Providing Air-time Usage Fairness in IEEE 802.11 Networks with the Deficit Transmission Time (DTT) Scheduler

R. G. Garroppo, S. Giordano, S. Lucetti, and L. Tavanti
University of Pisa
Department of Information Engineering
Via Caruso, 16 - 56126 Pisa, Italy
{*r.garroppo, s.giordano, s.lucetti, luca.tavanti*}*@iet.unipi.it*

Abstract

Wireless systems based on the IEEE 802.11 standard are known to suffer a performance degradation when just a single station in the network experiences bad channel conditions toward the Access Point (AP). This phenomenon, known as the "performance anomaly", is mainly due to the max-min throughput fairness of the CSMA/CA algorithm of the 802.11 MAC. The simple FIFO scheduling policy usually implemented in the AP also contributes to this problem. In order to overcome the performance anomaly, we propose the Deficit Transmission Time (DTT) scheduler. The aim of DTT is guaranteeing each station a fair medium usage in terms of transmission time. This feature, directly related to the proportional fairness concept, allows to ideally achieve exact isolation among the traffic flows addressed to different stations. The DTT achieves this goal taking advantage of measurements of actual frame transmission times. Experiments carried out using a prototype implementation of DTT are compared with analogous tests performed with a classic FIFO queue of a commercial AP and a recently proposed traffic shaping scheme aimed at solving the same 802.11 performance anomaly.

1 Introduction

In the past few years, the IEEE 802.11 standard [1] has become the most popular technology for indoor and outdoor broadband Wireless Local Area Networking (WLAN). This success has fostered the research in increasing system's capacity, by means of more efficient modulation schemes and/or exploitation of other frequency bands. This has led to the ratification of the *a*, *b* and *g* amendments to the first standard, which allow rates up to 54 Mbps.

More recently, the Task Group *n* (TG*n*) is pursuing the definition of physical and MAC layer enhancements to further increase the maximum data rate. Concurrently, however, it has become evident that the simple increase of available data rate is not sufficient to offer adequate support to services with strict Quality of Service (QoS) requirements, such as Voice over IP (VoIP), multimedia streaming or other delay (or bandwidth) sensitive applications. As a consequence, new IEEE Task Groups have been created with the aim of developing new solutions to satisfy the requirements of these increasingly demanding applications and users. In particular, Task Group *e* (TG*e*) is defining MAC mechanisms to offer differentiation of the transport service.

Hence, it is clear how the IEEE is spending much effort to improve 802.11 networks, both in terms of pure throughput and in terms of supporting more appealing services such as voice and video. Nevertheless, one of the most critical factors driving the efficiency of such systems still remains the ability of the stations to overcome the issues imposed by the wireless channel. As it will be better outlined in Section 2, the quite unreliable capacity of the wireless links coupled with the the simple "First-In First-Out" (FIFO) strategy employed at the AP, leads to a rather unpredictable frame delivery time, which dramatically hampers network performance. A solution to these problems has been reckoned to reside in a smart scheduling algorithm to be deployed at the AP.

Several schedulers have already been proposed in literature [2]. Most of them rely on a model of the wireless channel: the links between the base station and the user devices are independent of each other and are subject to bursty errors. Markov models are often used to imitate the quality of the link (a comparison among the different models is presented in [3]). However, these models could be sometimes distant from the actual channel behavior, thus a system based on it could become inefficient. Consequently, a more reliable solution would be centering scheduling decisions on a real measure of the channel.

Starting from these observations, we propose a scheduling algorithm, denoted as Deficit Transmission Time (DTT) scheduler, which is able to take into account the actual behavior of the channel. The quality of the links is not measured with common metrics, such as Signal to Noise Ratio (SNR), but is quantified evaluating the amount of time spent for the transmissions toward each station. Using this approach, the resource to share between the nodes is not the total capacity of the channel, but the time the channel is in use.

The rest of the paper is organized as follows: Section 2 describes performance issues with current 802.11 systems, motivating the need for a scheduling architecture as the one proposed in Section 3. In Section 4 we introduce an utility function and some results on its study. These point out that proportional fairness is the goal to pursue for achieving a good balance point between fairness and efficiency. The proposed DTT scheduler,

described in Section 5, has been designed taking into accounts these hints. Section 6 presents some experimental results achieved using a prototype implementation of the DTT and compares them with those obtained with a commercial AP and with a recent downlink traffic shaping scheme. Finally, concluding remarks are summarized in Section 7.

2 Rationale

An IEEE 802.11 WLAN supports two fundamental modes of operation: an *ad hoc* mode and an infrastructured mode. In the *ad hoc* mode all stations within range directly communicate to each other, whereas the infrastructured mode is characterized by the presence of the Access Point, a particular station that bridges the traffic between the wired infrastructure and the associated stations. Since all the stations belonging to the same cluster, or Basic Service Set (BSS), share the same channel, a medium access control is necessary. The 802.11 standard specification includes two medium access functions: a mandatory Distributed Coordination Function (DCF) and an optional Point Coordination Function (PCF). The DCF is based on a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism with binary exponential backoff; the PCF is based on polling, and was originally conceived to support delay-sensitive applications. In practice, however, the PCF has never been accepted by manufacturers, which have not released commercial products supporting such mode of operation; consequently, our work is focused on the DCF. Detailed description of the DCF mode of operation has already been included in several papers, thus, for the sake of brevity, the reader is referred to [4] or [5] for more details.

DCF provides a fair sharing of the medium in terms of channel access probability. In saturation conditions (i.e. all stations always have frames to transmit) the long term probability of accessing the channel is the same for all the competing stations. As detailed later in Section 4, and more exhaustively in [6], this behavior is due to the implicit *max-min fairness* property of the CSMA/CA protocol. In ideal conditions (i.e. when all frames are of equal length, are transmitted at the same data rate, and none are lost due to collision or low SNR at the receiver), this implies that all stations are provided with an equal share of medium occupancy time, which further reflects in an equal share of the overall bandwidth. However, since the relative position of the stations may vary during time, and a non-zero collision probability exists when more than one station is active, some frames may not be received correctly. In this case, the IEEE 802.11 standard provides for the retransmission of the frame up to a configurable Maximum Retry Limit (MRL). It also gives the possibility of employing more robust and less efficient modulation schemes, whose pattern of use is not specified in the standard but freely designed and

optimized by the manufacturer of each device.

The consequence of these factors is twofold: on one hand, the throughput actually perceived at the application layer by the station subject to retransmissions and/or rate adaptation is obviously lower than the value expected in ideal conditions. On the other hand, transmission attempts at lower data rates occupy the channel at the expenses of other stations, which also experience a significant reduction of their available bandwidth, regardless of the condition of their own channel state. This phenomenon is often indicated as the "performance anomaly" [4] of 802.11, and is typical of both the *ad hoc* and the infrastructure mode of operation.

Moreover, in infrastructure mode, a second and more relevant factor influences the performance of the network. This factor is directly related to the centralized figure of the AP. Data frames originated in the wired infrastructure, and addressed to the various associated stations, are queued at the AP and then transmitted once the competition for the medium has been won. This simple scheme has two consequences.

Firstly, the AP usually represents the bottleneck of the system since it is the main traffic injector in the BSS, but has the same chance of seizing the medium as any other active station. Let us consider an infrastructured BSS with N associated stations, each sustaining a bidirectional symmetrical traffic flow with a corresponding peer in the wired infrastructure (as it may be for e.g. a VoIP call). In this case, the AP must handle a traffic volume N times higher than any of the other N stations, although it has the same channel access probability. Even worse case if classical TCP services like web browsing or file retrieval are considered, since the traffic volume ratio is dramatically biased in favor of the downlink direction. This comes from the client-server paradigm: small uplink queries (HTTP or FTP GET commands) typically produce a large amount of data to be sent to the users (the requested web page or file). These considerations allow us to assume that, in infrastructure mode, the large majority of traffic is injected in the BSS by the AP. Consequently the contention coming from other stations has a reduced impact on the system performance.

Secondly, commercially available APs use a basic single FIFO queue. This solution is the cause of a second kind of "anomaly", whose effects are even more remarkable given the previous considerations. Due to MAC layer retransmission policy and automatic rate adaptation, the degradation of the channel quality towards any of the associated stations causes an increase in the time spent by the AP to successfully transmit a frame (or else drop the frame having exceeded the MRL) to that particular station. Since all frames are served from a single FIFO queue, all the stored frames must wait for the currently served frame to be dequeued, event that occurs after either a successful transmission or a MRL exceeded event (this phenomenon is sometimes referred to as the "head-of-

line" blocking problem). Thus the transmission delay experienced by each frame is not only a function of the particular station which the frame is addressed to, but also of other factors such as the position of other stations and the number of frames enqueued in front of it. Additionally, the transmission delay is also a function of the uplink load. However, uplink accesses may be assumed to be uniformly distributed for all the queued frames, and statistically affect all frames in the same manner. Hence, they can cause an increase in the average time spent in the queue, but do not differentiate between frames addressed to particular stations.

The ultimate result is that all the downlink flows are affected by increased delay, possible packet loss due to buffer overflow and reduced throughput at application level. Thus, in order to achieve a better utilization of the medium, and eventually guaranteeing the isolation among flows addressed to different stations, a smarter scheduling algorithm, aware of the channel state and working on top of a multiple queue system, must be designed and integrated into the MAC layer. Note however, that the use of a generic multi-queue scheduling algorithms, such as the Round Robin (RR) or others not aware of channel state, produces a virtual "head-of-line" blocking phenomenon, as highlighted in previous experimental analyses [7].

An optimal scheduling solution to overcome the "performance anomaly" should be based on a distributed algorithm, based on a coordination among all the stations and the AP. However, taking advantage of the assumptions on limited contention effect due to traffic asymmetry, adopted also in [8], it is possible to approach the scheduling problem in a centralized way. This solution, although sub-optimal, permits to focus our attention only on the AP queueing discipline, leading to a noticeable simplification of the problem formulation and solution approach.

2.1 Related Works

Channel state dependent packet scheduling (CSDPS) is one of the first scheduling algorithms facing the problems raised by the wireless channel [9]. Packets addressed to each station are held in a separate queue and served in FIFO order. The authors envisions several service policies to choose the queue where to pick the next packet. A module called link status monitor (LSM) tracks the link state for all hosts and when it determines that a link is in a bad state, it marks the related queue. A link is considered being in a bad state when the acknowledgment for a data packet is not received. The queue is unmarked after a time-out period, which may be, for example, the average link error duration. Marked queues are not served by the scheduler. Whereas simulations show that, compared to a single FIFO queue, CSDPS can achieve much higher data throughput and better channel utilization, it still has a major flaw. Links are "believed" to be in a bad state, and the queue is marked as unserviceable for a

period that is "believed" to reflect the real channel behavior. Therefore it is likely that a station receives much less (or much more) service opportunities than its actual link state allows, thus leading to unfairness among stations or bandwidth wastage (due to erroneous channel state estimation). Indeed, bandwidth wastage happens in two cases: the first occurs when the medium is idle because all queues are marked, and hence inactive, but not empty (non-work conserving behavior); the other occurs when a queue with a bad channel state is served because it is "believed" to have a good channel state, causing the frame to be corrupted. Also, CSDPS does not impose any limit on the amount of service received by each user.

Reference [10] describes the Server-based fair approach (SBFA). In this scheme, part of the bandwidth is allocated to a compensation server, called long-term fairness server (LTFS). An LTFS is a special data flow created for compensating flows whose packet transmissions are deferred because of link errors. The LTFS shares the wireless channel with other regular data flows. The scheduler maintains two queues, packet queue (PQ) and slot queue (SQ) for each flow. When a packet of flow i arrives, it is put into PQ_i and a virtual copy of the packet is put into SQ_i. Any service policy may be used to choose the next queue to be served among the PQs plus the LTFS. However, the chosen packet is actually transmitted and removed from the PQ only if the link it uses is in a good state. Otherwise, only the virtual copy is removed from the SQ and put in the LTFS. When the scheduler picks a "packet" from the LTFS, then the real copy of the packet in the corresponding PQ is transmitted and the virtual copy removed from the LTFS. Thus, flows losing their service share because of link errors will eventually be compensated. However, as for CSDPS, SBFA too relies on an estimate of the channel that may not mirror the actual state.

A more recent approach, that encompasses real link conditions, has been proposed by Portoles et al. in [8]. The authors introduce a limit that specifies, for each destination station, the maximum number of packets enqueued at the AP, competing for the wireless medium. The algorithm dynamically adjusts these limits based on the observations of recent transmission statistics. When a link to a particular station goes bad, i.e. the AP observes transmission failures, the algorithm lowers the corresponding limit (half its previous value). Conversely, when the link recovers, i.e. the AP observes transmission successes, the algorithm relaxes the restriction and doubles the limit. Thus, the algorithm can dynamically regulate the flow of packets into the device. It should be noted, however, that the proposed algorithm has no tight control on the order of the packets in the queue, hence it is not able to prevent the occurrence of sequences of packets addressed to a station with poor link quality. As a result, the traffic shape may be far from smooth and, above all, the head-of-line blocking problem is still present, since

frames are served according to the FIFO strategy.

3 System Architecture

In order to experimentally verify and measure the effectiveness of employing channel state dependent schedulers, we have developed a software framework to embed the schedulers in a prototype AP. Our framework is implemented as a set of Linux kernel modules, tightly integrated with the *Host AP* driver [11] for Intersil's Prism2.5 based WLAN devices. The most relevant components of the framework are highlighted in Figure 1. This architecture is integrated in the standard protocol stack of Linux systems to allow an easy and straightforward implementation without the need of custom-made MAC controllers. It also allows a simple software upgrade, needing no expensive hardware replacement. However, we foresee that in the future the same principles of operation constituting the basis of the framework may be integrated in customized programmable MAC controllers; as a side effect, such an approach would additionally reduce the scheduling architecture complexity, as it will become clear analyzing its operation.

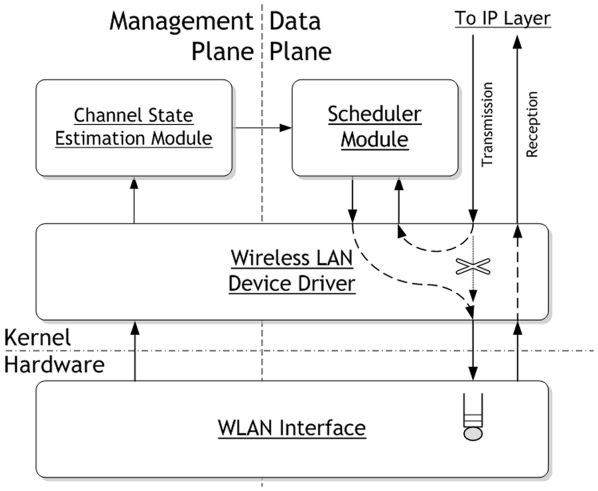


Figure 1. Overall System Architecture

In commercial APs, packets addressed to wireless stations are typically served according to a single-queue FIFO strategy. In simple Linux based APs, things are analogous: packets addressed to wireless stations are framed by the device driver, passed to the device, and queued in a hardware buffer waiting for all the preceding frames to be transmitted.

Our framework integrates the scheduler at the device driver level, as a kernel module, hence before the hardware queue. According to measurements presented in [12], this queue holds up to 50 frames (1500-byte long). This

queueing may actually reintroduce the head-of-line blocking, thus compromising the effect of the driver-level scheduling. Therefore it would be advisable to keep the hardware queue empty, fetching a single frame from the host memory only when the previous frame has been dequeued (after successful transmission or MRL exceeded). However, due to current hardware limitations, it is not convenient to have only one frame (the one competing for a transmission opportunity) stored in the hardware buffer. The time needed to transfer the next frame from the host memory to the device is much higher than the average idle time between two consecutive transmissions (DIFS plus backoff period). This will introduce an unnecessary delay which severely reduces the maximum achievable throughput. Hence we have implemented a mechanism that manages to keep, at maximum, only two frames in the hardware queue. One of them is the frame currently being served, waiting for a transmission opportunity or a positive acknowledgement, whereas the other frame is actually waiting in the queue.

