Design and performance analysis of the Real-Time HCCA scheduler for IEEE 802.11e WLANs

Claudio Cicconetti *, Luciano Lenzini, Enzo Mingozi, Giovanni Stea

Dipartimento di Ingegneria dell’Informazione, University of Pisa, Pisa, Italy

Available online 6 February 2007

Abstract

This paper presents a new scheduling algorithm, called Real-Time HCCA (RTH), devised to support Quality of Service (QoS) at the flow level in an IEEE 802.11e network using the Hybrid Coordinator Function (HCF) Controlled Channel Access (HCCA) function. RTH separates online activities which take place at the frame transmission timescale, from offline activities which take place at the flow lifetime timescale. Complex computations are relegated to offline activities, while online tasks are kept as simple as possible. More specifically, at admission control time, RTH computes a periodic schedule based on the well-known Earliest Deadline First algorithm for 802.11e Traffic Streams (TSs). In doing so, the Stack Resource Policy is applied to account for non-pre-emptability of frame transmissions. Furthermore, the parameters are configured so as to reduce the MAC overhead due to polling uplink TSs. On the other hand, online scheduling is enforced simply by reading the pre-computed schedule, at little or no computational cost. RTH performance is assessed in terms of the admission control limit and of the amount of channel capacity that is left for contention-based access. Under both criteria, RTH is shown to outperform the sample scheduler proposed in IEEE 802.11e.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Wireless LANs; Quality of service; Real-time scheduling; Admission control; 802.11e; HCCA

1. Introduction

The IEEE 802.11 [9] has established itself as the worldwide standard in terms of indoor and outdoor wireless LANs (WLANs). The growing interest by customers in applications requiring Quality of Service guarantees, such as Voice over IP (VoIP), has led to the standardization of the IEEE 802.11e standard [8]. This has in fact enhanced the IEEE 802.11 standard by adding QoS support. More specifically, IEEE 802.11e specifies the Hybrid Coordinator Function (HCF) Controlled Channel Access (HCCA) function, which requires centralized scheduling and allows applications to negotiate parameterized service guarantees. IEEE 802.11e also specifies the Enhanced Distributed Coordination Access (EDCA) function, which achieves statistical service differentiation in a distributed manner. HCCA is specifically devised to transmit traffic with real-time constraints. Scheduling in 802.11e HCCA is controlled by the Hybrid Coordinator (HC), which transmits downlink data frames and schedules
uplink transmissions by polling the associated stations. While the standard does not specify that a mandatory scheduling algorithm must be implemented at the HC, the latter is of obvious importance in terms of QoS provisioning capabilities, as discussed in [6].

In this paper we address the problem of real-time scheduling in HCCA. More specifically, our aim is to devise a scheduling algorithm that provides traffic flows with a fixed amount of granted capacity over a fixed period. This kind of service is well suited, for instance, to Voice over IP (VoIP) traffic, which generates constant size packets at fixed intervals. Such an algorithm should be flexible, i.e., able to cope with flows with different requirements (e.g., voice coding schemes with different packet generation rates). It should also be effective, so as to allow for a high utilization of the medium, and simple, so that scheduling decisions do not require an unfeasible amount of computations at the HC. To the best of our knowledge, there are only two HCCA algorithms in the literature that are designed to provide traffic flows with the aforementioned real-time guarantees: the sample scheduler, which has been included in the 802.11e document for information purposes, and Scheduling based on Estimated Transmission Time (SETT-EDD) [7]. However, neither fulfills all the above-mentioned requirements. In fact, the sample scheduler tends to perform poorly when scheduling flows with different service requirements. On the other hand, SETT-EDD may require as many as $O(n)$ operations per scheduling decision, $n$ being the number of admitted flows. Furthermore, no schedulability test has been provided, nor can it be straightforwardly derived. Thus it is not possible to decide a priori whether a flow can be admitted or not.

In this paper, we overcome the limitations of the above-mentioned algorithms by proposing Real-Time HCCA (RTH) as a new scheduling algorithm for real-time traffic in 802.11e HCCA. The basic idea behind RTH (see [3]) is to split the scheduling task into online and offline activities. The former occur at the MAC frame transmission timescale, and are kept as simple as possible. The latter occur at the flow lifetime timescale, which is expected to be several orders of magnitude larger than the MAC frame transmission’s timescale. At the latter timescale it is feasible to trade simplicity for effectiveness, i.e., to perform more complex computations in order to optimize the performance. In RTH, offline activities include: (i) checking the admission of a new flow, i.e., testing whether it is possible to provide the new flow with the required QoS guarantees without violating those of the already admitted flows, and (ii) computing a timetable, i.e., a periodic schedule of transmission opportunities for the admitted flows. The only online activity consists of looking up the pre-computed timetable, and granting transmission opportunities to the various stations accordingly. Thus, online scheduling requires $O(1)$ operations per scheduling decision with respect to the number of admitted flows. The RTH timetable is computed according to the Earliest Deadline First (EDF) scheduling algorithm [11,16]. The latter is a well-known real-time scheduling algorithm, whose main idea is to schedule the task with the earliest deadline. In other words, it assumes a fluid system model, where tasks can be scheduled arbitrarily small amounts of capacity. Such a model does not hold for 802.11e, where packets of finite duration require atomic transmission. For this reason, we also borrow the concept of blocking from the Stack Resource Policy (SRP) [1,16]: a higher priority task is not allowed to preempt the ongoing task whenever the latter is in a critical section. IEEE 802.11e frame exchange sequences are in fact regarded as critical sections. Moreover, we devise a simple, yet effective procedure to minimize the overhead of polling uplink flows within the above framework.

The paper is organized as follows. In Section 2 we describe the features of the IEEE 802.11e MAC protocol to support QoS, and review related research. The RTH scheduling algorithm is described in Section 3, and analyzed in Section 4. Finally, conclusions are drawn in Section 5.

