Computer Networks 53 (2009) 217–230

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

OpenCAPWAP: An open source CAPWAP implementation for the management and configuration of WiFi hot-spots

M. Bernaschi [a], F. Cacace [b], G. Iannello [b], M. Vellucci [b], L. Vollero [b,*]

[a] *Istituto Applicazioni del Calcolo-CNR, Rome, Italy*
[b] *Università Campus Bio-Medico, CIR – Centro Integrato di Ricerca, Via Alvaro del Portillo 21, 00128 Rome, Italy*

ARTICLE INFO

Article history:
Available online 4 October 2008

Keywords:
CAPWAP
Wireless networks
DTLS
Network management
QoS support
Performance evaluation

ABSTRACT

The Control And Provisioning of Wireless Access Points (CAPWAP) protocol is under definition within the IETF to enable an Access Controller (AC) to manage a collection of Wireless Termination Points (WTPs). CAPWAP aims at simplifying the deployment and control of large scale, possibly heterogeneous, wireless networks. We present the first open source implementation of the CAPWAP protocol along with early experimental results aiming at the assessment of its reliability and performances. The paper ends describing applications of the CAPWAP protocol to management and QoS scenarios and discussing benefits and technical issues concerning its use.

1. Introduction

The notion of Mobile Internet relies on scenarios where mobile users may have access anytime and anywhere to conventional and emerging Web applications requiring multimedia and interactive communications. Wireless access networks are the key element to implement these scenarios and much work has been done in recent years to develop and improve wireless technologies, including the introduction of support for mobility and Quality of Service. In this respect, special interest has been paid to wireless local area networks (WLANs) based on the IEEE 802.11 standard [4], that exhibits characteristics useful to pursue the goals of the Mobile Internet, including a large number of installations in a wide range of contexts.

Large deployments of Access Points pose serious problems in defining consistent strategies for their management, configuration and control. These issues forced network vendors to propose proprietary centralized solutions aiming at simplifying functionalities commonly requested by network administrators. All proposed solutions share two common elements: (i) they split functionalities that APs provide and (ii) they add more centralized functions for the monitoring and the remote control of the network. Splitting functionalities of APs allows the implementation of more flexible network infrastructures. Indeed, it allows centralizing the management of critical functions like channel selection, authentication and encryption. Whereas, leaving in the APs time-critical functions, like beacon generation and frames' acknowledgment, avoids the introduction of expensive components, like high performance interconnections, making proposed solutions competitive on the market. Moreover, the introduction of additional proprietary functions may further increase the level of flexibility in configuring and managing the network.

Recently, an Internet Engineering Task Force (IETF) Working Group, named Control and Provisioning of Wireless Access Points (CAPWAP), started its activity with the goal of defining standard solutions to such issues. In particular, the CAPWAP WG focused on problems like configuration, monitoring, control and management of large scale deployments of wireless networks in general, and of IEEE 802.11 networks [6] in particular, identifying a number

* Corresponding author. Tel.: +39 0622541 9631; fax: +39 0622541 9606.
E-mail address: vollero@ieee.org (L. Vollero).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.09.016

of functions that should be provided in such scenarios. The WG is currently working on the definition of a protocol, the CAPWAP protocol, capable of providing interoperability among devices supporting these functions.

In this paper we present our open source implementation of CAPWAP based on [1,2]. Section 2 describes the CAPWAP protocol and CAPWAP functionalities for the management of IEEE 802.11 networks. Section 3 describes the design and the implementation of OpenCAPWAP, our open source implementation [7]. Section 4 reports the results of some preliminary performance tests on the implementation, while Section 5 discusses scenarios where using CAPWAP may introduce improvements in network management and QoS support. Finally, Section 6 concludes the paper with the future perspectives of this activity.

2. The CAPWAP protocol

The increasing diffusion of Wireless Local Area Networks (WLANs), characterized by a number of simple Access Points, named in the following Wireless Termination Points (WTPs), and having a single point of control, called Access Controller (AC), suggested the definition of a standard protocol aiming at simplifying the deployment, management and control of such architectures. The Control And Provisioning of Wireless Access Points (CAPWAP, [1]) is a recent effort of IETF aiming at defining an interoperable protocol, enabling an AC to manage and control a collection of possibly heterogeneous WTPs.

Although originating from IEEE 802.11 architectures, the CAPWAP specifications aim at being independent of a specific WTP radio technology. The goals explicitly stated in current specifications of CAPWAP are the following:

(i) Centralize authentication and policy enforcement functions for a wireless network;
(ii) move processing away from the WTPs, leaving there only time critical functions and
(iii) provide a generic encapsulation and transport mechanism.

Currently the CAPWAP protocol defines the communication and general management functions among AC and WTPs. CAPWAP control and data messages are sent using UDP, over separate UDP ports, and secured using Datagram Transport Layer Security (DTLS, [3]). The CAPWAP protocol transport layer introduces resiliency using a request/response paradigm, where timeouts schedule retransmissions when a response does not follow a certain request. Two types of payload may be managed by this transport protocol: CAPWAP data messages and CAPWAP control messages. CAPWAP data messages encapsulate wireless frames forwarded by the WTP to the AC or by the AC to the WTP. CAPWAP control messages are CAPWAP management, control or monitoring messages exchanged among AC and WTPs.

CAPWAP also defines a discovery protocol for the automatic association of WTPs to the AC. As soon as the WTP is turned on, it sends a Discovery Request message (*Discovery* phase). Any AC receiving this request responds with a *Discovery Response* message. Hence, the WTP selects the AC, if any responded, with which it wants to interact and it establishes a DTLS session with it. Once the DTLS session has been established, both devices exchange their configurations and capabilities. The WTP is then ready to send and receive CAPWAP messages to/from the AC (*Run* phase).

2.1. Binding definitions and IEEE 802.11

The CAPWAP protocol has been designed to accommodate the needs of any wireless technology. Indeed, the specifications in [1] do not assume any specific message related to any specific technology. The implementation of CAPWAP for a specific wireless technology is called ”binding”. For instance, [2] defines the binding for IEEE 802.11 WLANs. Specifically, [2] devises two operational architectures: Split MAC (SM) and Local MAC (LM). The difference between the two architectures is based on where specific functionalities are implemented: in the WTP, in the AC or in both. Table 1 summarizes where specific functionalities may be implemented in the two cases.

In both Split MAC and Local MAC architectures the CAPWAP functionalities reside on the AC. In a Split MAC architecture, the Distribution and the Integration services reside on the AC, whereas in a Local MAC architecture, they reside on the WTP. The Distribution service, [4], enables the Medium Access Control (MAC) layer to transport MAC service data units (MSDUs) between stations, when those stations cannot communicate directly over a single instance of the wireless medium (WM). The Integration service, [4], enables the delivery of MSDUs between IEEE 802.11 and non-IEEE 802.11 devices. In Split MAC architectures, hence, all user data are tunneled between the AC and the WTP, while in Local MAC architectures not all station-generated frames are forwarded to the AC.

In both architectures, real-time IEEE 802.11 services, including beacon generation and probe responses, are implemented on the WTP. Remaining management frames are supported on the AC, when Split MAC is adopted, while

Table 1
802.11 functions mapping for Split MAC (SM) and Local MAC (LM) architectures.

