A Low Cost Atheros System-on-Chip and OpenWrt Based Testbed for 802.11 WLAN Research

Sushil Dutt, Daryoush Habibi and Iftekhar Ahmad
Centre for Communications Engineering Research
Edith Cowan University, Australia
{s.dutt, d.habibi, i.ahmad}@ecu.edu.au

***Abstract*—The IEEE 802.11 Wireless Local Area Network (WLAN) is a popular internet access technology, and researchers are continuously working on the quality of service (QoS) improvement by proposing new and efficient schemes. Performance analysis and benchmarking are an integral part of WLAN research. In most of the cases, this is done through computer simulation using popular network simulators such as Network Simulator – 2 (NS-2) or OPNET. The computer simulation of a dynamic system such as a communication network heavily depends upon a set of stochastic models for capturing the real-time behaviors of the system. The choice of a set of stochastic models that represent the most practical scenario is often debated and there is no universal consensus on the choice of these models. This is why, for proper assessment of the merits of a proposal, and any associated commercialization activities, a validation on a hardware platform is highly recommended. Researchers have developed few Field Programmable Gate Array (FPGA) and System-on-Chip (SoC) based hardware, but most of these development platforms are expensive and/or have limited support for networking layers software. This letter proposes an Atheros SoC based hardware platform, and an open-source Linux software stack, called OpenWrt, for WLAN research. The proposed hardware is available off-the-shelf as a WLAN access point (AP), and is a cost-effective way of generating results on a hardware system. A new MAC protocol is implemented on one such WLAN access point to prove the platform.**

***Index Terms*—802.11, Hardware platform, WLAN, Open Source, OpenWrt, Atheros.**

I. Introduction

WLAN is a widely adopted technology in communication networks for numerous applications such as internet access, data transfer, content sharing servers, Internet Protocol (IP) telephony and video surveillance. Access through the wireless medium, however, offers a range of research challenges. For example, it can cause a high percentage of packet collisions, and high packet delay variation [1]. WLAN researchers are working hard on devising new MAC techniques for efficient utilization of the wireless channel by minimizing the aforementioned issues. WLAN research is primarily driven by researchers in academia and many small research centres, and due to the limited support for practical communication infrastructures, they mostly use computer simulation as a tool to build, evaluate and verify their proposed performance-enhancement algorithms. The behavior and characteristics of the new models can be predicted with simulators in a controlled environment, which is governed by numerous stochastic models. The choice of a fixed set of stochastic models that depicts the most realistic scenario is a largely debated topic. For instance, random numbers used in computer simulation are generated by computers and since a computer is a deterministic system, it cannot produce true random numbers [2], [3]. For practical implementations and commercialization, providing a proof-of-concept on a real hardware platform is very essential. The process of selecting the right hardware is not trivial, and researchers need to carefully consider numerous attributes such as the cost, availability of open source code, maturity of hardware and time for development.

Currently, two hardware platforms are primarily used for WLAN research: i) Wireless Open Access Research Platform for Networks (WARPnet) developed by Rice university [4], and ii) CalRADIO1 [5]. CalRADIO1 is suitable for very basic prototype design and is unsuitable for implementing advanced MAC techniques. In comparison, WARPnet is an efficient platform for WLAN research, but it is very expensive and often not affordable for implementing a WLAN scenario that includes numerous nodes. These limitations lead us towards evaluating a hardware that can be modified without excessive burden and can easily be adapted in a budget and time constrained environment. In this letter, we present a framework to use an Atheros SoC based testbed and OpenWrt for real-time validation of the MAC algorithms. The hardware is easily available in the retail market and provides a convenient way to build a link with the personal computer (PC) for programming and debugging purpose. We have used the '*ath9k*' device driver [6], available as Open Source, to implement the TI-MAC (Traffic Intelligent-MAC) [7] and demonstrate the capabilities of the proposed hardware platform. The '*ath9k*' device driver comes with the IEEE 802.11e enhanced distributed coordinated access (EDCA) compliant QoS implementation. Therefore, this platform is also suitable for comparing the performance of newly developed MAC protocols against the 802.11e EDCA.

II. Background Study and Related Works

With the evolution of high-speed microprocessors, the MAC is typically implemented in software rather than like a hardware logic. OpenWrt supports various such soft-MAC and provides a framework to customize the software as per the user design. OpenWrt [8] is targeted for networking embedded platforms and offers tremendous support for off-the-shelf wireless routers, e.g. Netgear, D-Link, Linksys, Asus and Thomson.

In [9], [10], the authors have demonstrated the benefits of OpenWrt, and used OpenWrt to validate their new designs and systems in real-time platforms. However, the authors have not discussed the internals of OpenWrt, the hardware description and the steps to build a testbed.

As discussed in section I, two hardware platforms are primarily used for WLAN research: i) WARPnet [4], and ii) CalRADIO1 [5]. WARPnet is an FPGA based platform for real-time validation of new proposed schemes in the 802.11 WLANs. WARPnet has flexibility to customize all the protocol layers and comes with a custom software for the PHY and MAC layers. The MAC is implemented in C-code as a simple Carrier Sense Medium Access MAC (CSMA-MAC) [4] without any traffic prioritized queues. Researchers commonly use advanced MAC schemes such as the 802.11e EDCA and Hybrid Coordination Function (HCF) Channel Access (HCCA) as references to benchmark their own proposed solutions. Both schemes are based on the priority of application traffic, and multiple transmission queues are used for packet transmission. Therefore, researchers have to implement a priority queue based soft-MAC from scratch, which requires a reasonable amount of time and effort. Moreover, the cost of one FPGA kit is around US$6,500 and to implement a WLAN with one access point and 20 clients, it requires 21 FPGA kits, which costs around US$136,500 in total. This makes WLAN implementation using WARPnet an expensive proposition, which many academic researchers may not afford.

CalRADIO1 [5] is a low-cost platform for the 802.11b MAC layer research purposes. It runs on an ARM7 processor who executes the network layer and functions above. The MAC layer is implemented in C-code, which is executed by a TMS320VC5471 Texas Instruments digital signal processor (DSP). CalRADIO1 comes with a minimal software-development-kit (SDK) which includes a basic implementation of the MAC protocol. There are no prioritized queues for different categories of traffic as stated in the IEEE 802.11e. Therefore, the user has to implement a soft-MAC with prioritized queues, and a software control for priority based transmission. In summary, CalRADIO1 is a good low-cost platform for MAC layer development, but in terms of software availability, it comes with a limited and single queue MAC implementation.

There are other similar kinds of hardware, which are developed on SoC from Broadcom and Texas Instruments, but most often these come with a binary library of the MAC rather than full open source code.

The aforementioned limitations motivated this work towards evaluating Linux based hardware. These are, in general, available with open source code under the GNU public license (GPL). After careful consideration of different hardware and their suitability for new MAC development, this work found an AP manufactured by Netgear. Section III discusses this hardware and the steps to build a testbed.

III. