SNMP Agent for WLAN networks

Cristian Mihai Vancea, Virgil Dobrota
Communications Department
Technical University of Cluj Napoca
Cluj Napoca, Romania
e-mail: {Mihai.Vancea, Virgil.Dobrota}@com.utcluj.ro

***Abstract* — The solution presented is dedicated to WLAN monitoring using SNMP. The idea was to replace the existing equipment-oriented MIB with a new one that is network-oriented.. The solution is based on a specialized AirPcap interface, on Wireshark protocol analyzer and on a new software agent.**

***Keywords*- management SNMP, WLAN**

I. INTRODUCTION

Managing a WLAN with existing SNMP implementation does not allow a comprehensive vision of the status of all stations and access points. To get an overview of wireless networks, we should replace the existing MIB (which is focused on stations and access points) with another MIB that gives the global information. This requires creating a new SNMP agent, which will be able to offer the new information. The solution presented addresses the Windows operating system and relies on a hardware interface that can capture frames travelling across the network and a packet analyzer that will decode the captured frames. Besides these, one module is needed to extract from the captured frames, the information defined in the MIB.

II. DESCRIPTION OF THE PROPOSED SYSTEM

A. *Protocol analyzer Wireshark*

In 1998 Gerald Combs launched the first version of a tool for analyzing and decoding frames moving across computer networks, called Ethereal. In 2006 the project was renamed Wireshark due to copyright issues. Currently, he is the most popular protocol analyzer and is a "de facto" standard (often "de jure") in many sectors of activity and learning environments [1].

Wireshark runs on most Unix platforms, Apple and Windows. It is a "open source" project , which is distributed under GPL license. The number of protocols can be decoded is 96,000 (at time of writing this paper).

Wireshark uses LibPcap library (for Windows is called WinPcap) to effectively capture the network frames. For the decoding of frames modules are used (called by the authors - *dissectors*) that interprets the captured frame, according to information from the fields. These modules can be included directly in Wireshark or can be loaded dynamically (this allows a new protocol to be added without changing the application).

The user interface can be in graphical or text mode, the last giving the possibility to use Wireshark with other applications for obtaining information. This mechanism was used for by the SNMP agent described later in this paper.

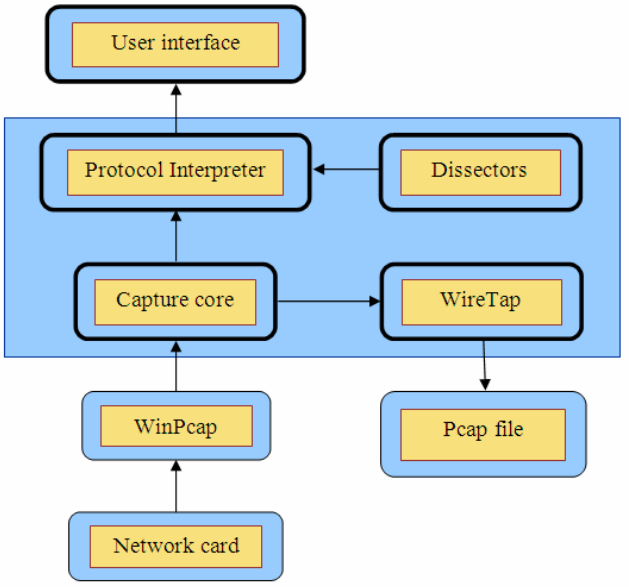

Figure 1. Wireshark Architecture.

B. *PCAP file structure*

Wireshark can be configured to save the captured frames in files in binary format by WireTap module or in predefined text formats. Binary format defined by the module Libpcap.

There are several versions of the binary format that it is used and described is 2.4. Last modified format was released in 1998, and is not expected to change anytime soon [2].

The files contain a global header including general information and a number of records for each frame captured.

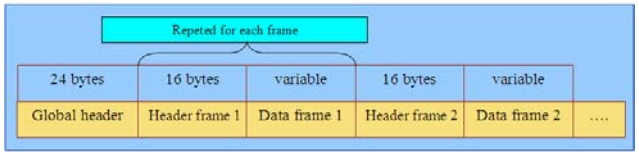

Figure 2. PCAP file structure.

