ABSTRACT
Sizing of IEEE 802.11 wireless LAN (WLAN), defined as the problem of finding the maximum number of users that can be supported, is essential for efficient performance of the WLAN. Although performance of the medium access control (MAC) of 802.11 WLANs has been studied extensively through analytical models, explicit models for sizing, based on the application profile, are not known. Since existing analytical models of 802.11 WLANs start with the MAC layer traffic specification, their usage for sizing requires the translation of application load to the traffic profile at the MAC. In this paper, we provide such mapping, from application usage profile to MAC traffic, and from MAC throughput (obtained using analytical models) to application throughput. We first evaluate analytical models for 802.11 MAC from the sizing perspective and then propose an approximate sizing method. We validate our method through extensive ns-2 simulations which show that the number of users suggested by our tool are within 13% of those derived from simulations for a majority of the operating parameters.

1. INTRODUCTION
Wireless Local Area Networks (WLANs) based on IEEE 802.11 standard [2] are fast becoming as ubiquitous as wired LANs. The 802.11 standard and its extensions 802.11b, 802.11a and 802.11g define the medium access control (MAC) as well as the physical layer protocols for wireless stations. Wireless hot-spots are emerging at public places such as hotel lounges, airport lobbies, university campuses, etc. Such hot-spots are characterized by a high concentration of users in a small geographical area. Although providing access to as many users as possible has been the initial objective, WLAN users have started expecting service levels similar to those with wired LANs. Commercial WLAN access services are also becoming common. Sizing of WLAN access networks therefore becomes essential in order to ensure that the perceived service quality does not deteriorate with increase in load and user population.

Unlike their wired counterparts, the performance of 802.11 WLANs varies widely, due to two important factors: wireless channel conditions and the effect of contention. For 802.11 WLANs, the collision domain is the Basic Service Set (BSS), served by one access point in the infrastructure mode of operation. Wireless channel performance depends on factors like noise, fading, attenuation, etc., that are determined by the environment in which a WLAN is being operated. The effect of contention within a BSS varies with parameters such as the number of users and the packet arrival rate, and can thus be controlled through careful selection of these operating parameters.

In this paper, we concentrate on one such parameter, i.e., the number of users. The problem, in the context of WLANs, can be defined as follows: given the usage profile of users (applications), find the maximum number of such users that can be supported within a BSS. Alternatively, given the number of users and their usage characteristics, estimate the number of BSSs (access points) required to support the users. This is essentially a WLAN sizing problem. Determining the capacity of the network (defined as the maximum throughput, subject to given constraints) is an important task in sizing. Since the capacity of 802.11 WLANs is itself dependent on the user population and the offered load, determining the maximum user population that can be supported becomes a non-trivial problem. In some cases, applications or users may assign constraints such as delay bounds or packet drop probability, complicating the problem further.

The use of analytical sizing models can result in simple tools for answering sizing questions. Such a tool would take as input the application usage profile along-with WLAN parameters and produce sizing recommendations using an analytical model. The analytical models available for 802.11 MAC require a low-level input, i.e., the specification of traffic offered to the WLAN interface. Thus the application load specification needs to be translated to the load at the MAC layer.

In this paper, we address this requirement, of the mapping from application usage profile to WLAN traffic profile, and a reverse mapping from the MAC performance predicted by analytical models to the application performance. We do this by first surveying existing analytical models, and comparing them with extensive simulations, to identify the model most suitable for sizing. We then derive a mapping between the application and the MAC by characterizing the
application traffic. Our mapping makes simplifying assumptions and hence is approximate. Results from simulations, however, show that we are still able to achieve reasonable accuracy.

The rest of the paper is organized as follows: In Section 2 we briefly describe the IEEE 802.11 protocol, related work and some important (from the sizing point of view) analytical models for 802.11 WLANs. We then present the results of our validation of the models. In Section 3, we present our approximate method for sizing using analytical models for 802.11 MAC. We illustrate our approach with an example of HTTP traffic and validate it against simulations. We conclude the paper with a discussion on sizing tools and application performance models for 802.11 WLANs.

