we consider two extreme cases in order to get (tight) bounds on the TCP throughput. First we consider a collision–free scenario, where a sort of round robin token–passing between different connections takes place. In such a case, given n_c connections, we get:

$$S_{TCP}^{nc}(n_c,1) = L_{TCP} \cdot \{ n_c \cdot [T_{TCP\_data} + T_{TCP\_ack} + \\ + (CW_{min}-1)T_{slot}] \}^{-1}. \quad (9)$$

In the second scenario, we take into account the mutual interference among competing connections. Let us assume that, for all TCP connections, $W^* = 1$. This assumption relies on the fact that in WLANs the bottleneck is represented by the shared radio channel, so that the network performance should not depend much on the actual value of W^* (an indepth discussion for the single station case is reported in the appendix). Hence, for any connection, the receiver has an ACK waiting to be sent with probability 1/2, which is also the probability that it contends for the channel. Considering the AP to be always backlogged, the average number of units contending for the channel is given by: $n_b = 1 + \frac{n_c}{2}$. Based on this approximation, and since, for any connection, an ACK is sent for every received TCP packet, we obtain:

$$S_{TCP}^{coll}(n_c,1) = L_{TCP} \cdot [n_c \cdot (T_{TCP\_data} + \\ + T_{TCP\_ack} + 2T_{tbo} + 2T_w)]^{-1}, \quad (10)$$

where T_{tbo} and T_w are computed assuming n_b backlogged nodes.

Intuitively, the "collision model" should well predict network performance for small values of n_c. On

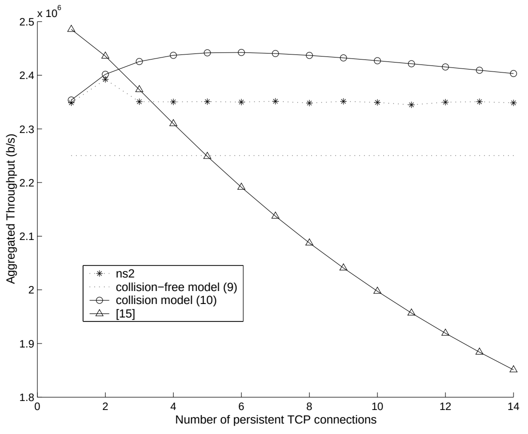

Figure 4: Aggregated throughput vs. number of persistent TCP connections.

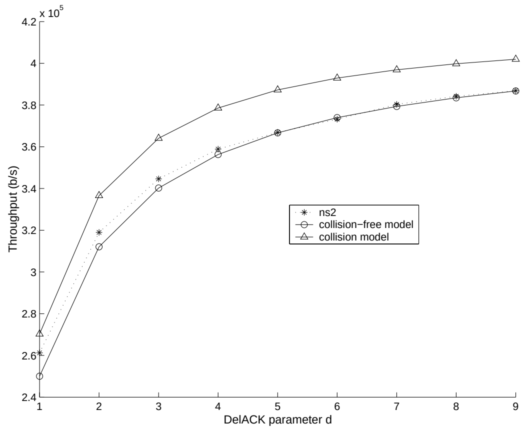

Figure 5: TCP throughput vs. DelACK parameter d, $n_c = 10$.

the other hand, for a large number of hosts, our assumptions (no buffer overflows and no timeout expirations) are not valid any more. In this situation TCP is expected to effectively reduce the congestion in the network, so that, in most cases, only one node will attempt transmitting. In this regime, the "collision–free model" is thus expected to provide a reasonable estimate of the network performance. Numerical results for the aggregated throughput are plotted in Fig. 4, where the two formulas above are compared with simulation results and with the model presented in [15] (the latter has been modified since it already accounts for $d = 2$). It is worth remarking that, for small values of n_c, the collision model predicts a larger throughput than the collision–free model. While this seems counterintuitive, it results from the more aggressive behavior nodes show in presence of contention.

