TWHTB: A Transmission Time Based Channel-Aware Scheduler for 802.11 systems

R. G. Garroppo, S. Giordano, S. Lucetti and E. Valori
Department of Information Engineering, University of Pisa
Address: Via Diotisalvi 2, 56126 Pisa, Italy
Email: {r.garroppo, s.giordano, s.lucetti}@iet.unipi.it

***Abstract*— 802.11 WLANs suffer severe performance degradation as soon as at least one of the stations in the Basic Service Set (BSS) experiments unfavourable radio channel conditions. This degradation is due to a so-called anomaly which manifests when one or some of the stations adopt a reduced transmission data rate, as well as to the reduced efficiency of radio channel exploitation due to retransmissions. The plain FIFO scheduler, typically implemented in commercial devices, is the main cause of the aforementioned anomaly (such behavior). To overcome such problem, and optimize radio resource utilization, a channel quality aware scheduler has been designed and integrated in a prototypal Access Point (AP). The developed scheduler is based on the Hierarchical Token Bucket (HTB), and aims at granting the configured transmission share towards all the STAs served by the AP, irrespective of their channel quality. This goal is achieved limiting the frame transmission rate towards the STAs experimenting bad channel conditions, which will be likely corrupted and which, as a general remark, lasts for a longer period of time, thus occupying the channel at the expenses of the other stations. Lastly, the performance of the presented prototype are experimentally evaluated and compared with those obtained with stanard scheduling algorithm, which do not take into account information on channel quality.**

I. Introduction

IEEE 802.11 [1] networks offer simplicity of operation and configuration in a number of operative scenario. However, the MAC design, associated to the intrinsic unreliability of the wireless channel, produces severe degradation of the throughput achievable by all the stations in a single Basic Service Set (BSS) if just one of them is characterized by unfavourable radio propagation conditions. Indeed, automatic data rate adaptation and multiple retransmissions cause such station(s) to occupy the channel for longer time, reducing the radio resources left available to the other stations. In particular, [2] has highlighted as, even in the case where all but one stations use the highest nominal data rate, the effective throughput of all the stations in a BSS is degraded below the lowest adopted data rate. The main cause of such behavior is the long term fairness of 802.11 MAC, based on CSMA/CA protocol, which guarantees an equal long term channel access probability to all stations.

Additionally, when working in infrastructured mode, the Access Point (AP) has to handle the traffic directed to all the associated stations. However, an AP contends for transmission opportunities with the same priority (CSMA/CA protocol parameters) of the other stations. Therefore, it is clear that the AP has the major bottleneck role of the system [3] [4]. Commercial APs manage the traffic towards the associated STAtions (STAs) according to a First-In-First-Out (FIFO) queueing discipline. As a consequence, when considering downlink direction, mutual throughput degradation occurs also within the same device, worsening the overall system performance.

The main contribution of the paper is the definition of TWHTB, a scheduler based on the Hiearchical Token Bucket (HTB) able to differentiate the transport service offered by an IEEE 802.11 AP taking into account the radio link quality experimented by the different mobile stations. Another example of channel-aware schedulers is presented in [5]; however, such approach does not take into account the downlink transport service differentiation issue. The present work is the evolution of [6], where the WHTB was introduced; some details on the motivations which led to TWHTB are given later in the paper.

II. Architecture of a Channel-Aware Scheduler

The aim of a channel-aware scheduler is to exploit information on the radio link quality experienced by the different users associated to the same AP in order to optimize the global goodput of the considered WLAN cell. The goodput is defined as the throughput perceived by the IP level; it takes into account the reduced efficiency due to MAC and PHY protocols overhead, data link layer retransmissions, current data rate, and so on. However, the actions taken to optimize the global goodput must consider also the required quality of service at IP level