2. PERFORMANCE MODELS FOR 802.11 WLANs

The IEEE 802.11 standard [2] defines the MAC layer protocol along with a few physical layer protocols. The MAC is a carrier sense multiple access with collision avoidance (CSMA/CA) protocol. The basic medium access method in 802.11 is the distributed coordination function (DCF). The polling-based point coordination function (PCF) is optional and is not supported widely. Channel access prioritization is through inter-frame space (IFS), described as the period of silence on the channel between transmissions. Three important IFSs that the 802.11 MAC employs, in decreasing order of priority, are: short inter-frame space (SIFS), point (coordination function) interframe space (PIFS), and distributed (coordination function) interframe space (DIFS).

A station with a packet to transmit senses the channel, and if the silence period on the channel equals or exceeds DIFS, the station is allowed to transmit. If the channel is found busy, the station defers transmission until the channel becomes idle for DIFS. The collision avoidance mechanism of the MAC mandates that a random backoff procedure be followed at this point of time. All deferred stations generate a backoff timer in the range $[0 - CW]$ for an additional deferment time before transmitting. The parameter $CW$ (contention window) takes an initial value $CW_{\text{min}}$, and is doubled after every unsuccessful transmission, subject to a maximum of $CW_{\text{max}}$. Failed transmissions for packets are re-attempted till the relevant frame retry limit is reached. The 802.11 MAC employs positive acknowledgements (ACKs) to detect whether the transmission was successful or not. The DCF also includes a four-way handshaking mechanism called RTS/CTS, which provides virtual carrier reservation.

2.1 Models for 802.11 MAC

Researchers have conducted performance analysis of 802.11 WLANs through analytical models [5, 7–9, 12–14, 19, 20, 22], simulations [3, 6, 11], and measurements [3, 4, 15, 21]. Application performance over 802.11 WLANs has been studied through simulations [3, 5, 6, 11], and measurements [3, 4, 15, 21]. Application performance models for 802.11 WLANs have been proposed in literature, but most of them are for bulk data transfer [18] and they require link-layer performance metrics (such as packet drop probability at the interface queue) [17] that are not available from existing WLAN models. File transfer times over WLANs has been studied analytically in [16] without modeling the TCP behavior.

Models to compute the throughput of 802.11 WLANs in saturated conditions have been proposed in [5] and [19]. Saturation is defined as the condition in which interface queues of all stations are non-empty at all times, indicating overload conditions. Saturation throughput, as a fundamental measure of capacity, is an important performance metric. However, at saturation, either the average delays tend to increase without bounds, or, as it happens in practice, packets get dropped at the interface queue because of a full queue. Therefore, networks are never intended to be used at saturation. Analytical models that do not assume saturation conditions have been proposed in [12], [8] and [20].

We identified a few analytical models for 802.11 MAC, to evaluate their suitability for use in sizing, in terms of their inputs, the assumptions they make, the performance metrics they compute and the accuracy of these predictions. Those are the models by Tay-Chua [19] and Foh-Zukerman [12]. We give a brief overview of these models in the following. Since Foh-Zukerman’s model uses results from Bianchi’s model [5], we also review Bianchi’s model.

2.1.1 Tay-Chua’s model

Tay and Chua [19] have proposed a simple analytical model for 802.11 DCF, which starts with the computation of the average backoff window size $W_{\text{backoff}}$ and the probability $p$ of collision:

$$p = 1 - \frac{2(1 - 2p)}{1 - p - (2p)^m} \frac{1}{W} n^{-1}$$

where $W = CW_{\text{min}}$ and $m =$ maximum backoff stage. Making a first order approximation that each collision is between exactly two stations, the rate of successful transmissions $r_{\text{success}}$ and saturation throughput $S$ are obtained from $p$.