The approximate formulas defined above can be easily extended to the case of delayed ACK. The extension of the collision–free model is trivial, and leads to:

$$S_{TCP}^{nc}(n_c, d) = L_{TCP} \cdot \{n_c \cdot [T_{TCP\_data} + \\ + \frac{1}{d} \cdot T_{TCP\_ack} + \frac{d+1}{d} \cdot \frac{(CW_{min}-1)T_{slot}}{2}\bigg]\bigg\}^{-1}. \quad (11)$$

As far as the "collision model" is concerned, we consider this time an average number of hosts contending for the channel equal to $n_b = 1 + \frac{n_c}{2d}$ (this corresponds to $W^* = d$). Then,

$$S_{TCP}^{coll}(n_c, d) = L_{TCP} \cdot \{n_c \cdot [T_{TCP\_data} + \\ + \frac{1}{d} \cdot T_{TCP\_ack} + \frac{d+1}{d} \cdot (T_{tbo} + T_W)\bigg]\bigg\}^{-1}. \quad (12)$$

The level of approximation we may get with the previous formula has been tested through extensive numerical simulations. In Fig. 5 we reported the TCP throughput, as a function of d, for $n_c = 10$ (in ns2 we set $W^* = 1000$).

4 Short–lived TCP flows: mean delay analysis

It is widely acknowledged that, although a study of throughput for persistent TCP connections is of interest to characterize the network performance, it does not reflect the end–user perception. In a web browsing scenario, the web page delay is indeed the performance metric influencing user perception. Assuming that a web page transfer corresponds to a TCP session, we characterize thus network performance in terms of mean session delay for short–lived TCP flows. As far as traffic patterns are concerned, we assume an ON/OFF model for HTTP traffic. Nodes alternate between activity and idle periods. Active periods correspond to the download of a web page, whose size, in bits, will be denoted by X. Off periods, referred to as think times, correspond to the time elapsed between the end of a download and the beginning of the next one. We assume that think times are exponentially distributed with mean $\frac{1}{\lambda}$. The file size distribution, where not otherwise stated, is assumed to be general with mean $E[X]$.

4.1 An affine model for the single station case

Let us start by considering a single TCP connection. In the previous section, we noted that the maximum congestion window size has a negligible impact on the

	ns2, $W = 1$	ns2, $W \gg 1$	analytical
Mean Session Delay (s)	0.1099	0.1064	0.1095
95% Conf.Int.(s)	$[0.0956, 0.1243]$	$[0.0928, 0.1199]$	–

Table 2: Session delay for short TCP connections, mean and 95% confidence interval.

connection throughput. Hence, we base our analysis on the assumption $W^* = 1$. Under such an assumption, 802.11 MAC provides, in our scenario, contention–free access to the wireless channel. Thus, the average round trip time is given by $T_{TCP\_data} + T_{TCP\_ack} + (CW_{min} - 1) \cdot T_{slot}$. Furthermore, we should take into account the time spent in the connection setup phase, based on a three–way handshaking. Neglecting the propagation delay, and considering that the third packet necessary to setup the connection may be piggybacked within a data frame, the mean connection setup time may be written as:

$$E[T_{setup}] = 2 \cdot \Big[T_{DIFS} + T_P + T_{PHY} + \\ + \frac{(CW_{min} - 1)T_{slot}}{2} + \frac{L_{IPH}}{R_{data}} + \\ + T_{SIFS} + T_P + T_{PHY} + \frac{L_{ACK}}{R_{control}}\Big]. \quad (13)$$

TCP encompasses a 4–way handshake to close down the connection; however, after the delivery of the last TCP segment, the file is taken to be sent to the destination, and hence we may neglect it in the calculations. For transmitting a message of size X (in bits), after some easy algebra we obtain an average session delay of:

$$E[T_{session}] = E[T_{setup}] + \left\lceil \frac{X}{L_{TCP}} \right\rceil (T_{TCP\_data} + \\ + T_{TCP\_ack} + (CW_{min} - 1)T_{slot}) - \left(\left\lceil \frac{X}{L_{TCP}} \right\rceil - \\ - \left\lfloor \frac{X}{L_{TCP}} \right\rfloor \right) \cdot \frac{L_{TCP} - X \bmod L_{TCP}}{R_{data}}. \quad (14)$$

In order to validate the model, we simulated 10000 file transfers over 802.11b[1]. The file size, X, is Pareto–distributed with mean 30 KByte and shape factor $\beta = 1.5$. The results are reported in Tab.2. As expected, the more aggressive behavior induced by a larger value of W^* leads to better performance in terms of session delay.