It will not always contain all bytes as were taken, sometimes only the first "n" bytes of the frame. In fact this number depends on a parameter called "snapshot length" which is defined by the user. The default value is 65535, usually higher than the current length of frames.

Global header is 24 bytes long and contains the following fields:

- magic_number (32 bits): it is used to identify the file format and to save the order of bytes in file. Application that saves data (in this case Wireshark) writes 8 bytes (0xa1b2c3d4) using byte order defined on that machine. It will read data 0xa1b2c3d4 (same as the application that saves) or 0xd4c3b2a1 (reversed version).
- version_major, version_minor (16 + 16 bits): - it defines the file format version used
- thiszone (32 bits): the number of seconds difference between GMT and local time zone
- sigfigs (32 bits): it defines the precision of capture time
- snaplen (32 bits): the maximum length used in capture
- network (32 bits): identifier for the protocol used in data link layer

At the beginning of each captured frame a header is added (header frame), which has a length of 16 bytes and contains the following fields:

- ts_sec (32 bits): date and time of capture frame. It is reported as the number of seconds passed from January 1, 1970 00:00:00 GMT
- ts_usec (32 bits): the number of microseconds (it is considered as an "offset" for ts_sec)
- incl_len (32 bits): the number of bytes actually captured and saved
- orig_len (32 bits): the number of bytes of network frame

The captured frame follows after the header.

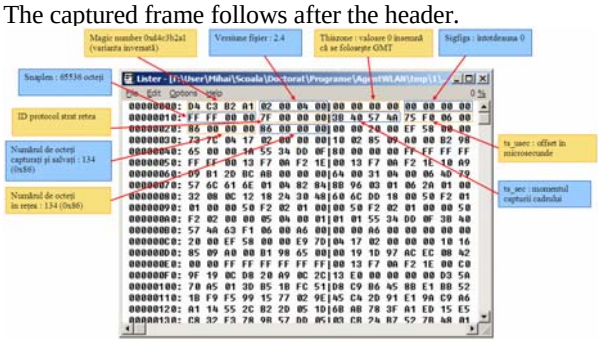

Figure 3. Example of the fields in the global header and frame header files Pcap.

C. *AirPcap Interface*

Capturing IEEE 802.11 frames in Windows, using Wireshark (or similar applications) is almost impossible. This is due to the implementation of WinPcap library and drivers used for network cards. Most times data frames are captured, without access to the control or management frames and their headers will be converted by network card driver in the Ethernet frame header. AirPcap interface was specifically designed for all IEEE 802.11 frames and offers them to Wireshark in Windows. More information can be found in [3].

There is an application that allows configuration of radio channel monitored, determines the type of capture and adds the RadioTap header to the WLAN frames.

D. *RadioTap Header*

This header RadioTap [4] was introduced to provide information about the received frame, initially implemented in the NetBSD operating system only. It has a fixed part of 8 bytes, followed by a variable number of bytes in length. The fields in the fix part are:

- it_version (8 bits) – version of header type used (current value is 0)
- it_pad (8 bits) - unused
- it_len (16 bits) – specific header length (fixed + variable part)
- it_present (32 bits) – Bit mask indicating the variable fields

The following fields may exist within the variable part:

- Antenna
- Antenna noise
- Antenna signal
- Channel
- FHSS
- Flags
- Lock quality
- RX flags
- Rate
- TSFT
- TX attenuation
- dB TX attenuation
- dB antenna noise
- dB antenna signal
- dBm TX power

III. MIB PROPOSAL FOR IEEE 802.11

The structure of proposed MIB contains information about physical and data link layers for WLANs. MIB description was made taking into account the rules specified by SMIv2. A single section called wLANStatistics, with 20 different objects was designed. We took the decision to enter the MIB under the *experimental* branch, choosing the random value 7330, from the numbers not allocated by IANA [5]. Actual traffic information described in this MIB may be used by network administrators for a better allocation of resources and can be combined with those obtained in theory by applying the traffic prediction algorithms in

wireless networks [6]. The objects defined can be seen in Figure 4.