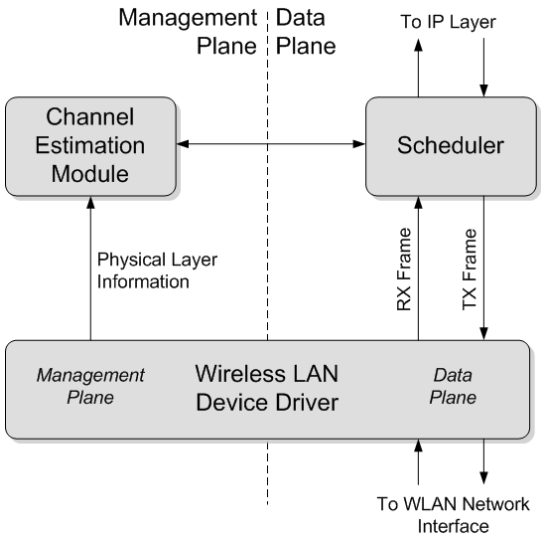

Fig. 1. Architecture of a channel-aware scheduler

chosen by the users. To take into account both aspects, a possible architecture can be obtained modifying a scheduler used in wired networks in order to take advantage of the information on the radio link quality experienced by the users.

The reference channel-aware scheduler architecture is represented in Fig. 1. On the right hand side, the Data Plane is devoted to the transmission and reception of the frames; it is composed of the WLAN device driver, which performs the actual transmission and reception, and the scheduler, which is driven by link quality information provided by the Management Plane. The Management Plane is composed of the part of the WLAN device driver deputed to export the information on the channel quality towards each receiver station, and of Channel Estimator module, which translates this information into values suitable to feed the scheduler. As far as the scheduler itself is concerned, it can be either an algorithm designed specifically for wireless network or a scheduler developed for wired networks opportunely modified to take into account the information on wireless channel.

III. From HTB to Wireless HTB

This Section briefly describes the main features of the HTB scheduler, and the extensions needed to introduce the support for channel-awareness in the scheduling process; for an extensive description of the HTB inner operations, the reader can refer to [7].

In short, HTB traffic classes are organized in a tree structure; each class is configured with an average rate to be guaranteed (*rate* parameter) and a maximum rate which cannot be exceeded (*ceil* parameter). To fulfill such requirements, each class is controlled by two internal token buckets, after which the scheduler is named. The HTB scheduler grants the right to transmit to classes which have not exceeded their allowed *ceil*, according to a Deficit Round Robin (DRR) algorithm. Classes which have not exceeded their *rate* can unconditionally transmit; classes which have exceeded their allowed *rate* but not their *ceil* can transmit only borrowing unused bandwidth, if available, from other classes.

A. Introducing channel-awareness in the HTB

The basic concept behind the introduction of channel-awareness in the HTB scheduler is to utilize an estimate of the actual effective goodput achievable towards each station to opportunely modify the scheduling algorithm. Schedulers developed for wired networks assume a constant available goodput, a fraction of the nominal data rate, which translates in well defined bandwidth guarantees to all the destinations. On the contrary, in a wireless domain, effective goodputs towards various destinations may not be exactly defined, because of unavoidable retransmissions or data rate adaptation due to fading, interferences or collisions. Consequently, it is necessary to obtain their estimates, and feed the scheduling algorithm with such knowledge.

Basically, with reference to the HTB scheduler, once the goodput is available, the evaluation of the radio resources occupied by each transmitted packet (of length Pkt_{length}) is performed taking into account an *effective* packet length, called *stretched*, which is evaluated according to the following expression:

$$stretched = \min(Pkt_{length} \cdot \frac{R_{MAX}}{\hat{R}(t_0)}, bound)$$

where $\hat{R}(t_0)$ is the estimated average goodput towards the specific destination at the time of packet's dequeue, and R_{MAX} is the maximum attainable goodput towards the same destination when channel quality is maximum. The *stretched* value is upper bounded by the value *bound* to reduce starvation issues when very poor channel conditions are experimented.