Notice that the above model is approximately affine in

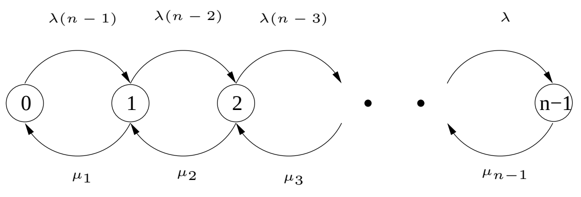

Figure 6: Transition probability diagram of the Markov chain $J(t)$.

the file size:

$$E[T_{session}] \approx E[T_{setup}] + \frac{E[X]}{L_{TCP}} \cdot (T_{TCP_{data}} + T_{TCP_{ack}} + \\ + (CW_{min} - 1)T_{slot}). \quad (15)$$

Furthermore, the mean session delay shows insensitivity properties with respect to the file size distribution, depending only on the mean file size. Simulation results have confirmed such a system feature.

4.2 Concurrent TCP sessions: a queueing model

Now we use the results obtained till now to compute the mean session delay in presence of concurrent TCP sessions.We consider a WLAN scenario, with one AP and $(n - 1)$ stations. We work under the following assumptions:

(i) for each connection, $W^* = 1$;

(ii) each session corresponds to a file transfer; file sizes are independent and identically distributed. In particular, X is exponential with mean $E[X]$;

(iii) after the completion of a file transfer, a node generates a random think time, which is exponentially distributed with mean λ^{-1}.

Let us denote by $J(t)$ the number of connections active at time t; $J(t)$ is a continuous time random process which takes values in the state space $\mathcal{S} = \{0, 1, \ldots, n - 1\}$. We assume that the active sessions are served according to a processor sharing discipline with state–dependent service rate. Furthermore, we denote by C_k the "system capacity" (in terms of effective aggregated TCP rate) when there are k connections active, expressed in b/s. Hence, any of the k active connections is served at a rate $\frac{C_k}{k}$. Since files are exponentially distributed, the rate at which each of the k active

[1]In reality ns2 implementation of TCP provides bit padding and, hence, all the $\left\lceil \frac{X}{L_{TCP}} \right\rceil$ packets generated present a payload of size equal to L_{TCP}. Furthermore, no explicit connection closing is provided.

sessions finishes is given by $\frac{C_k}{kE[X]}$. Hence, the rate at which one of the k active nodes moves to the OFF state is given by $\mu_k = \frac{C_k}{E[X]}$. Then, the process $J(t)$ is a continuous time Markov chain, whose transition structure is depicted in Fig. 6. This birth–death process can be solved to obtain the stationary probability distribution:

$$\pi_k = \frac{\frac{\lambda^k \cdot (n-1)(n-2)\dots(n-k)}{\mu_1 \dots \mu_k}}{\left(1 + \sum_{i=1}^{n-1} \frac{\lambda^i (n-1)(n-2)\dots(n-i)}{\mu_1 \dots \mu_i}\right)}, \; k = 0, 1, \dots, n-1. \tag{16}$$

The "system capacity" C_k is computed, under assumption (i), according to the collision model: $C(k) = k \cdot S_{TCP}^{coll}(k, 1)$. Note that, by using the collision–free model, our model reduces to a classical processor sharing system, for which $C_k = C \; \forall k$, leading to the model used in [16] for assessing the performance of HTTP traffic over EDGE. The average session delay can then be computed by means of Little's Law, summing the mean setup time:

$$E[T_{session}] = E[T_{setup}] + \frac{E[Y]}{\overline{\lambda}}, \tag{17}$$

where $E[Y] = \sum_{k=0}^{n-1} k\pi_k$ and $\overline{\lambda} = \lambda \sum_{k=0}^{n-2} (k+1)\pi_k$. Insensitivity of the stationary probability distribution π_k with respect to the file size and think time distributions follows from an approach similar to that in [6]. In particular, this implies insensitivity of the mean session delay.

