Event-Driven Simulation for IEEE 802.11e Optimization

Orlando Cabral, Alberto Segarra and Fernando J. Velez, *Member, IAENG*

***Abstract*— This work provides the specification of a PHY and MAC layers simulator that allows to evaluate the IEEE 802.11e service quality. A detailed state transition diagram is presented along with a description of the related attributes and methods, an identification of the associated events and input variables, and a description of functions for events. For validation purposes, results of the goodput as a function of the signal to interference noise ratio were obtained with an error margin lower than 10%. Higher values are found for the goodput for background and video services, mainly because the frames transmitted in these services are longer than the voice application ones. However, the number of supported users is higher for voice. This simulator will allow for tuning-up several parameters like the ones related to how to use BlockACK, normal ACK, and NO ACK policies.**

***Index Terms*— QoS, simulation, IEEE 802.11e, optimization.**

I. Introduction

In recent years, an amazingly rapid evolution in wireless local area networks (WLANs) as occurred. Due to the low cost, and easiness of deployment, IEEE 802.11 WLANs have been used so widely that they become the dominating WLAN technology. This is mainly because the technology is reaching an unprecedented maturity in regard to providing higher bit rates as the time goes by; however, it could not fulfil the increasing demand for quality-of-service (QoS) support from the increasingly popular multimedia applications yet.

To overcome this limitation, the IEEE 802.11e standard [1] is specified aiming to support QoS by providing differentiated classes of service in the medium access control (MAC) layer and to enhance the ability of the physical layer so that they can deliver time-critical multimedia traffic, in addition to traditional data packets.

This work addresses the service quality in IEEE 802.11 WLANs. We produced a simulator that enables simulations that analyse the performance and allows the improvement of IEEE 802.11e mechanisms, such as arbitrary inter frame spacing, differentiated backoff procedure, and the transmission opportunities for each service class, as well as experiments on acknowledgment policies. Besides these features, it was build accounting for inter-working with Worldwide Interoperability for Microwave Access (WiMAX) and High Speed Downlink Packet Access (HSDPA) simulators by means of improved scheduling algorithms and CRRM techniques. Scenarios that focus the interoperability among several wireless networks like Wireless-Fidelity (Wi-Fi), WiMAX, HSDPA, and Digital Video Broadcasting Handheld (DVB-H) are a final goal.

Manuscript received October 30, 2007. Orlando Cabral, Alberto Segarra, and Fernando Velez are with the Instituto de Telecomunicações, University of Beira Interior, Covilhã, Portugal (phone: +351 275329953; fax: +351 275329972; e-mail: orlandoc@ubi.pt, vyniard@gmail.com, fjv@ubi.pt).

The structure of this paper is as follows. In Section II, an overview of the IEEE 802.11e standard is addressed. In Section III, the hybrid coordination function is presented and details are given on the enhanced distributed channel access. Section IV includes the presentation of the state transition diagram, its variables, entities, events, and functions. In Section V, details are given on the physical layer of the IEEE 802.11a standard, the one considered in this work. Section VI presents the validation of our simulator after joining together PHY plus MAC functionalities in the same tool. This validation is performed based on extensive simulation results obtained. Section VII includes the hypothesis for system and scenarios, including details on traffic parameters. Section VIII presents simulation results for packet delay, goodput, and channel utilization. Conclusions are presented in Section IX as well as suggestion for future work.

II. IEEE 802.11e

The IEEE 802.11 architecture consists of several components that interact to provide a WLAN that supports station mobility transparently to upper layers. The basic service set (BSS) is the basic building block of an IEEE 802.11 LAN. Figure 1 presents two BSSs, each of which has two wireless stations that are members of the BSS. Instead of existing independently, a BSS may also form a component of an extended form of network that is built with multiple BSSs. The architectural component used to interconnect BSSs is the distribution system (DS). IEEE 802.11 logically separates the wireless medium (WM) from the distribution system medium (DSM). Each logical medium is used for different purposes, by a different component of the architecture. An access point (AP) is a station (STA) that provides access to the DS by providing DS services in addition to acting as a STA Data move between a BSS and the DS via an AP, Figure 1.

The IEEE 802.11 quality of service (QoS) facility provides medium access control (MAC) enhancements to support local area network (LAN) applications with QoS requirements. The QoS enhancements are available to QoS stations (QSTAs) associated with a QoS access point (QAP) in a QBSS.

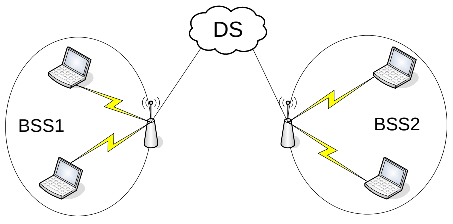

Fig. 1 - Non-roaming reference model.

The so-called enhanced distributed channel access (EDCA) mechanism delivers traffic based on differentiating user priorities (UPs), Figure 2. This differentiation is achieved by varying the following different UP values:

- Amount of time a STA senses the channel to be idle before backoff or transmission, or
- The length of the contention window to be used for the backoff, or
- The duration a STA may transmit after it acquires the channel.

The so-called hybrid coordination function (HCF) controlled channel access (HCCA) mechanism allows for the reservation of transmission opportunities (TXOPs) with the hybrid coordinator (HC), Figure 2.

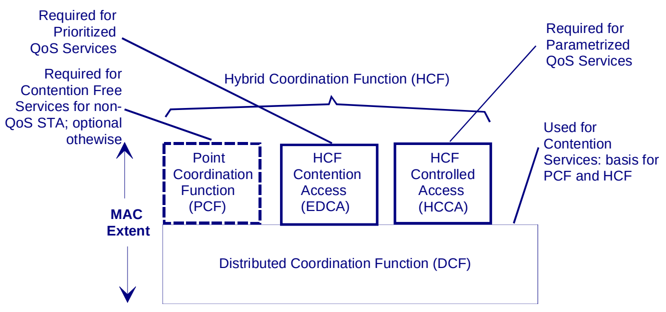
Required for Prioritized QoS Services
Required for Contention Free Services for non-QoS STA; optional othewise
Hybrid Coordination Function (HCF)
Required for Parametrized QoS Services
Used for Contention Services: basis for PCF and HCF
MAC Extent
Point Coordination Function (PCF)
HCF Contention Access (EDCA)
HCF Controlled Access (HCCA)
Distributed Coordination Function (DCF)

Fig. 2 - MAC architecture.

Details on the CSMA/CA protocol, and inter-frame spaces (IFS) are presented in [1]. The backoff time and the backoff procedure are addressed in [1] and [2], as well as the description of the details on the network allocation vector (NAV), and the use of RTS/CTS with fragmentation. The fragmentation is the process of partitioning a MAC service data unit (MSDU) or a MAC management protocol data unit (MMPDU) into smaller MAC level frames, MAC protocol data units (MPDUs).

III. Hybrid Coordination Function (HCF) Enhanced Distributed Channel Access (EDCA)

The QoS facility includes an additional coordination function called HCF that is only usable in QoS network (QBSS) configurations. The HCF shall be implemented in all QSTAs.

The EDCA mechanism provides differentiated, distributed access to the WM for QSTAs using eight different UPs. The EDCA mechanism defines four access categories (ACs) that provide support for the delivery of traffic with UPs at the QSTAs, Figure 3. The AC is derived from the UPs, as presented in Table I. For each AC, an enhanced variant of the DCF, called an enhanced distributed channel access function (EDCAF), contends for TXOPs using a set of EDCA parameters from the EDCA Parameter Set element or from the default values for the parameters when no EDCA Parameter Set element is received from the QAP of the QBSS with which the QSTA is associated.

Table I - UP mapping between user priorities and ACs.

