Modelling TCP flows over an 802.11 wireless LAN

Taka Sakurai and Stephen Hanly

ARC Special Research Centre for Ultra-Broadband Information Networks (CUBIN)\(^1\)
Dept. of Electrical and Electronic Engineering, The University of Melbourne
Vic 3010, Australia
\{t.sakurai, s.hanly\}@ee.uow.edu.au

Abstract: We present an analysis of the throughput of TCP flows transmitted over IEEE 802.11 wireless LAN connections, for the case when TCP flow control is restricted by the receiver’s advertised windows. Both persistent and non-persistent TCP connections are considered. We obtain an expression for the aggregate TCP throughput of multiple persistent TCP connections, and then use this result in a queueing model of non-persistent TCP connections. The theoretical models are validated with simulation. Our results show that the total bandwidth is shared fairly between TCP flows, and that the aggregate TCP throughput is insensitive to the number of flows.

1. Introduction

Wireless local area networks (WLAN’s) using the IEEE 802.11 standard are now in widespread use. As in the wider Internet, the bulk of traffic on a typical WLAN consists of applications such as web browsing that are carried over the TCP transport protocol. In this paper, we analyze the throughput of TCP, as a function of the number of flows, when transmitted over an IEEE 802.11 WLAN.

There have been a number of important models developed for predicting the performance of WLAN’s using the 802.11 MAC protocol. Bianchi [4] provided a Markov chain model, which was then simplified by Tay and Chua [12], who showed that a simple “mean value” analysis using fixed point methods, could be used to get accurate predictions of network performance. These models capture the behaviour of the 802.11 MAC protocol as it applies to “greedy” sources and do not consider additional constraints on packet transmission rates imposed by higher layer protocols such as TCP.

The flow control aspect of TCP strongly influences the MAC layer behaviour, including the probabilities of packet collisions between wireless stations. Recent work by Miorandi, Kherani and Altman [9] began an investigation of TCP over 802.11; we extend this work to provide a demonstrably accurate model that can handle a wide variety of network scenarios, including the coexistence of both upstream and downstream TCP flows.

In the present paper, we consider a 802.11 WLAN consisting of an access point (AP) and a number of wireless stations (STAs). The stations can be either TCP sources, generating upstream TCP traffic sent directly to the AP, or TCP sinks, receiving downstream traffic directly from the AP. We begin with the assumption that the TCP sources are persistent, in the sense that they always have something to send. Under the additional assumption that the TCP sending windows are constrained to moderate sizes by the receiver advertised windows, we develop an analytical expression for TCP throughput as a function of the number of flows. We validate our model with simulations, and demonstrate that the 802.11 protocol is fair and that the total throughput is quite insensitive to the number of flows.

Our results for persistent sources allow us to propose a processor sharing queueing model to handle the more realistic case of non-persistent sources. In this model, a node alternates between an active mode, when it is engaged in a TCP transaction over the wireless network, and an idle mode, when the user is engaged in think time. Using this model, we can compute average transaction latencies, which reflect the quality of service as perceived by the end user. This model can be used in network dimensioning. Note that we are not concerned here with finer-grained measures of quality of service, such as per-packet delay or jitter, as these measures are not directly relevant to TCP traffic.

The rest of this paper is organized as follows. In Section 2., we describe the IEEE 802.11 Medium Access Control (MAC) protocol and some salient aspects of TCP. In Section 3., we present our analysis of TCP throughput for persistent connections and compare our theoretical predictions with simulation. In Section 4., we present a queueing model for non-persistent sources, and validate it with simulation. Finally, in Section 5., we give some concluding remarks and hints for future work.

2. Overview of 802.11 MAC and TCP

2.1. MAC

The distributed coordination function (DCF) of the 802.11 MAC provides CSMA/CA access for multiple STAs sharing a common channel. This mode of operation is the focus of the present paper. We provide a brief description below; for details, refer to [7].

The MAC layer specifies exactly when a backlogged STA can send a packet across the wireless multiple-access channel. The first step is to wait a period of time during which the channel is sensed idle. The length of this period depends on the elapsed time since the previous data packet transmission from the same node. This is because a post-transmission backoff takes place after every packet transmission. A pre-transmission backoff also occurs when a new packet arrives and the carrier is sensed busy. Prior to a backoff interval, the channel must be sensed idle for a guard period known as the distributed interframe spacing time, \(t_{dfs}\). Backoff periods are slotted, with a slot time denoted by \(t_{slot}\). When backoff is initiated, a random backoff timer value is selected, representing the number of idle slots that must take place.