2. IEEE 802.11e and QoS support

In this section, we set out the necessary background to the paper. First, we describe the enhancements of 802.11e that are related to the support of QoS using HCCA, as well as introducing some notations that will be used throughout this paper. We then review the related work on real-time scheduling in HCCA.

2.1. IEEE 802.11e MAC protocol

As already mentioned, HCCA is a centralized access mechanism controlled by the HC, which resides in the QoS-Access Point (QAP). Each QoS-enabled 802.11e station (QSTA) may establish up
to eight HCCA Traffic Streams (TS). TSs are guaranteed a parameterized QoS access to the medium, and they can be either uni-directional (i.e., uplink or downlink), or bi-directional. In all cases, TSs are initiated and terminated only by QSTAs. However, the QAP performs an admission control procedure to ensure that the admission of a new TS does not violate the service guarantees of the already admitted TSs.

The QAP can take control of the medium whenever it needs to allocate transmission opportunities (TXOPs) to TSs. It does so by accessing the medium with a higher priority than QSTAs. Note however that the QAP is not allowed to interrupt ongoing transmissions. During TXOPs, the medium is accessed by only one QSTA. More specifically, a downlink TXOP consists of a burst of QoS Data (data, for short) frames transmitted from the QAP to a QSTA. On the other hand, an uplink TXOP is initiated when the QAP polls a QSTA, which takes control of the medium for a time interval smaller than or equal to the TXOP duration limit specified in the QoS CF-Poll (poll, for short) message. If the TS of a polled QSTA does not have data to transmit, or if the head-of-line packet does not fit into the remaining TXOP duration, the QSTA sends a QoS CF-Null (null, for short) frame to the QAP. If the polled QSTA does not use up the TXOP entirely, then the QAP has two options: (i) take back control of the medium, by immediately starting a new TXOP, or (ii) resume the contention-based access to the medium. A sample HCCA frame exchange sequence is illustrated in Fig. 1.

The time spent in polling QSTAs can be accounted for as a MAC overhead, which is added to the overhead due to direct acknowledgment of data frames. However, such overhead can be reduced by combining multiple control messages and/or piggybacking them on data frames. If the optional piggybacking feature is supported, the QAP may piggyback an acknowledgment and/or a poll on the ongoing data frame, as long as the control messages and the data frame are addressed to the same recipient. An example of how piggybacking reduces the polling overhead is shown in Fig. 1, where the second poll message from the QAP is piggybacked on a data frame. Furthermore, the optional QAck feature allows the QAP to combine control and data messages addressed to different QSTAs. More specifically, the QAP can piggyback a poll message directed to a QAck-enabled QSTA on a data frame addressed to another QSTA, or on a message that acknowledges the correct reception of a data frame from another QSTA. Additionally, the QAP can jointly transmit a data frame to a QSTA and acknowledge the correct reception of a data frame from another QAck-enabled QSTA.

A QSTA requests the admission of a new TS $i$ by sending an Add Traffic Stream (ADDTS) message to the QAP, which contains the Traffic Specification (TSPEC). In this study, we only consider the subset of five mandatory TSPEC fields, which are, for TS $i$: mean data rate ($R_i$), nominal Service Data Unit (SDU) size ($N_i$), minimum PHY rate ($C_i$), delay bound ($D_i$) and maximum service interval (MSI$_i$). The first four parameters characterize TS $i$. Specifically, the delay bound field specifies the maximum lifetime of an SDU at the QSTA: if an SDU experiences a delay greater than $D_i$, it is dropped by the QSTA. The parameter MSI$_i$ is a service requirement, and corresponds to the maximum time that can elapse between two subsequent polls at the same QSTA. The standard allows us to specify one parameter between $D_i$ and MSI$_i$, or both. If the QAP admits the TS, it responds to the ADDTS with a positive acknowledgment message. The latter includes the service start time, which specifies the

![Fig. 1. Sample HCCA frame exchange sequence.](image-url)
starting time from when the QSTA is allowed to send frames that belong to the admitted TS.

According to [8], the QAP is responsible for scheduling TXOPs so that the negotiated TSPEC parameters of admitted TSs are satisfied. More specifically, for any time interval \([t_1, t_2]\) greater than a minimum specification interval, which is selected by the QAP, the HCCA function is committed to scheduling a number of TXOPs to TS \(i\), whose cumulative duration \(W_i(t_1, t_2)\) is such that

\[
W_i(t_1, t_2) \geq \left\lfloor \frac{R_i}{N_i} (t_2 - t_1 - \Delta_i) \right\rfloor \cdot t_{Ni}, \tag{1}
\]

where \(\Delta_i\) is equal to MSI\(_i\), if MSI\(_i\) is specified, otherwise it is equal to \(D_i\), and \(t_{Ni}\) is the nominal transmission time for TS \(i\). In fact, \(t_{Ni}\) is the time to transmit an SDU that belongs to TS \(i\), including the time needed to receive the corresponding acknowledgment and the interframe spaces (see Fig. 1)

\[
t_{Ni} = \text{SIFS} + \left( t_{\text{PHY}} + \frac{h_{\text{MAC}} + N_i}{\Gamma_{\text{DATA}}} \right) + \text{SIFS} + \left( t_{\text{PHY}} + \frac{h_{\text{ACK}}}{\Gamma_{\text{ACK}}} \right),
\]

where \(t_{\text{PHY}}\) is the time duration of the PHY header, \(h_{\text{MAC}}\) (\(h_{\text{ACK}}\)) is the size of the MAC header (acknowledgment frame), and \(\Gamma_{\text{DATA}}\) (\(\Gamma_{\text{ACK}}\)) is the physical transmission rate of data (acknowledgment) frames. The notations are summarized in Table 1.