Priority	UP (Same as 802.1D user priority)	802.1D Designation	AC	Designation
Lowest	1	BK	AC_BK	Background
	2	—	AC_BK	Background
	0	BE	AC_BE	Best Effort
	3	EE	AC_BE	Best Effort
	4	CL	AC_VI	Video
	5	VI	AC_VI	Video
	6	VO	AC_VO	Voice
Highest	7	NC	AC_VO	Voice

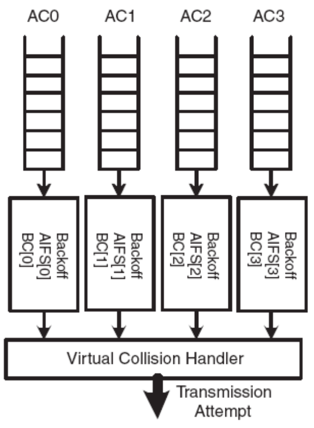

Fig. 3 – Access categories in EDCA [3].

The TXOP limit duration values are advertised by the QAP in the EDCA Parameter Set information element in Beacon and Probe Response frames transmitted by the QAP. Non-AP QSTAs shall ensure that the duration of TXOPs obtained by using the EDCA rules do not exceed the TXOP limit. The duration of a TXOP is the duration during which the TXOP holder maintains uninterrupted control of the medium, and it includes the time required to transmit frames sent as an immediate response to the TXOP holder's transmissions.

1) Obtaining an EDCA TXOP

Each channel access timer shall maintain a backoff function (timer), which has a value measured in backoff slots. The duration AIFS[AC] is a duration derived from the value AIFSN[AC] by the relation

$$AIFS[AC] = AIFSN[AC] \times aSlotTime + aSIFSTime$$

The value of AIFSN[AC] shall be greater than or equal to 2 for non-AP QSTAs and is advertised by the QAP in the EDCA Parameter Set information element in Beacon and Probe Response frames transmitted by the QAP. The value of

AIFSN[AC] shall be greater than or equal to 1 for QAPs. An EDCA TXOP is granted to an EDCAF when the EDCAF determines that it shall initiate the transmission of a frame exchange sequence. Transmission initiation shall be determined according to the following rules:

— On specific slot boundaries, each EDCAF shall make a determination to perform one and only one of the following functions [1]:

- Start the transmission of a frame exchange sequence for that access function;
- Decrement the backoff timer for that access function;
- Invoke the backoff procedure due to an internal collision;
- Do nothing for that access function.

— At each of the above-mentioned specific slot boundaries [1], each EDCAF shall start a transmission sequence if

- There is a frame available for transmission at that EDCAF;
- The backoff timer for that EDCAF has a value of zero;
- Initiation of a transmission sequence is not allowed to commence at this time for an EDCAF of higher UP.

2) Multiple frame transmission in an EDCA TXOP

Multiple frames may be transmitted in an acquired EDCA TXOP if there are more than one frame pending in the AC for which the channel has been acquired. However, those frames that are pending in other ACs shall not be transmitted in this EDCA TXOP. If a QSTA has in its transmit queue an additional frame of the same AC (as the one just transmitted) and the duration of transmission of that frame plus any expected acknowledgment for that frame is less than the remaining medium occupancy timer value, then the QSTA may start transmission of that frame after the completion of the immediately preceding frame exchange sequence plus a SIFS.

All other ACs at the QSTA shall treat the medium as busy until the expiration of the NAV set by the frame that resulted in a transmission failure, just as they would if they had received that transmission from another QSTA.

3) EDCA backoff procedure

Each EDCAF shall maintain a state variable CW[AC], which shall be initialized to the value of the parameter CWmin[AC].

The backoff procedure shall be invoked for an EDCAF when any of the following events occurs:

- A frame with that AC is requested to be transmitted, the medium is busy as indicated by either physical or virtual CS, and the backoff is zero for that AC;
- The final transmission by the TXOP holder initiated during the TXOP for that AC was successful;
- The transmission of a frame of that AC fails, indicated by a failure to receive a CTS, to receive an ACK, or to receive a BlockAck;
- The transmission attempt collides internally with another EDCAF of an AC that has higher priority, i.e., if two or more EDCAFs in the same QSTA are granted a TXOP simultaneously.

The backoff timer is set to an integer value chosen randomly with a uniform distribution taking values in the range [0,CW[AC]] inclusive. All backoff slots occur following an AIFS[AC] period during which the medium is determined to be idle.

IV. State Transition Diagram

The state transition diagram used to build the simulator is presented in Figure 4. Table II presents the actions related with each event.

Table II - UP Event actions.

	ACTIONS
1	State = LISTEN_DIFS. **Save:** Time creation packet. Length packet. Fragmentation (if it is required). Know destination. Know type of packet. Buffer [AC] !=NULL. **Schedule:** STOP_LTN_DIFS (clock + AIFS).
2	**Schedule:** STOP_TX (clock + packet_length). See the TXOPLimit to send if there is any packet more. Backoff_condition=1 for STA in LISTEN_DIFS. State = TX.
3	**Schedule:** STOP_RX_ACK (clock + ACK). State = LISTEN_SIFS.
4	State = WAIT_ACK.
5	Erase the packet. State = IDLE. **Save:** Increment number of packets transmitted. Delay of the transmission.

6	State = WAIT.
	Increase the number of collisions.
	Backoff_condition = 1;

7	**Schedule:** STOP_LTN_DIFS (clock + AIFS). State = LISTEN_DIFS.
8	Refresh NAV. State = WAIT. **Deschedule:** STOP_LTN_D. **Schedule:** STOP_RX (clock + NAV).
9	State = BACKOFF_TIMER. **If (backoff_condition == 1)** Generate backoff. Decrement backoff_value each time slot. **Else** Decrement backoff_value each time slot.
10	Suspend backoff procedure. Refresh NAV. State = WAIT. **Schedule:** STOP_RX (clock + NAV).
11	**Schedule:** STOP_RX (clock + RX). State = RX.
12	**Schedule:** START_TX (clock + SIFS). State = LISTEN_SIFS.

13	**Save:** Time creation packet. Length packet. Fragmentation (if it is required). Know destination. Know type of packet. Buffer [AC] !=NULL Backoff_condition = 1. State = WAIT.
14	**Schedule:** STOP_TX (clock + packet_length). See the TXOPLimit to send if there is any packet more. Backoff_condition=1 for STA in LISTEN_DIFS. State = TX.
15	State = IDLE.
16	**Schedule:** STOP_LTN_DIFS (clock + AIFS). State = LISTEN_DIFS.
17	**Schedule:** STOP_LTN_DIFS (clock + AIFS). State = LISTEN_DIFS.
18	State = IDLE
19	**Schedule:** STOP_LTN_DIFS (clock + AIFS). State = LISTEN DIFS
20	**Schedule:** STOP_RX (clock + packet_length). State = RX
21	**Schedule:** STOP_RX (clock + packet_length). State = RX
22	State = LISTEN_SIFS. **Schedule:** START_TX (clock + SIFS)
23	State = TX **Schedule:** STOP_TX (clock + packet_length).
24	State = RX **Schedule:** STOP_RX (clock + packet_length).
25	Decrement backoff_value **If (backoff_value == 0)** **Schedule:** START_TX (clock). **Else** **Schedule:** TIME (clock + SIFS).
26	Decrement backoff_value

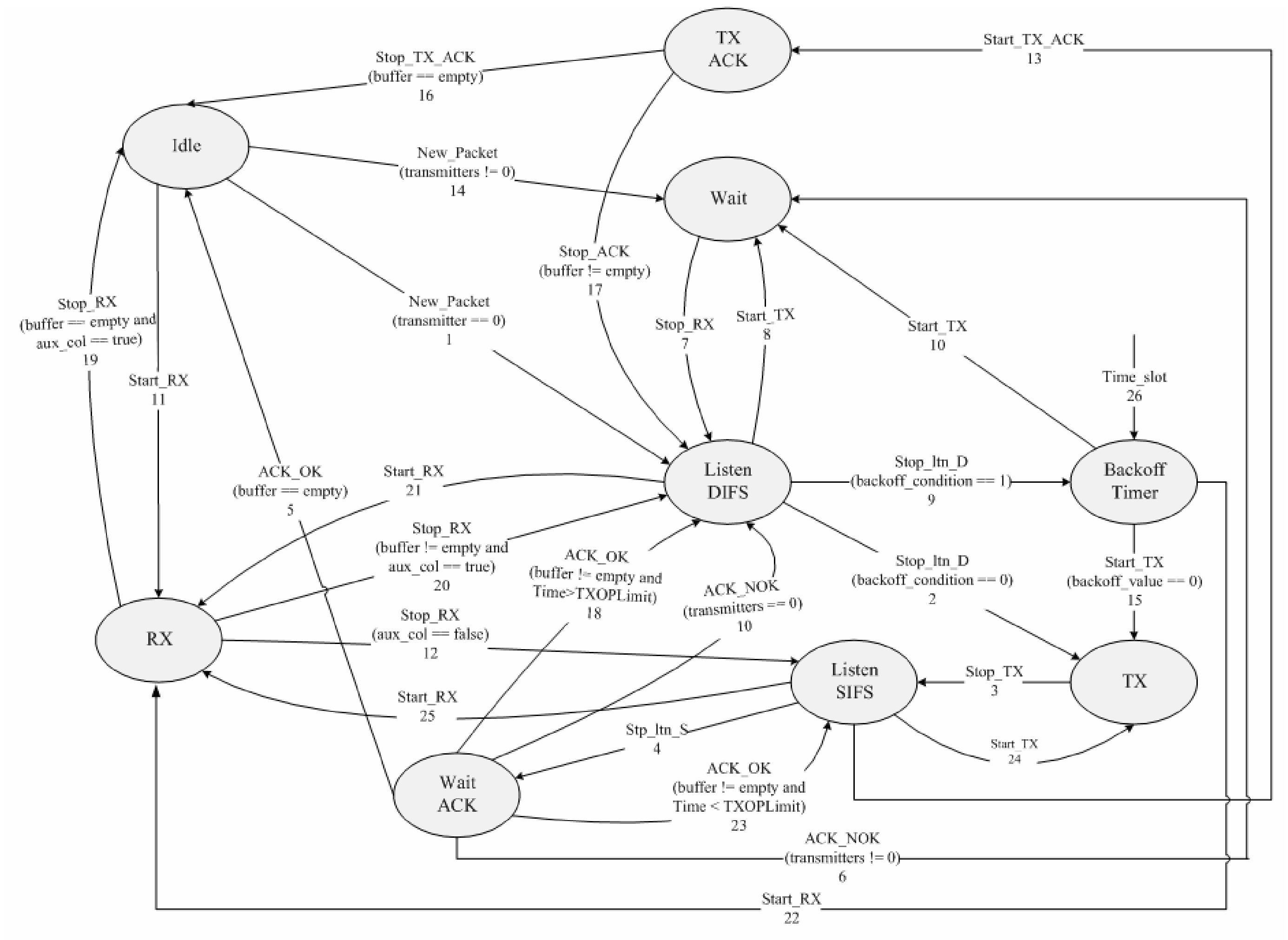
TX ACK
Start_TX_ACK 13
Stop_TX_ACK (buffer == empty) 16
Idle
New_Packet (transmitters != 0) 14
Wait
Stop_ACK (buffer != empty) 17
New_Packet (transmitter == 0) 1
Stop_RX 7
Start_TX 8
Start_TX 10
Time_slot 26
Stop_RX (buffer == empty and aux_col == true) 19
Start_RX 11
ACK_OK (buffer == empty) 5
Start_RX 21
Listen DIFS
Stop_ltn_D (backoff_condition == 1) 9
Backoff Timer
Stop_RX (buffer != empty and aux_col == true) 20
ACK_OK (buffer != empty and Time>TXOPLimit) 18
ACK_NOK (transmitters == 0) 10
Stop_ltn_D (backoff_condition == 0) 2
Start_TX (backoff_value == 0) 15
RX
Stop_RX (aux_col == false) 12
Listen SIFS
Stop_TX 3
TX
Start_RX 25
Stp_ltn_S 4
Start_TX 24
Wait ACK
ACK_OK (buffer != empty and Time < TXOPLimit) 23
ACK_NOK (transmitters != 0) 6
Start_RX 22

Fig. 4 - State transition diagram (the incoming arrow for the *Backoff_Timer* state means a transition from and to the same state).

The following events cause transition/change of the machine state:

NEW_PCK_BK	a new packet of BK is generated;
NEW_PCK_BE	a new packet of BE is generated;
NEW_PCK_VI	a new packet of VI is generated;
NEW_PCK_VO	a new packet of VO is generated;
STOP_LTN_D	end of the AIFS period for sensing the medium;
STOP_LTN_S	end of the SIFS period for sensing the medium;
TIME_SLOT	the STA decrements the backoff_value;
START_TX	the station starts to transmit;
STOP_TX	stop of the transmission;
START_RX	start of the reception;
STOP_RX	stop of the reception;
START_TX_ACK	start the transmission of an acknowledgement;
STOP_TX_ACK	stop the transmission of an acknowledgement;
ACK_OK	the acknowledgement was received;
ACK_NOK	the acknowledgement was not received.

A detailed description of the characterisation of possible states for the "machines" nodes, the simulation entities, the simulation variables, and the functions for events is given in [4]. The characterization of possible states for the "machines" nodes is the following [4]:

Machine

States

Idle: buffer is empty;
Wait: backoff counter is frozen;
Listen DIFS: waiting AIFS;
Backoff Timer: decreasing the backoff timer;
TX: transmitting packet;
Listen SIFS: waiting SIFS;
RX: receiving a packet;
Wait ACK: waiting an acknowledgement;
TX ACK: transmitting an acknowledgement.

Queues

Empty: buffer is empty;
Not empty: can have *p* packets waiting to be transmitted.

Medium states

Free: medium is free, no one is transmitting;
Not free: medium not free, there is someone transmitting and someone receiving.

States for the packet

Payload;
time of generation;
backoff condition: can be either 0, 1 or 2;
origin;
destination;
backoff value;
Ncollision: number of collisions the packet has suffered;
packet_type: type of packet, can be of BK, BE, VI and VO;
frag: it is a list that takes several fragments;
rts_first: serves to know if it is the first fragment to be sent or not.

There are the following entities in the simulations[4]:

Entities:

Machine: the objects of these class are the AP and the nodes

- attributes
  - `buffer[4]`: contains the packets for each access category;
  - `colisions[4]`: contains the collisions of each access category;
  - `location`: contains the 2D position;
  - `tx_power`: indicates the power when the machine is transmitting;
  - `tx_node`: indicates if the machine is transmitting or not;
  - `stae[4]`: contains the state of each access category;
  - `delay[4]`: contains the delays for each access category;
  - `num_packets[4]`: contains the number of packets transmitted of each access category:
  - `TXOP_limit`: the time that the machine shall not pass before listen an AIFS;
  - `SBlock_ACK`: if the machine has the Block Ack policy with another machin;e
  - `VBlock_ACK`: the value of the Block ACK's buffer;
  - `RBlock _ACK`: the receiver Block ACK's buffer;
  - `packet_loss[4];`
  - `retransmissions[4].`
- methods
  - `AddFragment(Possible_packets p, fragment f)`: adds a fragment to a packet in one of the queues;
  - `AddPacket(PacketWiFi p)`: adds a packet to one of the queues;
  - `AddPacketADDBAReq(PacketWiFi p)`: adds a ADDBA Request packet to one of the queues;
  - `AddPacketADDBARes(PacketWiFi p)`: adds a ADDBA Response packet to one of the queues;
  - `AddPacketRES(PacketWiFi p)`: adds a Response packet to one of the queues;
  - `ChangeSBlock(Possible_packets p, int d, StateBlock s)`: changes the state of SBlock_ACK buffer with a destination;
  - `ChangeVBlock(Possible_packets p, int d, double v)`: changes the value of VBlock_ACK buffer with a destination;
  - `DecVBlock(Possible_packets p, int d)`: decrements the value of VBlock_ACK buffer with a destination;
  - `Del_Frags(Possible_packets p, int d, int o, vector<Machine> *m)`: erases all the fragments sent by Block ACL policy to a destination;
  - `RemoveFirstPacketLost(Possible_packets p)`: removes a packet when the L_COL is reached from the queue `p;`
  - `RemovePacket(Possible_packets p)`: removes the packet from the queue `p;`

- RemovePacketsBlockSent(Possible_packets p, int d): removes the packets sent by Block ACK policy to a destination;
- RemovesPacketsEmpty(Possible_packets p, int d, double c, bool a): removes empty packets to a destination and update the delays;
- ResetTXOP_limit(): changes the value of the TXOP_limit to 0;
- SetState(Possible_packets p, State s): sets the state of the buffer p to state s;
- SetTXOP_limit(double x, Possible_packets p): changes the TXOP_limit to a value ;
- Buffers_size(Possible_packets p): returns the size of the buffer p;
- Buffers_size(): returns the number of packets in all buffers;
- DataPackets_to_i(Possible_packets p, int o, int d): counts the data packets to a destination;
- DataPackets_to_i(Possible_packets p, int o, int d): counts the data packets to a destination;
- Packets_to_i(Possible_packets p, int o,int d): counts the total packets to a destination;
- GetTXOP_limit(): returns the value of the TXOP_limit;
- GetVBlock(Possible_packets p, int d): returns the value of the VBlock_ACK buffer to a destination;
- GetFirstPacketLost(Possible_packets p): gets a packet which has waited the response during the ACK Time_out from the queue p;
- GetLastPacket(Possible_packets p): returns the last packet sent from the queue p;
- GetNextPacket(Possible_packets p): returns the next packet to send from the queue p;
- ADDBAReqPackets_to_i(Possible_packets p, int o, int d): returns a boolean if there is a ADDBA Request packet in the buffer p;
- CheckPacketEmpty(Possible_packets p, int d): returns a boolean if there is a empty packet in the buffer p;
- PacketsNotSent(Possible_packets p): returns a boolean if there is packets not sent in the buffer p;
- GetState(Possible_packets p): returns the state of the buffer p;
- GetSBlock(Possible_packets p, int d): returns the state of the buffer p to the destination d.

Packet: the objects of this class are the packets that fill in the buffer

- o attributes
  - payload;
  - time_gen: when it was created;
  - backoff_condition: 0 do not generate backoff, 1 generate backoff, 2 decrement backoff down to 0;
  - origin: the originator of the traffic;
  - destination: to whom it is to be transmitted;
  - backoff_value: number of slots to be decremented;
  - Ncolision: number of collisions the packet has suffered;
  - packet_type: it can be of VO (voice), VI (video), BE (best effort) and BK(background);
  - frag: queue of fragments of the packet;
  - rts_first: used to check if the fragment is the first or not.
- o methods
  - AddFrag(Possible_packets p, fragment f):adds a fragment to the packet;
  - ChSentFalse(bool x): changes the variable send of the last fragment sent to false;
  - ChangeSend(): changes the variable send of the first fragment not sent to true;
  - DelFrag(): erases the last fragment sent of the packet;
  - IncNcollisions(): increments the number of collisions the packet suffered;
  - SetBackoffcondition(int x): changes the variable backoff_condition of the packet;
  - FragsNotSent(): returns the number of the fragments not sent in the packet;
  - GetBackoffcondition(): returns the variable backoff_condition of the packet;
  - GetBackoffvalue(): returns the variable backoff_value of the packet;
  - GetDestination(): returns the variable destination of the packet;
  - GetPayloadNextFrag(): returns the variable payload of the first fragment to send;
  - GetPayloadLastFrag(): returns the variable payload of the last fragment sent;
  - GetPayloadSecFrag(): returns the variable payload of the second fragment to send;
  - GetNcolision(): returns the number of collisions of the packet;
  - GetModulationLastFrag(): returns the variable modulation of the last fragment sent;
  - SetModulationNextFrag(double x): changes the variable modulation of the fragment x;
  - SetrxDataNextFrag(double x): changes the variable rxData_Pr_db of the fragment to x;
  - SetSNRNextFrag(double x): changes the variable SINR_db of the fragment to x;
  - GetSNRLastFrag(): returns the variable SINR_db of the last fragment sent;
  - GetTimeGen(): returns the variable time_gen of the packet;
  - GetBlockLastFrag(): returns the variable Block of the last fragment sent;
  - GetRTSPacket(): returns the variable rts_first of the packet;
  - GetTypeLastFrag(): returns the variable type of the last fragment sent;

    - `GetTypeNextFrag()`: returns the variable type of the first fragment to send;
    - `GetTypePacket()`: returns the variable send of the fragment to false;
    - `ChSentFalse(bool x)`: changes the variable packet_type of the packet.

Event: the objects of this class are the events, they are stored in a list

- attributes
    - `event_time`: period at which the period is going to take place;
    - `t_event`: type of event already presented before;
    - `origin`: the originator of the event;
    - `destination`: the destination of the event `p` for which AC the event is for.
- methods
    - `set_event`: sets the event attributes to the new ones.

Channel: only one object of this class is created. It stores the data that has to be passed trough events

- attributes
    - `transmitters`: number of transmitters at that moment;
    - `aux_col`: when a collision occurs this variable is set to true;
    - `n_collisions`: total collisions suffered;
    - `clock`.

Output: one object of this class is created per simulation. It stores the main outputs of the simulation. As a simulation runs for some time, these outputs are stored in a vector. At the end the simulations the averages are calculated

- attributes
    - `delay_total_BK`;
    - `delay_total_BE`;
    - `delay_total_VI`;
    - `delay_total_VO`;
    - `delay_average_BK`;
    - `delay_average_BE`;
    - `delay_average_VI`;
    - `delay_average_VO`;
    - `colisions_total`;
    - `colisions_rate`;
    - `packets_total_BK`;
    - `packets_total_BE`;
    - `packets_total_VI`;
    - `packets_total_VO`;
    - `chann_utilization`;
    - `thr_total`;
    - `thr_per_sec`.

Lista: a object of this class (one list) is created to store the events.

- attributes
    - `lis`: contains al the events;
    - `inter`: iterator of the list.
- methods
    - `del_event_after`: erases all the events after some time;
    - `Get_next_event()`: returns the first event in the list;
    - `see_event_machine()`: checks if a given event is in the list.

Random_generator: the objects of this class serve the of generate a random number with uniform distribution
Distributions: the objects of this class generate random numbers with a given distribution

Simulation variables - the main simulation variables are the following:

event_list: is the list in which all the events are sorted by time;
stations: a vector which contains all the stations;
output: where the main outputs are saved.

Input variables - the input variables are the following.

MT: total number of stations;
SIMULATION_TIME 100000: simulation lifetime in ms;
DATA_RATE 20000000.0: data rate in bits;
PAYLOAD_BK 12000: payload of the BK packets;
PAYLOAD_VO 1280: payload of the VO packets;
PAYLOAD_VI 10240: payload of the VI packets;
PAYLOAD_BE 12000: payload of the BE packets;
FACTOR 1000.0: factor used for changing units of time to ms;
DIFS 0.034: DIFS size in ms;
SIFS 0.016: SIFS in ms;
INTER_ARRIVAL_BK 12.5: interarrival time for BK packets in ms;
INTER_ARRIVAL_VO 20: interarrival time for VO packets in ms;
INTER_ARRIVAL_VI 10: interarrival time for VI packets in ms;
INTER_ARRIVAL_BE 2: interarrival time for BE packets in ms;
SLOT 0.009: SLOT time in ms;
ACK_SIZE 112: acknowledgement size in bits;
DEGREES_FREEDOM 2: number of simulations that will be taken with different random seeds;
RTS_TH 3000: RTS threshold, to use a RTS procedure;
RTS_SIZE 160: RTS size in bits;
CTS_SIZE 112: CTS size in bits;
CW_MIN 31: CW_{min};
L_COL 8: collisions limit;
TXOP_BK 0: TXOP size for the BK traffic, being 0 means that only one MSDU can be transmitted in a TXOP;
TXOP_BE 0: TXOP size for the BK traffic;
TXOP_VI 3.008: TXOP size for the VI traffic in ms;
TXOP_VO 1.05: TXOP size for the VO traffic in ms.

Functions for events

Our simulator was build in standard C++. The definition of the functions used to initialize the traffic, to generate the backoff, and the AIFS, and of the functions that deal with the events (that change the state of the machines as well) is the following:

- `Initialize(list_events, ed, stations, fs)`: this function serves to add machines and schedule the traffic for them(`ed` is an event and `fs` is a object for writing);
- `AIFS(Possible_packets p)`: generates the AIFS for a given AC;
- `Generate_backoff(int colisoes)`: generates the backoff for a given AC;
- `New_pck_bk(list_events, ed, stations, chan, fs)`: this function adds packets to a given buffer (BK). The fragmentation of the MSDU into several MPDU (if required) is performed here. Also a new arrival is scheduled here;
- `Stop_ltn_d(list_events, ed, stations, chan, fs)`: the packet in a given buffer of one given machine is made available and the payload compared with the RTSLimit to answer if the packet will be sent with RTS/CTS method or not. With this function the simulator verifies the backoff_condition, and determines if the machine has to send the packet normally, to generate a backoff, or to continue decreasing the backoff_value;
- `Time_slot(list_events, ed, stations, chan, fs)`: this function will be applied to machines that are in the Backoff_Timer state. When the machine invokes the backoff procedure, all the machines that are in Backoff_Timer state are going to schedule an event time_slot to decrease the backoff_value by one unit. When the backoff_value reaches zero then the machine will start the transmission;
- `Start_TX(list_events, ed, stations, chan, fs)`: first of all, it is checked if there is an internal collision, if there is none, the machine will obtain a TXOP. Then, if there is enough time to send the packet the transmission will start. The other machines will update their NAVs, for this MPDU plus another, if there is one, and if it is possible to send (according to the TXOP limit policy);
- `Start_RX(list_events, ed, stations, chan, fs)`: the buffer of the machine which is going to receive is set to RX state;
- `Stop_TX(list_events, ed, stations, chan, fs)`: the buffer of the machine which is transmitting is set to LISTEN_SIFS state;
- `Stop_RX(list_events, ed, stations, chan, fs)`: this function detects if there is a collision. If the transmission is successful then the receiver will send an ACK to confirm the data. If a collision is detected then the receiver will not send the ACK. Then, after an ACKTimeout time, the sender will invoke the backoff procedure;
- `Stop_ltn_S(list_events, ed, stations, chan, fs)`: it sets the state of the transmitter machine to WAIT_ACK;
- `RES_ok(list_events, ed, stations, chan, fs)`: if the machine receives an ACK then checks if there are more MPDUs or MSDUs to transmit, and if the TXOP will allow sending them;
- `RES_nok(list_events, ed, stations, chan, fs)`: it will perform the procedures inherent to a retransmission.

VI. Physical Layer

The physical layer specification used in this work is the IEEE 802.11a standard [3], [5] that defines an Orthogonal Frequency Division Multiplexing (OFDM) based PHY layer that operates in the 5 GHz frequency band, being able to achieve bit-rates as high as 54 Mbps. It defines 8 non-overlapping channels of 20 MHz each across the low and middle 5 GHz bands (5.15-5.35 GHz) and four extra channels across the high 5 GHz band (5.725-5.825 GHz). Each of these channels is divided into 52 sub-carriers, with each sub-carrier being approximately 300 kHz wide. In each channel 48 sub-carriers are used for data transmission, while the remaining 4 sub-carriers are used as pilots for coherent detection. A high data rate is achieved by combining 48 lower bit-rate data streams transmitted in parallel, each modulating a different sub-carrier. The parallel transmission of 52 modulation symbols, one in each sub-carrier, forms an OFDM symbol. These are created by an Inverse Fast Fourier Transform (IFFT), which combines the sub-carriers before transmission.

IEEE 802.11a specifies 8 different transmission modes, obtained with different combinations of modulation and convolutional code rate. Each transmission mode corresponds to a different bit-rate. Within an OFDM symbol the same transmission mode is used in all data sub-carriers. The IEEE 802.11a transmission modes are listed in Table III, together with the respective number of bytes transmitted in one OFDM symbol (Bytes-per-Symbol, BpS). The convolutional encoder always encodes data with code rate 1/2. The 3/4 and 2/3 codes are derived from the original 1/2 code by a technique called puncturing. Puncturing is a procedure for omitting some of the encoded bits in the transmitter, and inserting a dummy “zero” metric into the convolutional decoder at the receiver, in place of the omitted bits. This technique is a simpler and more efficient way of generating a higher code rate.

Table III - IEEE 802.11a PHY modes.

Mode	Modulation	Code Rate	Bit-rate	BpS
1	BPSK	1/2	6 Mbps	3
2	BPSK	3/4	9 Mbps	4.5
3	QPSK	1/2	12 Mbps	6
4	QPSK	3/4	18 Mbps	9
5	16-QAM	1/2	24 Mbps	12
6	16-QAM	3/4	36 Mbps	18
7	64-QAM	2/3	48 Mbps	24
8	64-QAM	3/4	54 Mbps	27

In indoor radio environments, signals coming from multiple indirect paths added to the direct path induce delay spread. This may be large enough to cause Inter-Symbol Interference (ISI) if high rates are used. OFDM counters this effect within each sub-carrier by transmitting data in parallel using lower-rate sub-carriers. However, ISI can be further reduced with the introduction of a guard interval in the beginning of each OFDM symbol. A guard interval longer than the maximum channel excess delay ensures that ISI is eliminated. With a guard interval of length T_g, a new block duration is obtained as $T'_b = T_g + T_b$. In IEEE 802.11a, $T_g = T_b/4 = 800$ ns, which means that $T'_b = 4$ns.

The cyclic prefix has two drawbacks worth to be mentioned. One of them is the overhead transmitted over the radio channel, which reduces the maximum data rate achievable on top of OFDM. The other one is that the cyclic prefix of duration T_g leads to a power loss, as the receiver only uses the energy received during time T_b, discarding the energy that corresponds to Tg. A power loss αg must thus be taken into account,

$$\alpha_g = \frac{T_b}{T'_b} \qquad (1)$$

Consequently, the ratio between the effective average symbol energy and the noise power spectral density, $\frac{E_{av}}{N_o}$, is related to the SINR in the following way:

$$\frac{E_{av}}{N_o} = \alpha_g \cdot \frac{E_s}{N_o} = \alpha_g \cdot SINR \qquad (2)$$

where $\frac{E_s}{N_0}$ is the ratio between the raw average symbol energy (i.e., including the energy of the cyclic prefix) and the noise power spectral density.

The PPDU frame format is depicted in Figure 5. It consists of a PLCP preamble of 12 OFDM symbols, followed by the PLCP SIGNAL field and a variable size DATA field.

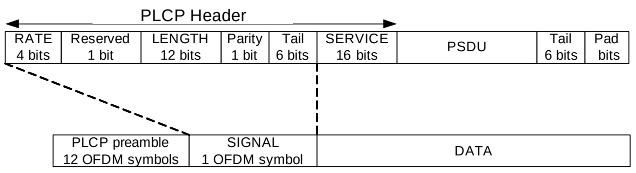

Fig. 5: Format of the IEEE 802.11a PPDU.

The preamble field is composed of 10 repetitions of a "short training sequence" (used for automatic gain control, diversity selection, timing acquisition, and coarse frequency acquisition in the receiver) and two repetitions of a "long training sequence" (used for channel estimation and fine frequency acquisition in the receiver), preceded by a guard interval. The PLCP header is composed of the SIGNAL field and the SERVICE field. The former constitutes an OFDM symbol and is transmitted with the lowest rate (BPSK-1/2), being composed of the payload RATE indicator, a reserved bit, the payload LENGTH, an even parity bit and six "zero" tail bits. The SERVICE field belongs to the DATA part and comprises seven "zero bits" used to synchronise the descrambler, followed by 9 bits reserved for future use. The DATA field also comprises the PSDU followed by 6 "zero" tail bits and a number of pad bits so that the total length of the DATA part corresponds to an integer number of OFDM symbols. The DATA part is encoded with the RATE specified in the SIGNAL field.

Link Adaptation

Link-adaptation can be based on several techniques:

1. Adaptation of frame size;
2. Automatic Repeat Request (ARQ);
3. Forward Error Correction (FEC);
4. Selection of coded modulation schemes with different bandwidth efficiency (bits/symbol) and thus different physical bit-rate. This technique is usually called rate-adaptation.

All these techniques incur on an overhead penalty, which must be weighted against the overall goodput and energy efficiency that can be achieved. There are several proposals of link-adaptation schemes presented in [6], [7], [8], [9].

Another rate-adaptation mechanism for IEEE 802.11a is proposed in [10], and further developed in [11]. In this algorithm, the sender chooses the bit-rate that achieves the highest goodput taking into account the channel state information (CSI) estimate and the number of transmission attempts. A PHY mode threshold table is calculated based on a conditional probability density function that models channel status variation. This rate-adaptation mechanism is compared with ARF by means of computer simulation, demonstrating its better performance. The algorithm considers no CSI feedback protocol. Instead the sender estimates the path loss at the receiver based on the received power of frames that come from the receiver (e.g., ACK and CTS frames) and the transmission power indication in the PPDU. Thus it is assumed that the path loss is the same in both directions, which may not be true due to multipath effects. The sender also estimates the noise at the receiver based on local noise power, which does not take into account the differences in terms of the experienced interference.

A requirement for the effectiveness of link-adaptation is to have a good estimate of the channel status. This can be accomplished with different techniques. In [6] and [12] the estimate of the packet error probability is obtained from the ratio between the number of failed transmissions and the total number of transmission attempts. The disadvantage of such approach is that it usually takes a significant number of transmission attempts to infer a good packet error probability estimate. When the channel is subject to fading effects, the reception quality may change faster than link-adaptation.

The Auto Rate Fallback (ARF) algorithm presented in [13] also bases its decisions on the number of missing acknowledgements. However, it is simpler as it does not seek to obtain an accurate packet error probability estimate. Instead, it lowers or raises the bit-rate based on a small number of losses or successful transmissions (2 or 10, respectively).

Another technique is based on the estimation of the received power and SINR experienced at the receiver [14], [15]. Such approaches are assumed in the link-adaptation techniques presented in [9], [10], [16], [11], where it is shown that link-adaptation based on SINR estimates is more efficient than ARF. However, they assume perfect estimates, which are usually difficult to obtain. In fact, the tolerance on the received power estimate considered in [17] ranges from ±5 dB to ±8 dB.

In the techniques proposed in [10] and [11], the error could even be magnified by the fact that the estimates assume that the path loss is symmetric and the noise experienced by the receiver is the same as that experienced by the sender.

In our simulations, we implement a procedure very similar to the one from [10]. We estimate the received power and SINR experienced at the receiver, based on the last reception of that machine. In this algorithm, the sender chooses the bit-rate that achieves the highest goodput taking into account the SINR estimate. A PHY mode threshold table (presented in next Section) is computed based on the simulations carried out. More details on the implementation of the physical layer in the simulator can be found in [18].

VII. Validation of PHY plus MAC

According to the IEEE 802.11 standard, the length of a MAC service data unit (MSDU) must be less than or equal to 2304 octets. The length of a MSDU shall be an equal even number of octets for all fragments except the last one of a fragment burst, which may be smaller. In this Section, we assume that the MSDUs to be transmitted are all 2304-octet long. Each MSDU might be either fragmented or not fragmented up to 10 equal-size MPDUs. The PHY and MAC characteristics of IEEE 802.11a OFDM are presented in Table IV.

Table IV: IEEE 802.11a OFDM PHY Characteristics.

Slot time	0.009 ms
ACK size	112 bit
SIFS	0.016 ms
DIFS	0.015 ms
RTS threshold	3000 bit
RTS size	160 bit
CTS size	112 bit
CW_{min}	31 slots
CW_{max}	1023
Collisions threshold	7
Fragmentation threshold	8000
Simulation time	10000 ms

Figure 6 presents results of goodput performance for different PHY modes without fragmentation while Figure 7 presents results with fragmentation. As expected, the highest rate PHY modes show better goodput performance for the highest SINR range, while the lowest rate PHY modes have improved goodput performance in the lowest SINR range.

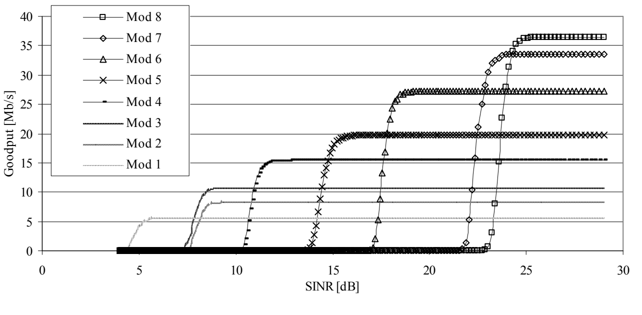

Fig. 6: Goodput *versus* SINR for the 8 different transmission modes without fragmentation.

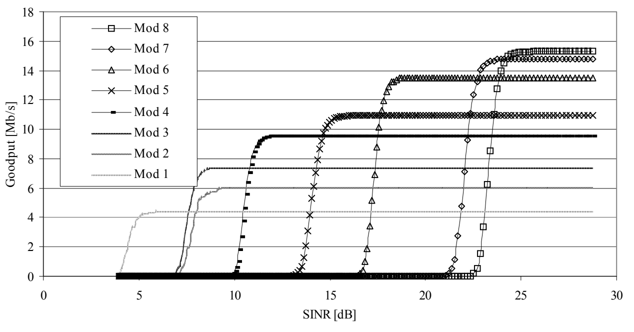

Fig. 7: Goodput *versus* SINR for the 8 different transmission modes with fragmentation.

One interesting observation is that the goodput performance of PHY mode 3 (QPSK modulation with rate ½ coding) is always higher than of PHY mode 2 (BPSK modulation with rate ¾ coding) under all SINR conditions. Although QPSK has worse error performance than BSPK, the worse performance of the ¾ rate convolutional code (compared to the ½ rate convolutional code) has a more dominating effect. Therefore, without the appropriate power control schemes, PHY mode 2 may not be a good choice if PHY mode 3 is present. From Figures 6 and 7 we can conclude that, for the same PHY mode, fragmentation decreases the maximum goodput due to the overheads; however, it improves the goodput performance at certain SINR range. The optimal combination of the PHY mode to achieve the highest goodput for different SINR conditions is presented in Figures 8 and 9, without and with fragmentation, respectively.

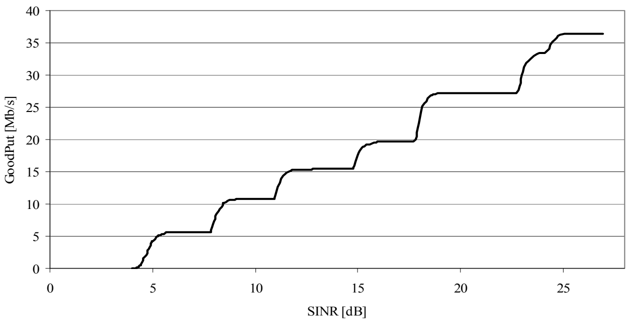

Fig. 8: Goodput *versus* SINR without fragmentation when an optimal transmission mode selection is used.