\(^1\)CUBIN is an affiliated program of National ICT Australia
before the next packet can be transmitted. A contention window variable, \( W \), is maintained and the random number of slots selected is drawn uniformly from the set \( \{0, 1, 2, \ldots, W - 1\} \). The contention window value, \( W \), is initially set to be \( W_{\text{min}} \), the minimum possible value for the window size. The backoff time counter is decremented as long as the channel is sensed idle. It is \textit{frozen} when a packet transmission is detected on the channel, and reactivated after the channel is sensed idle again for a guard period. The guard period is equal to \( t_{\text{ifs}} \) if the transmitted packet was error-free, and equal to the extended interframe spacing time if the packet was errored. The station transmits when the backoff time counter reaches zero. A collision occurs when the counters of two or more stations reach zero in the same slot.

If a packet transmission is successful, the receiving MAC layer sends an ACK after a short interframe spacing time, \( t_{\text{ifs}} \). If the packet transmission is unsuccessful (an event indicated by an ACK timeout at the sending STA), the contention window size is doubled, and another backoff period is initiated. This doubling continues until the window has reached its maximum possible value, \( W_{\text{max}} \). We define \( W_{\text{max}} = 2^{m'} W_{\text{min}} \), where \( m' \) is the number of window doublings allowed.

If the packet is still unsuccessful after \( m' \) attempts, the window will be held fixed at \( W_{\text{max}} \) for the remaining attempts, until the packet is successful, or until the maximum number of attempts, \( m \), takes place. If the packet is unsuccessful after \( m \) attempts, the MAC layer gives up, and any further attempts will be the responsibility of higher-layer error recovery, such as that of TCP.

In this paper, we choose MAC and physical layer parameters consistent with an 802.11b system. In Table 1, we list the principal parameters used. In addition to those parameters already defined, we list the raw transmission data rate \( r_{\text{data}} \), the transmission rate for control packets \( r_{\text{ctrl}} \), the length of an ACK packet \( l_{\text{ack}} \), the length of the MAC layer header \( l_{\text{mac}} \), the length of the TCP/IP header \( l_{\text{tcpip}} \), and the transmission time for the physical layer header \( t_{\text{phy}} \).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{\text{data}} )</td>
<td>11 Mb/s</td>
<td>( l_{\text{ack}} )</td>
<td>112 bits</td>
</tr>
<tr>
<td>( r_{\text{ctrl}} )</td>
<td>1 Mb/s</td>
<td>( l_{\text{mac}} )</td>
<td>224 bits</td>
</tr>
<tr>
<td>( l_{\text{tcpip}} )</td>
<td>320 bits</td>
<td>( t_{\text{phy}} )</td>
<td>192 ( \mu )s</td>
</tr>
<tr>
<td>( l_{\text{slot}} )</td>
<td>20 ( \mu )s</td>
<td>( t_{\text{ifs}} )</td>
<td>10 ( \mu )s</td>
</tr>
</tbody>
</table>

Table 1: 802.11b MAC and PHYS parameters

2.2. TCP

TCP is a window-based protocol, which means that the sender keeps track of how many packets are unacknowledged and controls this number in response to congestion feedback from the network [11]. It is “self clocking” in the sense that new packets are injected into the network when TCP ACK packets arrive back at the sender. During the lifetime of a TCP connection, the window is adjusted by the protocol to perform a flow control function. At all times, the receiver is able to constrain the window size via the advertised window (AWND) that it places in every TCP ACK passed to the sender. The TCP window size can never exceed the current value of AWND.

For many operating systems, the default AWND is not large (for example, for Microsoft Windows 95/98/NT it is 8760 bytes and for Microsoft Windows 2000, it is 17520 bytes); as a consequence, the sender may never be able to fill up the network pipe. On the other hand, a sufficiently small AWND will mean that buffer overflow at the network bottleneck can be avoided entirely. The equilibrium behaviour in this case is characterized by a fixed window size, fixed at the value of AWND. This issue has been considered by Miorandi, Kherani and Altmann [9], and in other works e.g. [2].