From (1), the authors derive a closed-form expression for $n^*$, the number of stations for which saturation sets in, for a given packet arrival rate $\lambda$.

$$n^* = \frac{1}{\lambda T'} - \frac{1}{3 + W \lambda T'}$$

where $T' = T_{\text{physical}} + T_{\text{SIFS}} + T_{\text{ACK}} + T_{\text{DIFS}}$ and $T_{\text{physical}}$ is the time to transmit the packet, including headers.

2.1.2 Bianchi’s model

The analytical model proposed by Bianchi [5] uses a discrete-time Markov chain (DTMC) to model the behavior of a single station at saturation. The model assumes a constant and independent probability $p$ of collisions, regardless of the number of retransmissions suffered. Solving the DTMC, the stationary probability $\tau$ that a station transmits in a randomly chosen slot time is obtained:

$$\tau = \frac{2(1 - 2p)}{(1 - 2p)(W + 1) + pW(1 - (2p)^m)}$$

From this probability $\tau$, the probability that there is at least one transmission in a slot time ($P_n$), the probability that
a transmission occurring on the channel is successful ($P_s$), and average length of the logical slot time ($E[\text{slot time}]$) are computed. The throughput $S$ is then given by

$$S = \frac{P_s P_a E[\text{Payload}]}{E[\text{slot time}]}$$  

(4)

2.1.3 Foh-Zukerman’s model

Foh and Zukerman [12] have presented an approximate model based on Bianchi’s model. According to their approach, of the $k$ stations sharing the channel, only $n$ stations are active and are assumed to be saturated. Saturation throughput $S$ and mean service rate $\mu(n)$ for $n$ stations is obtained from (4). The authors construct a continuous-time Markov chain single-server queue with a certain arrival and service process. The state of the queue represents the number of active stations. The arrival process is Poisson with rate $\lambda$ which is dependent on the number of active stations. Based on the service time distribution obtained from simulations, the authors recommend using Erlang-8 ($E_8$) as the service process of the queue. Solving the balance equations of the $M/E_8/1/k$ system, the throughput and mean service time are obtained. Further details on the model are provided in the Appendix.

2.2 Comparison and validation of 802.11 Analytical models

We validated the models described above through extensive simulations using the ns-2 simulator [1]. For the ns-2 simulations, identical nodes sent traffic to each other in a logically circular topology, over an error-free channel, using the exponential traffic generator. For both simulations and analytical models, we varied the number of stations $n$, while keeping the load per station, packet size and data rate fixed. We conducted numerical analysis with simulations and analytical models for different combinations of data rates, packet sizes and per-station loads.

Fig. 1(a) and Fig. 1(b) present MAC throughput vs. $n$ and average delay vs. $n$ respectively, obtained from simulations and Foh-Zukerman’s model. The per-station load, consisting of 1000B packets, is 1% of the channel capacity for these graphs. The plots clearly indicate that Foh-Zukerman’s model is able to predict throughput and delays accurately.

Table 1 lists values of $n^*$, the number of stations for which saturation sets in, obtained from simulations and Foh-Zukerman’s model. The per-station load, consisting of 1000B packets, is 1% of the channel capacity for these graphs. The plots clearly indicate that Foh-Zukerman’s model is able to predict throughput and delays accurately.

Table 1: MAC Congestion points $n^*$

<table>
<thead>
<tr>
<th>Pkt Size (bytes)</th>
<th>DataRate (Mbps)</th>
<th>%load/station</th>
<th>$n^*_\text{sim}$</th>
<th>$n^*_\text{Tay}$</th>
<th>$n^*_\text{Foh}$</th>
<th>$n^*_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1</td>
<td>1</td>
<td>80</td>
<td>64</td>
<td>75</td>
<td>84</td>
</tr>
<tr>
<td>1000</td>
<td>1</td>
<td>2</td>
<td>40</td>
<td>33</td>
<td>37</td>
<td>41</td>
</tr>
<tr>
<td>1000</td>
<td>11</td>
<td>1</td>
<td>46</td>
<td>43</td>
<td>47</td>
<td>46</td>
</tr>
<tr>
<td>1000</td>
<td>11</td>
<td>2</td>
<td>24</td>
<td>23</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>500</td>
<td>1</td>
<td>1</td>
<td>72</td>
<td>62</td>
<td>67</td>
<td>76</td>
</tr>
<tr>
<td>500</td>
<td>1</td>
<td>2</td>
<td>37</td>
<td>30</td>
<td>33</td>
<td>39</td>
</tr>
<tr>
<td>500</td>
<td>11</td>
<td>1</td>
<td>32</td>
<td>33</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>500</td>
<td>11</td>
<td>2</td>
<td>17</td>
<td>17</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>1</td>
<td>33</td>
<td>24</td>
<td>23</td>
<td>33</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>2</td>
<td>17</td>
<td>13</td>
<td>9</td>
<td>17</td>
</tr>
</tbody>
</table>