In Fig. 7 a first comparison between analytical and numerical results is presented. Think times are exponentially distributed with mean $\lambda^{-1} = 10$ s, the average file size is 30 Kbytes and two file size distributions are used, Pareto (with shape factor $\beta = 1.5$) and exponential. Simulations are run for both $W^* = 1$ and $W^* = 1000$; each stations download 300 files and buffers are dimensioned in such a way that no packet drops take place. Notice that, when $W^* = 1$, similar results, in terms of mean session delay, are achieved by Pareto and exponential distributions. Further, in this case the ns2 outcomes show a good match with the analytical results. For exponentially distributed file sizes, the mean session delay shows a very weak dependence on the value of W^*. On the other hand, in the case of Pareto distribution, the mean session delay shows a high variability with W^*. In particular, a very large value of W^* leads to a much higher mean session delay. This phenomenon can be understood as follows: assume there is only one session active, which corresponds to the transfer of a very large file (an "elephant"), and that the TCP window size has

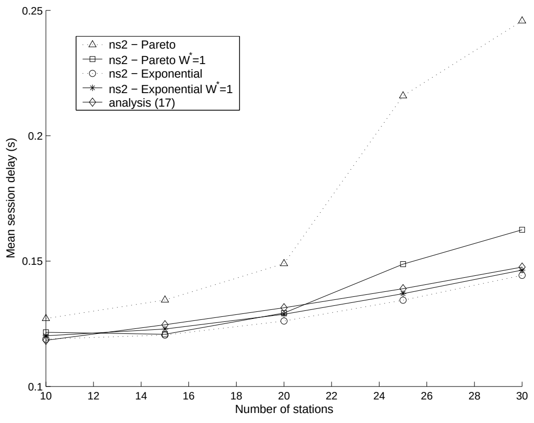

Figure 7: Mean delay for short–lived TCP flows vs. number of stations.

reached its stationary behavior and is thus equal to W^*. In such conditions, if the AP transmission queue is served according to a FCFS policy, the elephant is going to get most of the network resources, arising starvation problems for the other incoming sessions (a similar behavior has already been observed in [17] in the context of wireline networks). In practice, an increase in the value of W^* leads to a smaller mean session delay for elephants, while increasing it for small files transfer (the "mice"). This is basically due to the implicit unfairness of the FCFS policy according to which the packets waiting in the AP transmission queue are served. By setting $W^* = 1$ (or, alternatively, by changing the scheduling at the AP queue), we force a round–robin scheduling of the various active sessions, so that the increase in fairness leads to a decrease in session delay. Furthermore, note that, with $W^* = 1$, we achieve not only better performance in presence of short–lived flows, but also get insensitivity w.r.t. the file size distribution. It is indeed clear that, from a network designer point of view, having a system which behaves in a predictable manner regardless of the traffic characteristics (which indeed may change due to protocol evolution etc.) is a very desirable property, so that setting a low value for the advertised window size represents a good choice from an engineering perspective.

To check the ability of the model to track the system behavior for high loads (where the terms "load" refers to the queueing model), avoiding numerical instability problems related to $ns2$, we acted on the mean think time. Results are plotted in Fig. 8, where the analytical results are compared with the outcomes of numerical simulations (we reported the 95% confidence interval) for $n = 11$, $W^* = 1$ and Pareto–distributed file size.

