# A Software Architecture for Simulating IEEE 802.11e HCCA

Claudio Cicconetti, Luciano Lenzini, Enzo Mingozzi, Giovanni Stea

*Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy*
*Via Diotisalvi 2, 56122 Pisa, Italy*
E-mail: {c.cicconetti,l.lenzini,e.mingozzi,g.stea}@iet.unipi.it

## Abstract

*We present a software framework for simulating the HCF Controlled Channel Access (HCCA) in an IEEE 802.11e system. The proposed approach allows for flexible integration of different scheduling algorithms with the MAC. The 802.11e system consists of three modules: Classifier, HCCA Scheduler, MAC. We define a communication interface exported by the MAC module to the HCCA Scheduler. A Scheduler module implementing the reference scheduler defined in the draft IEEE 802.11e document is also described. The software framework reported in this paper has been implemented using the Network Simulator 2 platform. A preliminary performance analysis of the reference scheduler is also reported.*

## 1. Introduction

In recent years Wireless Local Area Networks have become very popular due to the increasing interest of residential and office customers in ubiquitous services. The IEEE 802.11 protocol [1] has established as the world-wide standard in wireless indoor and outdoor LANs. On the other hand, the high level of performance provided by the wired networks is driving the users toward an emerging set of applications with Quality of Service requirements, such as phone or videoconference over IP networks. In order to support applications with QoS requirements the upcoming IEEE 802.11e amendment [2] provides the IEEE 802.11 MAC with two additional access mechanisms: Enhanced Distributed Channel Access (EDCA), with distributed control which enables prioritized channel access, and HCF Coordination Channel Access (HCCA), which instead requires centralized scheduling, and allows the applications to negotiate parameterized service guarantees. The interested reader may refer to [3] for a full description of the 802.11e enhancements to support QoS. In this paper, we present a software framework for simulating HCCA. The proposed framework models the 802.11e system by means of three different modules, namely *Classifier*, *HCCA Scheduler*, *MAC*. The *MAC* module exports an interface to the *HCCA Scheduler* module. Thus, different scheduling algorithms can be easily integrated. As an example, we implemented the *reference* scheduler defined in the 802.11e draft specification [2]. The above software framework has been implemented using the Network Simulator 2 (ns2, [4]). A preliminary performance assessment of the *reference* scheduler is also reported, showing the isolation provided to delay-sensitive flows in the case of heavy-loaded system.

The rest of the paper is organized as follows. In Section 2 we describe the HCCA access function specified in [2]. Section 3 contains the main contribution of this paper, which is the software framework for simulating the 802.11e HCCA. In order to validate the framework that has been devised in Section 2, in Section 4 we discuss some preliminary results that have been obtained with a sample scenario. Finally, we draw conclusions in Section 5.

## 2. HCF Controlled Channel Access Description

The HCCA is a centralized access mechanism controlled by the Hybrid Coordinator (HC), which resides into the QoS-enabled Access Point (QAP). Each QoS-enabled station (QSTA) may have up to eight established Traffic Streams (TS); a TS is characterized by a Traffic Specification (TSPEC) which is negotiated between the QSTA and the QAP. Mandatory fields of the TSPEC include: Mean Data Rate, Delay Bound, Nominal SDU Size. For all established streams the QAP is required to provide a service that is compliant with the negotiated TSPEC under controlled operating conditions. 802.11e compliant stations must be able to process the additional frames reported in Table 1.

| QoS frames | QoS piggybacked frames |
|---|---|
| QoS Data | QoS Data + CF-Ack |
| QoS CF-Ack | QoS Data + CF-Poll |
| QoS Null | QoS Null + CF-Ack |
| QoS CF-Poll | QoS Data + CF-Poll + CF-Ack |

**Table 1. QoS frames**

The QAP enforces the negotiated QoS guarantees by scheduling Controlled Access Phases (CAPs). A CAP is a time interval during which the QAP may either transmit

MSDUs of established downlink TSs or poll one or more QSTAs by specifying the maximum duration of the transmission opportunity (TXOP): a QSTA is never allowed to exceed the TXOP limit imposed by the QAP, including interframe spaces and acknowledgments. If the traffic stream of a polled QSTA is not backlogged, then the QSTA responds with a Null frame. Fig. 1 shows a sample CAP during which the QAP transmits two frames and polls the QSTA, which in turn transmits two frames. It is worth noting that the scheduling of CAPs, i.e. of HCCA traffic streams, also affects the overall capacity left to contention-based traffic, i.e. EDCA and DCF.