$n^*_\text{sim}$: simulations with single-packet Poisson arrivals  
$n^*_\text{Tay}$: Tay-Chua’s model  
$n^*_\text{Foh}$: Foh-Zukerman’s model  
$n^*_8$: simulations with bursts of 8 packets

Table 1: MAC Congestion points $n^*$
the same offered load), emphasizing the point that the average packet size needs to be computed accurately for correct analysis.

![Graph showing throughput for different payloads, 1% load per station](image)

**Figure 2:** Throughput for different payloads, 1% load per station

We also conducted simulations for packet arrivals occurring in bursts of 8 packets (with exponential delay between bursts) and compared the results with those for single-packet Poisson arrivals. As indicated by the results in Table 1, the assumption of single-packet Poisson arrivals does not change throughput and $n^*$ values significantly.

### 3. SIZING OF 802.11 WLANS USING APPLICATION LOAD SPECIFICATION

A network user is mainly concerned with the performance of an application. Therefore the sizing of WLANs has to be performed with the goal of optimizing the application throughput or delays. The performance of an application is affected by the performance of all the layers below it. Deriving an exact model that takes into account the behavior at the application, TCP, MAC and physical layer is a complex task; in the absence of such a model, we need to look at approximate approaches. In this section, we propose an approximate method which makes use of existing models for the 802.11 MAC to predict the application throughput. Since the analytical models for 802.11 MAC require specification of traffic offered to the MAC, we provide a mapping from application usage profile to the MAC traffic profile. We also provide a reverse mapping from the MAC throughput to application throughput. While doing so, we make important assumptions regarding TCP behavior and hence our method is approximate. We can see from the validations that despite these assumptions, the method works quite well. We illustrate our method with HTTP traffic, but it is general enough to be used with all applications, provided that the traffic offered by the application can be characterized.

#### 3.1 Approximate sizing method using application traffic specification

Before we proceed, we mention a few assumptions that we have made to simplify the method: first, that there is no packet loss as perceived by the layers above the MAC. Second, MAC delays are not large enough to cause TCP to alter its offered load to the MAC. Third, the TCP flows are short lived due to small sized objects being transferred.

These assumptions are significant and need justification. Since 802.11 MAC employs MAC-level retransmissions, packet loss is reported to higher layers only if a) the limit on the number of retries is exceeded or b) the interface queue is full. The probability of any of the above happening is very low until the WLAN has reached saturation. From the simulations, we verified that there were indeed no TCP retransmissions (due to packet drops at WLAN interface), even after the application throughput had saturated.

Near saturation, however, the connection-oriented nature of TCP causes it to throttle the flow of packets to the WLAN interface, due to higher MAC packet transfer delays. For example, an HTTP session can be broken into three request-response pairs:

- TCP connection establishment
- HTTP GET request and HTTP object as the response
- TCP connection tear-down

Although TCP employs sliding-window flow control, a TCP connection remains blocked between the request-response pairs. Fig. 3(a) and Fig. 3(b), which compare the offered load to the MAC and the MAC transfer delays respectively, from simulations to those from our approximate method, illustrate this effect. While the offered load to the MAC for analytical model keeps increasing with increase in application load, that for simulations is throttled by TCP itself past application saturation point (27 nodes). Thus, our method assumes open arrivals to the WLAN interface while TCP exhibits a closed-loop behavior. Modeling this TCP behavior would add complexity to the sizing method, and since it does not significantly affect the performance before application saturation, we choose to ignore it without compromising the accuracy of our results.