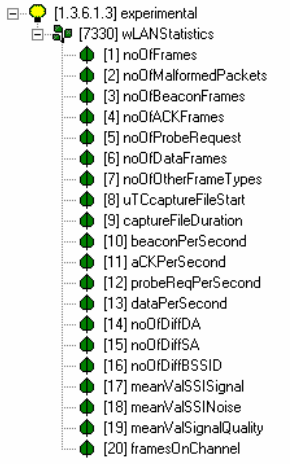

Figure 4. Objects defined in the MIB WLAN.

- 1.3.6.1.3.7330.1 – is the total number of frames captured and is obtained by counting all frames from captured file
- 1.3.6.1.3.7330.2 – is the total number of erroneous frames and is obtained by counting all erroneous frame from captured file
- 1.3.6.1.3.7330.3 – is the total number of Beacon frames and is obtained by counting all beacon frames
- 1.3.6.1.3.7330.4 – is the total number of acknowledge frame and is obtained by counting all acknowledge frame
- 1.3.6.1.3.7330.5 – is the total number of Probe Request frames
- 1.3.6.1.3.7330.6 – is the total number of data frames and is obtained by counting all data frames from captured file
- 1.3.6.1.3.7330.7 – the number of frames not included in other categories
- 1.3.6.1.3.7330.8 – is the time of capture of the first frame and is obtained by reading from file the moment of time saved for the first frame
- 1.3.6.1.3.7330.9 – is the length of the monitoring interval in seconds
- 1.3.6.1.3.7330.10 – is the number of beacon frames per second and is obtained by dividing the number of beacon frames with the number of seconds in the monitoring interval
- 1.3.6.1.3.7330.11 – is the number of acknowledge frames per second and is obtained by dividing the number of acknowledge frames by the number of seconds in the monitoring interval
- 1.3.6.1.3.7330.12 – is the number of Probe Request frames per second and is obtained by dividing the number of type Probe Request frames to the number of seconds in the monitoring interval
- 1.3.6.1.3.7330.13 – is the number of data frames per second and is obtained by dividing the number of data frames to the number of seconds in the monitoring interval
- 1.3.6.1.3.7330.14 – the number of different destination addresses
- 1.3.6.1.3.7330.15 – the number of different source addresses found in capture file
- 1.3.6.1.3.7330.16 – the number of different AP
- 1.3.6.1.3.7330.17 – the average value of SSI for signal
- 1.3.6.1.3.7330.18 – the average value of SSI for noise
- 1.3.6.1.3.7330.19 – the average signal quality
- 1.3.6.1.3.7330.20 – number of frames on each channel

IV. IMPLEMENTATION OF SOFTWARE SNMP AGENT FOR IEEE 802.11

The software agent developed for wireless networks consists on two software applications: NetAnalyzer for data collection and WLAN_Agent for communication with the SNMP manager [7]. Other approaches for SNMP usage in WLAN can be found in [8].

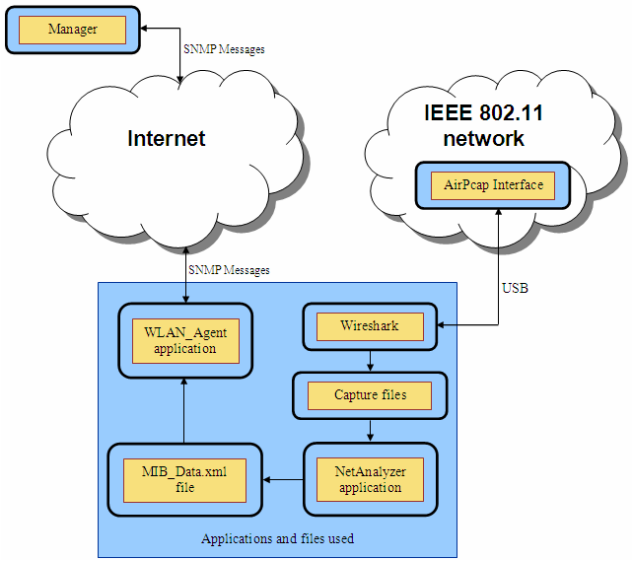

Figure 5. Connection of the modules for WLAN Agent .

A. *Installing and configuring software of WLAN agent*

The programs were developed to be used in Windows, and for proper operation they should be installed in the same directory. One is NetAnalyzer.exe, with the role of extracting information from files generated by Wireshark WLAN_Agent.exe, which is responsible for the communication with managers, and MIB_Data.xml file, which is intended to be the connection file between the two applications. In addition to these three files, application Wireshark need to be installed on this machine.