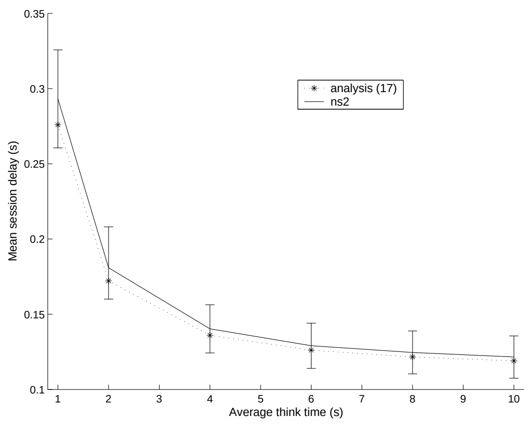

Figure 8: Mean delay for short–lived TCP flows vs. average think time.

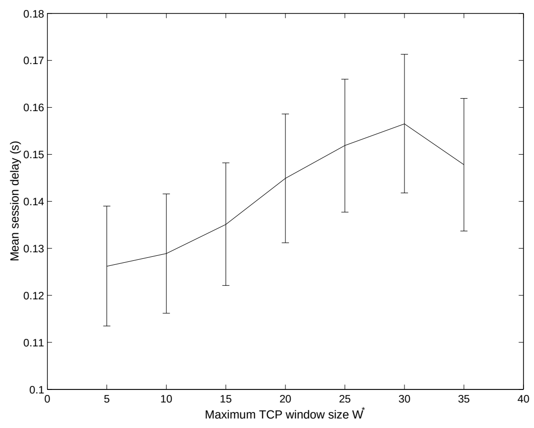

Figure 9: Mean delay for short–lived TCP flows vs. W^*, $B = 30$, $n = 21$.

All the analysis above relies on the assumption of infinite AP transmission buffer and (consequently) no timeout expiration. Let us now focus on a more realistic scenario, where the buffer size B (expressed in packets) is indeed finite. Then, we can distinguish among three different regimes. For $W^* \ll B$ we retrieve the behavior previously described. The case $W^* \approx B$ is the one which leads to the worst performance. In this regime, indeed, the unfairness of the FIFO policy according to which the buffer is served causes mice to experience losses, leading to a lower degree of fairness and higher session delay. For $W^* \geq B$, on the other hand, also elephants will experience losses; in this regime TCP effectively reduces the congestion and provides a "fair" sharing of the network resources. In this case, then, we expect the mean session delay of elephants to increase (owing to frequent losses) and that of mice to decrease (since the elephants are now frequently reducing their window sizes). As a result, an overall performance improvement can be expected. It is also natural to expect that any value of $W^* > B$ would give same performance.

In Fig. 9 we reported some simulation results of these regime. With $n = 21$ nodes, we set the AP transmitting buffer size to $B = 30$ packets, simulated the transfer of Pareto–distributed files and varied W^* from 5 to 35 (packets); note that, as expected, in this region the session delay is first an increasing function of $W^* \leq B$, and then flattens out when $W^* > B$.

4.3 On the use of delayed ACK for short TCP connections

The analysis (both the affine model for the single station case and the modified Engset model for the multiple stations case) can be extended to account for the use of delayed ACK, under the assumption $W^* = d$, yielding similar results. The problem is here to account for the interaction with the receiver–based delayed ACK timer, which can expire (under our assumptions) at the end of a file transfer. As we show in [11], the timeout interval may have a great impact on network performance and, in particular, the choice of a high timeout value may have a negative effect on session delay.

Results for the single station case (obtained with $\tau_{out} = 40$ ms) are presented in Tab.3, for d varying from 1 (i.e. standard TCP) to 5. Simulations consisted in the transmission of 10000 Pareto–distributed files, with $W^* = 1000$ and a large buffers. We think there is no real reason to go beyond $d = 5$, since, as it is apparent from simulation results, increasing too much the number of delayed ACK has not a beneficial impact on network performance. This may be understood by considering the effect of the timeout at the end of the file transfer. Indeed, by increasing d, we increase the probability that the receiver will be forced to send an ACK due to timeout expiration. To investigate the impact on session delay of the slower congestion window growth, we simulated also the dynamic delayed ACK scheme proposed in [5]. Surprisingly, we obtain results worse than the "standard" $d = 2$ option, namely an average session delay of 0.1080 s, with a 95% confidence interval of $[0.0980, 0.1180]$ s.