### 2.2. Related work

Current research challenges of 802.11e have recently been reviewed in [13]. The authors state that most existing work on HCCA is aimed at devising strategies to improve the performance of Variable Bit-Rate (VBR) traffic. In fact, the standard QAP commitment reported in Eq. (1) is based on the mean data rate. Thus, enforcing such a guarantee in the presence of VBR traffic would require the amount of capacity reserved at each period to be over-provisioned so as to absorb traffic peaks with small delays. However, the focus of this paper is to design an efficient HCCA scheduler which provides TSs with a guarantee that complies with the standard requirement. To the best of our knowledge, only two algorithms for this purpose have been proposed in the literature: the sample scheduler, which has been included in the 802.11e document for information purposes, and Scheduling based on Estimated Transmission Time (SETT-EDD) [7].

The sample scheduler serves each TS on a periodic basis, where the period (called Service Interval, SI) is the same for all TSs and is set to the smallest \(\Delta_i\). The TXOP duration of TS \(i\), namely \(\bar{C}_i\), is then set to the smallest value that satisfies Eq. (1)

\[
\bar{C}_i = \left\lfloor \frac{R_i}{N_i} \cdot \text{SI} \right\rfloor \cdot t_{Ni}, \tag{2}
\]

\(\bar{C}_i\) is rounded up to contain an integer number of SDUs of nominal size, which may produce a surplus bandwidth allocation to that particular TS. Note that the TXOP computation according to the standard is slightly different from the one in Eq. (2). Specifically, in Eq. (2) we do not take into account the maximum SDU size that can be specified in the TSs TSPEC. This is because in this study we only consider those QoS guarantees that provide each TS with a fixed amount of capacity granted on a periodical basis.

The schedulability test is as follows: a given set of TSs is schedulable with the sample scheduler if and only if

\[
\sum_i (\bar{C}_i + t_p) \leq 1,
\]

where \(t_p\) is the time to transmit a poll message, including the interframe space (see Fig. 1)

\[
t_p = \text{PIFS} + \left( t_{\text{PHY}} + \frac{h_{\text{POLL}}}{\Gamma_{\text{POLL}}} \right),
\]

where \(h_{\text{POLL}}\) is the size of poll frames and \(\Gamma_{\text{POLL}}\) is the physical transmission rate of poll frames. The polling overhead of TS \(i\), \(t_{pi}\), is equal to \(t_p\) if \(i\) is an uplink TS, otherwise it is zero.
The admission control algorithm and the method to derive the timetable are extremely simple, yet effective when all TSs have the same $\Delta$. However, if TSs with different $\Delta$ are admitted, they are then all served with the same period as the TS with the smallest $\Delta$, i.e., SI. Therefore, all TSs with $\Delta > SI$ are polled more often than needed, which unnecessarily increases the polling overhead, thus wasting bandwidth.

SETT-EDD is based on an online version of the Earliest Due Date (EDD), which is a well-known algorithm in the real-time literature [16]. Specifically, SETT-EDD requires that TSs additionally specify the minimum service interval (mSI), i.e., the minimum time that must elapse between two consecutive TXOPs, and the maximum burst size (MBS). The latter are used to set up virtual token buckets, which are used to check the eligibility of TXOPs. Furthermore, TXOPs are assigned a release time equal to mSI and a deadline equal to MSI. EDD is then used to select the next eligible TXOP to be scheduled. Therefore each scheduling decision entails (i) updating the set of eligible TXOPs according to the virtual token buckets and (ii) selecting the one with the nearest deadline according to EDD. Unless complex data structures are employed, a scheduling decision may then require as many as $O(n)$ operations, $n$ being the number of admitted TSs. To make matters worse, since EDD relies on preemption, which is not possible with packetized traffic, the theoretical results related to EDD cannot be straightforwardly, if at all, applied to devise an admission control test for SETT-EDD. As a matter of fact, no admission control test has been derived for SETT-EDD, which means that it is not possible to guarantee a priori that deadlines are met.

In the following we present RTH, which unlike SETT-EDD provides explicit a priori deadline guarantees. Furthermore, we show that – although it has the same online complexity as the sample scheduler – it is more effective than the latter, in that it better accounts for TSs with heterogeneous requirements.

3. Real-Time HCCA (RTH) scheduler

RTH is made up of three blocks, as shown in Fig. 2: an admission control algorithm, a timetable computation algorithm and an enforcement procedure. The admission control and the timetable computation are offline activities, performed within the MAC sublayer Management Entity (MLME) of the QAP. The MAC sublayer of the QAP only runs the enforcement procedure, which is the sole online activity. As already outlined, the enforcement procedure is used by the QAP to grant TXOPs to the admitted TSs, based on a periodical timetable prepared by the timetable computation. The timetable computation is only run when the admission control admits a new TS. The timetable computation is based on theoretical results from the real-time scheduling literature, with regard to the EDF scheduling algorithm [11] with the SRP policy [1]. The admission control procedure also produces sets of
parameters describing each admitted TS, which are required by the timetable computation procedure. Below we describe the three above-mentioned procedures and their interaction.

3.1. Enforcement procedure

A timetable is a list of entries \( <i, TXOP_i, t_i> \), each of which states that TS \( i \) can access the medium within \( [t_i, t_i + TXOP_i] \). Let us assume that the latter is made available at the MAC sublayer of the QAP. One step of the enforcement procedure, executed whenever a TXOP ends, consists of reading the next entry in the timetable, \( <i, TXOP_i, t_i> \), waiting until time \( t_i \) (possibly allowing stations to access the medium using contention in between), and then granting a TXOP of duration \( TXOP_i \) to TS \( i \). The block diagram of the enforcement procedure is reported in Fig. 2, along with a sample timetable and its associated schedule. Note that the enforcement procedure has \( O(1) \) computational complexity with respect to the number of admitted TSs. As mentioned above, the timetable is built offline by computing an EDF schedule based on the requirements of the admitted TSs.

3.2. Admission control

The admission control procedure is run by the QAP whenever a QSTA requests the admission of a new TS. The set of TSPEC parameters advertised by the QSTA is \( \{R_i, N_i, D_i, T_i\} \), which are the mandatory traffic specifications and requirements of IEEE 802.11e HCCA. Based on the admission control algorithm described in detail below, the QAP decides whether to accept or reject the TS, and immediately notifies the QSTA. If the TS is accepted, the QAP computes a new timetable, to be used by the enforcement procedure, as described in Section 3.3. Otherwise, the TS cannot be served using HCCA.

In RTH, a TS \( i \) is characterized by a capacity \( C_i \) and a period \( T_i \), both expressed in time units, which are computed from the TSPEC parameters as follows. Firstly, the capacity \( C_i \) is computed as \( C_i = \lfloor(R_i \cdot T_i)/N_i \rfloor \cdot t_{N_i} \), which is the minimum needed to comply with the TS requirements in Eq. (1). The ratio \( N_i/R_i \) is the expected average interarrival time of SDUs of TS \( i \). Then, if the delay bound is smaller than the expected average interarrival time, \( T_i \) is set to the delay bound itself. Otherwise, \( T_i \) is set to the largest multiple of \( N_i/R_i \) that is not greater than the delay bound, i.e., \( \lfloor(R_i/N_i) \cdot D_i \rfloor N_i/R_i \), since the constraint \( T_i \leq D_i \) must be adhered to.

Using the above TS characterization, and with the idea of employing EDF as a scheduling algorithm, we can regard TSs as tasks in a multi-programmed environment, and capitalize on the schedulability tests already devised for such a context. Let us assume, without loss of generality, that TSs are sorted by increasing period duration, i.e., \( i > j \Rightarrow T_i > T_j \). Essentially, \( C_i \) is the cumulative duration of the TXOPs scheduled by the QAP each \( T_i \) instant. In the case of uplink TSs, \( C_i \) has to take into account the overhead due to polling, which also depends on the availability of the QAck optional feature. Although the capacity \( C_i \) does not need to be allocated in a single TXOP, the minimum TXOP for TS \( i \) should not be smaller than the time needed to transmit a nominal size SDU, including a poll for uplink TSs. Should this constraint be violated, nominal size SDUs would not fit into scheduled TXOPs, and these TXOPs would thus be wasted. Therefore the minimum critical section \( b_i \) of TS \( i \) is its nominal size SDU transmission, i.e., \( b_i = t_{N_i} + t_{p} \). When TS \( i \) is in a critical section, and is thus scheduled instead of the highest priority TS \( j \), TS \( i \) is said to block TS \( j \). Let \( B_i \) denote the maximum critical section durations of TSs with a period longer than TS \( i \), i.e., \( B_i = \max_{j \neq i} \{b_j\} \). A sufficient condition for a set of \( n \) TSs to be schedulable with EDF under the above constraint is derived from SRP as follows:

\[
B_i + \sum_{j=1}^{\infty} \frac{C_j + \pi_j \cdot t_p}{T_j} \leq 1 \quad \forall i : 1 \leq i \leq n, \quad (3)
\]

where \( \pi_i \) is the maximum number of times that the QAP has to poll TS \( i \), during \( T_i \), with a dedicated poll message. If QAck is enabled on all stations, \( \pi_i = 1 \), because if TS \( j \) preempts TS \( i \) at a TXOP boundary during its service, the QAP can piggyback the poll message to TS \( j \) on the last acknowledgment message to TS \( i \). If QAck is disabled, then \( \pi_j = \lfloor(R_j \cdot T_j)/N_j \rfloor \), which is the number of nominal size SDUs served each period, i.e., the maximum number of pieces in which the capacity \( C_i \) can be split due to preemption without violating critical section durations. Note that inequalities in (3) assume that TS \( i \) is blocked for the maximum time during its execution, which might not actually happen. Thus, condition (3) is sufficient but not necessary.
Clearly, $B_i$ is a non-decreasing function of the critical section duration $b_j (j > i)$. However, as long as $B_i$ is such that (3) holds, increasing $b_j$ does not actually hamper the schedulability of a set of TSs. On the other hand, having longer critical sections yields lower MAC overhead, since the number of preemptions experienced by a TS $i$ might actually decrease. We thus compute for each TS $i$ an extended critical section duration, namely $\bar{b}_i$, which is the maximum value such that Eq. (3) holds. The latter will be used instead of $b_i$ to compute the timetable under EDF/SRP. For this purpose, we first define the extended maximum blocking times, which are denoted as $\bar{B}_i$ and computed as follows:\footnote{The maximum blocking time of the lowest priority TS is not defined, as it is never blocked by other TSs.}

$$\bar{B}_i = \left(1 - \sum_{j<i} \frac{C_j + \pi_j \cdot t_{p_j}}{T_j}\right) T_i \quad \forall i : 1 \leq i < n. \tag{4}$$

Note that, by definition of $\bar{B}_i$, all inequalities in (3) hold if we substitute $\bar{B}_i$ into $B_i$. Then, we set the critical section durations of all TSs as follows:\footnote{The critical section duration of the highest priority TS is not defined, as it never blocks any TS.}

$$\bar{b}_i = \min_{j<i} \bar{B}_j \quad \forall i : 1 < i \leq n. \tag{5}$$

We now have to prove that the critical section durations $\bar{b}_i$, computed according to (5), are the largest ones that satisfy (3). We can easily prove this proposition by contradiction. Assume that there exists a set of critical sections $b_j$ also satisfying (3), and an index $k$ such that $\bar{b}_k < b_k$. It immediately follows from the definition of $\bar{B}_i$ that $\bar{b}_i \leq \min_{j<k} \bar{B}_j$, which can be substituted into $\bar{b}_k < b_k$ so as to obtain

$$\min_{j<k} \bar{B}_j < b_k \leq \min_{j<k} \bar{B}_j. \tag{6}$$

However, we can rewrite each inequality in (3) as follows:

$$\frac{B_j}{T_j} + \sum_{h<j} \frac{C_h + \pi_h t_{p_h}}{T_h} \leq 1 \Rightarrow B_j \leq \left(1 - \sum_{h<j} \frac{C_h + \pi_h t_{p_h}}{T_h}\right) T_j.$$ 