A judicious setting of AWND can greatly increase the throughput of short TCP connections, by eliminating buffer overflow and reducing the delay experienced by packets waiting in the AP buffer. This approach may not be advantageous to flows with large propagation delays, but will certainly improve overall performance if most flows in the network have relatively short propagation delays. In our work, we assume that the AWND is fixed, and that end-to-end propagation delays across the Internet are short enough to be ignored. Thus, for downstream flows, a new packet arrives at the AP from the sender immediately after the AP sends a TCP ACK to the sender and for upstream flows, a TCP ACK arrives at the AP from the receiver immediately after the AP sends a packet that it obtained from the STA. Our analysis assumes that the MAC layer reliably retransmits lost packets, so that packet loss does not occur at the IP layer. IP-layer packet loss does occur occasionally in the simulations, but is rare enough to be neglected in the analysis.

3. Throughput analysis

In this section, we develop an analytical model for TCP throughput over IEEE 802.11 WLANs. Our throughput model can predict TCP throughputs in a scenario where there is one or more persistent upstream flows or one or more persistent downstream flows. We restrict attention to the case when there is one TCP flow per STA, but the model can be readily adapted to the general case. Below, we consider various network configurations, starting with the case of one STA and AWND = 1, then proceeding to the case of AWND > 1 and then finally generalizing to the case of multiple STAs. In the latter two cases, we make the simplifying assumption that the buffer at the AP is never empty. Simulation experiments with ns-2 confirm that this is a reasonable assumption.

3.1. One STA, AWND = 1

In this case, the behaviour of TCP reduces to that of a stop-and-wait protocol. Obviously, there are no collisions in this scenario, but there will be some delay due
to backoff activity, since both the AP and STA perform post-transmission backoffs.

The throughput $S$ can be expressed as

$$S = \frac{t_{data}}{t_{cycle}},$$

where $t_{data}$ is the TCP packet payload in bits and $t_{cycle}$ is the average time between two TCP data packet transmissions by the TCP sender. We refer to this time as a cycle. Equation (1) applies to both upstream and downstream situations. The time $t_{cycle}$ is given by

$$t_{cycle} = t_{data} + t_{ack} + t_b,$$

where the transmission times $t_{data}$ and $t_{ack}$ are defined to be the time from commencement of data transmission to reception of ACK, for TCP data and TCP ACK packets, respectively. In other words,

$$t_{data} = t_{phy} + \frac{t_{mac} + t_{cpp}}{r_{data}} + t_{sifs} + t_{phy} + \frac{t_{ack}}{r_{ctrl}} + t_{difs},$$

$$t_{ack} = t_{phy} + \frac{t_{mac} + t_{cpp}}{r_{data}} + t_{sifs} + t_{phy} + \frac{t_{ack}}{r_{ctrl}} + t_{difs}.$$ 

Finally, $t_b$ is the average delay contribution due to backoffs, which will be determined in the analysis below. The AP and STA backoff periods (measured in slots) are sampled uniformly from the set \(\{0, 1, \ldots, W_{min} - 1\}\). Denote by $B_{AP}$ and $B_{STA}$ the backoff samples of the AP and STA in a given cycle. Since the rival backoff clocks count down simultaneously, the effective backoff delay in any given cycle will be the maximum of $B_{AP}$ and $B_{STA}$. The delay $t_b$ is an average over all cycles and is measured in seconds; therefore, it is given by

$$t_b = t_{slot}E(\max(B_{AP}, B_{STA})).$$

Finally, it can be shown (see Miorandi and Altman [8]) that

$$E(\max(B_{AP}, B_{STA})) = \frac{W_{min} - 1)(4W_{min} + 1)}{6W_{min}}.$$

### 3.2. One STA, AWND > 1

The assumption AWND > 1 means that collisions between TCP data packets and TCP ACK packets are possible. It also means that the TCP sender, after transmitting a data packet, does not always need to wait for a TCP ACK before sending another packet, since it may already have another packet in queue. In fact, to simplify the analysis, we make the key assumption that the transmit buffer at the TCP sender is never empty. We approach the analysis of throughput by focussing on the average time $t_{cycle}$, between successfully transmitted packets by the TCP sending node.

Our second key assumption is that the system achieves an equilibrium state where the average rate of successfully transmitted packets from the TCP sending node equals the average rate of successfully transmitted TCP ACK’s from the TCP receiving node. In other words, the average time between successfully transmitted packets by the TCP receiving node is also given by $t_{cycle}$.