- The 802.11e provides three acknowledgment modes: *direct acknowledgment*: each Data frame is acknowledged by the recipient station immediately after it has been correctly received. The recipient station may piggyback the acknowledgment to an outgoing frame directed to the sending station in order to reduce the MAC overhead (see new frame types in Table 1). A further optimization consists in using of the *QAck* optional feature. If both the QAP and the sending QSTA are QAck-enabled, then the QAP may piggyback an acknowledgment into a frame directed to a different QSTA than the sending one
- *no acknowledgment*: data frames are never acknowledged by the recipient station
- *block acknowledgment*: several acknowledgments are aggregated into one frame. The 802.11e does not specify a standard procedure that the sending station should apply when fragmenting and concatenating the MSDUs in a burst of frames. Since, to the best of our knowledge, there is no previous work on this particular issue, we leave this optional mode for future investigation.

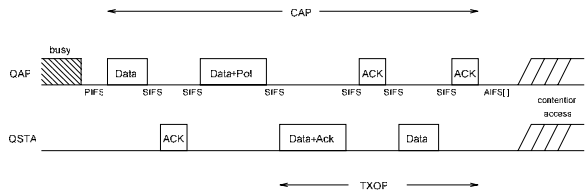


**Fig. 1. Example of HCCA frame exchange sequence**

The IEEE 802.11e standard does not define a mandatory HCCA scheduling algorithm; however, a reference scheduler is specified and reported therein for informational purposes. The *reference* scheduler requires that flows specify the following TSPEC parameters: Mean Data Rate, Nominal SDU Size, Maximum SDU Size and Maximum Service Interval (MSI). The MSI of a given flow is the maximum time that elapses from the start of two subsequent service periods to that flow. The reference scheduler produces TDM-like schedules: each TS is periodically allocated a fixed amount of capacity. The period is called Service Interval (SI) and it is the same for all traffic streams. It is computed as the smallest admitted MSI. The TXOP duration is then set to the time required to transmit the packets of Nominal SDU Size that arrive at the negotiated Mean Data Rate during the SI; the TXOP is rounded up to contain an integer number of Nominal SDU Size. In order to avoid head of line blocking, the actual TXOP value is the maximum between the value obtained with the above procedure and the time to transmit a packet with Maximum SDU Size. A sample schedule showing three admitted flows (*i*, *j* and *k*) is reported in Fig. 2.

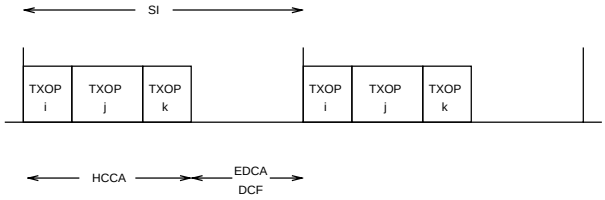


**Fig. 2. Sample schedule with the *reference* scheduler**

## 3. Software Architecture

The simulation framework is shown in Fig. 3 and consists of the following modules: *Classifier*, *MAC*, *HCCA Scheduler*. These are described later in this section and are functional to the simulation of 802.11e HCCA.

The *Link Layer* and *Measurement* modules are external to the 802.11e and depend on the simulation environment where the proposed framework is implemented. The former module is required to connect the MAC to upper layers. Performance is evaluated through the use of the *Measurement* module.

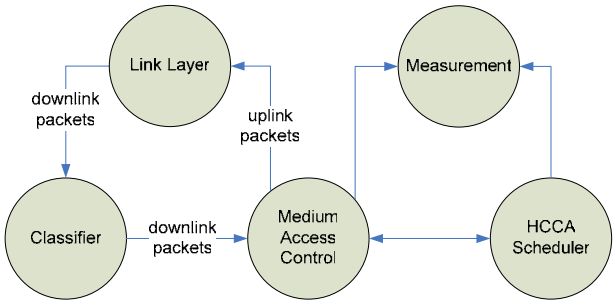


**Fig. 3. Software modules**

### 3.1. Classifier Module

The function of the *Classifier* module is to appropriately tag packets that belong to established traffic streams with a Traffic Identifier (TID). Only packets from the Link Layer to the MAC are tagged, because uplink packets are just passed to upper layers without any scheduling/differentiation treatment.

Each station runs a separate instance of the *Classifier* module, which may be further specialized in the following two types:

- Classifier for a QSTA: the tagging policy is based on a terminal-specific set of rules
- Classifier for the QAP: the tagging policy is based on the above set of rules and on the identifier of the destination QSTA

The *MAC* and *HCCA Scheduler* modules are able to retrieve the TID of any packet.

### 3.2. *MAC* module

In this subsection, we describe the main data structures, functions and events of the *MAC* module.

There are three piggybacking policies, which may be set on a per-station basis:

- *no piggybacking*: the only frames used are those listed in the left column of Table 1 under "QoS frames"
- *piggybacking on*: the *MAC* piggybacks an acknowledgment on outgoing Data frames directed to the same station only
- *QAck*: the optional QAck feature is turned on. Therefore, the piggybacking is used whenever it is possible.

We devised the following set of events that drive the *MAC* state machine:

- `HCCA_HAS_CONTROL`. This event notifies the station that it has control of the medium. A QSTA generates this event when it is polled from the QAP. The QAP generates this event when it senses the medium idle for a period larger than PIFS or when it receives the last frame from a QSTA during a TXOP burst.
- `HCCA_LOST_CONTROL`. This event notifies the station that it has not the control of the medium anymore. It is generated when the *HCCA Scheduler* does have any packets to send. Also, the QAP generates this event when a QSTA correctly responds to a polling frame and a QSTA generates this event when the QAP correctly acknowledges the last frame of the TXOP burst.
- `HCCA_DATA_RECV`. This event notifies the station that a frame carrying data addressed to this station has been correctly received.
- `HCCA_RECV`. This event notifies the station that a frame of any type (acknowledgment, Data and poll frames) addressed to this station has been correctly received.
- `HCCA_SUCCESS`. This event notifies the station that a downlink Data frame has been correctly acknowledged by the receiving station.
- `HCCA_TRANSMIT`. This event notifies the station that a downlink frame has been dispatched.
- `HCCA_TX_END`. This event is generated by the QAP when the TXOP granted to a QSTA expires
- `HCCA_CAP_HAND`. This event is generated by the QAP when it is time to start a new CAP, according to the *HCCA Scheduler* requirements.

Fig. 4 shows the above events in a sample frame exchange sequence: the transmission of two uplink Data frames, followed by the transmission of a downlink Data frame, assuming that neither collision nor frame corruption due to bad channel state occur.

In Fig. 5 the finite state machines of the QAP and QSTA *MAC* modules are depicted. After the initialization of the HCCA subsystem (`HCCA_START` event), a station alternates between two main states (HAS_CONTROL and LOST_CONTROL, shown in more detail in Fig. 6 and Fig. 7). The only difference between the QAP and the QSTA is that the former may access the medium at any moment, provided that ongoing frame exchanges are not interrupted. Instead, the only way for a QSTA to access the medium using HCCA is responding to a poll from the QAP. The *MAC* is notified by the *HCCA Scheduler* of the start time of the next CAP and uses a dedicated timer (`mhCap_`) to this purpose.