The third assumption is justified from the empirical evidence on Internet object sizes. Study of www object sizes reveals [23] that 90 % of objects were smaller than 12 Kbytes, with an average size of 5 Kbytes for a static content. The assumption of small object sizes implies that TCP window size does not affect the offered load to the MAC.

With these assumptions, the overheads of TCP can be approximated to be static for non-saturated WLANs.

Our approximate sizing method consists of the following steps:

**Step-1) Specification of application workload:** This is the specification of application usage, either provided by the user herself or extracted from measurement data. The parameters that need to be specified are: the type of application, the average size of application payload $L_{app}$ (in bytes), the number of users $n$ and their rates of requests $\lambda_{app,i}, i \in [1,\cdots,n]$.

**Step-2) Characterization of application traffic:** Each appli-
Sizing for HTTP traffic

3.2 Example: sizing for HTTP traffic

We illustrate our method by applying the steps discussed above to an example of a simple HTTP application. It should be noted that while uniform traffic from all stations is assumed by the analytical models for 802.11 MAC, their application for the WLAN where downstream traffic is concentrated at an access point needs to be validated through simulations. For the sake of simplicity, we will not model the network dynamics outside the BSS.

Step-1) Specification of HTTP workload: In a WLAN of size \( n \), there are \( n - 1 \) HTTP client nodes and one access point which the clients use for communication. Each HTTP client communicates with a server outside the WLAN, requesting objects of fixed size \( L_{\text{app}} \) (bytes), the request generation process being Poisson with rate \( \lambda_{\text{app}} \). Thus \( \lambda_{\text{app},i} \) is \( \lambda_{\text{app}} \) for all clients and is zero for the access point.

Step-2) Characterization of HTTP traffic: Each HTTP object request generates two application messages which constitute the application load: a GET request of a small size (approximately 40 bytes) and an HTTP response of a size equal to the object size. The response messages are broken into packets of size equal to the TCP Maximum Segment Size (MSS) plus TCP/IP headers (except for the last packet). In the following analysis, we have assumed an MSS of 536 bytes, and an MSS plus TCP/IP headers (except for the last packet) of 1,500 bytes.

Step-3) Computation of MAC performance metrics using analytical models: The analytical models for the 802.11 MAC require packet arrival rate \( \lambda_{\text{mac}} \), average packet size \( l_{\text{mac}} \), and the number of stations \( n \) as inputs. The average throughput of the WLAN (in bytes/sec), denoted as \( S_{\text{mac}} \), is computed from these inputs using the analytical models.

Step-4) Mapping the MAC performance to application performance: Our simplifying assumptions make this step straightforward. Given the MAC throughput \( S_{\text{mac}} \), the application throughput \( S_{\text{app}} \) (in bytes/sec) is obtained, after accounting for overheads (assumed as constant) at TCP and the application layer.

Step-5) Finding the application saturation point: The most useful result of the sizing exercise is the application saturation point, denoted by \( n^{**} \). It is defined as the number of stations for which the application throughput saturates, for a given application request rate \( \lambda_{\text{app}} \) per station. We define saturation as the condition where an increase in offered load does not result in corresponding increase in throughput. It is that point where the application throughput curve diverges from the offered load curve. Increasing \( n \) beyond this \( n^{**} \) will result in application requests being infinitely delayed or dropped. Formally, if \( S_{o} \) is the offered load,

\[
S^{**} = \text{largest } n \text{ such that } S_{o} - S_{\text{app}} < S_{o} \frac{\epsilon}{n^{**}} \tag{5}
\]

where \( S_{o} = \sum_{i=1}^{n} L_{\text{app},i} \) bytes/sec and \( \epsilon \) is the threshold fractional difference between the offered load and the throughput.