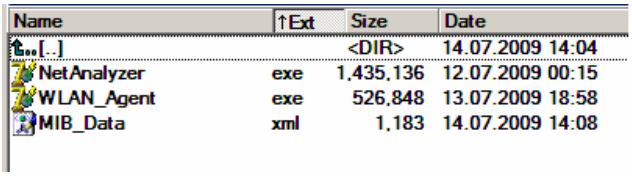

Figure 6. Files needed for WLAN agent

NetAnalyzer was created for processing captured files generated by the Wireshark. The application was created using Borland Delphi 7 development environment. This application will start automatically Wireshark at the begin of the capture, Wireshark beeing configured with the parameters introduced in NetAnalyzer. Among the configuration options are time interval used by Wireshark to save capture file and the name of the fiel. At the time specified NetAnalyzer read capture file and write values obtained in MIB_Data.xml file. The results obtained MIB_Data.xml file can be seen in Figure 8. Processing a file containing about 55,000 frames took about 500 milliseconds.

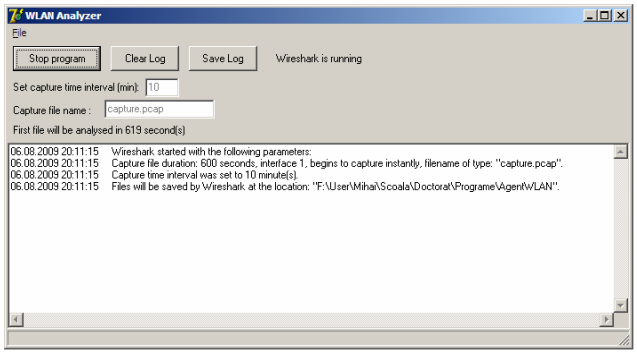

Figure 7. The GUI of the application NetAnalyzer

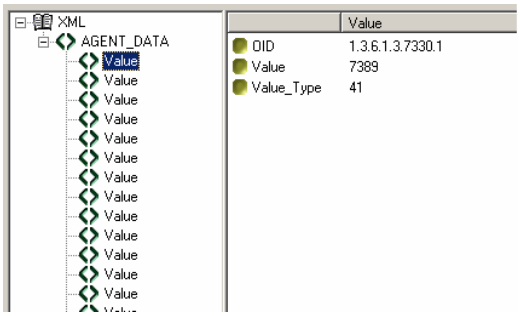

Figure 8. Sample values from the file MIB_Data.xml

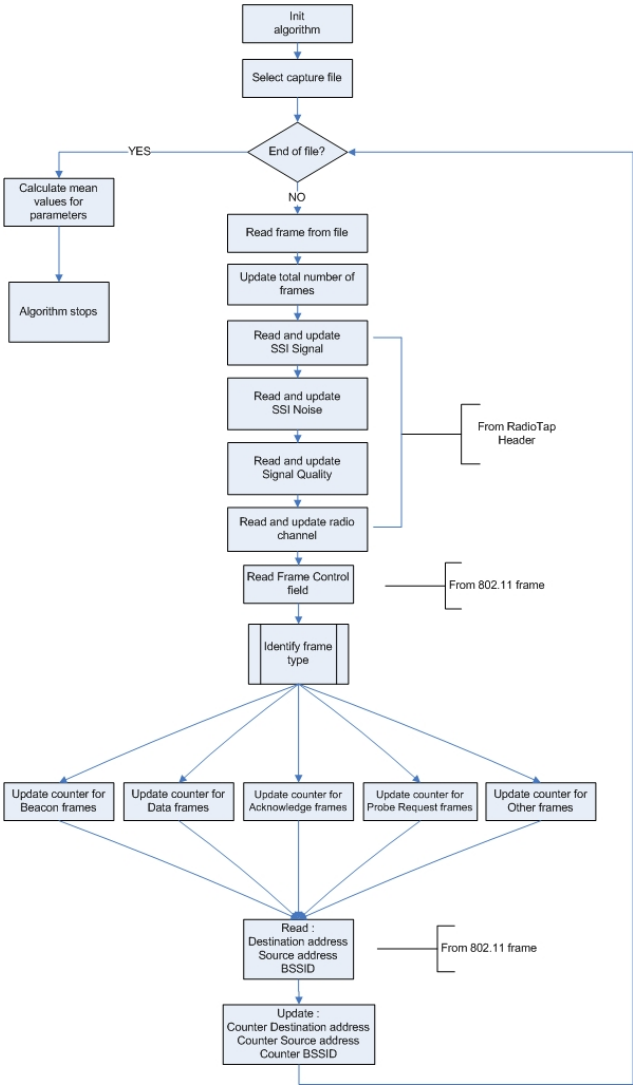

Figure 9. The algorithm used to obtain information from the MIB

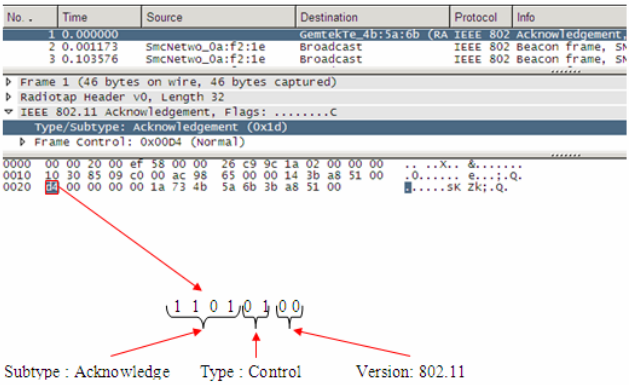

Figure 10. Information used to identify the type frame acknowledge.

B. *Test architecture*

To demonstrate the functionality and to validate the MIB developed for agent, we created a test architecture according to Figure 11. The complete test requires a PC that will run the application management and a second computer which installed all the applications required by wireless