In a previous work [6], the current maximum achievable goodput towards a destination STA was deducted by the value of the Signal to Noise Ratio (SNR) of the received frames at the AP. The SNR value is provided by the *Wireless Channel Monitor* (WChMon), assuming channel simmetry, and then mapped to a goodput value by means of a static table obtained by a precedent calibration session. The resulting scheduler was named Wireless HTB (WHTB), and experimental measurements proved its effectiveness in avoiding badly positioned destinations to monopolize the channel usage. The drawback

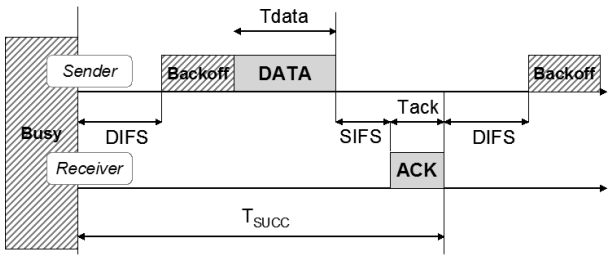

Fig. 2. Time budget of the transmission of a data frame

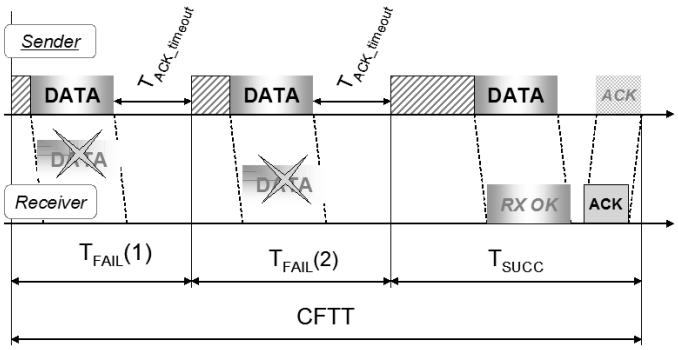

Fig. 3. Example of CFTT in the case of 2 retransmissions

of such approach is to rely on calibration (needed for each different wireless network adapter) and simmetry assumption.

In the next subsection we present a different algorithm to estimate the goodput towards a specific destination, aimed at avoiding preventive calibration and increasing the robustness of the goodput estimation process.

B. The Time-based Wireless HTB (TWHTB)

One of the most relevant issues in the design of channel-aware schedulers is the channel estimation procedure. The ultimate product of the channel estimator is the time-varying information on the effective goodput towards each station. Differently from the approach which led to the WHTB scheduler (based on SNR measurements), an analysis based on time needed to successfully transmit a frame (including retransmissions and reception of the ACK message) has been pursued.

Fig. 2 shows the various components of the time T_{SUCC} needed to complete a successful frame transmission. Note that, although the *DIFS* and *Backoff* periods are not strictly related to frame transmission (as far as the transmitter is concerned, they are idle periods), they have been included in the T_{SUCC} evaluation, since they limit the maximum frame rate, and have to be respected independently of the system conditions. Several works (see [8][9][10]) have used such time analysis to derive the maximum achievable goodput in saturation conditions of a single UDP source. The estimated effective goodput is obtained by the straightforward relation:

$$\hat{G} = \frac{Pkt_{length}}{T_{SUCC}}$$

optionally subtracting the $Pkt_{headers}$ size of the overhead bits which constitute IP and higher layers protocol headers if application level goodput is desired.

With similar considerations, we extend the analysis to the case of frame transmission requiring one or more retransmissions to define the Cumulative Frame Transmission Time (CFTT). The CFTT is defined as the amount of time needed to successfully transmit a frame, including retransmissions and possible rate reduction), and is measured from the instant of the transmission of the first bit of the frame, to the reception of the positive ACK which confirms its reception.