We calculate the application throughput \( S_{\text{app}} \) for increasing values of \( n \), and continue till \( n^{**} \) satisfying (5) is obtained.

Figure 3: Effect of the closed-loop behavior of TCP

- Offered load to the MAC
- Packet transfer delay at the MAC
of 536 bytes. For keeping the analysis simple, we classify packets in two sizes: small packets of size $l_s$ bytes, consisting of just TCP and IP headers, and large packets, of a size approximately equal to the TCP MSS plus $l_h$.

Assuming the HTTP/1.0 protocol (no keep-alive), each HTTP request/response sequence uses a new TCP connection. From the traffic traces, we observed the following packet statistics for each HTTP object request:

- In the upstream direction, 5 (for TCP connection setup and tear-down, and HTTP GET message) + $D$ (TCP ACKs to payload) small packets
- In the downstream direction, 4 (for TCP connection setup and tear-down) small packets, $D$ large packets (for application payload)

where $D = \lceil \frac{L_{app}}{MSS} \rceil$. Then,

$$\lambda_{mac} = \frac{\lambda_{app}(n-1)(9 + 2D)}{n} \text{ packets/s/station}$$

Average Packet size $= l_a = \frac{[9 + 2D]l_h + L_{app}}{(9 + 2D)}$ bytes

Step-3) Computation of MAC performance metrics: Knowing $\lambda_{mac}$ and $l_a$ (obtained from (6) and (7)), and the number of users $n$, we compute the MAC throughput $S_{mac}$ using an analytical model for the 802.11 MAC. Based on the results in the previous section, we chose to use Foh-Zukerman’s model [12], the details of which are provided in Appendix.

Step-4) Mapping the MAC performance to the HTTP performance: Since each HTTP request generates $l_h(9+2D)+L_{app}$ bytes of traffic at the MAC, of every $l_h(9+2D)+L_{app}$ bytes of MAC throughput, $L_{app}$ bytes constitute application throughput. Therefore, the application throughput $S_{app}$ is obtained as

$$S_{app} = \frac{S_{mac}L_{app}}{l_h(9+2D)+L_{app}}$$

Step-5) Finding the application saturation point: The value of $n$ satisfying (5) is the application saturation point $n^{**}$, where the offered application load

$$S_o = (n-1)\lambda_{app}L_{app}$$

The following algorithm summarizes the computation of $n^{**}$:

1. obtain $\lambda_{app}, L_{app}$
2. $n \leftarrow n_{start}$
3. repeat
4. compute $\lambda_{mac}, l_a$
5. compute $S_{mac}$ using analytical model for the MAC
6. compute $S_{app}, S_o$
7. $n^{**} \leftarrow n$, increment $n$
8. until $\frac{S_o-S_{app}}{S_o} < \epsilon_t$

3.3 Validation using simulations

To validate our method, we conducted extensive simulations using the ns-2 simulator [1]. The scenario simulated was the same as that considered for the analytical method: $n - 1$ stations, each having an HTTP client, sending requests to a server through an access point. We used the webcache module of ns-2 to simulate the HTTP client and servers, with modifications to allow concurrent TCP connections between a client-server pair. The requests were generated according to a Poisson process with rate $\lambda_{app}$, at each client. Simulations results were obtained through 14 sets of independent runs of 20 seconds duration each.

In each set, we increased the number of nodes $n$, while keeping the HTTP request generation rate per station and the payload size fixed. The per-station load value is the application payload per unit time, expressed as a percentage of the channel bit rate. We conducted the numerical analysis for different values of per-station load, HTTP object size $L_{app}$, and data rate.

