W-Meter : An open-source WLAN product evaluation framework

Mohamed Imran K R
AU-KBC Research Centre
MIT campus of Anna University
Chennai, India
mohamed.imran@au-kbc.org

Srikanth S
AU-KBC Research Centre
MIT Campus of Anna University
Chennai, India
srikanth@au-kbc.org

***Abstract*— Wireless LAN products based on the IEEE 802.11 standards are very popular across home and enterprise markets globally. The certification programme conducted by the wi-fi alliance is expensive and focuses on the inter-operability issues. It also uses proprietary tools for accomplishing the same. In this paper, we propose a framework for the comprehensive evaluation of 802.11 based products using free and open-source tools. The evaluation comprises of rigorous conformance, performance, and inter-operability issues.**

Index terms – IEEE 802.11, Wi-Fi, Certification, FOSS

I. Introduction to IEEE 802.11 and Wi-Fi

IEEE 802.11 is a set of protocols which describe the characteristics of operation of a wireless local area network (WLAN) [1]. They are defined by the the IEEE LAN/MAN Standards Committee (IEEE 802). It is the dominant standard for WLANs and is deployed in the ISM frequency bands namely, 2.4 and 5 GHZ. The first standard was released in 1997 and there have been many changes and amendments to the standards which includes advancement in features in terms of data rate, security, performance etc.

The 802.11 standards target the specifications of the Medium Access Control (MAC) Layer and the Physical (PHY) Layer of the TCP/IP Protocol Stack. The 802.11 base standard gave data rates of up to 2Mbps. Further amendments have been made to the standard in order to optimize bandwidth, use advanced physical layer techniques to improve throughput, security and other features. These include the widely popular 802.11b (11 Mbps), 802.11a (54 Mbps and OFDM on 5GHz band), 802.11g (54 Mbps/OFDM and 2.4GHz band). There have been amendments for improving security measures (802.11i), Quality of Service (QOS) (802.11e), collaborative routing (802.11s) and other amendments reflecting regulatory and protection features. The latest amendment, 802.11n targets data rates of upto 600 Mbps using a combination of advanced physical layer techniques like MIMO, Advanced coding schemes, MAC layer enhancements like aggregation, block-ack etc.,

Wi-Fi Alliance (WFA) is an industry consortium which promotes 802.11 based products and fosters inter-operability between vendors [2]. The alliance certifies products based on the 802.11 standard and gives products the trade name of "Wi-Fi". The certification process has spurred growth and has been influential in expanding growth in consumer and enterprise markets. Wi-Fi is predominantly available in laptops, PC cards, access points and wireless routers. Due to the prevalent usage of Wi-Fi for connectivity, it has also found a market in the enterprise segment through infrastructure products like wireless switches etc. The market base is evolving with an increasing number of smart-phones, PDAs, consumer electronics devices, printers, and all classes of mobile and stationary devices now available with embedded Wi-Fi connectivity. Wi-Fi alliance has certified over 5000 different products across various standards and its amendments.

II. Motivation

The motivation for our work can be viewed from two different perspectives. One is the need to provide the open-source community with the necessary tools to develop open-source drivers for WLANs and the other larger goal is to evolve a rigorous certification program which targets performance, conformance, and inter-operability issues.

There is a vibrant community of open-source developers who are actively contributing to open source device drivers. At present, there are drivers available in the public domain for popular WLAN chipsets like Intel, Atheros, Broadcom etc., There is active encouragement from the vendors with respect to the open-source efforts. The open source model of collaborative development involves developers contributing code from across the globe. They contribute new features in the protocol as soon as the specifications are released. The testing and debugging and certification of the open-source drivers serves as a barrier because of expensive proprietary tools and infrastructure required for the same. Our framework is a unique proposition which will help developers gain quick insight into the implementation and performance. We believe that this can fasten up the process of open-source driver development and attract the developer community. Normally, in a proprietary model of development, the developers use proprietary tools and utilize costly infrastructure to develop/test the code. They then take the driver across for certification which apart from being expensive does not give an insight into the details if the product does not pass through certification. One more important thing is that, this process is not suitable for

incremental development, where the developers do small contributions and test the same. An open-source developer works in a collaborative environment across the internet to develop code. In the WLAN device driver development context, an independent developer might be interested in a particular feature and he may be willing to spare time for the same. The challenge is to provide the developers with the right kind of tools to do the task without depending on expensive hardware or undertaking the certification process for the drivers which is hugely expensive. Let us take for an instance, a developer is working on the aggregation scheme which is mentioned in the 802.11n standard [3][4]. The simple scenario would be to write the code and instantly test the same, just like we do for any normal program rather than go through a lengthy and expensive process for the same. The need for such a framework which can help in the development of open-source code is significant.