The latter can be substituted into (6), in addition to (4), so as to obtain

$$\min_{j<k} \left(1 - \sum_{h<j} \frac{C_h + \pi_h t_{p_h}}{T_h}\right) T_j < \min_{j<k} \left(1 - \sum_{h<j} \frac{C_h + \pi_h t_{p_h}}{T_h}\right) T_j,$$

which is obviously false. This concludes our proof.

In summary, the admission control procedure consists of: (i) checking the schedulability, assuming that each TS has critical sections whose duration is $t_{N_i} + t_{p_i}$, i.e., the transmission of an SDU with nominal size at the minimum PHY rate (normative requirement); (ii) computing up to what extent those critical sections can be extended while still ensuring schedulability, by applying (5), and (iii) using the extended critical sections to build the timetable offline, taking advantage of enlarged TXOP durations to reduce the polling overhead.

### 3.3. Timetable computation

The timetable computation produces an EDF schedule of the admitted TSs, whose parameters $T_i$, $C_i$ and $\bar{b}_i$ are computed as described in the previous section. The schedule duration $H$ is equal to the Least Common Multiple (LCM) of the periods of the admitted TSs. The following working variables are used: $t$ is the virtual clock, i.e., the time at which the procedure schedules either a TXOP to the TS with the earliest deadline, or some capacity to contention-based access; the flag $q$, which is meaningful only if QAck is enabled, specifies whether a poll message can be piggybacked on the next TXOP or not; $C_i$ is the capacity that has to be granted to TS $i$ before its deadline expires; $D_i$ is the deadline of TS $i$; and finally, $R_i$ is the release time of TS $i$, i.e., the start time of the next period at which $C_i$ is replenished. The add() function inserts a new entry in the timetable that will be used by the enforcement procedure.

The pseudo-code for the timetable computation is reported in Fig. 3. The computation of $T_i$, $C_i$ and $\bar{b}_i$ is not shown since it only involves elementary operations, which have been described in the previous section. After the initialization of the working variables (lines 1–7), the main body of the timetable computation is started. The latter is a loop that lasts until the virtual clock $t$ reaches the schedule duration $H$ (line 8). Firstly, the procedure checks whether there is at least one TS $i$ with unfulfilled capacity (lines 9–10). If this is not the case, the
virtual clock is put forward until the earliest release
time, i.e., the expected arrival time of the next SDU,
and the loop restarts immediately (lines 10–13). In
this case, \( q \) is also set to false (line 12). This is
because it is not possible to piggyback a poll on
the next TXOP, since there is a gap between two
consecutive TXOPs, which is left for contention-
based access functions. If there are TSs with
unfulfilled capacity, then the TS with the earliest
deadline, say TS \( i \), is scheduled a TXOP. The TXOP
duration depends on the release time of TSs with a
higher priority, i.e., with an earlier deadline than TS
\( i \). If TS \( i \) has the highest priority at time \( t \), then
the whole of its unfulfilled capacity \( C_i' \) is scheduled
(lines 15–24). The time to poll TS \( i \) is added if either
QAck is disabled or if it is impossible to piggyback
the poll to TS \( i \) on the last scheduled TXOP, i.e., \( q \) is
false (lines 16–17). Then, the main loop restarts
immediately (line 24). If a TS exists with an unful-
filled capacity and a higher priority than TS \( i \), say
TS \( k \), then TS \( i \) can be preempted (on an SDU
boundary) by TS \( k \) depending on the temporal dis-
tance between \( R_i \) and \( R_k \) (lines 25–30). Before
updating the working variables of TS \( i \) (lines 34–
38), the procedure restores the unfulfilled capacities,
if any, of the TSs whose release times expire before
the end of the scheduled TXOP (lines 32–33).

Fig. 4 shows an example of timetable computa-
tion with three TSs, whose parameters are reported
on the right. The output with the sample scheduler
is also reported as a reference. Note that \( b_i \) is not
used by the timetable computation, and is reported
for explanatory purposes only. As can be seen, the
timetable is computed over a period \( H = 30 \) time
units, which is the LCM of 5, 10, and 15. According
to EDF, at time \( t \) the TS with the highest priority
(i.e., earliest deadline), among those with some
residual capacity, is scheduled a TXOP. However,
at time \( t = 5 \), TS 1, which has the highest priority,
cannot preempt TS 3, since the latter is in a critical
section, i.e., it has been scheduled part of a mini-
mum size TXOP starting at \( t = 4 \). If \( b_i \) were used
for the timetable computation, then TS 1 would
be scheduled a TXOP at \( t = 6 \). Using the extended
critical section duration \( h_i \), TS 3 is allowed to block
TS 1 until \( t = 8 \), while still guaranteeing that no
deadline will be missed. In this example, using \( h_i \)
instead of \( b_i \) thus saves the overhead of an addi-
tional poll to TS 3.