Function	Location	
	SM	LM
Distribution service	AC	WTP/AC
Integration service	AC	WTP
Beacon generation	WTP	WTP
Probe response generation	WTP	WTP
Power management/ packet buffering	WTP	WTP
Fragmentation/ Defragmentation	WTP/AC	WTP
Association/disassociation/ reassociation	AC	WTP/AC
IEEE 802.11 QoS		
Classifying	AC	WTP
Scheduling	WTP/AC	WTP
Queuing	WTP	WTP
IEEE 802.11 RSN		
IEEE 802.1X/EAP	AC	AC
RSNA key management	AC	AC
IEEE 802.11 Encryption/ decryption	WTP/AC	WTP

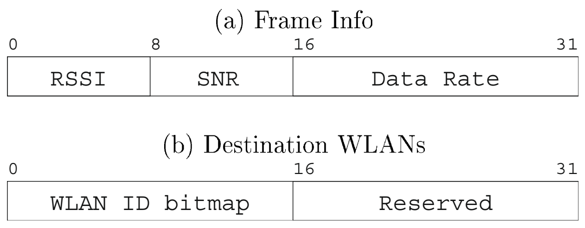


Fig. 1. Optional wireless specific information. (a) Frame info. (b) Destination WLANs.

they are supported on the WTP and, in some cases, forwarded to the AC, when Local MAC is in use. For instance, since the AC may need to be aware of mobility events within WTPs, the WTP has to forward to the AC all the Association Request messages it receives.

When needed, the transport of IEEE 802.11 frames is realized using the CAPWAP data message encapsulation rules. IEEE 802.11 header and payload are encapsulated, while the FCS checksum (see [4]) is excluded. An optional CAPWAP header may be added. When the frame is encapsulated by the WTP, the optional header is that depicted in Fig. 1a and it gathers information on the Received Signal Strength Information (RSSI), the Signal-to-Noise-Ratio (SNR) and the data rate used by the sending station. When the frame is encapsulated by the AC, the optional header is that depicted in Fig. 1b and it informs the WTP about the WLAN ID to be used when sending the frame.

CAPWAP control messages for IEEE 802.11 are enriched by the introduction of specific information for radio control, management and monitoring. Moreover, special control frames for the Quality of Service management using Wireless Multimedia (WMM) extensions are defined in [2]. WMM is a WiFi Alliance interoperability certification, based on the IEEE 802.11e standard [5].[1] WMM prioritizes traffic according to four Access Categories (ACs): (VO) Voice, (VI) Video, (BE) Best Effort and (BK) Background. The quality of each Access Category is controlled by five MAC layer parameters: $AIFS$, CW_{min}, CW_{max}, $TXOP_{limit}$ and q_{depth}. To be more specific, each AC within each station implements a slotted CSMA/CA channel access protocol. The $AIFS$ defines the fixed waiting time for carrier sensing that a specific AC has to use before any attempt to transmit. The CW_{min} and CW_{max} control the exponential backoff algorithm implemented by each AC. They aim at mitigating network congestion. The $TXOP_{limit}$ is a maximum channel holding time. When an AC transmits successfully, that AC can transmit other frames without contending with other ACs and STAs until this time has expired. The q_{depth} parameter is not directly specified by IEEE 802.11e, but it represents the maximum number of packet that may be buffered in each AC queue. It can be used to control the trade-off between delay and loss in the sending queue. CAPWAP allows the configuration of four of such parameters with the WTP Quality of Service message depicted in Fig. 2.

[1] WMM extensions are a subset of EDCA, the Enhanced Distributed Channel Access mechanism defined in the IEEE 802.11e standard.

3. Protocol implementation

There are several other projects aimed at developing an open source implementation of the CAPWAP protocol (*e.g.*, [10]). However, to the best of our knowledge, none of them has released code yet or, as in the case of [10], only a limited and outdated implementation is provided.

The state diagram reported in Fig. 3 represents the lifecycle of a WTP-AC session with a Finite State Machine (FSM), as defined in the protocol specification [1]. Use of DTLS by the CAPWAP protocol results in the juxtaposition of two nominally separate yet tightly bound state machines. The DTLS and CAPWAP state machines are coupled through an API consisting of commands and notifications. Certain transitions in the DTLS state machine are triggered by commands from the CAPWAP state machine, while certain transitions in the CAPWAP state machine are triggered by notifications from the DTLS state machine.

The same FSM is defined for both the WTP and the AC, although some states and transitions are implemented only on either the WTP or the AC. A complete description of the FSM is reported in the Section 2.3 of [1]. A normal session for the WTP begins with the `Start` state. After the initialization is complete, the WTP enters the `Idle` state. The WTP then proceeds to the `Discovery` phase to find an AC to connect with. The `Discovery` state is skipped if the WTP is instructed to try to connect to a fixed AC. In both cases the WTP moves to the `DTLS Setup` phase, to establish a secure DTLS connection with the AC. A transition from `DTLS Setup` to `Authorize` occurs when the DTLS session is being established, and the DTLS stack needs authorization to proceed with the session establishment. If all goes well, the WTP enters the `DTLS Connect` state, and when the DTLS connection is established the `Join` state is reached. In the `Join` state the AC and the WTP start communicating with each other and the CAPWAP session can begin. Upon the reception of a successful *Join Response* message from the AC, the WTP is instructed to either download new executable firmware (in which case it enters the `Image Data` state, and the session ends because the device is reset) or it is configured through the appropriate messages sent by the AC (in the `Configure` state). If the success of the configuration process is confirmed by messages exchanged in the `Data Check` state, the WTP eventually reaches its normal operating state, represented in the FSM by the `Run` state. There are a number of possible events that may occur in the `Run` state, the most relevant is the *Configuration Update Request* by which the AC requires the WTP to modify its configuration.

The other states illustrated in the FSM have intuitive meanings. The `Sulking` state is somehow a special case that is defined for situations in which the WTP can not communicate with an AC. The WTP exits from the `Sulking` state and starts a new `Discovery` phase (after a temporary transition to the `Idle` state) when the *SilentInterval* timer expires.

Our implementation of the CAPWAP protocol consists of two Linux applications, one running on the ACs and one running on the WTPs. Each WTP acts as a client of an AC. Each AC manages both the communication with WTPs

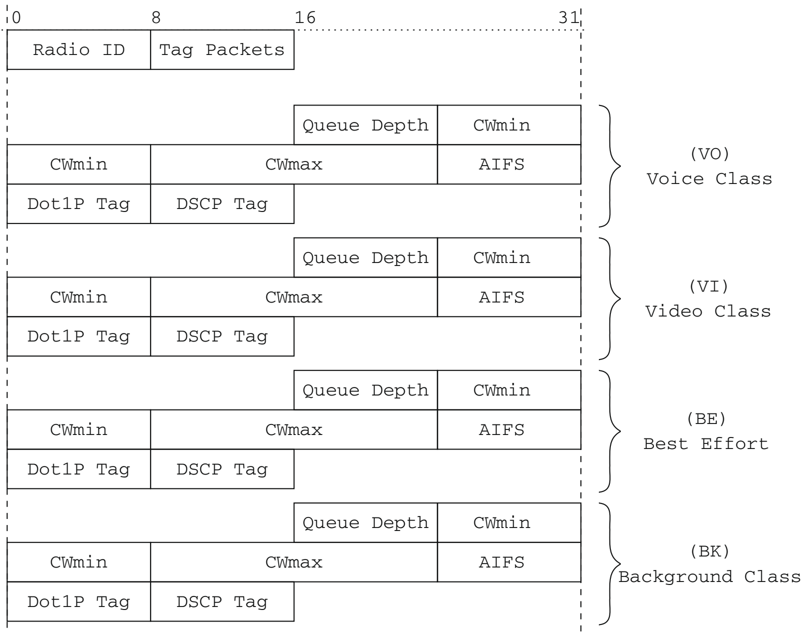


Fig. 2. Quality of Service message.

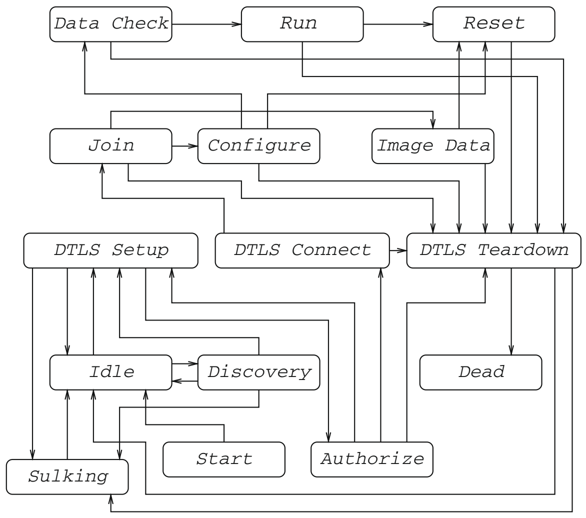


Fig. 3. CAPWAP Finite State Machine.

already registered and new requests sent by WTPs not-yet registered.