A detailed analysis of the related timings is outside the scope of this paper; however, as an example, the saturation throughput of a single UDP flow from an 802.11b AP to a station, using 1500 byte packets, falls from the ideal value of about 6.1 Mbps (analytically evaluated and confirmed by measurement on commercial devices [5]) to around 3 Mbps. Having a frame immediately available for transmission as soon as the previous frame is dequeued allows our system to reach the exact expected saturation limit, without interfering with the scheduling decisions performed at the device driver.

Apart from the Scheduler Module, which in fact incorporates the classifier, the various queues and the scheduler itself, the most important component of the framework is the Channel State Estimator (CSE) Module. This module is devoted to the estimation of an indicator of the quality of the link between the AP and any associated station. This is done by means of measurements and events originated by the device. Several kinds of channel estimates are possible, depending on the amount and nature of the available information, and the type of channel knowledge that the scheduler must have to operate properly. The various CSE realizations may be classified according to their stochastic nature (i.e. if they provide a statistical or a deterministic indication), or to the kind of knowledge they make available (e.g. time spent for a frame transmission, estimated SNR at each receiver, or saturation throughput towards each station). In this paper we describe two CSEs, differing in both their stochastic nature and channel description; a short outline of both, along with their respective strengths or weaknesses, is reported in the next subsections.

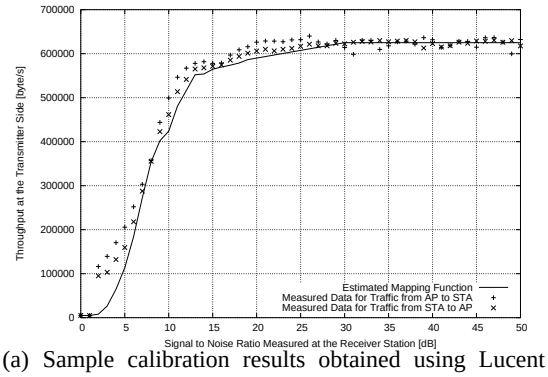
3.1 Wireless Channel Monitor

The Wireless Channel Monitor (WChMon) is a tool originally developed at the Washington University in St. Louis and described in [13]. It produces an indication on the maximum achievable throughput towards a specific destination station, using SNR measurements. It is based on the presence, in almost all currently available wireless chipsets, of an Automatic Gain Control (AGC) unit that monitors the signal condition and adapts the RF circuit of the card. This information is also used to compute three values indicating the signal level, the noise level and the signal quality (other designs only provide one value indicating the signal level). These values are stored in the Parameter Storage Area (PSA) of the wireless card and can be read by a device driver which exports them to the kernel memory. The reported values are related to the signal and noise levels in dB (allowing to estimate a SNR indicator), and are computed for each received packet.

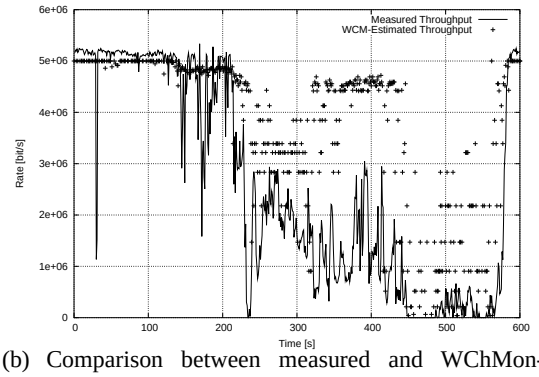
SNR and bit error rate are strongly correlated, so are the SNR reported by the card and the probability of receiving a frame with errors (*Frame Error Rate*, *FER*). As a consequence, channel quality degradation is involved in the reduction of the attainable throughput on the channel. Since a successful frame transmission is detected by the transmitter through the reception of a positive ACK, the actual link quality in a transmission from the AP to a STA is a combination of the data frame SNR (at the STA) and the ACK frame SNR (at the AP). Such a complete knowledge is rather difficult to obtain without modifying some fields in the frames sent by the stations, because the AP should be informed of the SNR measured at each station. However, channel symmetry hypothesis, given the principle of reciprocity, justifies the assumption of the same SNR value at both ends of each AP–station link.

The driver first computes the SNR associated to the received frame transmitted by a particular STA. Then it transfers this information to the WChMon which, by means of a table, estimates the maximum throughput the AP expects to reach towards that mobile station. This information is then used to feed the scheduler module, which utilizes it according to its specific scheduling policies.

The main drawback of this monitoring system is represented by the tight dependence on specific network cards and corresponding drivers: although all manufacturers shall conform their products to the IEEE 802.11 standard, current implementations may differ (and actually do) in several ways, both in the hardware and software [14]. As a consequence, receiver sensitivity may change, so different cards may offer different transmission capacity for a given SNR; therefore, an initial calibration of WChMon, according to the procedure described in [7], is necessary in order to build the table of SNR to throughput. The calibration procedure represents a key phase, because the scheduler performance completely relies on throughput estimates. On the other hand, once calibration



(a) Sample calibration results obtained using Lucent *Orinoco* 802.11b devices



(b) Comparison between measured and WChMon-estimated saturation throughput

Figure 2. Performance of Wireless Channel Monitor (WChMon)

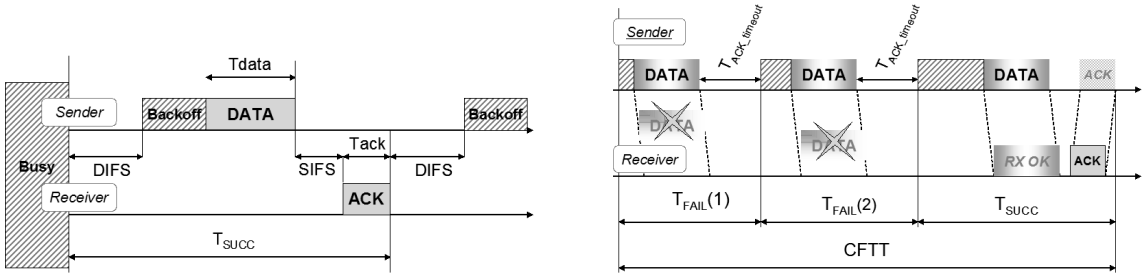
has been performed, WChMon has proven to be fairly good at approximating the attainable throughput towards a destination, on the sole basis of the SNR value of the received frames from that destination. In the case of heterogeneous cards, one SNR-to-throughput table for each different card is necessary. This obviously poses a noticeable limit on the possibility to actually deploy a channel state dependent scheduler based on WChMon.

As a reference result, Fig. 2(a) shows a sample calibration outcome for Lucent *Orinoco* 802.11b devices, highlighting the symmetry of the two paths. Fig. 2(b) compares the saturation throughput estimated using WChMon and the measured throughput, in the case of a single station moving back and forth from the AP; it shows that, once calibrated, the WChMon provides a fairly good approximation of the attainable throughput towards a destination, on the sole basis of the value of SNR of the received frames from that destination.

3.2 Cumulative Frame Transmission Time Measurements

In designing our CSE module, we have chosen a different approach, based on the exact measurement of the time needed to transmit (either successfully or not) a frame. Neglecting the propagation delays, a single frame transmission attempt is composed of a number of different time contributions [5][15], as Fig. 3(a) highlights for T_{SUCC} in the case of successful delivery. Note that, although the *DIFS* and *Backoff* periods are not strictly related to frame transmissions (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 must be respected independently of the system conditions.

With similar considerations, we have extended the analysis to the case of frame delivery requiring one or more retransmissions: so we have defined the Cumulative Frame Transmission Time (CFTT) as the overall amount of



(a) Time budget of a data frame transmission

(b) Example of CFTT in the case of 2 retransmissions

Figure 3. Timings of 802.11 frame transmission

time needed to send a frame, regardless of the outcome, including possible retransmissions (up to the MRL) and rate reductions. Fig. 3(b) illustrates the case of a successful frame transmission composed of two unsuccessful and one successful events (the third). The overall CFTT can be expressed as $CFTT = \sum_{k=1}^{N} T_{FAIL}(k) + T_{SUCC}(N)$, where N is the number of unsuccessful transmission attempts. With $T_{FAIL}(k)$ we indicate the time associated to the k–th failed attempt. The index k takes into account the different values of $T_{Backoff}$ that come from the 802.11 exponential backoff algorithm; since $T_{Backoff}$ is a random variable, these durations are random variables too. Furthermore, if any rate adaptation algorithm is employed, each retransmission attempt may occur at a different bit rate. Thus, $T_{SUCC}(N)$ too, if present (i.e. if $N < MRL$), is not constant and depends on N as well as on the particular rate adaptation strategy adopted by the device manufacturer. Finally, if N reaches the MRL, the frame is removed from the queue and discarded. The time of such an event defines the end of the CFTT period for unsuccessful transmissions. Note that the transmitter assumes the frame transmission attempt to be successful only at the reception of an ACK. Should an ACK get corrupted, the transmitter behaves as if the data frame had been lost, i.e. it schedules a retransmission.

CFTT measurement has two main advantages over WChMon: it does not need any calibration, and it produces deterministic channel information, rather than average values. An analytical evaluation of achievable CFTT values in saturation conditions is presented in [16]; typical values (in the case of MRL equal to 4 and a packet size of 1024 bytes) range from about 1.3 ms to about 47 ms.

With regard to the practical implementation, we have extended the device driver to inform the CSE module of any ACK frame reception or MRL exceeded event. Using this information, the CSE is able to compute the CFTT for any frame. Once the CFTT for a given frame is evaluated, its value is sent to the scheduling module, for proper processing of queued frames.

These considerations are based on the assumption of a single active transmitter in the BSS (i.e. the AP); if

another station performs a frame transmission between two transmission attempts (related to the same frame) from the AP, the CFTT will incorporate that duration too. In principle, this time should be subtracted from the CFTT. Practically, given the traffic asymmetry in the downlink direction, these events are limited and, in order to keep the CSE module simple, the current implementation does not perform such an operation. We will show in Section 6 that this simplification does not affect scheduling performance in a noticeable way.

4 Utility Function

In order to properly design a channel dependent scheduler to be integrated at the AP, it is necessary to define an objective to be maximized. In wired networks, the design of traffic control algorithms has been carried out optimizing two relevant parameters: fairness among different flows and efficiency in link utilization. In wireless networks, and particularly in 802.11 WLAN due to the specific MAC protocol, these two parameters are somewhat conflicting. Recent experimental analyses (e.g. [17]) have shown that, in multirate wireless networks, throughput fairness leads to bandwidth underutilization. Focusing on WLANs, this result is confirmed by [4], which analytically derives that global performance of a BSS is determined by the stations using the lowest data rate. The same effect manifests itself at the AP for downlink flows addressed to stations with different channel quality, as indicated in Section 2.

In this scenario, it is a fundamental choice whether one should strive to achieve max-min fairness (optimize "fairness"), maximize the total throughput (maximize "efficiency"), or strike a balance by adopting a proportional fairness objective [18]. Typically, achieving one of these goals is directly related to the maximization of a particular "utility" metric, different for each goal. A widely adopted general expression for such a metric is $\sum_j U(x_j)$, where x_j is the rate of flow j and $U(\cdot)$ is a concave function (called the utility function). Obviously, the characteristics of the utility function $U(\cdot)$ directly affect the properties of the utility metric, and consequently the particular fairness objective which is pursued. The most used class of utility functions is the one proposed in [19], which is described by the following expression:

$$U(x,\alpha) = \begin{cases} (1-\alpha)^{-1}x^{1-\alpha}, & \text{if } \alpha \neq 1; \\ \log(x), & \text{if } \alpha = 1. \end{cases} \tag{1}$$

In this equation, x represents the throughput allocated to each station, and α is a parameter that can be modified to tune the tradeoff between efficiency and fairness in order to achieve various optimization objectives. In

particular, when $\alpha = 0$, the utility function leads to the extreme goal of throughput maximization, at the complete expenses of fairness. A scheduler which pursues such a goal would allocate the medium to the station with the highest data rate, whereas low data-rate stations would go into starvation. Typically, this is not a desirable solution. Note that here (and in the rest of this Section) the term "scheduler" may indifferently refer to medium access contention solving or to actual downlink frame scheduling at the AP.

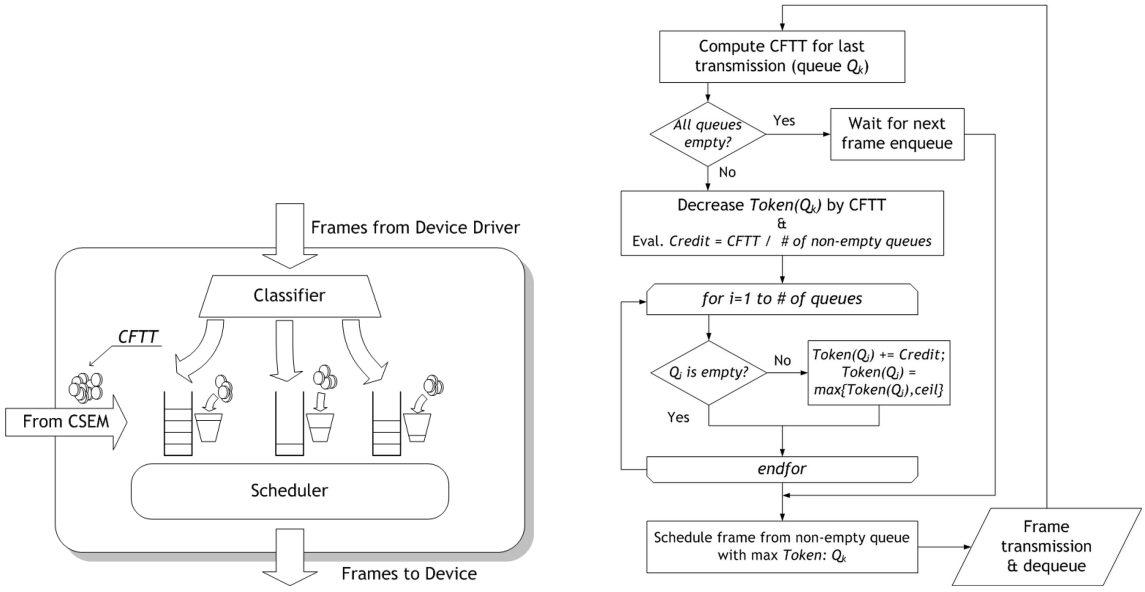
When $\alpha \rightarrow \infty$, the utility function leads to the max-min fairness in bandwidth usage. Contention solving mechanism of 802.11 MAC implicitly adopts this utility function, leading to a long term fairness of channel access probability among all the competing stations. Even egalitarian scheduling algorithms at the AP, such as FIFO or Round-Robin, turn out to produce the same effects of a max-min fair bandwidth allocation, as exposed in Section 2. Since transmissions at lower physical nominal bit rate occupy the channel for longer time, the max-min fairness in bandwidth usage does not translate to max-min fairness in air-time usage.

One of the results of the study presented in [6] is that, in *ad hoc* WLANs, proportional fairness, obtained using $\alpha = 2$ in $U(\cdot)$, allows to obtain a reasonable tradeoff between efficiency and fairness. Furthermore, authors in [20] demonstrate that in an infrastructured WLAN, given a fixed number of stations, the throughput of any of them is independent of the data rates used by the others if proportional fairness (in bandwidth) is achieved. This property is deduced from the demonstration of the following proposition: ignoring the protocol overhead such as PHY header, MAC header, backoff time, ACK, IFS, etc, proportional fairness is achieved when the fractions of air-time usage by the stations are equal, with air-time usage taking into account both uplink and downlink transmissions. The preposition and the related property lead to deduce that proportional fairness in bandwidth (or equivalently max-min fairness in air-time usage) yields a good trade off between fairness and efficiency in a multi-rate WLAN, as well as leads to the very desirable property of flow isolation. In the case the transmissions are either in the uplink or downlink direction only, proportional fairness is equivalent to air-time usage fairness.