In this work, the values used for transmission mode selection are extracted from Figure 8. If we call SMT_k to the SINR threshold for mode k, the transmission mode m is selected as follows

m=8	if $SINR\_max \geq SMT_8$
m=7	if $SMT_8 > SINR\_max \geq SMT_7$
m=6	if $SMT_7 > SINR\_max \geq SMT_6$
m=5	if $SMT_6 > SINR\_max \geq SMT_5$
m=4	if $SMT_5 > SINR\_max \geq SMT_4$
m=3	if $SMT_4 > SINR\_max \geq SMT_3$
m=1	otherwise.

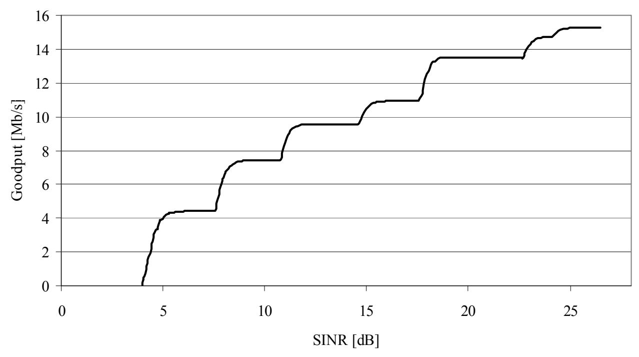

Fig. 9: Goodput *versus* SINR with fragmentation when an optimal transmission mode selection is used.

After selecting the transmission mode, the sender always uses the maximum transmission power. The PHY mode distribution in the square coverage zone is presented in Table V. PHY mode 2 does not have any percentage since the goodput performance of PHY mode 3 is always better than of PHY mode 2 under all SINR conditions.

Table V - IEEE 802.11a PHY modes.

Mode	Bit-rate	Distribution [%]
1	6 Mbps	28.22
2	9 Mbps	-
3	12 Mbps	21.51
4	18 Mbps	18.10
5	24 Mbps	9.27
6	36 Mbps	10.08
7	48 Mbps	1.83
8	54 Mbps	10.99

The bird's-eye view of cell area is presented in Figure 10.

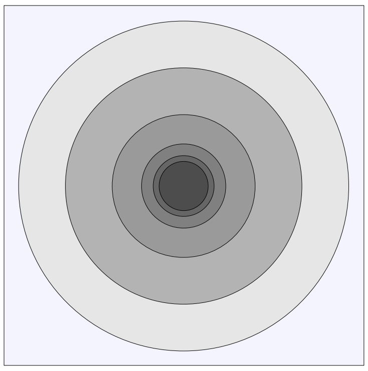

Fig. 10 – Bird's view of cell area.

VIII. System, Scenario and Assumptions

Lets consider a cellular WiFi system where each cell has a set N+1 IEEE 802.11e stations communicating through the same wireless channel. While station 0 is the Access Point or QoS Access Points (QAP), the other N wireless terminals or QoS stations (QSTA). The propagation time is assumed to be absorbed by some mechanisms of the IEEE 802.11. Each station has four buffers whose size depends on the kind of service being dealt in order to guarantee a given value for the goodput (payload of the packet).

Simulations have to be undertaken in order to get the best buffer size to be used. One of the approaches is to consider the buffer with infinite size. This buffer will be filled with a MAC Service Data Unit (MSDU) generator that characterises the service being dealt in the given buffer. If the MSDU is bigger than a fragmentation threshold, it will be fragmented. In order to cope with service quality the packet transmission follows the Enhanced Distributed Channel Access (EDCA) IEEE 802.11e MAC procedure. Due to collisions or interference a packet may not be correctly received. The number of collisions is represented by a global variable that checks whether there is more than one user transmitting simultaneously. The interference issues are addressed by using a radio propagation model. Each packet exits the buffer only after the reception of an acknowledgement, or if it has suffered more than a collision threshold.

In this first phase the users are assumed to be static, and are distributed uniformly in a square area of 2500 square meter.

The topology to be implemented consists of several wireless stations and an Access Point (AP). Three types of traffic sources were chosen, namely high priority voice, medium priority video and low priority FTP data. The traffic sources parameters are shown in Table VI. In this Table the Access Categories (AC) are also presented of each type of traffic.

Table VI – Traffic Parameters [3].

	Voice	Video	Background (FTP)
AC	VO	VI	BK
Packet size	1280 bit	10240 bit	18430 bit
Packet interval	20 ms	10 ms	12.5 ms

IX. Simulation Results

Our simulations consider one access point with several client machines. Results include packet delay, goodput, in bit per second, and channel utilization. A performance measure combining throughput and delay into single function will be proposed in a near future.

Results considering MAC and PHY layers are obtained, for each parameter, as the average of 20 simulations (each with different random seeds). Results with MAC layer alone are presented in [19].

Packet delay is the period of time between the moment at which the packet arrives to the buffer and the moment at which the packet is successfully transmitted. The results for stand alone voice (VO) uplink and downlink traffic, background (BK) and video (VI) are presented in Figure 11.

In terms of grade of service, from [20] the voice application supports delays up to 30 ms, the video application supports delays up to 300 ms, while the delay for the background applications can be up to 500 ms. Hence, our QAP supports up

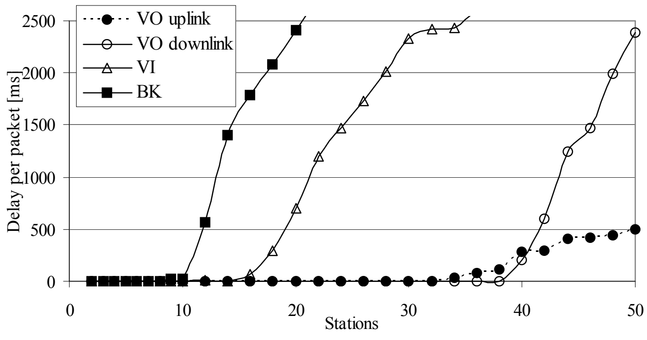

Fig. 11 - Delay as a function of the number of nodes for VO, VI, and BK applications.

to 40 voice users, 18 video users, and 11 background users with an appropriate degree of QoS. The system is limited by the downlink connection.

Another performance measure is the maximum goodput achievable for a given channel capacity. It is certain that a fraction of the channel capacity is used up in form of overhead, acknowledgments, retransmission, token delay, etc.

Channel capacity is the maximum possible data rate, i.e., the signalling rate on the physical channel. It is also known as the data rate or transmission rate, assumed to be variable, between 6 and 54 Mb/s. Goodput is the amount of "user data" that is carried by the wireless network. The results for goodput as function of the number of stations are presented in Figures 12 and 13. The maximum achieved goodput is 16 Mb/s.

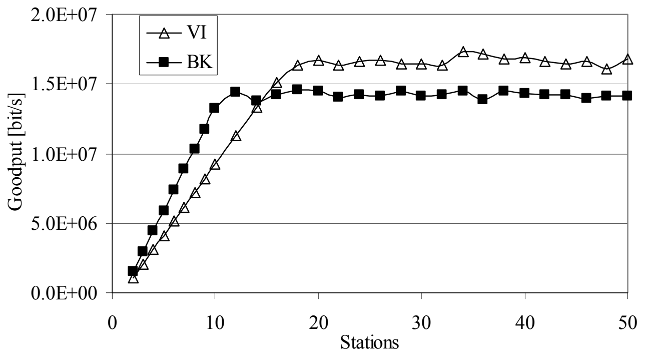

Fig. 12 - Goodput as a function of the number of nodes for VI, and BK applications.

Figure 13 presents the goodput for VO applications in both directions. When the number of stations is higher than 40 the goodput decreases; this is due to small CW size. The goodput in the downlink is equal to 64kb/s up to 38 stations. For more than 38 stations, the collisions start to occur very often and the goodput of each station decreases.