For both applications we followed a multi-threaded programming model which represents a reasonable trade-off between modularity and efficiency. We describe the structure of our implementation starting from the AC component. In the beginning, there is a single *receiver* thread that is in charge of receiving any packet arriving from the WTPs. When a packet arrives, the *receiver* thread extracts the source address and checks if it corresponds to an already established session. In this case the packet is queued to a list of pending requests associated with that session. If it is a request coming from an unknown WTP, the *receiver* thread analyzes the message to determine if it is a *Discovery Request*. In this case, it sends in reply a *Discovery Response* message. The reason why it is the *receiver* thread that directly manages the *Discovery Request* messages is that it is not worth starting a new thread unless the WTP continues the registration procedure (a WTP may send *Discovery Request* messages in broadcast so that more than one AC receives them). If the message is a *Client Hello*, that is the first message sent by a WTP when it wants to establish a DTLS session with the AC, a new *session manager* thread is created. The *session manager* thread replies to the message and manages any future request (including the sending of possible responses) coming from the same WTP. The flow of messages among WTPs and AC threads is shown in Fig. 4.

The number of threads used on the WTP is always limited to three. During the *Discovery* phase, there is only one *principal* thread in charge of the communication with potential ACs. After receiving the *Discovery Response* messages and having chosen one of the ACs, another *receiver* thread is created. This thread sends to the selected AC the *Client Hello* message to establish the DTLS session, but this is the only message sent by this thread. All other requests are sent by the *principal* thread which also manages the responses received by the AC. The need for an independent thread dedicated exclusively to the reception of packets arises because during the *Run* phase the AC may send its own requests to a WTP. The *receiver* thread shares packets with the *principal* thread using a list as in the AC case. The third, *receiver-from-STA*, thread is required for intercepting packets sent by the STAs connected to the WTP. This operation is necessary because some packets, in particular the *Association Requests*, sent by the STAs may have to be forwarded to the AC (see Section 3.1). The relation

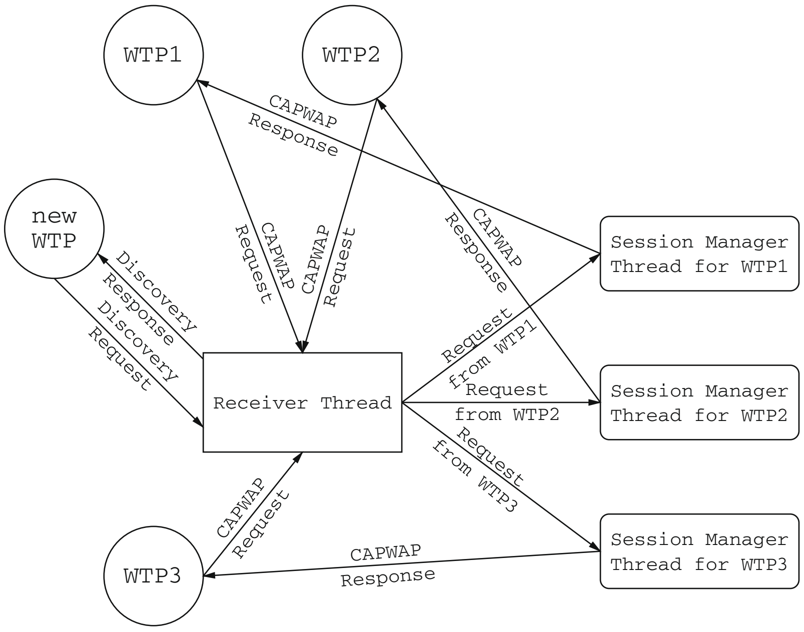


Fig. 4. Flow of messages among WTPs and AC threads.

among the three threads of the WTP application is shown in Fig. 5.

The CAPWAP protocol, like other existing control and management protocols (*e.g.*, the SNMP), relies on UDP as transport protocol. Since UDP does not guarantee reliable communications, we implemented, as required by the CAPWAP protocol, a retransmission mechanism for requests whose response does not arrive within a specified timeout. As to the security requirements imposed by the CAPWAP protocol, our implementation resorts to the OpenSSL implementation of the DTLS protocol [8] to fulfill them. In case of timeouts and consequent re-transmission of packets it is necessary to cipher them again to avoid that the *receiver* could assume that a *replay attack* is in progress.

All code is organized as a set of modules to make easier the replacement of single components. This is a fundamental requirement for the implementation of a protocol which is not completely and definitely specified yet. From the very beginning, we included a logging mechanism with multiple levels of verbosity to ease troubleshooting activities.

At the current stage of development we assume that each WTP has a single radio component. Moreover both Data and Control CAPWAP messages use the same port

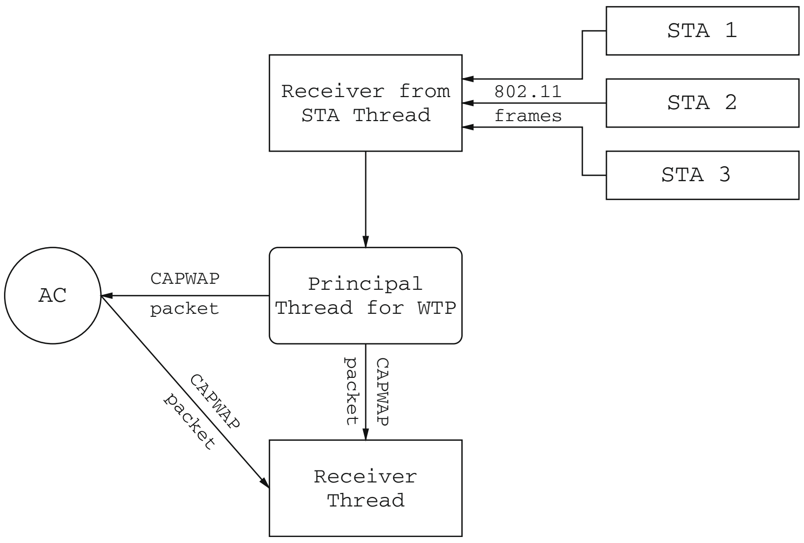


Fig. 5. The relation among threads of the WTP application.

although the protocol expects that, in the *Data Check* phase, WTP and AC negotiate a new communication channel. Actually, such choice is due mainly to the uncertainty that it is still present within the CAPWAP working group about the possible benefits of this feature. Finally, in the *Run* state all the CAPWAP messages are currently managed. Among the others, the *Configuration Update Request* is fully supported, allowing for the management of IEEE 802.11 WTP Quality of Service messages that have been used to test the implementation.

Two components of the WTP application are, to some extent, platform dependent. The first is the module in charge of "capturing" the 802.11 frames. The second is the module in charge of modifying one or more parameters of the WTP (*e.g.*, the *AIFS*) when the AC sends a *Configuration Update Request*. Currently, our platform for development and testing (see Section 4) is based on a wireless card equipped with an Atheros chip and the open source device driver MADWiFi. The structure of the interface is reported in Fig. 6.

With this organization, some of the most critical real-time functionalities required by the 802.11 protocol, like the generation of RTS/CTS/ACK frame controls are implemented directly in firmware. On top of the firmware, a *Hardware abstraction layer*, distributed in binary form, is used to separate the device driver from the card and the firmware itself. For any physical interface, the MADWiFi driver allows defining multiple virtual interfaces. Each virtual interface may work in one of five possible operational modes: *Station, Adhoc, Access Point, Wireless Distribution System* and *Monitor*. Usually, a WTP requires a single virtual interface working in *Access Point* mode and a copy of all packets received at device driver level can be obtained by using the *packet socket* API. However, to have access to *all* 802.11 frames, including those managed at firmware level, we define an additional virtual interface working in *Monitor* mode and use *packet sockets* on it. This solution, although not general, is portable to any hardware platform for WTPs that supports multiple virtual interfaces and Linux as operating environment.

For the change of WTP parameters, we make use of both the classic `ioctl` and the more specific `subioctl` primitives available in any Linux driver for wireless devices that supports the so-called *Wireless Extension* API [9].

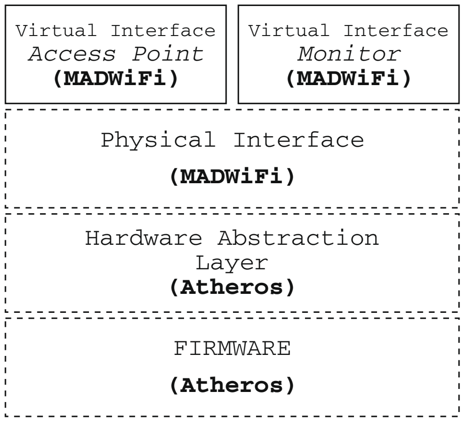


Fig. 6. Organization of a wireless interface based on a Atheros chip and managed by the MADWiFi driver.

3.1. Local MAC vs. Split MAC

As described in Section 2, the CAPWAP protocol defines two modes of operation: Split and Local MAC. Hereafter, we provide some additional information about the difference between the two architectures and explain why we currently support only the Local MAC mode.

The Local MAC mode of operation allows for the data frames to be either locally bridged, or tunneled as 802.3 frames. In either case the Layer 2 wireless management frames are processed locally by the WTP, and then forwarded to the AC. The integration service exists on the WTP, while the distribution service may reside on either the WTP or the AC. When it resides on the AC, frames generated by end-user wireless devices are not forwarded to the AC in their native format, but encapsulated as 802.3 frames.

In Split MAC mode all Layer 2 wireless data and management frames are encapsulated via the CAPWAP protocol and exchanged between the AC and the WTP. The wireless frames received from an end-user device are directly encapsulated by the WTP and forwarded to the AC. Moreover the distribution and integration services reside on the AC, and therefore all user data are tunneled between the WTP and the AC. Real-time IEEE 802.11 services, including the Beacon and Probe Response frames, are handled on the WTP. All remaining IEEE 802.11 MAC management frames are supported on the AC, including the Association Request frame which allows the AC to be involved in the access policy enforcement portion of the IEEE 802.11 protocol.

With respect to the IEEE 802.11 QoS functionalities, the Local MAC architecture requires the WTP to support them. In Split MAC mode, the queuing function is also in charge of the WTP whereas the real-time scheduling can be managed either by the WTP or by the AC.

For both architectures, the IEEE 802.1X and RSNA Key Management functions reside in the AC. Therefore, the WTP needs to forward all IEEE 802.1X/-RSNA Key Management frames to the AC and forward the corresponding responses to the end-user device.

In summary, in Local MAC mode the whole 802.11 MAC resides on the WTPs, including all the 802.11 management and control frame processing for the STAs whereas in Split MAC mode only real-time MAC functions should be managed by WTPs. Such distinction appears quite sharp but actually in the IEEE 802.11 specifications there is no a clear definition of which 802.11 MAC functions are considered real-time, so that each vendor may decide to use his own interpretation. An unpleasant side effect of this situation is that the implementation of a Split MAC architecture is possible, currently, only for the equipment of few vendors. The major problem, as already mentioned above for the access to all 802.11 frames received by a WTP, is the lack of standard interfaces that makes most operations on a WTP platform-dependent. However the difficulties in this case are more serious since on many WTPs there is no way at all (not even proprietary) to prevent them from carrying out tasks that in a Split MAC architecture should be in charge of the AC. As a consequence, we expect to support the Split MAC mode on a specific platform in the near future as a proof of concept, but full support for heterogeneous environments will be possible, probably, only when vendors understand the issue and address it.

4. Experimental results

In this section we report some preliminary experimental results about the performance of our CAPWAP prototype implementation. The testbed included one AC and five WTPs. All the machines were HP tc4200 tablet PC equipped with a Pentium M 1.73 GHz processor and Linux Ubuntu OS, kernel version 2.6.15-26-386. The AC was connected to the WTPs through a switch on a 100 Mb/s 802.3 link. The WTPs were all equipped with a wireless PCI NIC, a NETGEAR GW511T using an Atheros chip, capable of working in master mode. We chose this NIC, that supports a subset of the IEEE 802.11e protocol, since the MADWiFi 0.9.2.1 open source driver for this card allows to dynamically adjust the QoS MAC parameters (CW_{min}, CW_{max}, $AIFS$, and $TXOP_{limit}$) for the WTPs and the STAs. We could thus experiment the remote control of the QoS settings of the wireless cell from the AC through the CAPWAP protocol. Vendor specific features of the wireless cards were disabled.

With this testbed we performed three sets of measurements:

(i) The delay for the association between WTP and AC. This is the time elapsed between the WTP request and the AC response,
(ii) The delay for the configuration of IEEE 802.11e QoS MAC parameters on the WTP (the time elapsed between the AC request and the WTP response),
(iii) The delay for the *echo* request from the WTP to the AC (the time elapsed between the WTP request and the AC response).

In all cases the measures refer to the round-trip time of the requests. Since the contribution of the Ethernet link to the delay is negligible, the delay measured corresponds to the sum of the time needed to generate the request on the sender side *plus* the time needed to compute the answer on the *receiver* side. In the second and third case we measured the performance for both a single request and a flood of requests, whereas in the third case we measured also the delay for a growing number of WTP. The association delay experimented by the WTPs across 15 experiments was 86.51 ms, with a 90% confidence interval of 4.81 ms.

Table 2 reports the measurements for the configuration case. In the first row the average delay (across 15 experiments) is reported. The second row is the “flood” case, with the AC sending a new configuration request immediately after the WTP response. Similarly, Table 3 reports the delays for the “*echo*” requests when the requests are immediately sent by the WTPs after each response. This table shows the AC performance with requests from a growing number of WTPs. The last column is the average of the total number of requests received by the AC in 1 s.

Table 2
Configuration delay between WTP and AC.

Requests per s	Delay (ms)	90% Confidence int. (ms)
1	5.78	0.83
117.07	8.56	0.39

Table 3
Echo delay between WTP and AC.

WTPs	Messages per second	Delay (ms)	Total messages
1	604.55	1.65	604.55
2	619.20	1.62	1238.40
3	446.47	2.24	1339.71
4	350.24	2.85	1400.97
5	286.30	3.48	1431.50

5. Discussion

CAPWAP is a valuable protocol enabling smart strategies for the management of Hot-Spots. In this section we present a simple management architecture based on CAPWAP and we show its using in solving typical configuration problems of Hot-Spots. In particular, we focus on: (i) the Frequency Planning problem, (ii) the Load Balancing problem and (iii) the Automatic Adaptation of WMM Parameters.

(i) Frequency Planning deals with the problem of assigning communication channels to WTPs. An optimal or sub-optimal planning is fundamental in order to reduce cross-WTP interferences and increase the resources for users.
(ii) Load Balancing deals with the problem of distributing users to WTPs. The goal is to balance the resource allocation in order to avoid congestion and improve performance.
(iii) Automatic Adaptation of WMM Parameters deals with the problem of tuning WMM parameters in order to fairly distribute the resources or to introduce resource guarantees.

In the following subsections we discuss first the architecture, then the three configuration problems presenting simple configuration solutions. Simulation analysis and experiments are used to evaluate quantitatively the impact of the proposed solutions.

5.1. Management architecture

In the following sections we consider a simple management architecture. The architecture, depicted in Fig. 7, is based on CAPWAP functionalities and messages. The architecture does not require a specific operation mode, being based on functionalities that may be present into the AC for both Local and Split MAC. In the architecture the Hot-Spot Manager, i.e. the agent implementing the management strategies, is co-located with the AC and the communication among it and the AC is performed through an API interface.[2]

[2] Other communications strategies may be devised, but the analysis and the comparison of different communication paradigms is out the scope of the analysis done in this section.

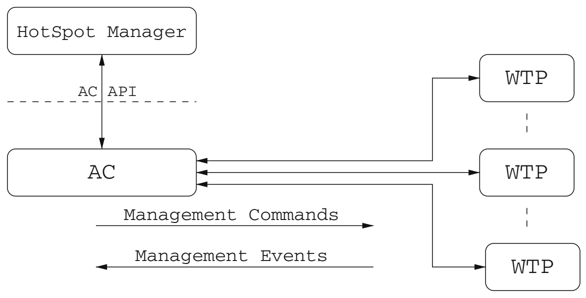


Fig. 7. Management architecture using CAPWAP functionalities.

The API allows the Hot-Spot manager to send configuration commands to the AC. The Hot-Spot manager uses the same API to collect information and statistics. The set of messages and their mapping on CAPWAP functionalities is described in details in each of the following subsections, with proposed solutions and results.

5.2. Frequency planning

IEEE 802.11b defines eleven[3] transmission channels for wireless communication, but at most three of those channels can be used simultaneously without cross-interference (namely Channel 1, Channel 6 and Channel 11). When configuring or upgrading large deployments of WTPs, the configuration of frequencies used by every WTP may be a major problem in optimizing overall network performance. Optimal or sub-optimal frequency reuse is desirable to reduce interference among adjacent cells, but it is not always possible. In this context several works proposed solutions to the optimal allocation of frequencies to WTPs. For instance, in [11] an architecture based on CAPWAP is proposed.

Based on assumptions similar to those presented in [11], we present a simple management strategy for the sub-optimal solution of the Frequency Planning problem. We consider a flat square area whose side is set to 2 km (Fig. 8 shows a simulation snapshot for 100 WTPs). This area is populated with a growing number of WTPs, where each WTP uses a communication channel in the set {Channel 1, Channel 6, Channel 11}. The position of each WTP is randomly located in the area. When placing a new WTP, the position is obtained picking within a uniform distribution the first place not closer than 100 m to previously installed WTPs. We consider two configuration strategies for the communication channels:

(i) RANDOM: Each WTP uses a channel randomly chosen in the set of allowed configuration channels.
(ii) CLOSEST: When a new WTP is placed, the AC monitors the communication activities of neighbors WTPs. Based on the power of beacons from WTPs in its range, the AC identifies the channel to use as the channel orthogonal to those of the two closest WTPs.

[3] We consider the US regulations. In Europe the number of channels is thirteen. Moreover the maximum number of non-overlapping channels is three also in Europe, leaving valid all the results present in this and the followings subsections.

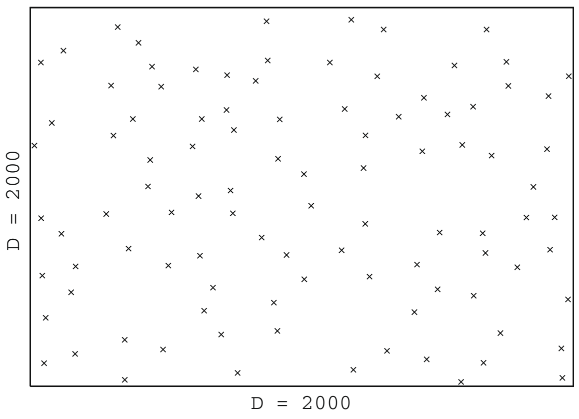


Fig. 8. WTPs spatial configuration in the Frequency Planning scenario (simulation snapshot).

The RANDOM configuration represents a completely blind configuration strategy, using only the IEEE 802.11 Direct Sequence control message of CAPWAP to force a specific WTP to use a given channel. Conversely, the CLOSEST strategy requires a more complex interaction between the AC and the managed Hot-Spot, involving monitoring functionalities based on IEEE 802.11 Information Element and the IEEE 802.11 Direct Sequence Control messages of CAPWAP.

Once installed and configured, we introduce in the previously presented scenario an increasing number of users (STAs). The position of each STA is random with uniform distribution in the area. Each user communicates using the closest WTP in the area with the constraint to be closer than 300 m to it, otherwise it does not communicate at all. To evaluate the two configuration strategies in this scenario, we consider the following metric: for each WTP and for each STA, we measure the number of WTPs and the number of STAs interfering with it (N_{int}), where we say that:

(i) A WTP interferes with another WTP when they both use the same communication channel and their distance is less than 300 m,
(ii) A STA interferes with a WTP when they both use the same communication channel, their distance is less than 300 m and the STA is associated with another WTP,
(iii) A WTP interferes with a STA if the STA interferes with the WTP,
(iv) A STA interferes with another STA when both use the same communication channel, their distance is less than 300 m but they are associated with different WTPs.

Based on this metric we consider two particular scenarios with $N_{wtp} = 50$ and $N_{wtp} = 100$ and we evaluate the average number of interfering WTPs and STAs, for an increasing number of STAs in the Hot-Spot. Fig. 9 shows the interference suffered by a generic STA. For each experiment, that has been repeated 20 times, average values with 95% confidence intervals are shown. The experiments prove that the CLOSEST strategy consistently reduces the

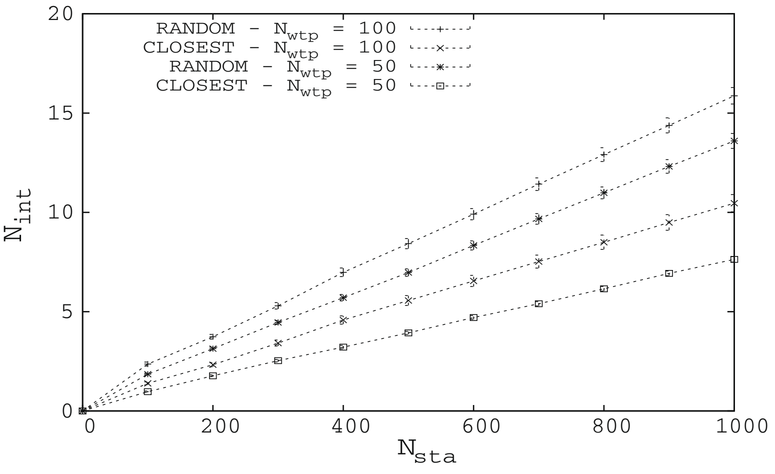


Fig. 9. Frequency Planning results (Interference suffered by the generic STA.).

number of interfering groups by almost 50%. There is also a small decrease of WTP–WTP cross-interference (not shown in the figure).

Fig. 10 shows the impact of the CLOSEST strategy in the distribution of interfering WTPs and STAs for a configuration composed of 100 WTPs and 1000 STAs (similar results are obtained with different simulations or analyzing the same scenario with different seeds). The plots show the probability of finding an interfering device closer than a given distance. The gain is particularly evident in case of WTP-over-WTP interference. Since down-link flows are usually dominant in wireless communications, this gain may translate in a significant performance improvement in typical use cases.

5.3. Load balancing

Several studies showed that in Hot-Spot scenarios, users are often unevenly distributed in space and, hence, the number of associated users may vary widely from WTP to WTP. This may translate in an uneven load distribution which can severely degrade the quality of services that users access especially if some WTPs result highly congested. Load Balancing aims at mitigating this problem by forcing some users to roam toward less loaded neighbor WTPs, avoiding congestion.

IEEE 802.11 does not provide any explicit mechanism to control the association of users to a specific WTP. In order to implement Load Balancing strategies, hence, a set of extensions to the standard has been proposed in the literature. Cell Breathing [12] gained a lot of consensus due to its simple and standard implementation. Cell Breathing is based on the assumption that stations choose to associate with the WTP from which they receive beacons with the highest power. Hence, Cell Breathing tries to redistribute users association varying the power that each WTP uses for transmitting beacons, by increasing the power of beacons sent by low loaded WTPs, and by reducing that of beacons generated by highly loaded WTPs.