On the whole, we may thus conclude that, by choosing an initial window size equal to d, and by appropriately tuning the timeout interval, the delayed ACK option, with $d = 2, 3, 4$ may overperform standard TCP and also the more complex multi–threshold scheme proposed in [5].

We evaluated also the impact on network performance

$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
0.1064 $[0.0928, 0.1199]$	0.1044 $[0.0934, 0.1154]$	0.1053 $[0.0951, 0.1156]$	0.1043 $[0.0945, 0.1141]$	0.1090 $[0.0993, 0.1188]$

Table 3: Session delay for short TCP connections with delayed ACK, mean and 95% confidence interval (s), $\tau_{out} = 40$ ms.

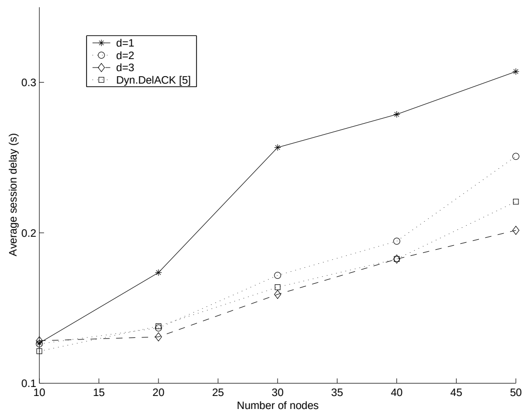

Figure 10: Average session delay vs. number of stations.

of the delayed ACK option in presence of concurrent TCP connections. By taking advantage of the results for the single station case, we limited ourselves to the cases $d = 1, 2, 3$ and to the scheme of [5]. The initial congestion window size was varied accordingly. HTTP traffic is modelled as done for the standard TCP case, with Pareto–distributed file sizes and exponential think times. Simulations were run with $W^* = 1000$. Results are presented in Fig. 10. Notice that, for a large number of stations, delayed ACK lead to a considerable reduction of the mean session delay, making it an attractive choice for hot–spot scenarios.

5 Conclusions

In this paper we have analysed the network performance of an 802.11–based WLAN in presence of HTTP traffic. We have derived bounds for the TCP throughput in presence of n_c competing persistent connections, for both standard TCP and generalized delayed ACK techniques. These results have then been employed to develop a queueing model which has enabled us to estimate the mean session delay for short–lived flows. Insensitivity of the results with respect to the file size distribution has been shown to hold in presence of a small value of the advertised window size. In presence of a large advertised window size, unfairness problems do arise, leading to an overall performance worsening for heavy–tailed distributed file size. The effect of limited buffers size has been investigated, and its impact on both fairness and session delay has been discussed. Generalized delayed ACK techniques have also been analysed, and it has been shown that, under a careful tuning of some system parameters, they can lead to an overall performance enhancement in terms of both throughput and session delay.
These results provide precious guidelines for network designers, showing in particular the positive effect of a small advertised window size option on network behavior and the benefits deriving from the use of generalized delayed ACK techniques.

6 Acknowledgments

This work was supported by the *Indo–French Center for Promotion of Advanced Research (IFCPAR)* under research contract number 2900–IT and by the EURO NGI network of excellence. The work of D. Miorandi was partially supported by *Fond. A. Gini.*

References

[1] *IEEE standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std., Aug 1999.

[2] V. Jacobson and M. J. Karels, “Congestion avoidance and control,” in *Proc. of Sigcomm*, Stanford, CA, 1988.

[3] R. Braden, “Requirement for Internet hosts – communication layers,” RFC 1122, Oct 1989.

[4] M. Allman, “On the generation and use of TCP acknowledgment,” *ACM Computer Commuication Review*, vol. 28, Oct 1998.

[5] E. Altman and T. Jimenez, “Novel delayed ACK techniques for improving TCP performance in multihop wireless networks,” in *Proc. PWC*, Venice, Italy, 2003.