As a reference result, it is possible to analitically evaluate the CFTT, in saturation conditions, as a function of the number of frame transmission attempts before reception of the corresponding ACK message. Neglecting the propagation delays, a single transmission attempt duration can be splitted in several components [8], easily identifyable in Fig. 2. In the case of unsuccessful transmission of the frame, the sender recognizes such event if it does not receive back the ACK message within a defined $T_{ACK\_Timeout}$. We indicate with $T_{FAIL}(k)$ the amount of time associated to the k-th failed transmission attempt. The index k takes into account the different values of the $T_{Backoff}$ for each retransmission, due to the exponential Backoff update algorithm of 802.11; furthermore, since $T_{Backoff}$ is a random variable, these durations are random variables too.

Basing on these considerations, and as illustrated by Fig. 3 the overall CFTT can be expressed as $CFTT(N) = \sum_{k=0}^{N-1} T_{FAIL}(k) + T_{SUCC}$, where N is the number of transmission attempts needed to successfully transmit the frame. Table I collects the minimum, mean and maximum values of the CFTT (in the cases of 1 to 4 transmission attempts), for the transmission of a 1024 Byte UDP packet using the four nominal data rates available in 802.11b. The issue of rate adaptation between consecutive attempts is not covered in the present tractation for sake of simplicity. However, Table I gives an indication on the expectable range of values of CFTT which can be measured in actual networks.

C. TWChMon, a channel estimator based on CFTT

As indicated above, the knowledge of the CFTT for a given frame allows to estimate the achievable goodput towards the corresponding destination. We have

# of transmission attempts			
1	2	3	4
min-**avg**-max CFTT @ 11 Mbps			
1.29-**1.60**-1.91	2.58-**3.52**-4.46	3.88-**6.09**-8.30	5.17-**9.93**-14.7
min-**avg**-max CFTT @ 5.5 Mbps			
2.08-**2.39**-2.70	4.17-**5.11**-6.05	6.25-**8.46**-10.7	8.33-**13.1**-17.9
min-**avg**-max CFTT @ 2 Mbps			
4.85-**5.16**-5.47	9.70-**10.6**-11.7	14.6-**16.8**-19.0	19.4-**24.2**-28.9
min-**avg**-max CFTT @ 1 Mbps			
9.26-**9.57**-9.88	18.5-**19.5**-20.4	27.8-**30.0**-32.2	37.0-**41.8**-46.6

TABLE I

CFTT VS NUMBER OF TRANSMISSION ATTEMPTS BEFORE SUCCESSFUL RECEPTION OF ACK FRAME (MILLISECONDS)

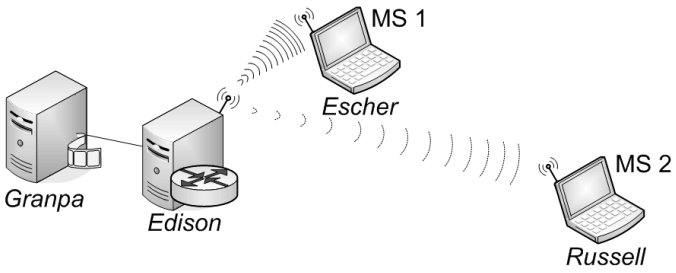

Fig. 4. Experimental testbed

integrated a real-time CFTT measurement procedure in the architecture depicted in Fig. 1; the resulting goodput estimates are used to feed the HTB scheduler to compute the *stretched* values of the Pkt_{length}. With respect to WHTB [6], the only difference in architecture is the channel estimation module. In the present approach, with reference to Figure 1, it is indicated with Time-based Wireless Channel Monitor (TWChMon)and operates as detailed below.