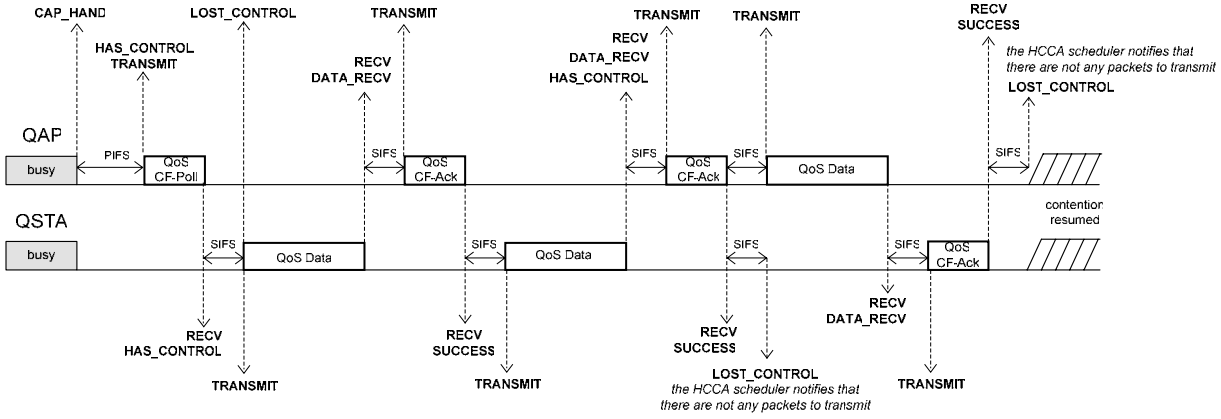


**Fig. 4. Events during a sample frame exchange sequence**

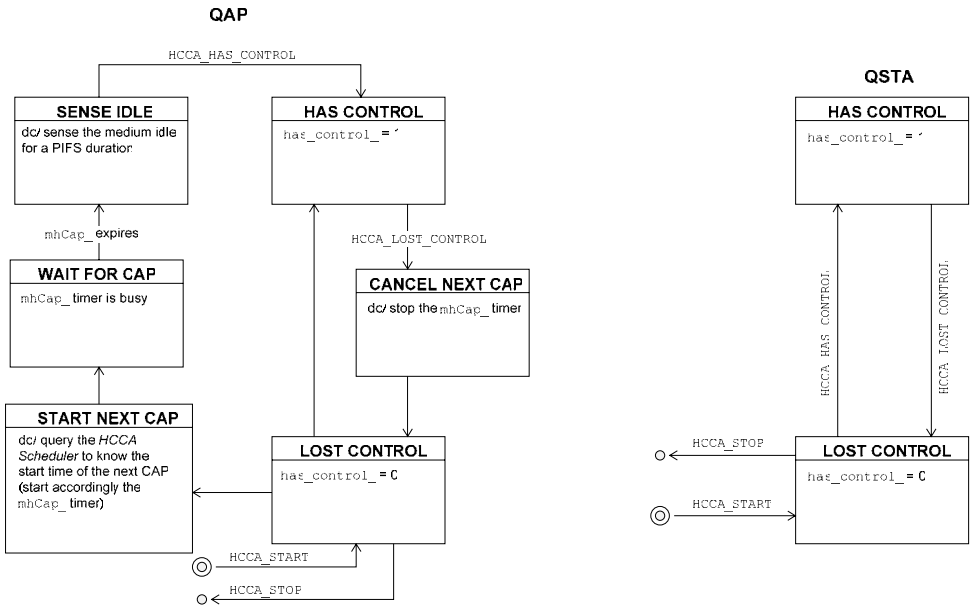


**Fig. 5. State chart of the QAP and QSTA *MAC* modules**

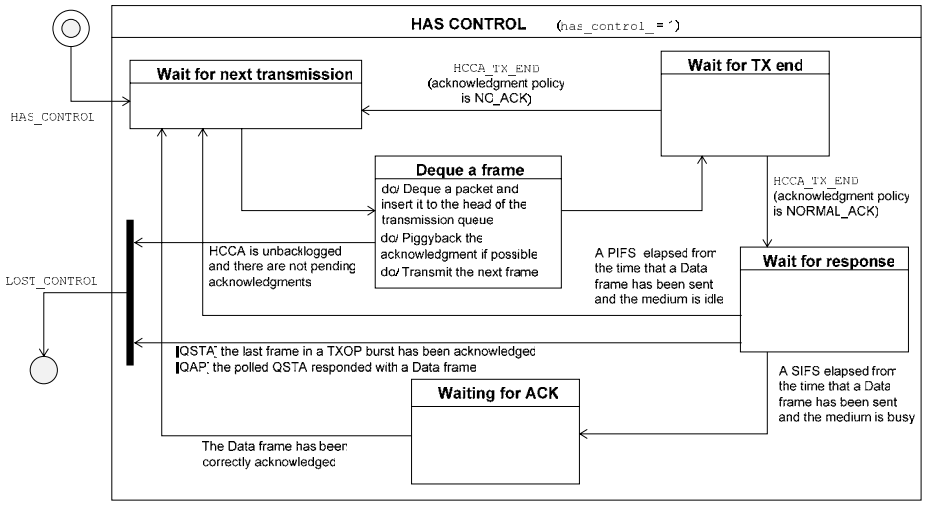


**Fig. 6. State chart of the *MAC* module (HAS_CONTROL case)**

When the HAS_CONTROL block is entered (Fig. 6) the station waits for the medium to become idle. Then the *MAC* requests the head-of-line packet to the *HCCA Scheduler*. If the HCCA is currently unbacklogged, then the station immediately loses the control of the medium. Otherwise, the frame is sent to the physical layer. If there is a pending acknowledgment the station may piggyback it to the outgoing frame, provided it is allowed by the sending and receiving capabilities. The outgoing frame may or may not require an explicit acknowledgment. In the first case, the cycle restarts immediately. In the second case, the station waits for the acknowledgment after transmitting the frame, and:

- if an acknowledgment is correctly received after a SIFS the station returns to its initial state
- if the station is the QAP and the medium is idle for a period greater than or equal to PIFS, then it claims the control of the medium, so that contention-based traffic cannot use time slots reserved to HCCA during a recovery phase
- if the station is the QAP and there is a response to the last poll frame, then the station loses the control of the medium
- finally, if the station is a QSTA and the last frame in the current TXOP burst has been acknowledged, then the station loses the control of the medium

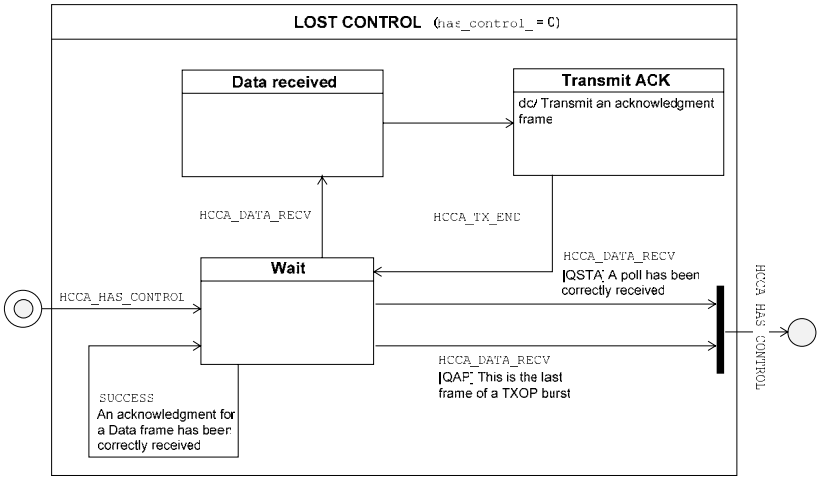


**Fig. 7. State chart of the *MAC* module (LOST_CONTROL case)**

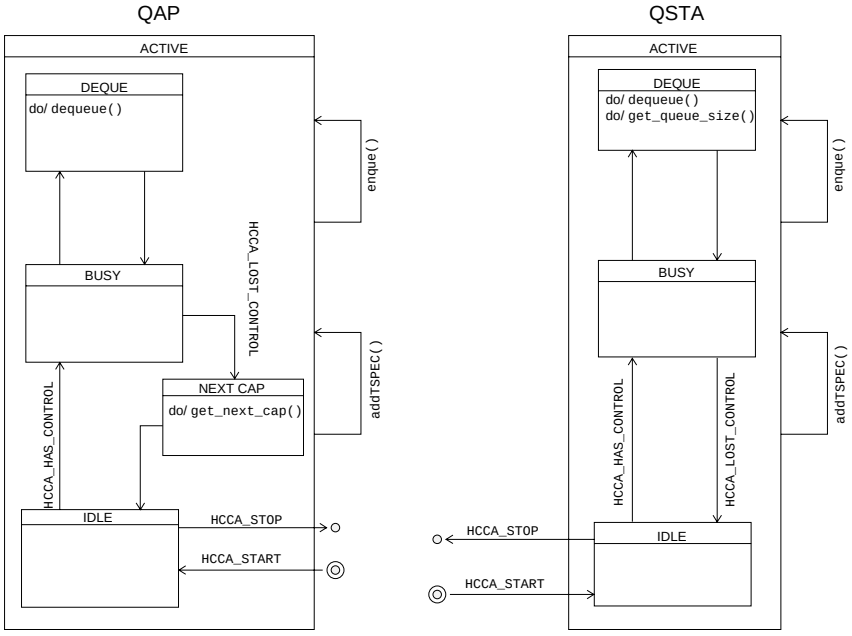


**Fig. 8. State chart of the QAP and QSTA *HCCA Scheduler* modules**

When the LOST_CONTROL block (Fig. 7) is entered the station continuously listens to the medium. Four events may occur:

- an acknowledgment to a previously transmitted Data frame is received. In this case the station immediately returns to the initial state
- a Data frame that requires direct acknowledgment is received, in which case the acknowledgement is transmitted after a SIFS duration
- if the station is a QSTA it may receive a poll frame from the QAP. In this case the station enters the HCCA_HAS_CONTROL block
- finally, if the station is the QAP it may receive the last frame of a TXOP burst. In this case the station enters the HCCA_HAS_CONTROL block.