Another important issue is the certification process. The widely recognized certification program for 802.11 based devices is conducted by the WFA. The focus of the program is inter-operability testing which translates into co-existence of hardware devices manufactured by different vendors. There are a few limitations in the certification programs. One is the absence of a rigorous and comprehensive testing of protocol details, which can lead to poor implementation of the device drivers. The second is the dependence on proprietary tools and platforms for conducting the certification programs. The issue with the latter limitation is that the certification process becomes expensive and out-of reach for free/open-source developers, who develop the code individually and network across the internet for building software systems. FOSS (Free and Opensource Software) gives the developers the liberty to use/modify the software. This methodology helps the developers to be unencumbered of any restrictions on their efforts to develop the tool. The absence of an free/open-platform can be a handicap for a free/open-source developer. One more pertinent factor in the certification process is that there is no independent performance evaluation of the WLAN devices in terms of speed, bandwidth, Quality of Service (QoS) etc. This hinders consumers from making an informed choice and they are left to vendor studies to gain insights into the devices. The need for an independent end-user evaluation is highly desirable.

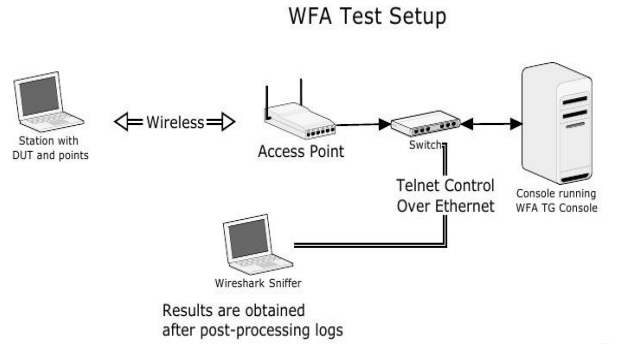

Fig 1: WFA Test Setup

Fig 1 illustrates a sample scenario from the WFA test bed. One can notice that there is a dependence on the platform for conformance testing. This means that we cannot get a stand-alone evaluation of the card and need to develop specific end-points across different platforms. One more issue is the absence of open-source based solution which is completely automated.

Our work plans to target these issues and design a low-cost certification framework which can be helpful to the developers in writing new code while also serving the larger community by evolving a comprehensive test framework.

III. Overview of w-meter

We have designed and implemented w-meter, a FOSS (Free and Open Source Software) framework for comprehensive WLAN testing. The objective of the tool is to give an integrated frame-work which will certify the product for conformance, benchmark the performance and provide an in-depth analysis of the product. The framework would also test the co-existence of the product with other well established products. The frame-work is designed to be extensible to addition of new protocols and programmes. This will help in keeping up with the evolution of the 802.11 standard and the consumer need for programmes which reflect a specific area of application like voice etc., This will also help the developer community incrementally add features to the driver code base. The novelty of the implementation is the integration of open tools to perform the certification. Open source tools have been used for testing purposes but there are a number of tools for the same and maintaining the same for a creating a coherent test environment is a huge challenge. For example, tools like wireshark for packet inspection, iperf for performance measurement are very popular but setting up the tools, configuring metrics in client as well as AP and running the tests iteratively for every change can be a laborious process. Similarly for some validation, the tester has to execute the wireshark program on one machine for sniffing frames and also have a traffic generator for generating specific traffic apart from the machine having DUT (Device Under test). Then for every test case, the configuration has to be changed. All this leads to complexities and imagine if the same process has to be repeated iteratively after every functional improvement (or a new release) to the driver code. We are looking at automation of the whole process in an integrated manner. A typical exercise using our framework would be to configure end-points and then w-meter would run all the

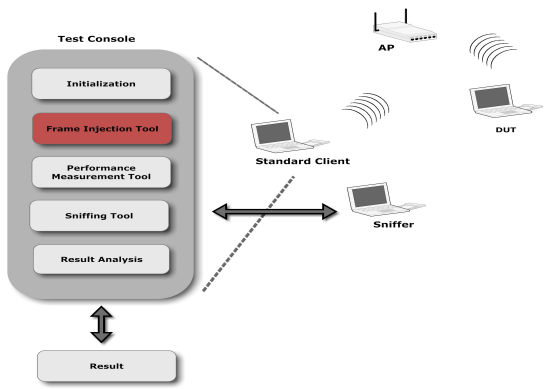

Fig 2: W-meter Components

tests with different environmental conditions and analyze the result.

Fig 2 illustrates the components of our tool. It has integrated all the important tools required and built a GUI (Graphical User Interface) around so that normal end-users and developers can utilize the tool without being bothered about internal details. The GUI is developed using qt, a cross-platform application development framework[6]. A typical setup would be one machine containing our program and another machine having the DUT. Once we start our tool, it does all the configuration of devices, runs all the experiments, and presents a report for the end-user for easy diagnosis of the product. This helps the developer/tester to completely focus on the driver development rather than worry about the design of the test experiments and executing them.

The framework also allows specialized testing of protocol for specific issues like security and other vulnerabilities. For an illustration wve.org, a portal for wireless vulnerabilities and exploits, has documented well known issues with the current drivers[7]. One of the issues mentioned is the vulnerability in Block Ack Handling, where in an attacker can impersonate ADDBA(Block Ack ADD) frames and modify the starting sequence numbers causing all other frames to be dropped. Testing of these scenarios cannot be done by conventional methods. Our framework allows us to customize the frame which also includes customization of source address to impersonate another station. This facility can be used to test how the driver responds to an ADDBA impersonation. Similarly lots of other vulnerabilities, exception handling can be tested.

IV. Components of w-meter test console

A. Initialization

The initialization module has the task of managing wireless interfaces and creating suitable monitor mode interfaces for frame injection. The monitor mode is necessary for the radio tap injection method to work and is also required for sniffing packets from the air. This module also establishes connections with the sniffer tool to extract all the sniffed packets from the network. This will also take care of configuration of the AP settings, station settings, session establishment with DUT etc., and stores results in appropriate files. The overall objective is to initialize and manage the entities for conducting the experiments.

The driver for the whole setup is based on Ath5k/Ath9k, an open-source wireless driver for Atheros based 802.11n chipsets [8][9]. It is based on the mac80211-based drivers which is a new architecture for soft MAC based drivers [10]. It is integrated into the mainline Linux wireless tree and hence it is easy to configure and setup using commodity hardware and works out of the box for most 11n based chipsets from Atheros [11] .

B. Conformance Measurement component

The conformance part of the framework validates the features of 802.11 and their protocol functioning. This will serve as a benchmark for putting the cards for conformance tests. This component relies on the frame injection tool which is described in detail in the subsequent section.

1) Introduction to frame injection tool

One of the major concepts which we have explored and implemented is the usage of injection of arbitrary frames for conformance testing. Using this methodology, we can extensively test the conformance aspect of the protocol by generating frames which are strictly conforming to the protocol and is not dependent on the driver to properly generate frames. This takes away the need to have a traffic generator which is needed to emulate the frames for a specific test case. This approach gives us very strict control to the whole conformance testing paradigm. Another salient feature of this approach is that it also allows packets which are not conforming to protocol to be injected in the air. This allows us to study the behaviour of the card in such cases where there is a mis-behaving node. It also allows us to explore the security aspects of the deployment and find out how the infrastructure is responding to these kinds of challenges. The concept of frame injection is based on radiotap, the de-facto standard for 802.11 frame injection and reception [12]. It acts as a method for interacting between user-space and driver space applications. This is a mechanism for supplying additional information to user space applications like libpcap and for sending message from user space to driver space for transmission. The tool w-meter uses the frame injection paradigm to inject specific frames into the medium to elicit responses appropriately from the DUT.

The frame injection component, which implements the injection paradigm, is concerned with generation of custom and arbitrary frames to evoke responses from the DUT. We have built a GUI wrapper over the radio tap frame injection functionality and using the same, frames can be injected in the air with a few clicks.