For each transmitted frame, both the instant of transmission begin and the instant of the reception of the corresponding ACK frame are detected. The difference between these two time values represents the frame CFTT. On the basis of such value, the TWChMon evaluates the estimated goodput (useful for the scheduling procedure) as

$$\hat{G} = \frac{Pkt_{length}}{CFTT}$$

To smooth the quickly varying channel dynamics (and to average the random values assumed by T_{BO}), a first order IIR low pass filter is applied to the result before letting it available for the modified HTB:

$$\hat{G}_{AVGD}(k) = 0.75 \cdot \hat{G}_{AVGD}(k-1) + 0.25 \cdot \hat{G}$$

As far as implementation issues are concerned, in the first phase of development of a channel-aware scheduler prototype we have used two network devices at the AP. One of them behaves normally, and is used to perform current transmission and reception of the frames, whereas the other is put in *monitor* mode. The *monitor* mode is a special operating state allowed by some wireless devices by means of proper kernel support [11], which allows the user to access (in read mode) all the frames on-air received by the interface; these include Management Frames (such as Beacons) as well as Control Frames (which include RTS, CTS and ACK frames). In order to be able to export ACK reception events to the TWChMon, the HostAP [12] driver used in the prototype has been modified accordingly.

IV. PARAMETERS OF EXPERIMENTAL ANALYSIS

The experimental testbed is depicted in Fig. 4; the functions and characteristics of each device are described in the following. All the devices are Linux based PCs or laptops running a 2.4 series kernel; both the MSs and the PC which acts as AP and channel-aware scheduler (Edison) run a modified version of the kernel, which includes the HTB patches for the Linux Traffic Control [13]. The second PC is used to generate two unidirectional data flows towards the two MSs, adopting the BRUTE [14] traffic generation tool. In this set of experiments, BRUTE has been configured to produce two CBR flows, characterized by constant inter-departure times and packet size. In order to highlight the efficiency of the proposed scheduler to react to channel quality variations, most of the experiments have been performed slowly moving away one of the MSs (MS2) from the AP, while keeping the other in optimal radio visibility with the AP.

For each measurement session, two performance parameters have been collected. The first one is the goodput received by the MSs, whereas the second one is the percentage share of the radio resources occupied by the transmissions towards each of them. The former is a performance metric related to the user perspective of the obtained service, whereas the latter is related to network perspective, since it indicates the efficiency of bandwidth utilization. Indeed, the channel occupancy share constitutes an indicator on the scheduler efficiency in avoiding MS2 to occupy the medium with many unsuccessful retransmission attempts, at the expenses of MS1.

The goodputs are measured at the receivers simply counting the number of received frames passed to the upper layer. The estimation of the channel occupation

share between the MSs is less straightforward. We suppose, and verify by measurements, that the vast majority of transmission attempts towards MS1 are successful, since MS1 is in very favourable radio conditions. Consequently, little, if any, bandwidth is wasted for retransmission, and it is possible to estimate the channel occupation share by MS1 simply as the ratio of the achieved goodput versus the saturation goodput of the system in ideal conditions (single source, no retransmissions). The remaining channel capacity is then consumed by transmissions towards MS2.

V. Performance analysis of TWHTB

In order to give an insight on the performance achievable using the proposed TWHTB scheduler, a reference test case, among all those analyzed, is presented in this Section. The experimental testbed is the one depicted in Fig. 4; each receiving MS is assigned to an HTB class, and the saturation goodput is evenly divided between the two classes. Since we have used 1024 byte long packets, the saturation goodput is about 5 Mbps; each class is then configured with a *rate* parameter equal to 2.5 Mbps. Furthermore, *ceil* has been set equal to *rate*.

Each set of measurements has been performed keeping MS1 close to AP with good channel quality, and moving MS2 away from the AP, stopping in four different positions, characterized by different values of channel quality. The four positions have been chosen on a heuristic basis, using the link quality indicator available with the device driver, ranging from Good (on which no retransmissions occur) to intermediate states Medium, Bad and Very Bad. The same testbed has been configured using classical schedulers for wired networks, in order to compare the TWHTB performance and verify the effectiveness of the approach. The reference schedulers are the Class Based Queueing (CBQ) discipline [15], the DRR and the plain HTB.