To understand the various components that comprise $t_{cycle}$, it is instructive to view $t_{cycle}$ from the perspective of the TCP sending node. In a $t_{cycle}$ period, the TCP sending node successfully transmits one packet, freezes its backoff counter when the TCP receiving node successfully transmits one packet, and increases its backoff stage whenever there is a collision. We can write

$$t_{cycle} = t_{data} + t_{ack} + t_{bc},$$

where $t_{bc}$ is the average time occupied by backoffs and collisions. Following Bianchi [4], we assume that each transmitted packet has a constant and independent collision probability $p$. With this assumption, the quantity $t_{bc}$ can be written as

$$t_{bc} = (1-p)A_0 + p(1-p)A_1 + \ldots + p^{m-1}(1-p)A_{m-1},$$

where $m$ is the maximum retransmission count and $A_k$, $k = 0, 1, \ldots, m - 1$, is the sum of the collision occupancy and average backoff time in the event of $k$ collisions. It can be shown that for $0 \leq k \leq m'$,

$$A_k = k t_{coll} + t_{slot} \sum_{j=0}^{k} \frac{2^j W_{min} - 1}{2},$$

and for $k > m'$,

$$A_k = k t_{coll} + t_{slot} \sum_{j=0}^{m'} \frac{2^j W_{min} - 1}{2} + (k - m') \frac{2^m W_{min} - 1}{2}.$$

Here, $t_{coll}$ represents the time duration of a collided packet plus the MAC acknowledgement timeout. The MAC acknowledgement timeout parameter defines the time that must elapse before the sending MAC layer deems that a collision has occurred. The value this parameter should take is left open by the 802.11 MAC specifications. We assume that $t_{coll} = t_{data}$, which is equivalent to assuming that the MAC acknowledgement timeout is equal to the time it would have taken to receive the ACK had the packet been successful. This is an assumption made in the ns-2 simulator.

In order to evaluate (5), we must first compute $p$. To do this, we again follow the fixed point approach of Bianchi [4], with minor modifications to account for the limit $m'$ on the number of window doublings and a limit $m$ on the total number of transmissions. The collision probability is defined as

$$p = 1/\tau,$$

where $\tau$ is the average backoff window and is given by

$$\tau = \frac{W_{min}(1-p)(1-(2p)^m')}{2(1-2p)} - 1 - p^{m'}$$

$$+ \frac{(2^m W_{min} - 1)(p^{m'} - p^m)}{2}.$$ 

We calculate $p$ from (9) and (8). The throughput can then be found from

$$S = \frac{t_{data}}{t_{data} + t_{ack} + t_{bc}}.$$
where $l_{data}$ is the TCP packet payload in bits.

### 3.3. Multiple STAs

We now develop a model for the case of $n \geq 2$ active STAs. In this case, either all TCP data packets (in the downstream case) or all TCP acknowledgement packets (in the upstream case) flow through the AP transmit buffer. Therefore, it is reasonable to assume that the AP transmit buffer never empties. We denote by $t_{cycle}$ the average time between packets successfully transmitted by the AP, and we assume that an equilibrium state is reached such that the combined effect of the $n$ active STAs is to yield a sequence of successfully transmitted TCP packets (either data packets or acknowledgement packets) with average spacing that is also equal to $t_{cycle}$.

We assume that at each transmission attempt of a given AP packet, there is a probability $p$ of colliding with STA packets, resulting in an average backoff window $\tau$. Our assumption on the average packet spacing for the STA population as a whole implies that each individual STA produces a sequence of successfully transmitted packets with an average spacing of $\tau/n_{cycle}$. Given this assumption, it is natural to assume that each STA has an average effective backoff time of $\frac{1}{\tau}$. Since the STAs may not always have a packet to send, the effective backoff time accounts for the durations of idle times as well as the true backoff durations. We then use $p = 1 - (1 - (n/\tau))^{-1}$, together with equation (9) for $\tau$, to determine $p$.

Another difference from the case studied in Section 3.2. is that we must now account for the time that the shared medium is occupied by collisions between packets from different STAs. Consider a tagged STA. We wish to determine the probability $q$ that in any given slot, a packet from the tagged STA has a collision with a packet from at least one other STA but not with a packet from the AP. The probability of a collision with a packet from at least one other STA is $1 - (1 - (1/\tau))^{-n} = 1 - (1 - (n/\tau))^{-1}$, and the probability of a collision with an AP packet is $1/\tau$. Therefore, we obtain

$$q = (1 - (1 - \frac{1}{\tau}))^{n-1}(1 - \frac{1}{\tau}).$$

For simplicity, we assume that the retransmissions of the tagged STA are performed according to a geometric distribution (in truth, it is a truncated geometric distribution). The total number of retransmissions per packet from the tagged STA is then given by $q/(1 - q)$, and the total occupancy of the channel $\beta$ due to collisions between STA packets is given by