The application and MAC throughput, normalized to the channel data rate are plotted in Fig. 4 for HTTP object sizes of 536B and 8576B. Fig. 5 compares application throughput from our method and simulations for different data rates and HTTP request rates, for 4288B objects. As can be seen from the plots, our sizing method is able to track the variations in application throughput up to saturation 1 reasonably well. The MAC throughput plotted is the throughput obtained from Foh-Zukerman’s model [12]. It can be seen from Fig. 4 that the application payload size has a significant effect on the application and MAC throughput. TCP overheads consume as much as 50% of the MAC throughput for a 536B payload, reducing the maximum application throughput to 0.21, while that for 8576B payload is 0.5, other parameters being the same. Low values of normalized throughput at 11 Mbps are because of the dominant physical layer overheads since we did not use the “short preamble” option of 802.11b.

The application saturation points $n^{**}$ for different values of channel bit rate, HTTP request rate and HTTP object size are tabulated in Table 2. The $n^{**}_{sim}$ and $n^{**}_{ana}$ values are $n^{**}$ values obtained from simulations and from our method respectively, for a $\epsilon_t$ value of 0.15 . Despite the approximation of symmetric traffic distribution across all wireless stations and our simplifying assumptions regarding TCP behavior, the $n^{**}$ values suggested by our method are within 13% of those derived from simulations for a majority of the operating parameters. The results validate our simplifying assumptions regarding the TCP behavior.

4. CONCLUSION

In this work, we have addressed the sizing problem, i.e., the problem of finding the maximum number of users that can be supported (subject to given constraints), in the context of IEEE 802.11 WLANs. We have proposed an approximate sizing method which provides the required mapping from the application-layer workload specification to the traffic at the WLAN interface, required by the existing analytical models.

1 Beyond saturation, however, the closed-loop behavior of TCP increases the HTTP object transfer time, decreasing the throughput. This explains the sharp drop in application throughput past saturation for simulation results.
Payload size: 536 B, 1 Mbps data rate

Payload size: 8576 B, 1 Mbps data rate

Payload size: 4288 B, 1 Mbps data rate

Payload size: 4288 B, 11 Mbps data rate

Figure 4: Application throughput vs \( n \) for different HTTP object sizes at 1% application load per station

(a) 0.58 requests/sec (2% load per station)
(b) 0.88 requests/sec (3% load per station)

(c) 0.583 requests/sec (0.18% load per station)
(d) 0.88 requests/sec (0.27% load per station)

Figure 5: Application throughput vs \( n \) for different request rates, 4288B Object
The computation of our simplifying assumptions.

5. REFERENCES

extended for delay-constrained sizing. The computation of application-layer delays, it can also be turned into a full-fledged sizing tool. By incorporating workload and a combination of different applications, can be extended for a more detailed specification of application usage profile, using the application characterization information. It also provides a reverse mapping from the MAC throughput to the application throughput and recommends the maximum number of users, $n^*$. While doing so, our method makes simplifying assumptions regarding TCP behavior in response to packet flow dynamics.

We have illustrated our method using an example of HTTP traffic and validated it with extensive simulations using the ns-2 simulator. Despite the simplifying assumptions, we found a close match between the results from our method and the simulations, particularly the value of $n^*$, the number of stations for which the application throughput saturates. This value of $n^*$ is found to be sensitive to the choice of the threshold throughput-load gap $\epsilon_t$, due to the effect of our simplifying assumptions.

The computation of $n^*$ can be made more accurate and less dependent on the choice of $\epsilon_t$ by modeling the closed-loop behavior of TCP in response to MAC delays. Our method can be extended for a more detailed specification of application workload and a combination of different applications, to be turned into a full-fledged sizing tool. By incorporating the computation of application-layer delays, it can also be extended for delay-constrained sizing.