5 Proposed Scheduler

The results presented in the previous Section led us to design a scheduling algorithm whose goal is to achieve proportional fairness in infrastructured WLANs. As discussed in Section 2, since the volume ratio of offered traffic is biased towards the downlink direction, the scheduler focuses on the fair air-time usage among the flows originated in the wired portion of the network and addressed to the associated wireless stations. The architecture of the proposed scheduler is depicted in Fig. 4(a). The scheduler is implemented at the AP and operates according

to a centralized policy. It splits outgoing traffic into several queues, on the basis of predefined classification rules. Currently, classification is performed according to the destination MAC address, but it can be easily extended to support different user priorities, as in 802.1Q and planned for adoption in 802.11e draft. Each queue has an associated "bucket" that accounts for the air-time usage taken for the previous frame transmissions. The rules for filling/emptying the buckets are specified in the flowchart of Fig. 4(b).



(a) Architecture of DTT. The contents of the buckets are adjusted after receiving the feedback on previous frame transmission.

(b) Flowchart

Figure 4. General architecture and flowchart of the DTT scheduler

For each frame the CSE Module computes the CFTT, i.e. the time spent for its transmission. This period shall include all MAC retransmission attempts, if any, and is also computed in case of transmission failure, i.e. when the MRL is exceeded. The bucket associated to the destination of the frame is then drained by a quantity of tokens equal to this CFTT (measured in CPU cycles). Next, this same number of tokens is divided among all the non-empty queues, filling their corresponding buckets up to a predefined ceiling value, which in the current implementation is assumed infinite. At the current stage of development, we do not consider traffic differentiation issues, and hence tokens are divided in equal shares, according to the air-time fairness goal identified in Section 2. Note that the filling phase also includes the queue whose frame has just been sent, if non-empty, to grant it the same transmission possibility of other queues. A relevant exception is when all queues are empty: in this case, tokens are not removed from the queue, in order to keep a short-term constant number of tokens in the system.

Concurrently, all the buckets associated to queues that have been empty for an inactivity timeout value are cleared (set to zero). This avoids that some stations, idle for a long period of time, inherit a token credit or debit which would reduce short-term fairness once they have again some frames to transmit. Once these tasks have been completed, the scheduler picks the next frame to be transmitted from the queue whose associated bucket has the highest number of tokens, choosing randomly in case of two or more stations with equal credit.

The algorithm automatically manages the association and disassociation of stations. It initializes a new queue, setting to zero the number of tokens of the corresponding bucket when a station associates, and marks as inactive the queue of disassociated stations, until the expiry of a further timeout; after that period from disassociation (currently zero) the queue is flushed and the memory freed. The scheduler has no impact on management and beacon frames, that are generated directly by the firmware, which resides in the hardware device, hence well below our scheduler.

A few things are worth noting. First, the tokens used to fill/empty the buckets are directly related to a transmission time, thus the bucket with the highest number of tokens is also the one whose associated queue has transmitted less. Hence the name of the scheduler: Deficit Transmission Time (DTT). Second, the tokens are an actual measure (not an estimate, nor a prediction) of the channel quality. More retransmissions, possibly at lower bit rates, are needed to deliver the frame to the stations whose channel quality is poor. As a consequence, such destinations get their buckets emptied by larger quantities of tokens, thus having to wait longer before being chosen again. In contrast, easily reachable destinations get their buckets lightly drained (when their frames are sent out) and heavily filled (when difficult-to-reach stations' frames are on air). Third, this scheme is conservative, in the sense that the number of tokens in the buckets (that might also be negative) does not diverge, quite the reverse it always tends to zero. Finally, as noted above, by introducing some weights when distributing the tokens, traffic can be easily differentiated on the basis of various parameters, such as MAC address destination or, in combination with an extended classification scheme, IP Traffic Classes (if such knowledge is available) or 802.1Q VLAN Priority tags.

6 Experimental Results

The experimental trials have been performed over a testbed of one AP and a variable number of associated stations, depending on the specific try. The stations change their position, moving closer and farther from the AP, until they exit its radio coverage range, and thus are disassociated. Movement was controlled defining a path, and establishing a number of stops. The duration of each stop (and the walking speed between consecutive stops) has

been mostly measured using a reference timetable. A qualitative indication of the received throughput and the signal strength (although not needed) was collected during the tests using on-line network monitoring tools.

In order to increase the system performance, timeout values to force disassociation of poorly connected stations has been reduced. The inactivity timer before probing for loss of connectivity has been lowered in the *Host AP* driver from 300 to 30 seconds; thus we have a faster response of the scheduler when a station is in a very bad position, as the disassociation occurs sooner.

All the devices are equipped with Prism2.5 IEEE 802.11b network cards, providing a maximum data rate of 11 Mbps. UDP traffic is generated using the MGEN traffic generator [21]. TCP traffic is obtained via an FTP file transfer session. The experimental testbed is depicted in Fig. 5: one host (Wired Host) acts as a corresponding peer, which communicates with each station through the AP. The host may generate (or be the sink for) UDP flows, as well as act as FTP server. The actual movements of the stations, as well as the traffic patterns, are described for each specific measurement run.

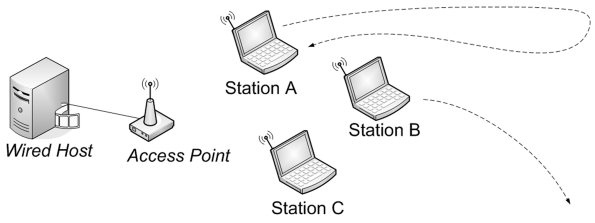


Figure 5. Experimental testbed

For the purpose of having a touchstone for our scheduler, we compared its behavior with another solution to the performance anomaly, proposed in [8] and already described in Subsection 2.1, which we will refer to with the acronym PZC from the authors' names. Then, to have a real baseline, we also employed a commercial AP, the HP Procurve 420, which natively implements a simple FIFO discipline. This AP is b/g capable; to reproduce the exact scenario we have configured it (using administrative tools) to support only the b standard.

Experimental results are presented in the following. Each set of measurements has been focused on verifying particular scheduler properties, such as flow isolation, borrowing of spare bandwidth among queues and scalability. These however are not the only tests we carried out on the proposed scheduler. Experiments with different packet sizes and with more than three stations have also been led. They confirmed the effectiveness of the scheduler, but added few extra information with respect to the tests discussed in the following Subsections. Hence, for the sake of brevity, they have not been included.

By the way, in a separate work [22] we also evaluated (via simulation) the DTT in presence of voice connections, thus with symmetric uplink/downlink traffic. The performance has been measured in terms of the number of VoIP calls that can coexist in a BSS with an acceptable perceived speech quality. Similarly to the present paper, we registered a noteworthy improvement over a simple FIFO-based AP, thus concluding that the presence of uplink traffic does not significantly affects the performance of our scheduler.

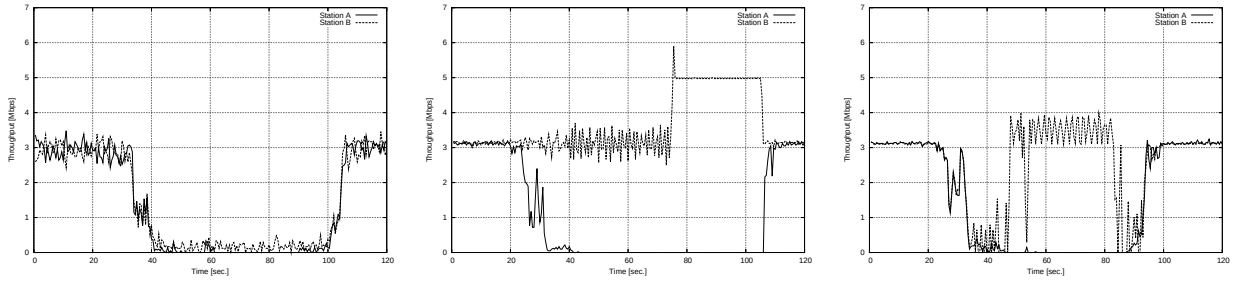
6.1 Flow isolation

The tests presented in this subsection show the flow isolation feature of the proposed scheduler. Flow isolation is the consequence of max-min fairness in air-time usage, or, in other words, of proportional fairness in bandwidth. It basically means that the throughput achievable towards a station is not influenced by the position of other stations. In order to highlight the DTT effectiveness with respect to the other schedulers, we present Fig. 6. Here, the wired host generates two 5 Mbps UDP flows, with an IP packet length of 1500 bytes, addressed to two wireless stations; the traffic offered to the AP is then higher than the value that can be served in ideal conditions (about 6.2 Mbps), thus the system is saturated. After some time, one station (station A) moves away from the AP, worsening its link quality, until it disassociates. Afterwards, the station returns to its starting position, with almost ideal channel conditions. The other station (B) is constantly kept in optimal radio visibility with the AP.