[6] D. P. Heyman, T. V. Lakhsman, and A. L. Neidhardt, “A new method for analysing feedback–based protocols with applications to engineering

web traffic over the Internet," in *Proc. of Sigmetrics*, Seattle, 1997.

[7] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Sel. Areas in Comm.*, vol. 18, pp. 535–547, 2000.

[8] *Supplement to 802.11-1999,Wireless LAN MAC and PHY specifications: Higher Speed Physical Layer (PHY) extension in the 2.4 GHz band*, IEEE Std., Sep 1999.

[9] M. Allman, S. Floyd, and C. Partridge, "Increasing TCP's initial window," RFC 2414, Sep 1998.

[10] Y. C. Tay and K. C. Chua, "A capacity analysis for the IEEE 802.11 MAC protocol," *Wireless Networks*, vol. 7, pp. 159–171, 2001.

[11] D. Miorandi and E. Altman, "On the effect of feedback traffic in IEEE 802.11b WLANs," INRIA, Tech. Rep. RR4908, 2003. [Online]. Available: http://www.inria.fr/rrrt/rr-4908.html

[12] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New insights from a fixed point analysis of single cell IEEE 802.11 WLANs," INRIA, Tech. Rep. RR5218, 2004. [Online]. Available: http://www.inria.fr/rrrt/rr-5218.html

[13] K. Chen, Y. Xue, and K. Nahrstedt, "On setting TCP's congestion window limit in mobile ad hoc networks," in *Proc. ICC*, Anchorage, USA, 2003.

[14] The network simulator ns2. [Online]. Available: http://www.isi.edu/nsnam/ns

[15] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," in *Proc. INFOCOM*, S. Francisco, US, 2003.

[16] N. K. Shankaranarayanan, Z. Jiang, and P. Mishra, "Performance of a shared packet wireless network with interactive data users," *Mobile networks and applications*, vol. 8, pp. 279–293, 2003.

[17] A. A. Kherani and A. Kumar, "On processor sharing as a model for TCP controlled HTTP–like transfers," in *Proc. of ICC*, Paris, France, 2004.

APPENDIX

Impact of the maximum TCP congestion window size W^*

Under the assumptions reported in §3 (no timeout expiration and no packet losses), the congestion window would grow indefinitely. Assume that there is a maximum limit, W^* on the TCP congestion window size (this indeed may be thought as the limit imposed by the receiver by means of an appropriate setting of the advertised window). Due to the aforementioned assumptions, after some period (transient phase), TCP congestion window will stabilize at W^*. For the moment, let us focus on a single connection: the couple (X_S, X_D) will denote the number of packets queued at the sender and receiver side, respectively. Clearly, we have that $X_S(t) + X_D(t) = W^*$. Furthermore, assume that the reception of a MAC layer ACK represents a renewal instant for the process $(X_S(t), X_D(t))$[2]. Then, embedding at such instants, $(X_S(n), X_D(n))$ represents a Markov chain, as depicted in Fig. 11 for the case $W^* = 4$. Due to the particular transition struc-

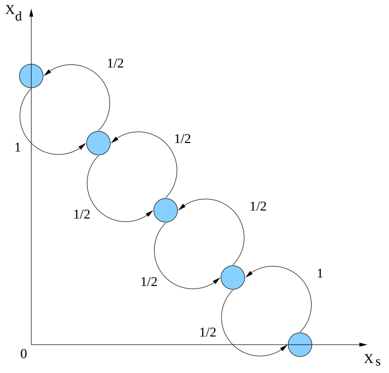

Figure 11: Structure of the embedded Markov chain, $W^* = 4$.

ture, the resulting Markov chain can be easily solved, leading to the following stationary probability distribution:

$$P[(X_S(n), X_D(n)) = (\alpha, \beta)] = \pi_{\alpha,\beta} = \\ = \begin{cases} \frac{1}{W^*} & (\alpha,\beta) = (a, W^* - a),\ a = 1, \dots, W^* - 1, \\ \frac{1}{2W^*} & (\alpha,\beta) = (a, W^* - a),\ a = 0, W^*, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

To compute the throughput we may work in the following way: except for the "extremal" states, in all the other states a packet is acknowledged with probability 1/2, due to the fair access to the medium provided by the 802.11 DCF [3]. In the state $(0, W^*)$ this occurs with probability one. However, the time it takes to

[2]This is strictly true only if we assume a geometric backoff is present at the MAC layer, which is clearly just an approximation of the complex 802.11 MAC behavior. Note that, by appropriately expanding the state space, we could account for the real protocol behavior.