## 3.3. HCCA Scheduler Module

The main component of the HCCA simulator architecture is the *HCCA Scheduler* module. Unlike those operating at the network layer, schedulers operating at the MAC layer heavily depend on the underlying layer-2 and physical layers. Thus, we have defined an interface which is general enough to adapt any sort of scheduling algorithm to the specified framework. The interface is shown in Fig.

8, which reports the block diagrams of the QAP and QSTA *HCCA Scheduler* modules.

When in active state (i.e., when the user turns the simulation of HCCA on) both the QAP and the QSTA schedules alternates between two main states:

- `IDLE`. This is a *passive* state: enqueued packets from downlink TSs are not transmitted using HCCA and no polls are generated (QAP only). The only actions allowed in the current state are: (i) enqueue a new downlink packet of an established TS (ii) add a new TS to the set of established TSs
- `BUSY`. This is an *active* state. All the allowed action in the previous state may take place in the current one. Moreover, downlink packets that belong to established TSs may be transmitted or the QAP may poll QSTAs with established TSs. Note that the QSTAs reach this state only on receipt of a poll from the QAP, whereas the QAP itself is always in this state except during the transmission opportunities that it granted to the QSTAs.

The following functions are specific to the scheduling algorithm that is used at the QAP and QSTAs:

- `enque()`. This function is called by the Link Layer whenever a new downlink packet has been received from upper layers. The incoming packet should be enqueued in the scheduler-specific data structures of the *HCCA Scheduler* module. There are cases in which the scheduler may drop packets, for example if the buffer allocated to downlink packets is finite or according to time-to-live packet policies.
- `deque()`. This function is called by the MAC when the station has the control of the medium. Its effects depend on the scheduling algorithm and the current status of the scheduler. The allowed actions include: (i) the scheduler passes to the MAC a downlink packet to be transmitted (ii) the scheduler communicates to the MAC that a given QSTA should be polled with a specified TXOP duration (iii) a poll to itself should be sent (iv) no action is to performed. Of course, actions (ii) and (iii) are only performed by the QAP scheduler.
- `get_next_cap()`. [QAP only] This function is called by the MAC when the QAP scheduler loses the control of the medium. It is used by the scheduler to notify the MAC of the start of the next CAP. Until that time, the HCCA function of the QAP is idle.
- `addTSPEC()`. [QAP only] This function is called by the MAC when the admission of a new traffic stream is requested
- `get_queue_size()`. [QSTA only] This function is called by the MAC before transmitting a Data frame. It returns the queue size of a specified TS. This information is piggybacked to all Data frames generated by a QSTA that belong to an established TS.

Since the scheduler procedure may require some status information, we made available the full set of events described in the previous section to the *HCCA Scheduler* module. However, it is quite unlikely that any scheduler will require all event types. Thus, we allow a scheduler to explicitly register the subset of events that it wishes to be notified of according to its own requirements.

### 3.3.1. Example HCCA Scheduler Module for the QAP: *reference* Scheduler

The *reference* scheduler defined in [2] perfectly fits the contributed framework. Fig. 9 shows the main data structure used by *reference*, i.e. a circular list of descriptors. Each descriptor contains information related to an established TS. In case of downlink TS, then the descriptor also stores the corresponding transmission queue.

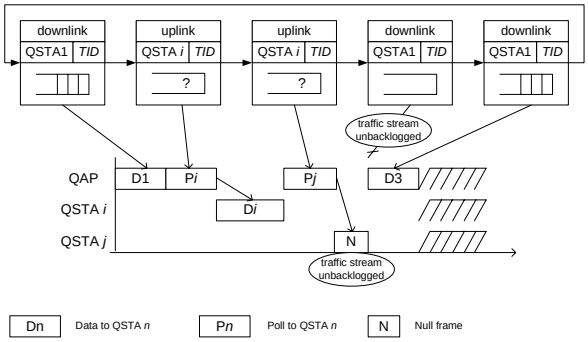


**Fig. 9. Sample schedule with *reference* scheduler**

The implementation of the interface methods between the *MAC* and the *HCCA Scheduler* modules is quite simple in the *reference* case:

- `enque()`. This method enqueues the packet into the appropriate transmission queue.
- `deque()`. If this is the last descriptor in the list, then the QAP refrains from accessing the medium using HCCA until the next SI. If this the current descriptor is not the last, then check the direction of the associated TS: (i) if downlink and the transmission queue is not empty, send as many frames as allowed by the maximum duration granted to that TS; (ii) if uplink, poll the corresponding QSTA for a duration equal to the maximum TXOP granted to that TS.
- `get_next_cap()`. This method just returns the remaining time to the next SI.
- `addTSPEC()`. This method performs the admission control procedure described in [2]. If the traffic stream is admitted, it creates a new descriptor and inserts it into the circular list.