Throughout the whole test, when both stations are in a good position, the two flows evenly share the available bandwidth; each station receives about 3.1 Mbps, which is half of the available saturation throughput. When station A starts moving, the differences become noticeable. Measurements under the commercial AP (see Fig. 6(a)) clearly reveal the anomaly of the 802.11 (and the max-min fairness in throughput): station B is heavily hindered by the simple FIFO policy and cannot exploit its bandwidth share.

The enhancements introduced by the other two schemes are evident. As for the DTT scheduler (see Fig. 6(b)), station B keeps receiving its data flows at roughly 3.1 Mbps almost irrespective of the other station's position. We can note just some oscillations, due to the unavoidable attempts to deliver the frames to the far terminal. As soon as station A is disassociated, station B acquires the full control of the channel. By the way, the 6 Mbps peak in the throughput, present in some of the following graphs as well, is merely due to the backlogged packets in the AP transmission queue. From the figure it is therefore evident that our scheduler allows each station to get the maximum of its bandwidth share independently of the other; in other words, it is able to isolate the flows, achieving proportional fairness.

With regard to Fig. 6(c), when station A begins experiencing link degradation, the PZC scheme reacts trying to allocate more and more bandwith to the closer station, reducing the number of queued frames addressed to station A. However, it cannot avoid a non negligible transient period. This is due to a couple of reasons. The first is that all the flows share the same queue. The second is that, following to previous successful transmissions, the PZC has raised the number of enqueued frames to the maximum. Consequently, the AP must get through a backlog of frames before handling the new situation. Also, the head-of-line blocking phenomenon is still perceivable in the oscillating throughput values. Additionally, as soon as a frame addressed to station A is correctly transmitted, further frames are allowed to be enqueued. Some improvement in the throughput towards station B is noticeable only when link quality of station A is severely reduced (after about 45 seconds). In this case, very few (one or two) enqueued frames belong to the UDP flow to A, causing only a limited impairment on transmissions towards station B, which thus receives several consecutive frames. Remarkably, the maximum throughput that B can experience depends on the maximum and minimum number of frames addressed to B that the PZC allows to be enqueued between two transmissions to A. The higher this value, the higher the throughput, but also the higher the reaction time. In conclusion, although in some periods the overall efficiency of PZC exceeds that of DTT, flows are not isolated, and fairness is reduced.



(a) Commercial AP (b) DTT Scheduler (c) PZC Scheme

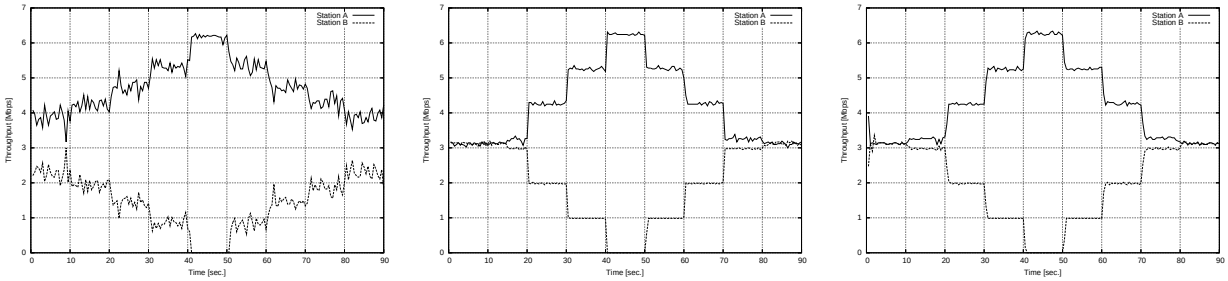
Figure 6. Isolation between two 5Mbps downlink UDP flows for the different APs

6.2 Spare bandwidth borrowing

The DTT scheduler has been designed to grant an equal share of air-time to all the associated stations. However, if some of them have no frames to transmit, their unused air-time share is distributed to the stations with non-empty queues, according to the work-conserving principle associated to the token distribution algorithms presented in Section 5. Fig. 7 reports the outcome of a test in the simple case of two stations. Two downlink UDP flows, composed of 1500 byte packets, are generated. The first is addressed to station A, has a data rate of 6.5 Mbps, and

is able to saturate the wireless link. The second follows a piecewise step function: every 10 seconds it switches in sequence among the values 4 Mbps, 3 Mbps, 2 Mbps, 1 Mbps, 0 Mbps, and then back to 4 Mbps at steps of 1 Mbps. Both stations are kept in optimal channel conditions.

Very similar results are achieved by DTT and PZC: as long as the traffic offered to station B keeps above 3.1Mbps (half of the saturation throughput), the two flows evenly shares the available transmission capacity. When some air-time is left free by transmission towards station B, because of reduced offered load, such time is exploited by the other station, which eventually manages to saturate the channel on his own. In contrast, the commercial AP is not able to guarantee a fair and stable channel share even in this simple case. This is due to the different amount of enqueued frames, as a consequence of the different offered load. The resulting throughput just reflects this point. This case is clearly a further evidence of the convenience of employing a multi queue scheduler at the AP. The PZC scheme, although adopting a FIFO strategy, has a finer control of queue occupancy, allowing an equal number of frames to be enqueued, avoiding such unfair behavior.

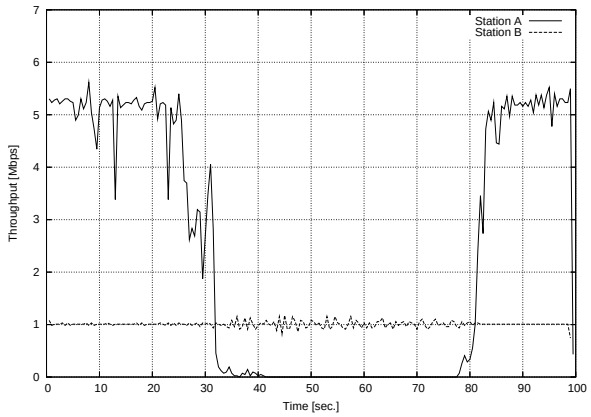


(a) Commercial AP (b) DTT Scheduler (c) PZC Scheme

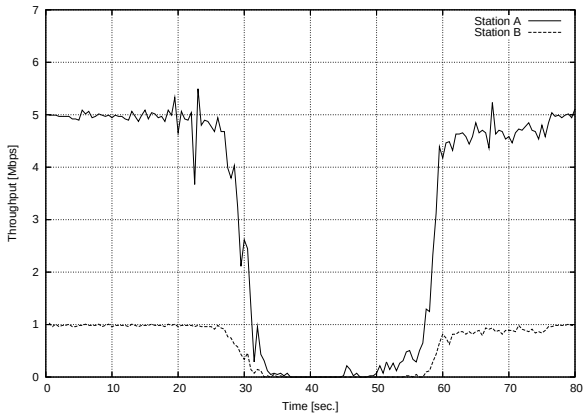
Figure 7. Spare bandwidth borrowing with the three APs

The dynamics of spare bandwidth borrowing can be further analyzed considering Figg. 8 and 9. The scenario is still the base one, with one station moving away from the AP, until disassociation occurs, and then coming back in its original position. The difference with Subsection 6.1 resides in the offered traffic: the flow addressed to station A has a data rate of 6 Mbps, whereas the other is just 1 Mbps. Measurements reported in Fig. 8(a) refer to station A moving away from the AP. Clearly, when both stations are in a good position, bandwidth borrowing occurs. Station A receives 5.2 Mbps, that is the sum of its fair share of available throughput (3.1 Mbps) plus the 2.1 Mbps that are not used by station B. When station A moves away from the AP, its throughput decreases, but the flow towards station B is unaffected, thanks to the flow isolation property of the DTT scheduler.