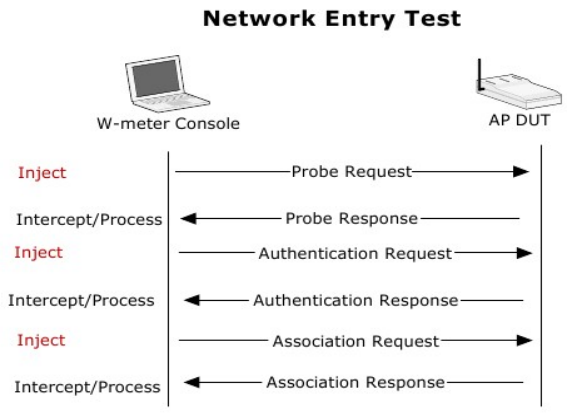

Fig 3: Sequence Injection

Fig 3 illustrates the sequential generation of frames. In such a sequence, every successive frame is dependent on the results of reception of previous frame. Hence, this component also interacts with the sniffing tool to query the received frame and processes it. A unique feature that we have designed includes designing test cases which includes sequences. For example, a network entry test case would include basic exchange, authentication with the AP etc. Consider the scenario where we are testing an AP for conformance. The case we are testing is the network entry process. The setup would include a w-meter console and an AP. In the illustration, all request frames are generated frames from w-meter while responses are generated frames from AP which is the DUT. The main thing to note is that w-meter processes each and every frame before injecting another frame and behaves like a normal station trying to associate with an AP

Fig 4 show the base screen of the tool with combo-boxes for selection of the type of the 802.11 frame. Once the type is selected, the subtypes for that type is automatically listed out for selection. In Fig 4, we have selected Management as the frame type and RTS as the subtype

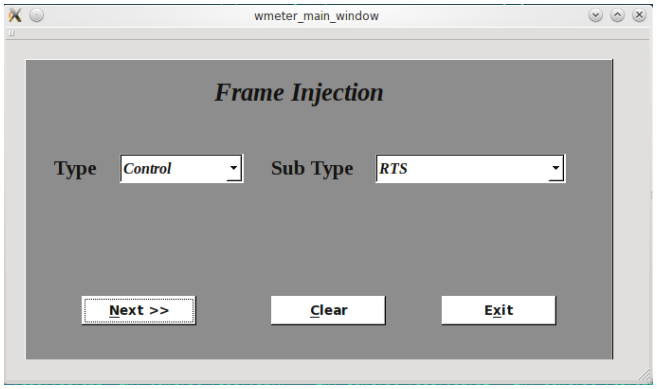

Fig 4 : Screenshot of the base screen

The default template frames for all the frame-types are defined. Once we make the selection for the type of the frame we would like to send, we can then manipulate the default definitions that we have.

Fig 5 shows the screenshot of a window obtained by pressing "Next" button from the window shown in Fig 4.

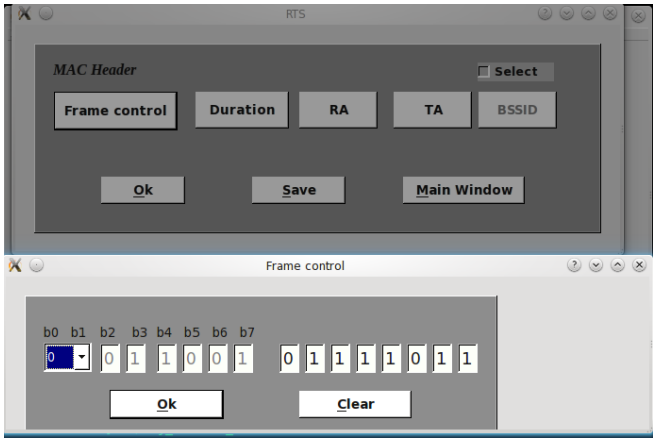

Fig 5 RTS Frame control

Fig 5 also illustrates how frame control field can be manipulated to suit our requirements. Similarly addresses, duration etc can be changed from the default templates.

Fig 6 shows the screenshot of a data frame

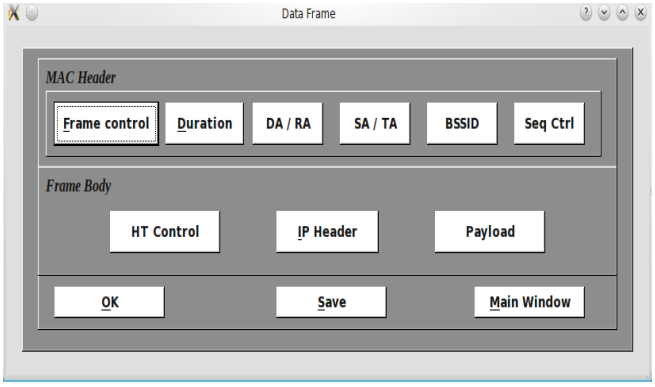

Fig 6 Data Frame

It shows the various options which can be changed to form an arbitrary frame. Fig 7, shows how HT control field of a data frame can be arbitrarily set.