The only registered event is `HCCA_SUCCESS`. This way the MAC notifies the scheduler that the last dequeued packet has been correctly received by the recipient station, and hence it may be removed from the queue of the corresponding TS.

### 3.3.2. Example HCCA Scheduler Module for the QSTAs: *oneflow* Scheduler

We implemented an example scheduler (*oneflow*) to be used by QSTAs that only establish one uplink TS with the QAP. Outgoing packets are managed in a First-In-First-Out manner. In this assumption, the required methods may be trivially implemented as follows:

- `enque()`. This method adds the packet received from the Link Layer to the tail of the transmission queue.
- `deque()`. This method returns the packet at the head of the transmission queue.
- `addTSPEC()`. This method checks that there is not a previously established TS.
- `get_queue_size()`. This method returns the sum of the size of the enqueued packets.

The only registered event is `HCCA_SUCCESS`, which causes the head-of-line packet in the transmission queue to be actually dequeued.

### 3.3.3. Performance Evaluation

Our contribution to the ns2 simulation environment consists in the C++ source code of the *Classifier*, *MAC*, *HCCA Scheduler* and *Measurement* modules. The legacy ns2 implementation of IEEE 802.11 consists of one `Mac802_11` class containing the complete state of the station at the MAC level and including: receiving/sending buffers, physical and MAC configuration (stored in two separate MIB data structures), timers to detect transmission failure, to synchronize with interframe spaces and to perform backoff procedures. The only access function supported is DCF. The *MAC* class inherits from the 802.11 one, so that the stations using the *MAC* module can use both the HCCA and the DCF to access the medium.

The simulation environment consists of one QAP and 6 QSTAs operating in IEEE 802.11b mode. The MAC and physical parameters are shown in Table 2. We assume that the channel is error-free and that there are no hidden stations. RTS/CTS protection mechanism, MAC-layer fragmentation and piggybacking features have been turned off during simulation.

We show that the *reference* scheduler is able to provide the established traffic streams with QoS guarantees, even though the system is heavy loaded. Since we are not interested in evaluating the performance of contention-based access, we assume that the stations with data traffic operate in asymptotic condition, that is the transmission queue is never empty. The packet size if fixed and equal to 1500 *bytes*. Three stations operate in such condition, with the service starting at time 420 *s* and terminating at the end of the simulation.

| | | | |
|---|---|---|---|
| **Data rate** | 11 *Mbps* | **SIFS** | 10 $\mu s$ |
| **Basic rate** | 1 *Mbps* | **PIFS** | 30 $\mu s$ |
| **PHY header** | 192 $\mu s$ | **DIFS** | 50 $\mu s$ |
| **Retry Limit** | 7 | **CWmin** | 31 |
| | | **CWmax** | 1023 |

**Table 2. MAC and physical parameters**

Each of the remaining QSTAs has a bi-directional video streaming session active during the whole simulation. In the first time interval (from 20 to 820 *s*) the EDCA function is used to access the medium, while in the last interval (from 820 *s* to 1220 *s*) one TS is admitted by the QAP for each video session. The video stream is a medium-quality version of the MPEG4 encoded *Jurassic Park* movie [5], [6]; the average bit rate is about 270 *Kbps* with a frame rate of 25 *fps*, corresponding to a frame interarrival time of 40 *ms*.