In a second try, station B moves away from the AP (see Fig. 9(a)). When it starts suffering bad channel conditions, less and less spare bandwidth (i.e. air-time) is left for transmissions to station A, until all the excess

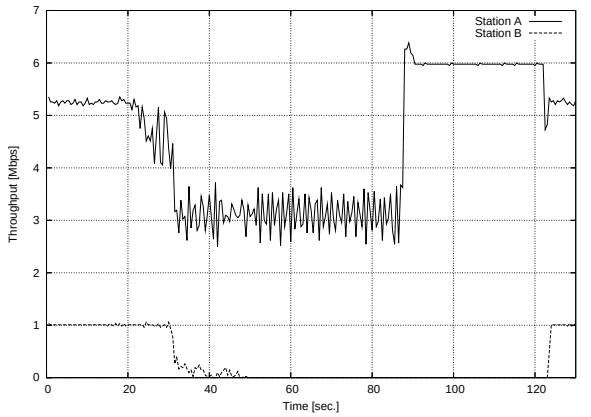


(a) DTT scheduler

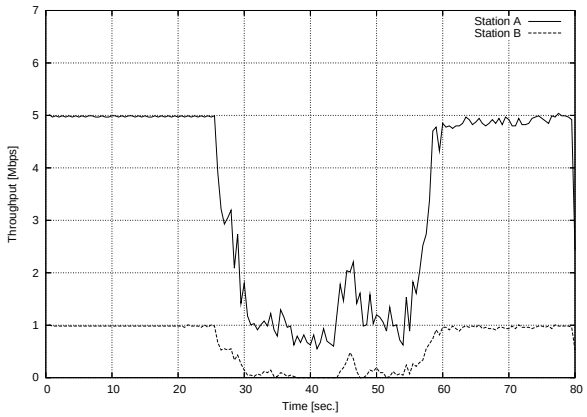


(b) FIFO scheduler

Figure 8. Combination of flow isolation and spare bandwidth borrowing - Station A moving away from the AP



(a) DTT scheduler



(b) FIFO scheduler

Figure 9. Combination of flow isolation and spare bandwidth borrowing - Station B moving away from the AP

air-time is taken back by station B to sustain its 1 Mbps data rate, and the data rate perceived by station A drops to 3.1 Mbps, since the DTT scheduler enforces an even share of air-time usage. When B is finally disassociated, station A can enjoy the whole channel.

Fig. 8(b) and Fig. 9(b) show the results obtained in the same scenarios employing the commercial AP. The improvements introduced by DTT are apparent.

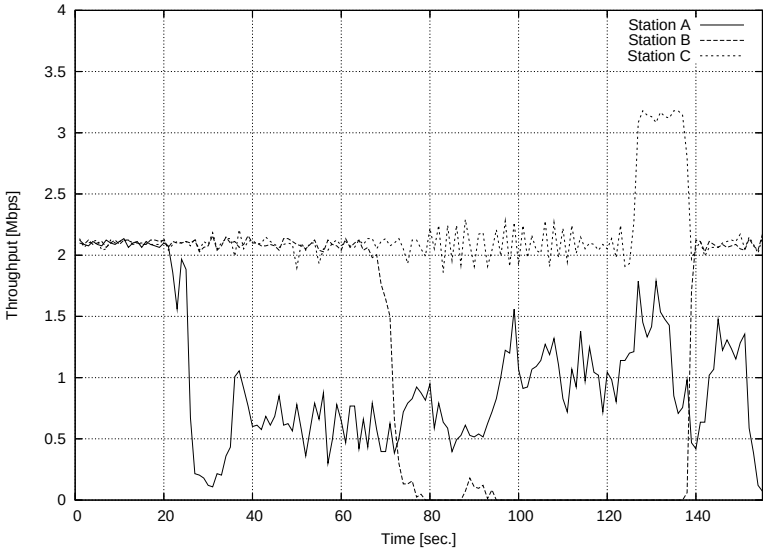


Figure 10. Scenario with three associated stations and downlink UDP flows

6.3 Scalability

The principle of operation of the DTT scheduler, and the corresponding implementation in the Linux kernel, are completely general and do not make any assumption on the number and position of the associated stations. Yet, to increase the confidence on the achieved results, some measurements involving more than two stations have been carried out. The set of possible patterns of movements, timings and traffic load is extremely vast; as a reference test case, we present just a simple example that is used to verify the ability of the proposed scheduler to scale from two to three stations. Addressing the large scale scalability issue is outside the scope of the paper, and rather difficult to be analyzed through real experiments. This is mainly due to the lack of availability of a considerable number of devices. A simulation analysis for a large number of bidirectional VoIP peers (up to 25) has been performed and the results collected in [22].

In the present example, one station (say A) starts moving away from the AP until it reaches a position in which its link becomes weak, but is never cut. Later on, a second station (B) departs from the AP, until it is disassociated. Towards the end of the test, B re-enters the AP coverage area. The third station (C) always has a good connection to the AP. We have thus tried to create a scenario that encompasses more complex features than the basic cases of previous Sections. For instance, we have a station (A) that is constantly half the way between the good and bad link states, which may represent a user standing in an unfavorable place.

The behavior of DTT is reported in Fig. 10. As expected, when station A starts moving, the scheduler man-

ages to make this event unnoticeable to the other stations, which perceives no significant changes in the received throughput. A similar behavior has been observed also when B moves. At last, when B is disassociated (at 125 s), both A and C can get more channel capacity, but only C is able to exploit it at its best, reaching 3.1 Mbps. Station A only improves slightly, being still subject to poor channel conditions. Finally, when B re-enters the network, it can have its air-time share back, which is equally subtracted from those of A and C. In conclusion, apart from some short-term oscillations, the three flows are still isolated. This is a valuable confirmation of the goodness of DTT even for more complex scenarios.

6.4 Mixed UDP and TCP traffic

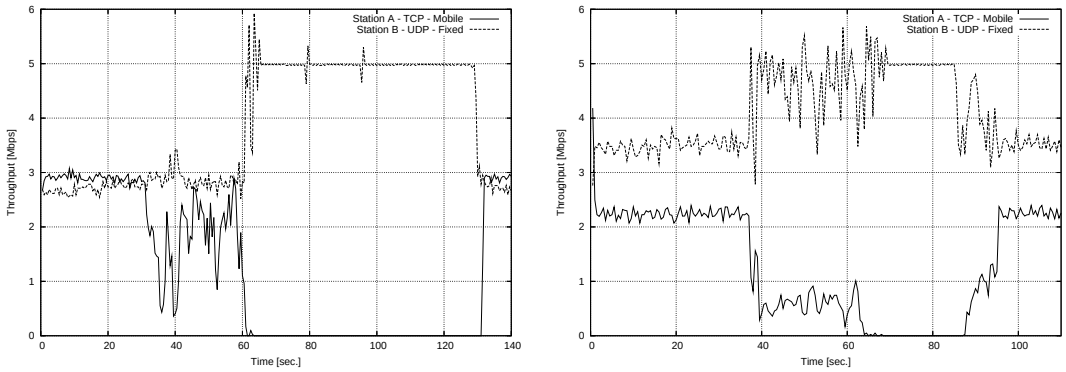
Up to now, all the measurements have been carried out in scenario where all the traffic is composed of downlink UDP flows. In order to extend the analysis to more realistic operation conditions, mixed UDP flows and TCP sessions should be considered. For the sake of simplicity, and to clearly highlight the effect of the DTT scheduler, we present two simple experiments in which one station is the destination of a downlink UDP flow with a data rate of 5 Mbps, while the other is the endpoint of a downlink TCP session. Incidentally, in this scenario we also have a "fourth station" accessing the medium. The station receiving TCP downlink traffic is requested to send (TCP) acknowledgement packets, thus subtracting bandwidth to all other flows. Consequently, UDP flows will not exactly reach one N-th of the saturation throughput as in the previous test case. However, given the limited size of TCP ACK packets, the degradation of performance is limited, strengthening the assumptions of the validity of a centralized scheduler, as will be clear analyzing measurement results.