Each iteration of the main loop in Fig. 3 may
require as many as \( O(n) \) operations per TXOP,
where \( n \) is the number of admitted TSs, because of
the searches in lines 9, 14, and 32. The number of
iterations is equal to the overall number of sched-
uled TXOPs, which in turn depends on the total
number of preemptions, plus the number of idle
HCCA intervals, i.e., those intervals while the
QAP refrains from scheduling TXOPs, which are
left for contention-based access functions. In gen-
eral, the number of preemptions with EDF is not
known \emph{a priori}, and deriving a relatively tight upper
bound requires complex computations [15]. However, with RTH, the number of iterations of the main loop in Fig. 3 is upper bounded by
\[
\sum_i \left( \left\lfloor \frac{C_i}{b_i} \right\rfloor + 1 \right) \left( \frac{H}{T_i} \right),
\]
(7)
since each TS \(i\) can be preempted at most \(\left\lfloor \frac{C_i}{b_i} \right\rfloor\) times in one period. This is because of its extended critical section duration \(b_i\), and because there can be at most one idle HCCA interval for each period of any TS \(i\). Therefore, the timetable computation may be computationally expensive. However, as already observed, such an overhead is not an issue, since the timetable is computed on a timescale of a higher order of magnitude than that of the enforcement procedure. Moreover, when a QSTA requests the admission of a new TS, the QAP can provide it with a quick yes/no response based on the admission control algorithm. The latter requires \(O(n)\) simple computations, \(n\) being the number of TSs involved. Furthermore, according to the standard, the actual start of the service for the newly admitted TS can be deferred to a later time (which is conveyed in the ADDTS response message). This allows the scheduler to take the time needed to produce the new timetable.

The QAP does not need to compute a new timetable when a QSTA deletes one of its admitted TSs. In fact, it can be easily proved that any subset of a set of admitted TSs is also feasible. Therefore, the QAP can simply remove the entries related to the deleted TS from the current timetable.

We conclude the section by discussing the storage complexity of RTH. The timetable consists of a list of entries \(\langle i, TXOP_i, t_i \rangle\), whose number is upper bounded by \(\sum_i \left\lfloor \frac{C_i}{b_i} \right\rfloor \left( \frac{H}{T_i} \right)\). The latter is obtained by subtracting the upper bound from the number of idle HCCA intervals from expression (7). Let us assume, as a rough computation, that each entry can be represented as follows: TS \(i\) as an 8-bit identifier, which allows up to 256 TSs to be served by the QAP at the same time; the TXOP as an 8-bit field, expressed in multiples of 32 \(\mu s\), as specified by the standard; the start time of the TXOP \(t_i\) as a 32-bit field, in \(\mu s\), which allows the schedule duration to be more than 71 min. In this way, one entry requires merely 6 bytes. Given that APs are currently shipped with the main memories in the order of megabytes, the storage complexity of RTH is not an issue.

4. Performance evaluation

In this section we evaluate the performance of RTH, in terms of the admission control limit of HCCA traffic and of the channel capacity left for traffic served via contention-based access functions, i.e., DCF and EDCA. First, we compare the admission control limit of the RTH and sample schedulers, with different mixes of QoS TSs. This analysis is carried out using an ad-hoc computational tool. Then, we employ packet-level simulation to assess the performance of contention-based traffic under different patterns of VoIP models. For both RTH and the sample scheduler we report the results with and without the QAck feature.

4.1. System parameters and analysis tools

The physical layer parameters are those specified by the High Rate Direct Sequence Spread Spectrum (HR/DSSS) PHY [9], also known as 802.11b, and are reported in Table 2.

The results reported in Section 4.3 were obtained with a stand-alone C++ program, which emulates the behavior of the offline timetable computation
algorithm and the HCCA admission control unit of 802.11e, with both RTH and the sample scheduler. Those in Section 4.4 were obtained using the ns2 network simulator [12], where we implemented RTH and the sample scheduler using the HCCA implementation framework described in [4]. The simulation was carried out using independent replications [10]. Specifically, we ran a variable number of independent replications of 600 s each, with a 100 s initial warm-up period. In all the simulation runs, we estimated the 95% confidence interval for each performance measure. Confidence intervals were not drawn whenever negligible.

4.2. Traffic models and metrics

We considered two types of QoS traffic transmitted through HCCA: VoIP and videoconference. Both traffic types are bidirectional,\(^3\) thus QSTAs request the admission of an uplink and a downlink TS at the same time. Data traffic, which possesses no specific QoS requirements, was transmitted using DCF. Stations with data traffic operate in asymptotic conditions, i.e., they always have a frame to transmit. The packet length of data traffic is constant and equal to 1500 bytes.

We simulated a VoIP stream of packets as an ON/OFF source: during the ON (talkspurt) periods the traffic is CBR with parameters that depend on the encoding scheme; during the OFF (silence) periods no packets are generated. The encoding schemes that we employed are [5] G.711, which generates packets of 160 bytes (including IP/UDP/RTP headers) at a constant inter-arrival time of 20 ms, and G.723, which generates packets of 70 bytes (including IP/UDP/RTP headers) at a constant inter-arrival time of 45.5 ms. For both encoding schemes, we set the TSPEC delay bound to the packet inter-arrival time, the nominal SDU size to the packet size, and the mean data rate to the peak rate during talkspurts. Talkspurt and silence periods were distributed according to Weibull distributions, so as to model four different types of conversations, as reported in Table 3 [2]: conference (i.e., many-to-many, M2M), lecture audience (i.e., many-to-one, M2O), lecture speaker (i.e., one-to-many, O2M), and two-way conversation (i.e., one-to-one, O2O).

We simulated videoconference traffic based on a pre-encoded MPEG-4 trace file [14], with an average rate of 158 kb/s and a peak rate of 2.7 Mb/s. The delay bound of videoconference TSs was 100 ms, and the mean data rate the smallest rate such that the 95th percentile of the delay was smaller than the frame inter-arrival time. This was done assuming that the video stream was transmitted by an HCCA TS which is periodically granted a TXOP of fixed duration, with the period equal to the delay bound, i.e., 364 kb/s. Finally, the nominal SDU size was set equal to the MTU employed by the IPv4 network layer, i.e., 1500 bytes.