$$\beta = t_{coll} q/(1 - q),$$

where $t_{coll}$ is the time occupied by a collided packet plus the extended interframe spacing time. In the case of downstream flows, this packet is a TCP ACK so

$$t_{coll} = t_{phy} + \frac{t_{mac} + t_{tcpip} + l_{data}}{r_{data}} + t_{sifs} + t_{phy} + \frac{t_{ack}}{r_{ctrl}} + t_{disfs},$$

while for upstream flows, the packet is a TCP data packet, and so we have

$$t_{coll} = t_{phy} + \frac{t_{mac} + t_{tcpip} + l_{data}}{r_{data}} + t_{sifs} + t_{phy} + \frac{t_{ack}}{r_{ctrl}} + t_{disfs}.$$

The aggregate or total throughput is calculated from

$$S = \frac{l_{data}}{l_{data} + t_{coll}} + l_{bc} + \frac{l_{data}}{r_{data}},$$

where $t_{bc}$ can be calculated from (5), (6) and (7). Obviously, the per-flow throughput is simply $S/n$. In the case of upstream flows,

$$t_{coll} = \max(t_{ack}, t_{phy} + \frac{t_{mac} + t_{tcpip} + l_{data}}{r_{data}} + t_{disfs}),$$

since the AP must wait for a timeout for the packet it transmitted (a TCP ACK) but it must also wait until the colliding STA’s data packet is finished before the collision period is over. In the case of downstream flows,

$$t_{coll} = t_{data}$$

since the duration of the TCP data packet plus the timeout defines the collision period.

### 3.4. Numerical results

To verify the accuracy of our model for TCP throughput, we used the popular ns-2 simulator [1] (version 2.27) configured to model an IEEE 802.11b WLAN with perfect wireless links. An advertised window size $AWND = 17$ packets was used, and the TCP packet payload was $l_{data} = 8000$ bits. The interface buffers at all nodes were set large enough to ensure zero buffer overflow.

In Figure 1, we plot analytical and simulation results for total TCP throughput and individual TCP flow throughput when there are $1 - 20$ STAs, each with an upstream TCP flow. All the simulation results are plotted with 95% confidence intervals. The simulated per-flow TCP throughput results are those for an arbitrarily selected flow; we found that the spread in throughputs across flows to be small. Figure 2 gives the corresponding results for downstream flows.

We observe that in spite of the many simplifying approximations made, the model is remarkably accurate. We note that there is a close similarity between uplink and downlink throughputs. Another interesting feature to emerge is that the total TCP throughput is insensitive to the number of flows. This implies that the collision probability does not increase significantly when the number of flows is increased. The explanation for this lies in the fact that all flows pass through the AP, which means that the AP constitutes a system bottleneck. Since the 802.11 MAC tries to give fair access to all contending nodes, the AP will, on average, receive its fair share of the channel but no more, and therefore will develop a large backlog of packets. The acknowledgement nature of TCP means that if the packets are in the buffer of the
AP, they cannot be in the buffers of the stations. As a result, only a small fraction of the STAs will have packets to send. Hence, the contention level in the network cannot increase significantly when the number of flows (stations) is increased.

4. Queueing analysis

The results of the previous section confirm that under certain conditions, concurrent persistent TCP flows share the available bandwidth fairly in a long-term average sense. In this section, we make use of these results to analyze the performance of a WLAN with a finite population of non-persistent TCP sources. We propose a model for this system based on an egalitarian processor sharing queue configured in a closed queueing network. The processor sharing queue has recently been recognized as a suitable modelling abstraction for elastic Internet traffic (see [10] and references therein). The closed queueing network model that we use has previously been applied by Heyman et al. [6] and Berger and Kogan [3] to a generic bottleneck link shared by a finite population of TCP sources. Miorandi, Kherani and Altman [9] propose a related closed queueing network with a state-dependent processor sharing queue to model an IEEE 802.11 WLAN with downstream TCP sessions. They use the state-dependent service rate feature of this queue to model the variation in the total TCP throughput as the number of active STAs is varied. As we saw in Section 3, this variation is small, so in the interests of simplicity, we choose to ignore the variation and use an egalitarian processor sharing queue. In an extension to [9], we are able to consider scenarios involving upstream TCP sessions as well downstream sessions.