5. REFERENCES


<table>
<thead>
<tr>
<th>Req. Rate</th>
<th>% Load</th>
<th>Bit Rate</th>
<th>Payload</th>
<th>$n^*_{ana}$</th>
<th>$n^*_{ana}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(req/stn/s)</td>
<td>/stn</td>
<td>(Mbps)</td>
<td>(bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.332</td>
<td>1.0</td>
<td>1</td>
<td>536</td>
<td>27</td>
<td>23</td>
</tr>
<tr>
<td>0.292</td>
<td>1.0</td>
<td>1</td>
<td>4288</td>
<td>59</td>
<td>49</td>
</tr>
<tr>
<td>0.146</td>
<td>1.0</td>
<td>1</td>
<td>8576</td>
<td>61</td>
<td>53</td>
</tr>
<tr>
<td>4.664</td>
<td>2.0</td>
<td>1</td>
<td>536</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>4.664</td>
<td>0.182</td>
<td>11</td>
<td>536</td>
<td>30</td>
<td>29</td>
</tr>
<tr>
<td>0.583</td>
<td>2.0</td>
<td>1</td>
<td>4288</td>
<td>29</td>
<td>25</td>
</tr>
<tr>
<td>0.583</td>
<td>0.182</td>
<td>11</td>
<td>4288</td>
<td>87</td>
<td>93</td>
</tr>
<tr>
<td>0.292</td>
<td>2.0</td>
<td>1</td>
<td>8576</td>
<td>31</td>
<td>27</td>
</tr>
<tr>
<td>0.292</td>
<td>0.182</td>
<td>11</td>
<td>8576</td>
<td>99</td>
<td>107</td>
</tr>
<tr>
<td>6.996</td>
<td>3.0</td>
<td>1</td>
<td>536</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>6.996</td>
<td>0.273</td>
<td>11</td>
<td>536</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>0.875</td>
<td>3.0</td>
<td>1</td>
<td>4288</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>0.875</td>
<td>0.273</td>
<td>11</td>
<td>4288</td>
<td>60</td>
<td>63</td>
</tr>
<tr>
<td>0.437</td>
<td>3.0</td>
<td>1</td>
<td>8576</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>0.437</td>
<td>0.273</td>
<td>11</td>
<td>8576</td>
<td>66</td>
<td>73</td>
</tr>
<tr>
<td>11.660</td>
<td>0.455</td>
<td>11</td>
<td>536</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>1.458</td>
<td>0.455</td>
<td>11</td>
<td>4288</td>
<td>35</td>
<td>37</td>
</tr>
<tr>
<td>0.729</td>
<td>0.455</td>
<td>11</td>
<td>8576</td>
<td>41</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 2: Saturation Points $n^*$, $\epsilon_t = 0.15$


**APPENDIX**

**Obtaining $S_{mac}$ using Foh-Zukerman’s model**

Foh-Zukerman’s model [12] is based on the assumption that, out of $k$ stations, only $n$ are active ($0 \leq n \leq k$) and are saturated. The number of active stations $n$ is modeled with a continuous time Markov chain single server queue; $n$ increases according to the arrival rate for a particular state, given by

$$\lambda(n) = \lambda_{ind} \cdot (k - n), \quad n = 0, 1, \ldots k \quad (10)$$

where $\lambda_{ind}$ is the arrival rate (at the MAC) at each station. The number $n$ decreases according to a state-dependent service process, which is modeled as Erlang-8. The state of the system is thus represented by a tuple $(n, i)$ where $i$ is the stage of Erlang-8 process, $1 \leq i \leq 8$. The mean service rate for $n$ stations, $\mu(n)$ is obtained as

$$\mu(n) = \frac{S(n)}{d_t} \quad (11)$$

where $S(n)$ is the saturation throughput for $n$ stations (obtained from Bianchi’s model) and $d_t$ is the mean transmission time of payload. The service rate for state $(n, i)$ is thus $8 \cdot \mu(n)$.

Knowing the arrival and service rates, the balance equations for the $M/E_8/1/k$ system can be written. We solve the balance equations for the Markov chain using the SHARPE solver [10] and obtain $p_{n,i}$. Then the probability that the system is in state $n$ is

$$p_n = \sum_{i=1}^{8} p_{n,i}$$

The mean arrival rate $\bar{\lambda}$ of the system is then

$$\bar{\lambda} = \sum_{n=0}^{k} (\lambda(n) \cdot p_n) \quad (12)$$

and the throughput $S_{mac}$ is

$$S_{mac} = \bar{\lambda} \cdot d_t \quad (13)$$