According to the set of TSPEC parameters and the traffic characterization, any VoIP SDU will experience a delay lower than or equal to the packet inter-arrival time, whereas a videoconference SDU will experience a delay lower than or equal to 100 ms with probability 0.95.

Several metrics were defined to assess the performance of RTH. With regard to the admission control analysis, we considered the number of TSs that can be admitted, and the amount of channel capacity that is not used by HCCA. The latter is defined as the percentage of idle time within a scheduling duration, and is derived from a schedulable set of TSs by applying the timetable computation algorithm in Fig. 3. The performance metrics of the packet-level simulation analysis are the null rate, defined as the number of null messages sent in the unit of time, and the throughput of DCF traffic, which is the amount of bits correctly acknowledged by the QAP in the unit of time.

\(^3\) The downlink and uplink traffic flows are not correlated.
4.3. Admission control analysis

We first evaluated the performance of RTH in terms of the admission control limit. Note that the admission control test for the sample scheduler does not take into account the presence of the QAck. However, the admission control for RTH changes if QAck is enabled, as shown in Fig. 3.

Fig. 5 shows the number of admitted VoIP G.723 TSs as a function of the number of admitted VoIP G.711 TSs. Fig. 6 shows the number of admitted videoconference TSs, as a function of the number of admitted VoIP G.711 TSs. In both cases, the sample scheduler curve lies significantly below the RTH curve. This behavior confirms that the sample scheduler cannot efficiently accommodate TSs with different TSPECs. In fact, firstly, it polls TSs with $D_i > SI$ more often than needed, by setting the scheduling duration to the smallest TS period. Secondly, it overestimates the capacity needed by TSs because of the ceiling operator in Eq. (2). With regards to mixed VoIP traffic only (i.e., Fig. 5), the RTH admission control limit does not change if QAck is enabled. This is because the expected number of SDUs generated for each period is equal to one, for both G.711 and G.723 TSs. Therefore, in Eq. (3) the value of $\pi_i$ is always one regardless of the QAck feature. This is not true with mixed VoIP and videoconference traffic, where the latter is expected to transmit many SDUs for each period. Hence they can be preempted by TSs that have a higher dynamic priority at a given virtual time $t$ in the timetable computation. However, as can be seen in Fig. 6, enabling QAck warrants a negligible improvement, i.e., at most one additional videoconference TS in only a few cases.

Even though the QAck has a negligible impact on the HCCA admission control limit, enabling it can produce a significant increase in the capacity available for contention-based traffic. In Figs. 7 and 8 we report an estimate of the lower bound of this capacity, by plotting the fraction of a schedule duration that is not allocated to HCCA transmissions. Note that the actual capacity available for contention-based traffic can only be evaluated through packet-level simulation, which we do in the next subsection. Our scenario has one VoIP G.711 TS and an increasing number of either VoIP G.723 or Videoconference TSs, with one G.711 TS.

---

4 As already said in Section 2.2, no admission control test has been derived for SETT-EDD. Therefore, it is not possible to include it in the comparison.
videoconference TSs, respectively. In both cases, the sample scheduler reserves much more capacity than RTH for HCCA transmissions. This is because VoIP G.723 TSs and videoconference TSs are unnecessarily polled with the same frequency as VoIP G.711 TSs, which have a shorter period (20 ms vs. 45.5 ms and 20 ms vs. 100 ms, respectively). Thus, the capacity available for contention-based access with RTH is significantly higher than that available with the sample scheduler. Moreover, enabling QAck further reduces the overhead of polling uplink TSs, thus saving more capacity for contention-based access. This is especially true with mixed VoIP traffic (Fig. 7), whereas the capacity improvement with mixed VoIP G.711 and videoconference traffic is quite modest (at most 4% of channel capacity). This is because VoIP TSs have stringent delay requirements, i.e., short periods, with low rate requirements, i.e., small capacities. Therefore, polling is a critical overhead factor, which cannot be captured by $\pi_i$ in Eq. (3), which assumes that each TS consumes at least one poll message during its period. Actually building the timetable allows TXOPs to be concatenated so that only few TSs really need to be polled individually. Such an improvement is not achieved with mixed VoIP and videoconference traffic (Fig. 8) because the total number of TSs, and hence the number of poll messages needed, is smaller than that needed with VoIP traffic.

### 4.4. Packet-level simulation analysis

In this section we discuss the results obtained through simulation. Firstly, we observe that RTH produces timetables that are much more “fragmented” than those obtained with the sample scheduler. This is because the sample scheduler selects a common schedule duration, and allocates all the TXOPs back to back starting from the beginning of the schedule. On the other hand, RTH serves TSs based on their actual periods, which may be different, and hence may generate schedules with many “gaps” of capacity left for contention-based access (see Fig. 4). Thus, contention-based access functions, which in fact can access the channel in those gaps only, may have to contend for medium access in shorter intervals with RTH than with the sample scheduler. One might wonder whether this negatively affects the throughput of contention-based traffic. We now show via simulation that this is not the case. We simulate a system where the QAP keeps the channel busy periodically for a fixed amount of time. The HCCA service is thus completely characterized by two parameters: the duration of the HCCA period, and the ratio of the busy HCCA interval to the HCCA period, called $\delta$. We vary the period while keeping $\delta$ constant, so as to vary the absolute duration of the gaps left for contention-based access functions. In Fig. 9 we show the throughput of DCF traffic against the HCCA period, which is varied from 5 ms to 1 s. Results are reported in the case of $\delta = 0.2$, 0.5, 0.8, and with a variable number of QSTAs (i.e., 1, 5, and 10 QSTAs). As can be seen, in all cases the curves are almost constant, i.e., the throughput of DCF traffic does not depend significantly on how large the contention intervals are. This is because, according to the 802.11e standard, the QAP does not notify QSTAs of the forthcoming TXOP start time. Therefore, even though idle HCCA intervals

![Fig. 8. Unreserved HCCA capacity vs. number of videoconference TSs, with one G.711 TS.](image)

![Fig. 9. Throughput of DCF traffic with increasing period duration of the HCCA schedule.](image)
are smaller than the DCF/EDCA frame exchange sequence, QSTAs still initiate transmission. In this case, the TXOP start will be delayed until the frame exchange sequence is completed. Note that the throughput of DCF traffic slightly decreases when the number of QSTAs increases, due to the increasing number of collisions.