[3]Note that DCF provides only long–term fairness, but suffers from short–term unfairness.

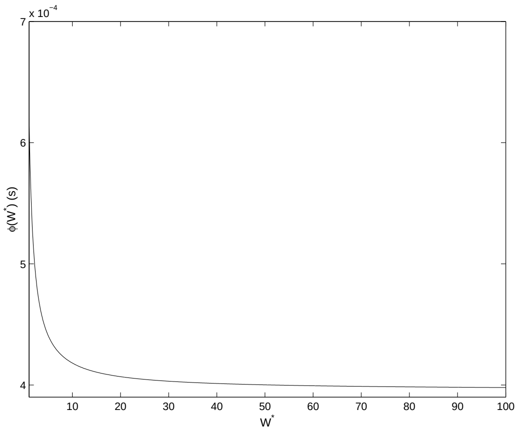

Figure 12: Behavior of $\phi(\cdot)$.

accomplish the transition is state–dependent. In particular, considering that collisions are possible in all but the extremal states, we have, using Markov renewal reward theorem:

$$S = L\cdot\left[\frac{1}{2}\sum_{i,j=1,\ldots,W^*-1}\pi_{i,j} + \pi_{0,W^*}\right]\cdot\left\{\sum_{i,j=1,\ldots,W^*-1}\pi_{i,j}\cdot\right.$$

$$\cdot\left[\frac{1}{2}(T_P+T_A)+T_{tbo}+T_w\right] + \pi_{0,W^*}\left(T_A + T_{slot}\frac{CW_{min}-1}{2}\right)$$

$$\left.+\pi_{W^*,0}\left(T_P + T_{slot}\frac{CW_{min}-1}{2}\right)\right\}^{-1}. \quad (19)$$

After some algebra, the equation above reduces to:

$$S = \frac{L}{T_A + T_P + \phi(W^*)}, \quad (20)$$

where $\phi(W^*) = \frac{2(W^*-1)(T_{tbo}+T_w)+T_{slot}(CW_{min}-1)}{W^*}$ is the average time spent in collision resolution. It is then routine to check that, if $W^* \geq 1$:

$$\frac{\partial\phi}{\partial W^*} = \frac{2(T_{tbo}+T_w) - T_{slot}(CW_{min}-1)}{(W^*)^2} < 0,$$

so that $\phi(\cdot)$ is monotone decreasing. The behavior of the function $\phi(\cdot)$ is plotted in Fig. 12. As a result, an increase in W^* leads to a throughput enhancement; the analysis may be extended to the case of delACK with similar results. Theoretical and simulation results for $W^* = d$ and $W^* \gg d$ are plotted in 13. These results complete the analysis of [13] for the case of a single–hop network.

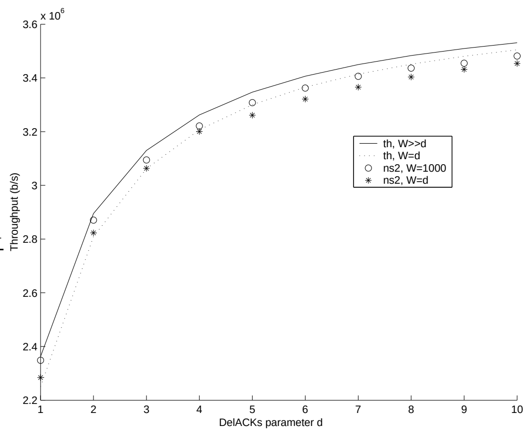

Figure 13: TCP throughput for the single–station case vs. delACK parameter d for two values of W^*.