agent. For validation several experiments have been performed, all having the same configuration, the difference between them consisting of WLAN traffic. In experiments that were performed the number of available AP was different, experiments with more than 10 AP, experiments in which the number of available AP was between 5 and 10 and experiments in which the number of AP was below 5.

Another parameter that was changed during various tests was the interval used by Wireshark to save file and NetAnalyzer to process the saved files. It was varied between 5 minutes and 30 minutes. NetAnalyzer application can be configured to process files with data from up to 24 hours but at greater interval is more difficult to run tests (test duration can reach several days). The management application used was SNMPManager.

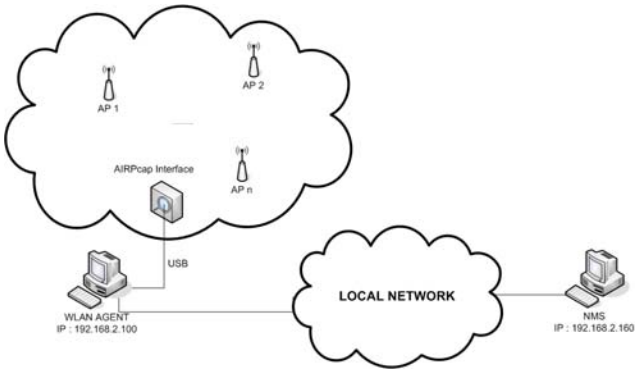

Figure 11. Test architecture used for WLAN software agent.

A first set of experiments was carried out in an area where the number of AP was relatively small (under 5 APs available). We've monitored parameters defined in the MIB and the monitoring interval was up to several hours.

In TABLE I some of the values of monitored parameters from this set of experiments can be seen.

Nr. Crt.	OID	Mean Value	Min Value	Max Value
1	1.3.6.1.3.7330.1	8500	7821	20411
2	1.3.6.1.3.7330.2	2	2	17
3	1.3.6.1.3.7330.3	5850	4293	6769
4	1.3.6.1.3.7330.6	1620	550	7031
5	1.3.6.1.3.7330.14	30	14	71
6	1.3.6.1.3.7330.16	3	2	5

TABLE I. VALUES OF PARAMETERS FROM FIRST SET OF EXPERIMENTS

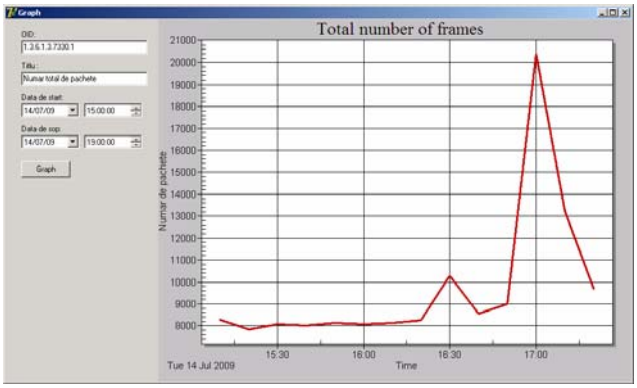

Figure 12. Changes in total number of packets over time.

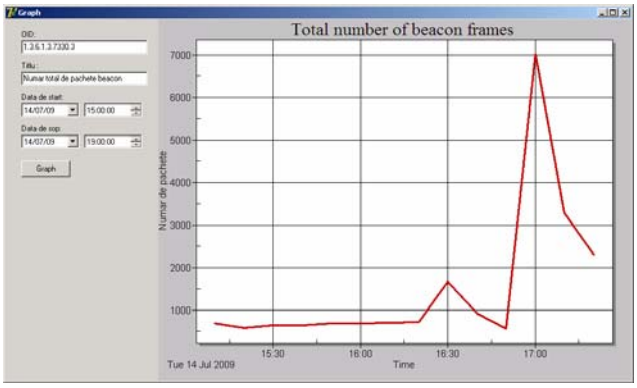

Figure 13. Changes in the number of beacon packets.

A second set of experiments was carried out in an area where the number of APs was medium (between 5 and 10 AP available). The parameters defined in the MIB were monitored for several hours.

The third set of experiments was carried out in an area where the number of APs was relatively high (at least 10 available APs), being monitored all the parameters defined in the MIB, a monitoring interval of several hours.

The results obtained in the set 2 and set 3 of experiments are similar to those obtained in the set 1, changes in the number of frames being similar.

V. CONCLUSION AND FUTURE WORK

Managing a WLAN with existing SNMP implementation does not allow a comprehensive vision of the status of all stations and access points. We have designed a software agent to be independent to the version of the standard IEEE 802.11 a, b, g, n, s and to provide statistical information on traffic captured from the wireless network. The solution is based on the collection with a specialized interface AirPcap (Cace Technologies) and interpretation of data with Wireshark protocol analyzer, running under Windows. The results are stored in .xml files and are accessed by the agent which implements communication with SNMP manager (version 1 and 3). Note that the version of SNMP agent implemented by Windows could not be used because it was based on MIB-II.

Although the standard was developed for usage SNMP in wireless networks, MIB used (i.e. MIB-II) does not provide specific information to WLAN. For this reason we have created a MIB based on 20 items (characterized by the type and value) to provide relevant information about: the number of frames (total, Beacon, ACK, data, ProbeRequest, wrong, other), the instantaneous values (number beacon frames/s, ACK/s ProbeRequest/s, date/s), number of different destination addresses, number of different source addresses, number of access points. The management information can be used to improve network design, to justify the enlargement/ reduction of the network and to improve the network security.

REFERENCES

[1] ***, About Wireshark, 2009, http://www.wireshark.org/about.html

[2] ***, Libpcap File Format, 2009, http://wiki.wireshark.org/Development/LibpcapFileFormat.

[3] ***, AirPcap Family, *CACE Technologies*, 2009, http://www.cacetech.com/products/airpcap.html

[4] J.Berg, „Radiotap Header Description", 2009, http://www.radiotap.org/

[5] ***, Network Management Parameters, *IANA*, 2009, http://www.iana.org/assignments/smi-numbers

[6] H.Feng, Y.Shu, S.Wang, M.Ma, „SVM-based models for predicting WLAN traffic", *Proc. IEEE ICC*, 2006, pp. 591 - 596, June 2006.

[7] C.M.Vancea & V.Dobrota, "Retrieving Call Detail Records from Asterisk using SNMP", *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, ISSN 1221-6542, Vol.50, No.3, 2009

[8] J.Kerdsri, „*SNMP Over Wi-Fi Wireless Networks"*, Storming Media, 2003