Due to the scarcity of wireless bandwidth, we also studied the medium utilization, and we computed the average data rate for the client stations. Since the distribution of users in the square is uniform, it is easy to compute the probability of a given transmission mode, being easy to compute the average data rate.

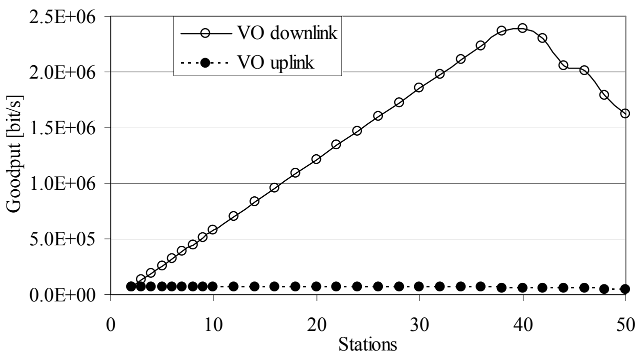

Fig. 13 - Goodput as a function of the number of nodes for VO, downlink and uplink directions.

The channel utilization is the ratio of goodput over the average data rate, and is presented in Figure 14 (as a function of the number of stations). The highest obtained value for utilization is around 80%, and is obtained for VI. The lowest one is obtained for VO traffic. This occurs because the packet size is much higher for VI than for VO.

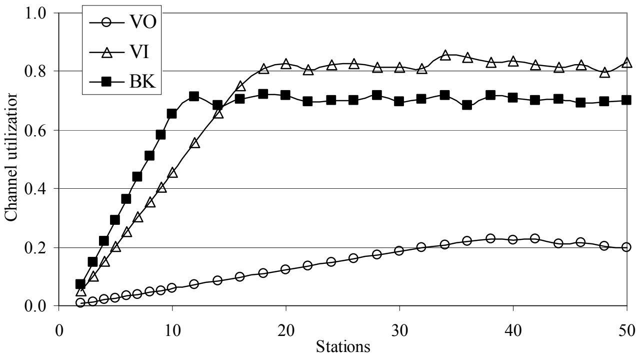

Fig. 14 - Channel utilization, as a function of the number of nodes, for VO, VI, and BK applications.

X. Conclusions and Future Work

Our IEEE 802.11e event driven simulator is a tool that allows for tuning-up several parameters like the ones related to how to use block acknowledgement, normal acknowledgement, and no acknowledgement policies. Policies that provide access to the medium, that ensure some degree of service, based on the channel SINR, delays, and bit error rate can be tested.

In the simulations, higher values of goodput are found for the VI and BK applications, mainly because the frames transmitted in these services are longer than the ones for the voice application but also the application data rate is higher. As a consequence, the number of supported stations of VI and BK is lower than the number of supported VO stations.

It should be noticed that the use of small CW for the VO access category may not be a very good idea since when the number of stations is higher than 38 the goodput starts to decrease due to a small CW size. By suffering successive collisions the retransmission threshold is overcome causing the increase of packet losses. As CW is longer for the BK traffic, there is a longer period for the random backoff generator to generate a backoff. The collisions do not exist in BK and VI applications because the data traffic is only present for downlink direction (from access point to stations). As the VO traffic is bidirectonal, collisions occur very frequently because, apart of the access point, all the stations are contending to access to the medium.

For future work, we suggest to extract results with mixtures of applications and to suggest block acknowledgement policies. Handover policies between APs will also be an objective to be fulfilled, where a scenario supporting more than one cell can be used. Later, our simulator will be integrated into the IT-MOTION simulator, and further work will be performed to optimise inter-working among different systems (e.g., WiFi, HSDPA, and WiMAX).

ACKNOWLEDGMENT

The authors would like to acknowledge the fruitful contributions given by António Grilo for the implementation of the PHY part of the simulator. This work was partially funded by CROSSNET (Portuguese Foundation for Science and Technology POSI and POSC projects with FEDER funding), by IST-UNITE, and by "Projecto de Re-equipamento Científico" REEQ/1201/EEI/ 2005 (a Portuguese Foundation for Science and Technology project).

REFERENCES

[1] IEEE Std. 802.11e; Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications, 2005.
[2] IEEE Std. 802.11; Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.
[3] Quiang Ni, "Performance Analysis and enhancements for IEEE 802.11e Wireless Networks," *IEEE Network*, Vol. 19, No. 4, July/Aug. 2005, pp. 21-27.
[4] Jonathan Rodriguez (editor), "Intermediate Specification of Algorithms for Cross-Layer Optimisation," IST-UNITE CEC deliverable 4-026906/IT/ DS/4.1.1, IST Central Office, Brussels, Belgium, 2007.
[5] IEEE Std. 802.11a; Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band, 1999.
[6] P. Lettieri, B. Srivastava, "Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency," in *Proc. of IEEE INFOCOM'98 –17th Conference on Computer Communications*, San Francisco, USA, Mar. 1998.
[7] D. Xu, B. Li, K. Nahrstedt, "QoS-Directed Error Control of Video Multicast in Wireless Networks," in *Proc. of IEEE ICCCN'99 - 8th IEEE International Conference on Computer Communications and Networks*, Boston-Natick, MA, USA, pp. 257-262, Oct. 1999.
[8] A. Majumdar, D. Sachs, I. Kozintsev, K. Ramchandran, "Multicast Unicast Real-Time Video Streaming over Wireless LANs," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 6, June 2002, pp.524-534.
[9] G. Holland, N. Vaidya, P. Bahl, "A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks," in *Proc. of ACM/IEEE Int. Conf. on Mobile Computing and Networking (MOBICOM'01)*, Rome, Italy, July 2001.
[10] D. Qijao, S. Choi, "Goodput enhancement of IEEE 802.11a wireless LAN via link adaptation," in *Proc. of IEEE ICC'2001 - Int. Conf. on Communications*, Helsinki, Finland, June 2001.
[11] D. Qijao, S. Choi, K. Shin, "Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs," *IEEE Transactions on Mobile Computing*, Vol. 1, no. 4, Oct.-Dec. 2002.
[12] M. Radimirsch, "An Algorithm to Combine Link Adaptation and Transmit Power Control in HIPERLAN Type 2," in *Proc. of PIMRC'2002 - Personal Indoor and Mobile Radio Conference 2002*, Sep. 2002.
[13] A. Kamerman, L. Monteban, "WaveLAN-II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal*, Summer 1997, pp. 118-133.
[14] K. Balachandran, S. Kadaba, S. Nanda, "Channel Quality Estimation and Rate Adaptation for Cellular Mobile Radio," *IEEE Journal on Selected Areas in Communications*, Vol. 17, no. 7, July 1999.
[15] K. Olszewski, "Channel Quality Measurement Method for 802.16ab PHY Layers," IEEE 802.16abc-01/45, Oct. 2001.
[16] D. Qijao, A. Soomro, K. Shin, "Energy-Efficient PCF Operation of IEEE 802.11a Wireless LAN," in *Proc. of IEEE INFOCOM'02 – 21st Conference on Computer Communications*, New York, New York, USA, June 2002.
[17] A. Myles, "IEEE 802.11h Draft Normative Proposal," IEEE 802.11-02/402r0, June 2002.
[18] A Grilo, *Quality of Service in IP-based WLANs*, Doctoral thesis, Instituto Superior Técnico, Technical University of Lisbon, Lisbon, Portugal, June. 2004.
[19] O. Cabral, A. Segarra, F. J. Velez, "Simulation of IEEE 802.11e in the context of interoperability," in *Proc. of ICWN´ 2007 - The 2007 International Conference of Wireless Networks (held in the context of WCE´ 2007 - World Congress on Engineering)*, London, U.K., July 2007.
[20] N. Anastácio, F. Merca, O. Cabral, F. J. Velez, "QoS Metrics for Cross-Layer Design and Network Planning for B3G Systems," in *Proc. of IEEE ISWCS'2006 - Third International Symposium on Wireless Communication Systems*, Valencia, Spain, Sep. 2006.