The details of the network configuration are as follows. Let there be a total of $N$ STAs communicating with an AP. Suppose that $N_u$ of the STAs operate as TCP sources and $N_d = N - N_u$ STAs operate as TCP sinks. The TCP sources are involved in sessions that alternate between flows and idle periods, where the flows represent transfers of files and the idle periods correspond to “think” times. The sizes of the uploaded files are assumed iid with mean $l_u$ bytes, the sizes of the downloaded files are iid with mean $l_d$ and all idle times are iid with mean $1/\lambda$ seconds.

Our closed queueing network model consisting of an egalitarian processor sharing (PS) node and an infinite server (IS) node. The PS node models the fair sharing behaviour of active sources and the IS node models the sources when they are idle. We let $S$ denote the total service rate in bits/sec of the PS node; the processor sharing assumption means that if there are $k$ customers at the node, each customer receives a rate $S/k$. We consider two classes of customers, corresponding to the sets of upstream and downstream TCP sessions. The average service time in the PS queue when there is one downstream flow is then $1/\mu_d := l_d/S$, and the corresponding quantity for a single upstream flow is $1/\mu_u := l_u/S$.

The queueing model described above is analyzed in [5]. From equation (5.21) of [5], we obtain the stationary joint probability of $n_u$ upstream flows and $n_d$ downstream flows in the PS queue:

$$
\pi_{n_u,n_d} = \frac{\binom{N_u}{n_u} \binom{N_d}{n_d} \rho_u^{n_u} \rho_d^{n_d} (n_u + n_d)!}{\sum_{k_u=0}^{n_u} \sum_{k_d=0}^{n_d} \binom{N_u}{k_u} \binom{N_d}{k_d} \rho_u^{k_u} \rho_d^{k_d} (k_u + k_d)!},
$$

where $\rho_u = \lambda/\mu_u$ and $\rho_d = \lambda/\mu_d$.

In our application, the service rate $S$ of the processor sharing queue corresponds to the TCP throughput of a WLAN; therefore, we obtain the numerical value of $S$ from the model of Section 3, specifically from equation (10). We are able to use a single value of $S$ to represent any number of upstream and downstream traffic because, as we saw in Section 3, the total TCP throughput is insensitive to the number of users and the direction of the flows.

4.1. Verifying the processor sharing model

In this section, we compare predictions of the processor sharing model with results from the ns-2 simulator. First, we consider a scenario with downstream sessions only, then with both upstream and downstream sessions. For each scenario, we perform two simulation experiments with different flow size distributions and means. For all experiments, we used exponential idle time distributions with $1/\lambda = 5$ secs.

For the downstream-only experiments, we set $N_u = 0$ and $N_d = 10$. For experiment A, the flow size distributions were exponential, with flow size mean of 100 kBytes. For experiment B, the flow size distributions were Pareto with shape factor 1.5, and a flow size mean of 500 kBytes. In Figure 3, we display the probability mass function (pmf) of the number of active flows for experiment A. The x-axis represents the number of active flows. The corresponding results for experiment B are displayed in Figure 4. For both experiments, we observe good agreement between our model and simulation. Among other things, these results support the flow size distribution insensitivity property predicted by the model.

For the experiments with both upstream and downstream sessions, we set $N_u = 1$ and $N_d = 10$. For experiment C, the flow size distributions were exponential, with upstream and downstream flow size means of 50 kBytes and 100 kBytes, respectively. For experiment D, the flow size distributions were Pareto with shape factor 1.5, and upstream and downstream flow size means of 200 kBytes and 500 kBytes, respectively. In Figures 5 and 6, we display the probability mass function (pmf) of the number of active flows for experiments C and D, respectively. To distinguish upstream and downstream flows, we let the x-axis represent the number active downstream flows and we plot analysis and simulation results for the event of no upstream flows and the event of one upstream flow. Once again, we find good agreement between the model and simulation.

5. Conclusions

We developed an analytical model to estimate throughput of persistent TCP flows over an IEEE 802.11...
WLAN when the TCP AWNDs are constrained such that there is zero buffer overflow. We found excellent agreement between the model predictions and simulation results from ns-2. We observed that when the AWNDs are of uniform size, there is fair sharing of the total bandwidth. The aggregate TCP throughput was found to be insensitive to the number of flows, and upstream and downstream throughputs were virtually identical. For non-persistent TCP sources, we found that a simple closed queueing network with an egalitarian processor sharing queue provides a very good model of system behaviour.

In this work, we considered static receiver advertised windows. In future work, we intend to investigate methods for dynamically controlling the receiver windows to avoid buffer overflow and underflow. We also hope to validate our models with measurements on an experimental test-bed.

REFERENCES


Figure 6: Experiment D: pmf of the number of active flows