To conclude our analysis, we compare RTH with the sample scheduler, in terms of the DCF throughput carried with different Voice Activity Detection (VAD) models for VoIP TSs. The performance of SETT-EDD depends on some TSPEC parameters, such as mSI and MBS. No criterion for selecting these parameters has been specified in [7], nor can it be straightforwardly inferred from the traffic model employed. In fact, SETT-EDD is best tailored to serve bursty traffic, while the simulated scenarios only involve CBR VoIP TSs. On the other hand, with CBR traffic, the sample scheduler is a stronger competitor, since it serves such traffic with a perfectly periodic schedule. The simulated scenario includes ten QSTAs, five with G.711 VoIP TS, the other five with G.723 VoIP TS; additionally, the QAP serves a QSTA with DCF traffic. All four VAD models were considered separately.

In Fig. 10 we show the null rate of VoIP TSs. As can be seen, regardless of the VAD model employed, the null rate is almost the same in both schedulers with G.711 TSs. This is because QSTAs advertise a smaller delay bound for G.711 TSs than for G.723 TSs (i.e., 20 ms vs. 45.5 ms). Therefore, in both RTH and the sample scheduler, the service period of G.711 TSs is equal to their packet inter-arrival time. In this case, null messages are only due to the silence periods. In fact, the lower the ratio of the average ON period over the average ON + OFF period in Table 3, the higher the chance that the QAP polls an idle TS, and hence the null rate. However, the null rate with RTH is significantly smaller than it is with the sample scheduler with G.723 TSs. This is because RTH polls them at the same pace of packet arrival, while the sample scheduler polls all TSs each 20 ms. Thus, besides the unavoidable contribution to the null rate due to the silence periods, the sample scheduler adds a significant overhead because it polls G.723 TSs at a higher rate than the packet generation.

In Fig. 11 we compare the actual throughput achieved by DCF traffic, with and without QAck, in the same settings as above. The throughput of DCF traffic with RTH is higher than with the sample scheduler, due to the reduced overhead when polling G.723 TSs. Clearly, the throughput increases when QAck is enabled in both schedulers. However, RTH alone outperforms the sample scheduler with QAck enabled. Furthermore, enabling the QAck entails a higher throughput increase with the RTH scheduler than with the sample scheduler.

5. Conclusions

In this paper we have presented RTH, a scheduling algorithm to provide real-time guarantees under 802.11e HCCA. RTH applies to those context rules that are derived from EDF scheduling with the SRP policy. Moreover, it defines a procedure that maps the TSPEC parameters of TSs to those that are required to carry out the scheduling procedure. This is done to reduce the number of polls to uplink TSs for each period, and hence the resource wastage.
RTH is simple, in that complex operations are performed offline with respect to packet transmission, while scheduling decisions require a constant number of operations compared to the number of admitted TSs.

We have evaluated the performance of RTH analytically, using a stand-alone implementation of the admission control algorithm and of the offline timetable computation procedure. We also assessed it through packet-level simulation, using ns2. In both cases, we compared the results to those obtained with the sample scheduler. Our performance evaluation showed that (at least within the limits of the analyzed scenarios) RTH is more efficient than the sample scheduler in terms of resource utilization. Not only can RTH admit more traffic, it also uses capacity sparingly, leaving more available for contention-based access functions. Furthermore, we have shown that using the optional QAck feature provides an effective means of reducing the polling overhead. While this yields little or no improvement on the admission control limit of HCCA traffic, it further increases the throughput of contention-based traffic. The increase, however, is larger with RTH than it is with the sample scheduler.

Acknowledgement

This study was carried out within the framework of the QUASAR project, funded by the Italian Ministry of Education, University and Research.

References

Enzo Mingozzi has been an associate professor at the Faculty of Engineering of the University of Pisa, Italy, since January 2005. He received a Laurea (cum laude) degree and a PhD in Computer Systems Engineering in 1995 and 2000, respectively, from the University of Pisa. His research activities span several areas, including design and performance evaluation of multiple access protocols for wireless networks, QoS provisioning and service integration in IP networks. He has been involved in several national (FIRB, PRIN) and international (Eurescom, IST) projects, as well as research projects supported by private industries (Telecom Italia Lab, Nokia). He actively took part in the standardization process of HIPERLAN/2 and HIPERACCESS networks, in the framework of the ETSI project BRAN (Broadband Radio Access Networks). He has served on the Technical Program Committee of several international conferences. He served as the Tutorial Chair for the European Wireless 2002 (EW2002) Conference Committee, and guest-edited a special issue of the Wireless Networks journal on selected papers from EW2002.

Giovanni Stea is an assistant professor (Ricercatore) at the Department of Information Engineering of the University of Pisa. His current research focus is design and performance evaluation of computer networks. He has coauthored several international publications and international patents. He has been and is involved in national and European research projects, such as IST WINEGLASS, Wireless IP NEtwork as a Generic platform for Location Aware Service Support, and IST EuQoS, End-to-end Quality of Service support over heterogeneous networks. He has served as a Member of the Technical Program Committee and/or Organization Committee for several international conferences, including European Wireless, VALUETOOLS and SIGCOMM.