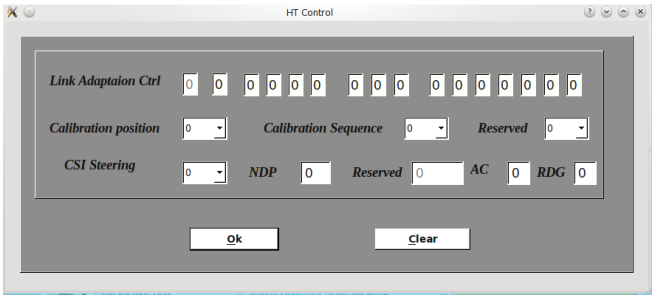

Fig 7 Data frame HT control field

C. *Performance Measurement Tool*

The component is concerned with running appropriate test cases for performance. This component uses iperf, a popular performance measurement tool [13]. The component will be managing an iperf server console either locally or on a backend network and the DUT will be connected to and configured for appropriate test cases as a client. The tests are run with different parameters and settings as defined in the performance test suite. The tool being used for the same is iperf, a widely used open-source tool for performance measurement. Iperf is a very efficient client-server utility which has a number of configuration options to setup and run TCP and UDP traffic with various parameters. The observations that can be made using iperf include throughput, jitter, and datagram losses. The reports contain updated results at user-defined intervals along with the metrics measured. The differentiation with plain vanilla iperf experiments is the automated tests that are being done in this framework. The automation takes care of configuring both the client and the server at every iteration for parameters

(TCP window size for example) and also configuring wireless metrics like block ack.

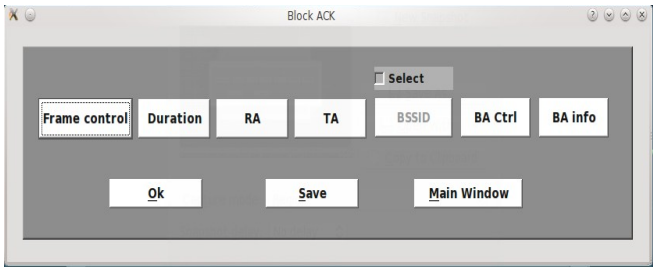

Fig. 8 Block Ack

Fig 8. Shows the screen for custom creation of block ack frames. The BA info button can be clicked to view the screen for entering BA seq control and BA bitmap field

D. Sniffing Tool

This component runs the frame collection using a sniffer on the local test console itself or parses data gathered from other sniffers by connecting to them as outlined in the initialization module. Wireshark, one of the widely used, free and powerful network packet analyzers is used [14]. It has excellent features for packet capture, filtering and analysis.

E. Result Analysis

The proposed module takes the results of both frame injection and performance measurement modules and generates complete report for the DUT including compliance and performance. The output of the module is a report which highlights the compliance of the product to the certification program. The report also includes diagnostics. For example, if a frame is non-compliant, the report will highlight the missing information etc., In relation to performance, the report will list out all the observed results from the performance measurement component and also a diagnostic of which configuration metric can be changed and suggestions for a value for threshold or other settings etc.,

V. Conclusions

We have built an open-source framework for the comprehensive evaluation of an 802.11 based product in terms of performance, conformance and inter-operability. The framework also provides insight into the working of the driver and can provide diagnostic information regarding the same.

VI. References

[1] http://grouper.ieee.org/groups/802/11/ retrieved on 09/01/09
[2] http://wi-fi.org/ retrieved on 09/01/09
[3] 802.11 Wireless Networks: The Definitive Guide, Second Edition
[4] www.ieee802.org/11/Reports/tgn_update.htm
[5] *E. Perahia and R. Stacey, Next Generation Wireless LANs, Cambridge University Press, 2008.*
[6] http://qt.nokia.com/ retrieved on 09/01/09
[7] http://wve.org/ retrieved on 09/01/09
[8] linuxwireless.org/en/users/Drivers/ath5k retrieved on 09/01/09
[9] linuxwireless.org/en/users/Drivers/ath9k retrieved on 09/01/09
[10] http://linuxwireless.org/en/users/Drivers/ath9k#supported_chipsets
[11] linuxwireless.org/en/developers/Documentation/mac80211 retrieved on 09/01/09
[12] www.radiotap.org/ retrieved on 09/01/09
[13] http://sourceforge.net/projects/**iperf**/ retrieved on 09/01/09
[14] www.wireshark.org/ retrieved on 09/01/09