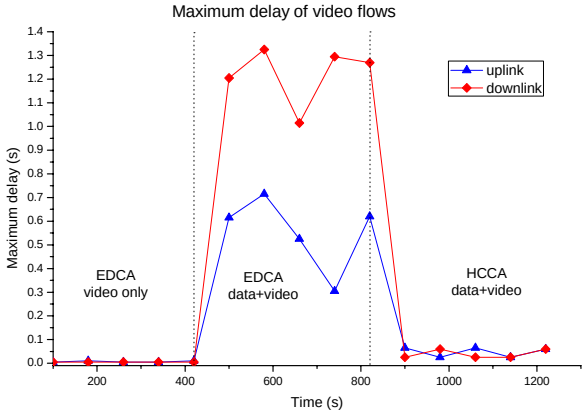


**Fig. 10. Maximum delay (video flows)**

In Fig. 10 we show the maximum delay experienced by correctly received packets (computed over time windows of 40 *s* for the sake of readability). In the first simulation period the asymptotic stations are inactive. In this case, the system is lightly loaded; in fact, the offered load is below 1.5 *Mbps*, and so very few collisions occur. Thus, the delay of video flows is bounded by a small value. However, this is not also true in the second period, when the system is overloaded by the stations in asymptotic condition. During this interval, the video flows experience very high maximum delay resulting in poor performance at the application layer. Furthermore, we note that even though the QAP has to deliver downlink video packets to all QSTAs, it accesses the medium with the same transmission probability as the latter ones. This is a known problem in 802.11 networks operating in infrastructure mode. Finally, after 820 seconds the QAP accepts the admission of the uplink and downlink video streams of all the QSTAs. We set the negotiated Mean Data Rate of the TSPEC element so that the video frames

may be served in one or two service intervals. Thus, during the third simulation period the maximum delay of video packets is bounded by 80 ms (2 × MSI = 2 × 40 *ms*).

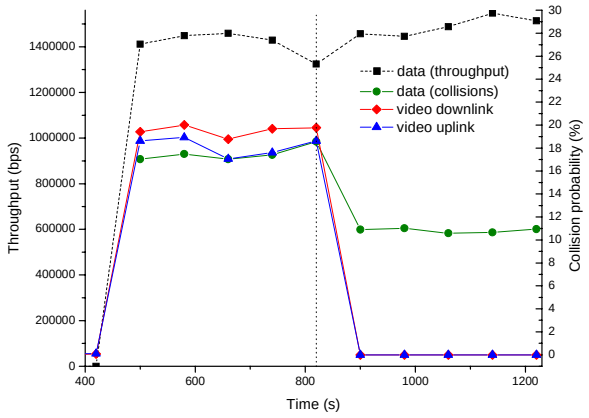


**Fig. 11. Throughput and collision probability**

We also show that servicing video flows with HCCA does not degrade the performance of contention-based traffic. In fact, the throughput that can be reached by a tagged station in asymptotic condition is slightly higher in the third simulation period (Fig. 11). This counterintuitive behavior may be explained by the fact that the collision probability when using a random backoff access procedure depends on the number of contending stations: this number is higher when all the stations use the EDCA function. As soon as the HCCA is used, the collision probability of data packets drops of about 7%, leading to a better efficiency in the medium access. This overcompensates the effect of reduced bandwidth due to transmission of CAPs. Finally, we note that the collision probability of the downlink video packets is higher than that of the uplink ones, which confirms the asymmetry of the 802.11 infrastructure mode.

## 4. Conclusions

In this paper we presented a software framework for simulating the IEEE 802.11e. In particular, we designed the following modules to model the MAC layer of an 802.11e system: *Classifier*, *MAC*, *HCCA Scheduler*. A set of MAC-related events, driving the *MAC* module finite state machine on both QAP and QSTA, has been devised. These events are listened by the *HCCA Scheduler*, so that any scheduling policy can be devised and fit into the proposed architecture. As an example, we implemented an instance of *HCCA Scheduler*, namely the *reference* scheduler described in the 802.11e draft. Finally, we described a working implementation of the presented framework in the *Network Simulator version 2* and tested it simulating a scenario involving the transmission of a mix of best-effort and delay-sensitive traffic. These preliminary results show that the *reference* scheduler is able to provide the simulated delay-sensitive streams with QoS guarantees, without degrading the performance of best-effort traffic.

## Acknowledgements

This work was partially funded by the European Union 6th Framework Programme under contract IST FP6 IP 004503 EuQoS Integrated Project. The authors would like to thank Federico Stentella for his substantial contribution to the ns2 implementation.

## References

[1] IEEE Computer Society LAN MAN Standards Committee. IEEE 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications, August 1999.

[2] IEEE Computer Society LAN MAN Standards Committee. IEEE 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications. Medium Access Control (MAC) Quality of Service (QoS) Enhancements, Draft D??, July 2004

[3] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, and L. Stibor. IEEE 802.11e wireless LAN for quality of service. In EW2002 - Proc. European Wireless, volume 1, pages 32–39, February 2002.

[4] http://www.isi.edu/nsnam/ns/

[5] Frank H.P. Fitzek, Martin Reisslein. MPEG-4 and H.263 Video Traces for Network Performance Evaluation. IEEE Network, Vol. 15, No. 6, pages 40-54, November/December 2001.

[6] http://www-tkn.ee.tu-berlin.de/research/trace/trace.html