The Hot-Spot Manager can be used also to implement Load Balancing strategies based on Cell Breathing. Those strategies can be implemented by using monitoring functionalities and commands for setting transmission power provided by CAPWAP. A simple example is presented hereafter. We consider the same scenario used for the Frequency Planning problem, but with a smaller number of WTPs ($N_{wtp} = 4$ and $N_{wtp} = 8$) placed in a smaller square area (300 m side). Both WTPs and STAs are placed following the same rules as above.

To assess the impact of Load Balancing, we consider two different configuration strategies: a FIXED distribution of power level to WTPs and a LOAD BALANCING driven distribution. In the FIXED strategy, each WTP sends beacons with the highest available power level. In the LOAD BALANCING strategy we follow a Cell Breathing approach to control the number of STAs associated to each WTP.[4] We assume that each WTP can transmit beacons using one out of three possible power levels. The three power levels (Pt_{300}, Pt_{275} and Pt_{250}) enable a successful receipt by stations at a distance from the WTP of, respectively, 300 m, 275 m and 250 m.

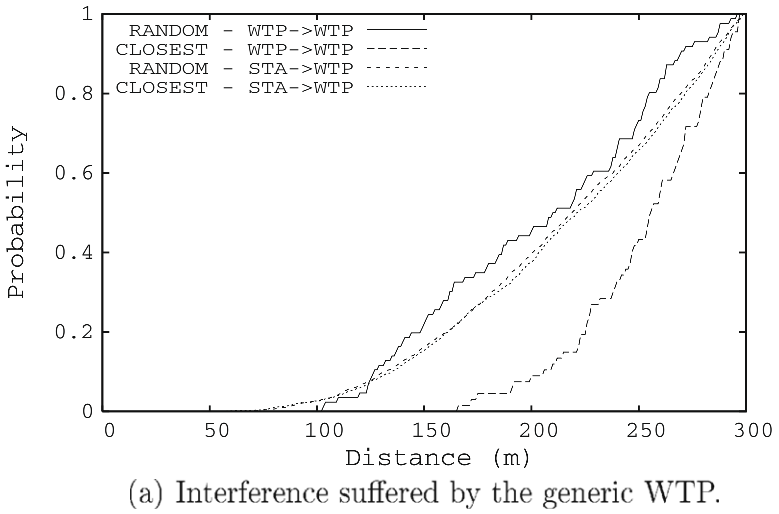
When LOAD BALANCING is used, the power levels are configured to meet two requirements:

(i) Maximize the number of STAs associated.
(ii) Minimize the maximum number of STAs associated to a single WTP (N_{max}).

In applying the LOAD BALANCING strategy, we assume that the Hot-Spot Manager knows the position of each STA.[5] Hence, it applies an exhaustive search among all possible configurations, choosing the one with the lowest maximum number of STAs associated to a single WTP. The exchange of CAPWAP messages is shown in Fig. 11.

[4] This congestion metric is agnostic of the load that each station may attempt to transmit.

[5] This assumption is based on the observation that WTPs in highly populated networks allow for a quite precise localization of users. Moreover, the localization of STAs can be implemented using monitoring functionalities based on the IEEE 802.11 Information Element message of CAPWAP.



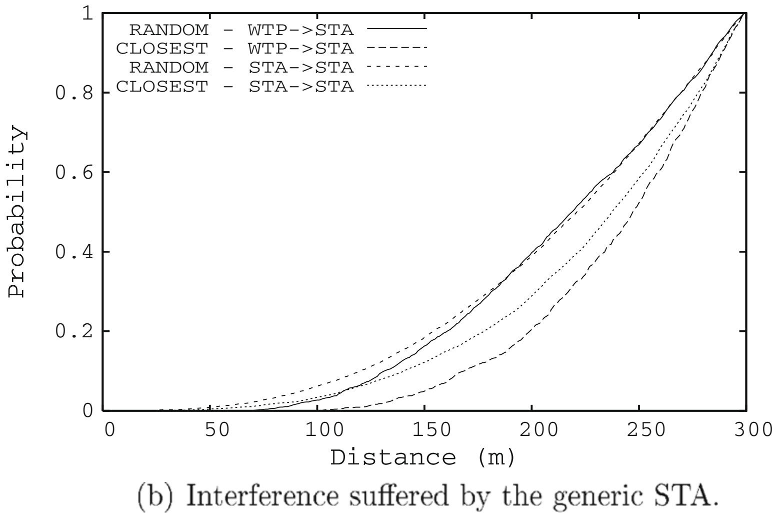


Fig. 10. Frequency Planning distance probability ($N_{wtp} = 100, N_{sta} = 1000$).

The data frames forwarded from the WTP to the AC include the optional "IEEE 802.1 Frame Info" CAPWAP header field. This field is used to include radio and PHY specific information associated with the frame. This information is used as an input for the LOAD BALANCING algorithm. The corresponding configuration is then enforced by using the IEEE 802.11 Tx Power message element within a CAPWAP *Configuration Update Request* from the AC to the WTP.

A comparison among the two power allocation strategies is presented in Fig. 12 for a number of STAs ranging from 20 to 200. For each network configuration we performed 20 trials and the results are shown as average values with 95% confidence intervals. In all the cases, as expected, the LOAD BALANCING strategy outperforms the FIXED configuration strategy.

5.4. WMM parameters adaptation

An important application of the Hot-Spot Manager can be the implementation and enforcing of QoS mechanisms

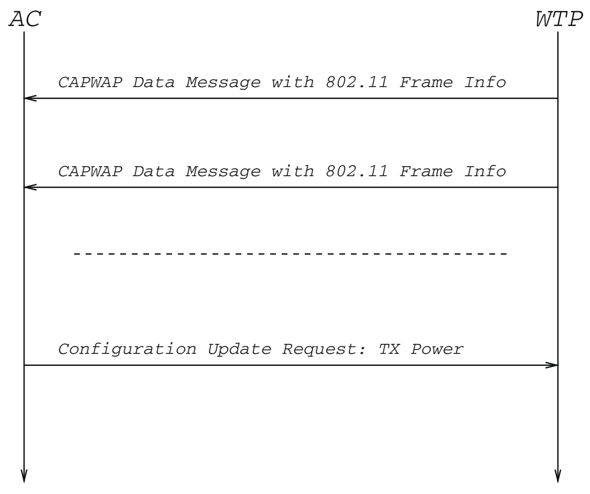

AC
WTP
CAPWAP Data Message with 802.11 Frame Info
CAPWAP Data Message with 802.11 Frame Info
Configuration Update Request: TX Power

Fig. 11. CAPWAP messages for Load Balancing.

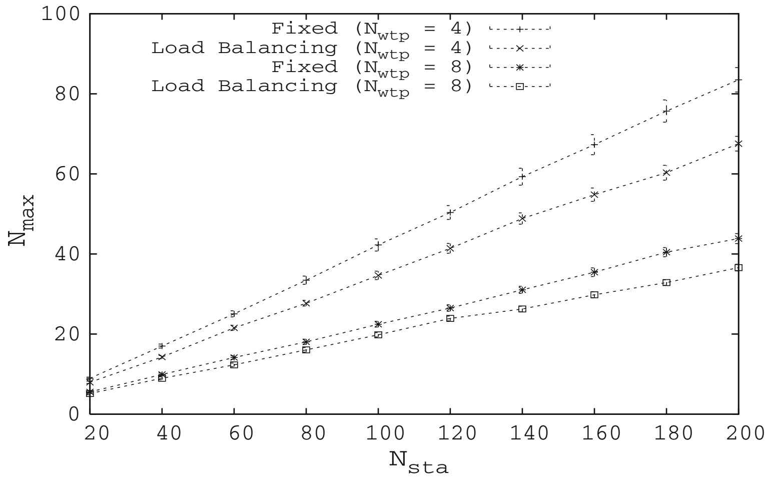


Fig. 12. Load Balancing results.

based on the WMM extensions described in Section 2.1. The dynamic tuning of IEEE 802.11e MAC parameters has been recognized as a powerful adaptive technique to provide QoS guarantees [13–18]. In this section we illustrate the integration of the Hot-Spot Manager in a distributed architecture that uses CAPWAP messages and dynamic setting IEEE 802.11e MAC parameters for the reactive control of QoS on a WLAN. We also provide experimental results that refer to a specific QoS problem.

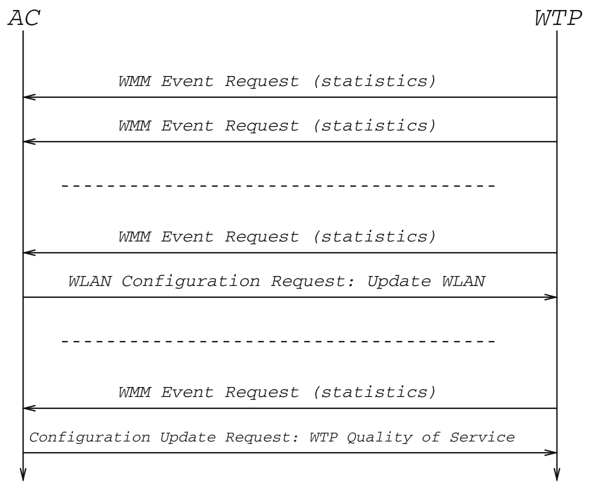
The distributed architecture includes two elements:

- A simple monitoring application that runs on the WTP and periodically sends to the AC statistics about the length of sending queues for the 4 WMM Access Categories;
- a control algorithm, executed by the Hot-Spot Manager, that integrates the WTP statistics with the throughput information available at the AC. Through this algorithm, the Hot-Spot Manager performs admission control on real-time traffic and dynamically sets 802.11e MAC parameters in order to pursue the QoS goals.

The Hot-Spot Manager can obviously use a variety of algorithms to perform its task. In the experiments reported here we adapted to a distributed implementation the method presented in [19].

Fig. 13 shows the use of CAPWAP messages for the dynamic setting of WMM parameters. The WTP uses the CAPWAP WTP Event Request control message to send its statistics: the message contains an element called WTP Operational Statistics, with information about the queue lengths (actually, we used four of such message elements, one for each Access Category). The control application running on the AC uses this information to compute the delay of down-link traffic. It can also detect losses due to the lack of buffer space on the WTP caused by the congestion on the Access Category. The control application may react to QoS-critical situations in different ways:

- it can perform admission control, to prevent other STAs from joining the WLAN or to stop ongoing flows; in the latter case, for example, a CAPWAP STA *Configuration Request* with a *Delete STA* message element can be sent by the Hot-Spot Manager;
- it can change WMM parameters on the WTP through a *Configuration Update Request* message. This command contains a WTP Quality of Service message element with the new set of parameters for the WTP (see Section 2.1);
- it can change the WMM parameters on the STAs through a *WLAN Configuration Request* message which contains an *Update WLAN* message element with the new set of WMM parameters. These parameters will be sent to the STAs using the 802.11e beacon.



AC
WTP
WMM Event Request (statistics)
WMM Event Request (statistics)
WMM Event Request (statistics)
WLAN Configuration Request: Update WLAN
WMM Event Request (statistics)
Configuration Update Request: WTP Quality of Service

Fig. 13. CAPWAP messages for the reactive QoS control.

In summary, the CAPWAP protocol supports the configuration of the MAC 802.11e parameters on both the WTPs and the STAs. It makes also possible to perform admission control of new flows and STAs, as well as deciding which Access Category a specific STA can use, etc. The CAPWAP protocol is therefore a valuable tool to enforce centralized QoS policies.

We illustrate an application of the aforementioned architecture based on CAPWAP, for the specific QoS problem of protecting real-time flows in presence of Best Effort (BE) traffic. In this example we assume that initially no BE traffic is present and the maximum number of VoIP STA is admitted. Then, saturated BE sources start. Usually, 802.11e default parameters provide enough differentiation to protect RT QoS, but there are situations where even a single BE flow causes QoS problems to real time traffic. We report experimental data, collected in our test-bed, about one such situation. In this experiment, we initially have 8 symmetric real-time flows at 400 kbps (at the application level), and 40 packets per second. This traffic is sent using the VO access category with default WMM setting ($AIFS = 2, CW_{min} = 8, CW_{max} = 16, TXOP_{limit} = 3264$ μs). Since 802.11b is used at the physical layer and the overall throughput is 6.4 Mbps, the residual capacity of the wireless channel is very scarce. In this situation, two down-link TCP flows and one up-link UDP flow are started, using the BE Access Category ($AIFS = 3, CW_{min} = 32, CW_{max} = 1024, TXOP_{limit} = 1$ packet). The sources of BE traffic are saturated. Due to the increased contention, the QoS of down-link real-time traffic degrades, since there are more losses and delays caused by collisions. Fig. 14 shows the effect on the down-link delay. BE traffic is started at $t \approx 50$ s. The plot refers to the service delay of down-link packets for the VO Access Category in the WTP, sampled at intervals of 1 s. In the whole time interval (0–120 s) the average service delay is about 50 ms, but after BE traffic starts the delay fluctuates widely, with spikes of more than 150 ms.

We have introduced a control application as part of the HotSpot Manager that monitors the service rate for each Access Category of each WTP. Data about packet rate and throughput can be collected directly from the AC, whereas the service delay on the WTP is estimated from the statistics about the queue length that the WTP periodically sends (in this experiment we set the frequency of these reports to 1 per second, but higher frequencies can be used). The WTP and the AC are connected by a 100 Mbps Ethernet link. When the control application detects that the VO queue is building up in presence of BE traffic it reacts by increasing the BE *AIFS* value for the STAs. The effect is shown in the 'adaptive' plot of Fig. 14. The change of the WMM parameter is triggered at $t \approx 62$ s. The average service delay in the whole time interval (0–120 s) is now below 10 ms, but, more importantly, the service delay never exceeds 50 ms after the WMM parameter adaptation. It is worth noticing that, at least in this case, the change of the *AIFS* BE parameter also increases the efficiency of the wireless channel by reducing the number of collisions. As a matter of fact, not only the VO traffic has a lower delay but also the aggregate BE throughput (not shown in the plots) increases from 300 kbps to about 400 kbps. The main contribution to this increase comes from the throughput of the down-link TCP traffic (previously starved by up-link UDP).

A reactive control running on the AC can deal with several QoS issues, such as fairness between real-time and best effort traffic, fairness between up-link and down-link traffic, admission control, etc. What is relevant in the present context is that the performance of our CAPWAP implementation is fully adequate to the requirements of the control application. As reported previously in Table 2, the configuration delay due to the exchange of CAPWAP messages from the AC to the WTP is under 10 ms. This delay is very small when compared to the dynamics of the wireless channel where the start/end of new flows happens every few seconds. Moreover, a new flow takes several seconds before producing an effect, as in the example reported in Fig. 14. Therefore, the delay introduced by the distributed architecture and CAPWAP does not prevent the control application from reacting quickly enough.