In the first experiment the TCP station moves away from the AP. The first thing to note is that, when the PZC scheme is employed (see Fig. 11(b)), the TCP flow is always unfavored, and never gets close to the 3 Mbps data rate that represents its maximum theoretical value, as it happens when the DTT scheduler is running. Secondly, when the TCP station starts moving away, only the DTT is able to grant it some degree of fairness, as the PZC immediately shows better regard for the closer station. Also note that, due to the congestion control mechanism of TCP, some idle periods of transmission from the queue related to the TCP station are used by the other queue, causing the small spikes visible (at around 40 s) in Fig. 11(a). When the TCP station is disassociated, both DTT and PZC let the whole bandwidth be captured by the UDP flow. When the TCP station reassociates after having come closer to the AP, it can re-acquire its original share of air-time (and throughput).

Same measurements have been performed also using the commercial AP; however, the UDP flow quickly

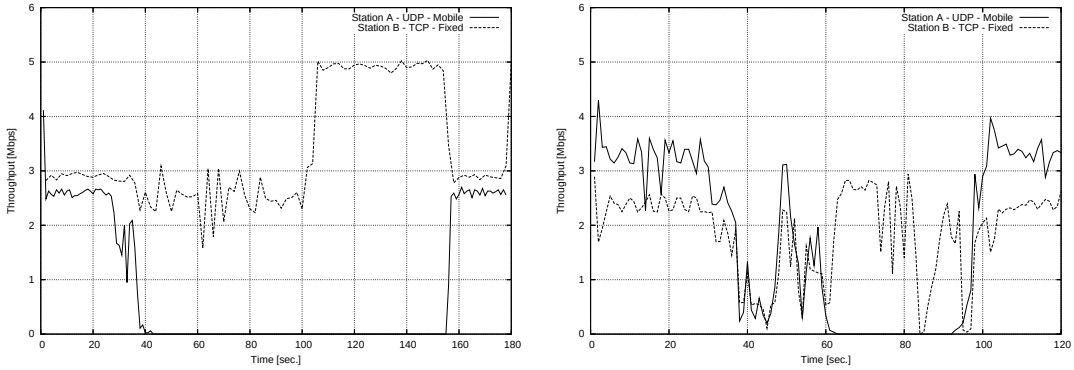
saturated the AP transmission queue, causing almost all TCP frames to be discarded. Consequently, TCP flow starved almost immediately, the connection dropped, and the UDP flow held at 5 Mbps. Therefore, the throughput plot is not as meaningful as those related to DTT and PZC and is not displayed.

In the second experiment, roles are inverted; the UDP sink station moves away. This is a further proof of the good flow isolation properties of DTT. When DTT is running (see Fig. 12(a)), the TCP session is almost completely unaffected by the movement of the other station, and continues transferring bits at a data rate very close to the limit. Conversely, the PZC (see Fig. 12(b)) shows once again that it is not able to refrain the two flows from influencing each other. As in the previous case, measurements involving commercial AP present the same behavior (TCP starvation) and are not reported.



(a) DTT Scheduler (b) PZC Scheme

Figure 11. Mixed UDP and TCP traffic - TCP station moving away



(a) DTT Scheduler (b) PZC Scheme

Figure 12. Mixed UDP and TCP traffic - UDP station moving away

7 Concluding remarks

Basic 802.11 systems suffer a severe performance impairment when even a single station in the BSS experiments bad channel conditions towards the AP. This phenomenon is due to the max-min throughput fairness of the 802.11 MAC algorithm, and to the simple FIFO scheduling policy usually implemented in the AP. In order to overcome this problem, we have proposed the DTT, a centralized scheduler to be integrated in the AP, able to provide a fair share of air-time usage to all the flows addressed to the associated stations. The air-time fairness avoids that a single station monopolizes the medium at the expenses of the others.

We have realized a working implementation of the DTT scheduler in a Linux-based AP, and performed some measurements to prove its effects in a number of operational conditions. Furthermore, the performance of FIFO scheduling policy and of a downlink traffic shaping scheme proposed in literature have been compared with those of DTT. Results clearly show that the DTT scheduler is able to provide the required flow isolation, in presence of both UDP and TCP traffic, hence representing a valid solution to 802.11 performance anomaly problem.

Nevertheless, several improvements may still be possible, and will be investigated in the future. First of all, it is necessary to further test the DTT operation, increasing the complexity of the testbed and adopting traffic originated by real applications (such as interactive voice calls and web browsing). Additionally, analysis of delays should be considered along with the throughput, to better characterize the quality of the transport service offered by DTT powered BSSs. As already mentioned, part of these studies has already been led in a simulative way [22].

Some other enhancements may regard the choice of the utility function, which may incorporate other parameters to account for traffic differentiation or pricing, thus producing different token distribution strategies.

Acknowledgements

The authors wish to thank Dr. Giuseppe Risi and Dr. Nicola Bonelli for their invaluable help in the implementation, testing and measurement phases of this work. This work has been partly supported by the Italian Ministry of Research and Instruction, under the research program TWELVE (ToWards Enhancing 802.11 support of differentiated service LeVEls).

References

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

[2] Y. Cao, V.O.K. Li, *Scheduling algorithms in broadband wireless networks*, Proceedings of the IEEE, Volume 89, Issue 1, Jan. 2001, pp. 76-87.

[3] J. Hartwell, A. Fapojuwo, *Modeling and Characterization of Frame Loss Process in IEEE 802.11 Wireless Local Area Networks*, Proc. of IEEE VTC Fall 2004, Los Angeles, CA USA, Sept. 2004

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

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

[6] B. Radunovic, J. Le Boudec, *Rate Performance Objectives of Multihop Wireless Networks*, IEEE Trans. on Mobile Computing, vol. 3, no. 4, pp. 334- 349, Oct.-Dec. 2004

[7] R. G. Garroppo, S. Giordano, S. Lucetti, E. Valori, *The Wireless Hierarchical Token Bucket: a Channel Aware Scheduler for 802.11 Networks*, Proc. WoWMoM2005, June 2005, Giardini Naxos (ME), Italy

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

[9] P. Bhagwat, A. Krishna, and S. Tripathi, *Enhancing throughput over wireless LANs using channel state dependent packet scheduling*, Proceedings of InfoCom '96, Mar. 1996, pp. 1133-1140.

[10] P. Ramanathan and P. Agrawal, *Adapting packet fair queueing algorithms to wireless networks*, in ACM MOBICOM '98, Dallas, pp. 1-9.

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

[12] Y. Choi, S. Park, S. Choi, G. Lee, J. Lee, H. Jung, *Enhancement of a WLAN-Based Internet Service*, in Mobile Networks and Applications (MONET), Vol. 10, N. 3, pp. 303-314, January 2005

[13] L. Wischhof, J. W. Lockwood, *Packet Scheduling for Link-Sharing and Quality of Service Support in Wireless Local Area Networks*, Technical Report WUCS-01-35, Applied Research Laboratory, Washington University in St. Louis, November 2001

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

[15] 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

[16] R. G. Garroppo, S. Giordano, S. Lucetti and E. Valori, *TWHTB: A Transmission Time Based Channel-Aware Scheduler for 802.11 systems*, Proc. of 1st Workshop on Resource Allocation in Wireless Networks (RAWNET 2005), Riva del Garda (TN), Italy, April 2005

[17] G. Tan, J. Guttag, *Time-based Fairness Improves Performance in Multi-Rate WLANs*, Proc. of USENIX 2004 Annual Technical Conference, July 2004, Boston (USA)

[18] F. P. Kelly *Charging and rate control for elastic traffic* European Transaction on Telecommunications, Volume 8, pp.33-39, 1997

[19] J. Mo and J. Walrand, *Fair End-to-End Window-Based Congestion Control*, IEEE/ACM Trans. Networking, vol. 8, no. 5, pp. 556- 567, Oct. 2000

[20] Li Bin Jiang, S. C. Liew, *Proportional Fairness in Wireless LANs and Ad Hoc Networks*, Proc. of IEEE Wireless Communications and Network Conference (WCNC), Mar. 2005

[21] B. Adamson, H. Greenwald, *MGEN User's and Reference Guide*, 2004, http://mgen.pf.itd.nrl.navy.mil/mgen.html

[22] R. G. Garroppo, S. Giordano, S. Lucetti and L. Tavanti, *A measurement-based channel aware scheduler to lessen VoIP capacity degradation in 802.11 networks*, submitted to the IEEE International Conference on Communications (ICC 2006), http://netgroup-serv.iet.unipi.it/reports/dtt-voip-icc06.pdf