Measurement results obtained according to the procedure described in the previous subsection are reported in Table II and Figure 5. Table II collects the effective received goodput by the two stations with the adoption of three standard scheduling algorithms (CBQ, HTB and DRR) as well as the proposed TWHTB. The results highlight the different behavior of the four scheduling algorithms as far as the isolation of the two flows is concerned. In particular, the three classical schedulers fail to fulfill the bandwidth reservation associated to MS1 when MS2 starts to experiment bad channel conditions not managing to deliver the requested 2.5 Mbps to MS1 in any configuration except the ideal one (both MS1 and MS2 in the Good channel quality state). Indeed, the CBQ, DRR and HTB operate to try and maintain

MS2 Position	MS	CBQ	HTB	DRR	TWHTB
Good	1	2545	2534	2546	2562
	2	2545	2512	2543	2552
Medium	1	2000	1925	2038	2125
	2	2000	1855	2037	1636
Bad	1	1462	1492	1471	2020
	2	1460	1153	1469	795
Very Bad	1	714	778	573	1676
	2	676	369	562	350

TABLE II

Performance comparison among the different schedulers: Mean Received Goodput in Kbps with 1:1 bandwidth share

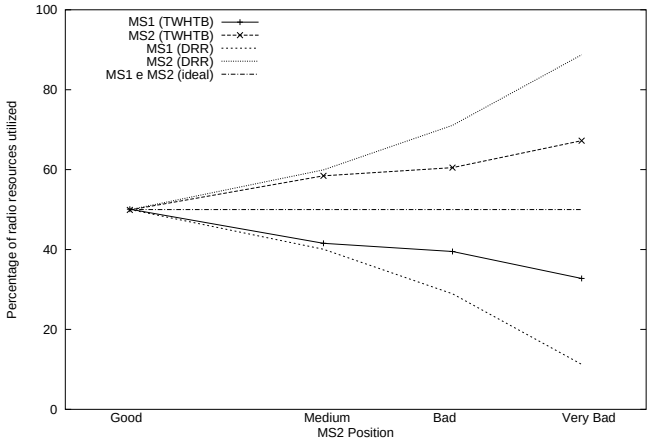

Fig. 5. Comparison between the radio resources used to transmit to MS1 and MS2 with DRR and TWHTB schedulers

the imposed ratio between the two assigned bandwidth. Consequently, the goodput received by MS1 degrades as long as MS2 suffers an increasing number of losses and retransmissions. On the other hand, the TWHTB reduces the influence of MS2 position on the goodput received by MS1, limiting the goodput reduction to about 35% versus the over 75% of the other schedulers.

When adopting the TWHTB scheduler, although the received goodput values are below the requested 2.5 Mbps, they are still much higher than those achieved by the other schedulers, and represent a considerable fraction (from 65 to over 80%) of the nominal assignment, highlighting the goodness of the proposed approach. Further improvements are under investigation to reach the goal of almost complete isolation between the two traffic flows.

Fig. 5 highlights the different consumption of transmission resources between the TWHTB and DRR schedulers (behavior of HTB and CBQ is analogous to DRR one). The latter allows the transmission towards MS2

to almost monopolize (up to 90% share) the use of the radio resources; indeed, the AP uses only 10% of the available bandwidth to transmit packets to MS1. On the contrary, the TWHTB scheduler allows to limit the unfairness between the resources used by each class, with a maximum unevenness of about 65% to 35%, regardless of the position of MS2 among the Medium, Bad or Very Bad conditions. Although it does not guarantee a complete decoupling between the two flows, the TWHTB represents a noticeable enhancement with respect to the standard schedulers.

At the current stage of development, the performance of TWHTB are degraded, when some destinations are in Very Bad position, by the limited number of positive ACK messages that can be used to estimate the channel quality. Ongoing work is devoted to optimize interaction of the device driver with the wireless interface registers, aiming at an increased knowledge of transmission attempt outcome.

VI. Conclusions

To overcome the performance anomaly of the 802.11 system, evidenced firstly in [2], and to achieve a fixed bandwidth share between the users receiving downlink traffic, an appropriate scheduling algorithm has been designed and implemented in a Linux based AP prototype. In particular, the developed scheduling algorithm is able to consider both the information on radio channel condition of the different stations and the transport service class required by the destination user.

The performance of the proposed scheduler have been experimentally analyzed and compared with those obtained by means of classical scheduling algorithms, such as DRR, CBQ and HTB, defined for wired network scenario. The performance analysis has highlighted the efficiency of the TWHTB; the experimental results show that classical scheduling algorithms are unable to maintain the assigned bandwidth shares to the stations. On the contrary, the proposed TWHTB scheduler permits not to penalize the STAs experimenting good channel condition and, as a consequence, their experimented goodput, independently of the channel conditions of other STAs.

Such result is due to the capacity of the proposed scheduling algorithm to limit the difference between the channel occupancy time of each STA with respect to the assigned bandwidth shares, independently of its channel condition. On the contrary, when classical schedulers are used, stations in bad channel conditions increase their channel occupancy time to the detriment of the STAs in good channel conditions.

Acknowledgment

This work has been sponsored by the Italian COFIN research program TWELVE. The authors wish to thank Dr. Giuseppe Risi for his help and support in setting up the experimental scenario.

References

[1] http://grouper.ieee.org/groups/802/11/

[2] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, *Performance Anomaly of 802.11b*, Proc. of IEEE Infocom 2003, San Francisco, April 2003

[3] D. P. Hole, F. A. Tobagi, *Capacity of an IEEE 802.11b Wireless LAN supporting VoIP*, Proc. of IEEE ICC 2004, Paris, June 2004

[4] R. G. Garroppo, S. Giordano, S. Lucetti, *Admission Region for Multimedia Services in IEEE 802.11e Systems*, Proc. Networks 2004, vol. 1, pp. 411-416, Wien 2004

[5] M. Portoles, Z. Zhong, S. Choi, *IEEE 802.11 Downlink Traffic Shaping Scheme For Multi-User Service Enhancement* Proc. IEEE PIMRC'03, Beijing, China, Sept. 7-10, 2003

[6] R. G. Garroppo, S. Giordano, S. Lucetti, E. Valori, *The Wireless Hierarchical Token Bucket: a Channel Aware Scheduler for 802.11 Networks*, Tech. Report TR-2004-12-01.pdf, http://netgroup-serv.iet.unipi.it/reports/TR-2004-12-01.pdf

[7] M. Devera, *Hierarchical token bucket theory*, May 2002, http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm

[8] R. G. Garroppo, S. Giordano, S. Lucetti, F. Russo, *IEEE 802.11b Performance Evaluation: Convergence of Theoretical, Simulation and Experimental Results*, Proc. of Networks 2004, vol. 1, pp. 405-410, Wien 2004

[9] G. Anastasi, E. Borgia, M. Conti, E. Gregori, *IEEE 802.11 ad hoc networks: performance measurements*, Proc. 23rd Int'l Conf. on Distributed Computing Systems Workshops, Providence, Rhode Island, USA 2003

[10] M. G. Arranz, R. Aguero, L. Munoz, P. Mahonen, *Behavior of UDP-based applications over IEEE 802.11 wireless networks*, Proc. of PIMRC 2001, San Diego, CA, USA 2001

[11] J. Tourrilhes, *Wireless LAN resources for Linux*, 1996-2003, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/

[12] J. Malinen, *Host AP driver for Intersil Prism2/2.5/3*, http://hostap.epitest.fi

[13] M. Devera, *HTB Linux queuing discipline manual - user guide*, May 2002, http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm

[14] BRUTE: Brawny and Rough Udp Traffic Engine, http://netgroup-serv.iet.unipi.it/brute

[15] S. Floyd, V. Jacobson, *Link-sharing and Resource Management Models for Packet Networks*, IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, August 1995