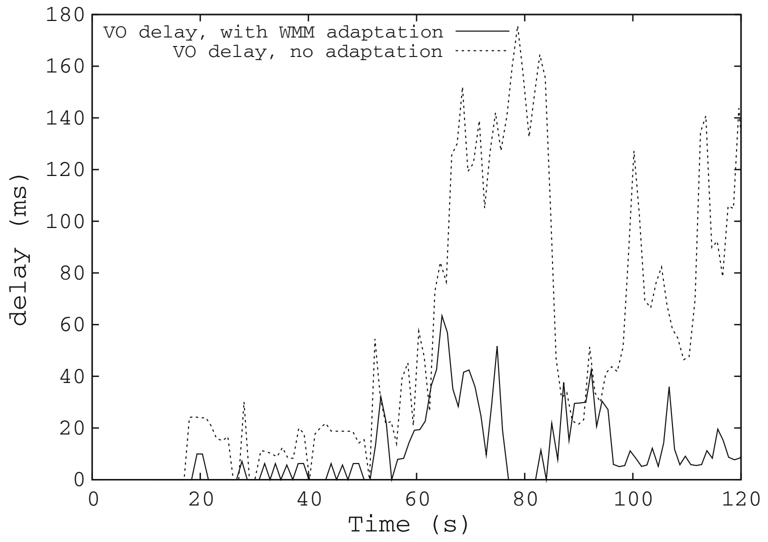


Fig. 14. Automatic WMM parameters adaptation scenario.

6. Conclusions

We presented OpenCAPWAP, an open source implementation of the CAPWAP protocol. The paper described the software architecture and the performance delivered, proving that our implementation can be effectively used in real world situations. Moreover, a set of scenarios have been presented where CAPWAP may be used for network management. Indeed, CAPWAP functionalities for network monitoring are a valuable input to algorithms for network management and configuration, while CAPWAP functionalities for network control are fundamental in the enforcement of policies triggered by these algorithms. Our present work aims at using our CAPWAP implementation as a building block of a comprehensive and autonomic architecture for Hot-Spot management, monitoring and configuration. A first step in this direction is the porting of the WTP CAPWAP client described in this paper to a commercial *off-the-shelf* access point. We are currently testing the porting to 802.11g–802.11e commercial devices. Other issues in this effort include devising efficient and sound algorithms for control applications on the AC, smart methods of estimating the wireless state from the data available at the WTPs, as well as mobility and security issues.

A concluding remark concerns the implementation of the Split MAC architecture. Split MAC moves several low-level functions to the AC. As detailed in Section 3.1, a Split MAC implementation requires more modular devices than those currently available, as well as more standard interfaces between real-time and non real-time MAC functions. It is therefore necessary that suitable WTP devices, specifically designed for the purpose, be available before an open implementation of Split MAC architecture becomes really feasible.

Acknowledgements

This work is partly funded by the 6th Framework Program, Information Society Technologies, under Contract Nos. FP6-IST-038423 (CONTENT), FP6-IST-034819 (ONELAB) and FP6-IST-033516 (NETQOS). We thank Andrea Del Moro, Federica Giovannini and Ludovico Rossi for the development of the first prototype implementation of the protocol. We thank Mauro Bisson and Daniele de Sanctis for their efforts in improving and updating the implementation of the protocol.

References

[1] P. Calhoun, M. Montemurro, D. Stanley, CAPWAP Protocol Specification (v. 11), Internet Draft, IETF, July 2008.
[2] P. Calhoun, M. Montemurro, D. Stanley, CAPWAP Protocol Binding for IEEE 802.11 (v. 7), Internet Draft, IETF, July 2008.
[3] T. Dierks, E. Rescorla, The transport layer security (TLS) protocol version 1.1, RFC 4346 (2006). April.
[4] Information technology – Telecommunications and Information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, IEEE Standard 802.11, 1999.
[5] IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, IEEE Standard 802.11, 2005.
[6] B. O'Hara, P. Calhoun, J. Kempf, Configuration and Provisioning for Wireless Access Points (CAPWAP) Problem Statement, RFC 3990 (Informational), February 2005.
[7] <http://capwap.sourceforge.net>.
[8] <http://www.openssl.org>.
[9] <http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html>.
[10] <http://opencapwap.org>.
[11] A. Levanti, F. Giordano, I. Tinnirello, A centralized approach for automatic frequency planning in WLAN, in: Proceedings of MediaWiN, 2007.
[12] Y. Bejerano, S.-J. Han, Cell breathing techniques for balancing the access point load in wireless LANs, in: Proceedings of Infocom, 2006.
[13] A. Banchs, G. Iannello, P. Serrano, L. Vollero, A CAPWAP architecture for dynamic configuration of QoS MAC parameters, in: Proceedings of MediaWiN, 2006.
[14] I. Dangerfield, D. Malone, D.J. Leith, Experimental evaluation of 802.11e EDCA for enhanced voice over WLAN performance, in: Proceedings of the Second International Work On Wireless Network Measurement (WiNMee 2006), Boston, April 2006.
[15] Y. Tanigawa, J. Kim, H. Tode, K. Murakami, Proportional control and deterministic protection of QoS in IEEE 802.11e wireless LAN, in: Proceedings of International Wireless Communications and Mobile Computing Conference (IWCMC 06), Vancouver, July 2006.
[16] S. El Housseini, H. Alnuweiri, Adaptive contention-window MAC algorithms for QoS-enabled wireless LANs, in: Proceedings of 2005 International Conference on Wireless Networks, Communications and Mobile Computing, vol. 1, June 2005, pp. 368–374.
[17] S. Acanfora, Filippo Cacace, Giulio Iannello, Luca Vollero, Dynamic configuration of MAC QoS mechanisms in 802.11 access networks, in: Proceedings of 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking, NEW2AN 2006, St. Petersburg, Russia, May 2006, pp. 542–553.
[18] J. Freitag, N.L.S. da Fonseca, J.F. de Rezende, Tuning of 802.11e network parameters, IEEE Communications Letters 10 (8) (2006).
[19] F. Cacace, G. Iannello, M. Vellucci, L. Vollero, A reactive approach to QoS Provisioning in IEEE 802.11e WLANs, in: Proceedings of 4th EURO-NGI Conference on Next Generation Internet Networks, Krakow, Poland, April 2008.

Massimo Bernaschi graduated in physics in 1987 at "Tor Vergata" University in Rome. After that he joined the IBM European Center for Scientific and Engineering Computing (ECSEC) in Rome. He spent ten years with IBM working in the field of parallel and distributed computing.

Currently he is with the National Research Council of Italy (CNR) as Chief Technology Officer of the Institute for Computing Applications.He is also an adjunct professor of Computer Science in "La Sapienza" University in Rome.

Filippo Cacace graduated in Electronic Engineering at Politecnico di Milano in 1988 where he received a Ph.D. in Computer Science. His research interests include wireless and heterogeneous computer networks, network applications, database programming languages and logic programming. He is currently an Assistant Professor at the University Campus Bio-Medico in Rome.

Giulio Iannello graduated in Electronic Engineering at Politecnico di Milano in 1981 and received a Ph.D. in Computer Science and Computer Engineering from the University of Napoli Federico II in 1987. Currently he is Full Professor of Computer Science and Computer Engineering at the University Campus Bio-Medico di Roma.

His current research interests include wireless and high performance computer networks, multimedia data processing, biomedical data and image processing, design and analysis of parallel algorithms, performance evaluation of parallel and distributed systems. He has published over 100 journal and conference papers in these and related areas. He is member of several Program Committees of international conferences. Dr. Iannello is a member of the IEEE and ACM.

Massimo Vellucci graduated in computer science in 2006 from "La Sapienza" University in Rome. Currently, he is working as senior engineer and developer at UNIDATA S.P.A. His research interests include VoIP, wireless, and mobile networks.

Luca Vollero received the M.Sc. and Ph.D. degrees in Telecommunications and Computer Science from the University "Federico II" of Napoli respectively in 2001 and 2005. In 2001 he joined the DIS – Dipartimento di Informatica e Sistemistica – of University "Federico II" of Napoli. In 2005 Luca worked at the Consorzio Nazionale CINI in Napoli as a researcher. In 2005–2006 Luca worked at the NEC Network Labs in Heidelberg (Germany) as a researcher. From 2006, Luca is an Assistant Professor at University Campus Bio-Medico of Rome. His general research interests are in wireless networks and image processing. He served and is serving as reviewer for international journals and conferences. He has/had actively participated in Italian and European projects: CADENUS, DAIDALOS, E-NEXT, CONTENT, NETQoS, OneLAB, Quasar, WebMinds. Luca co-authored over 35 research papers.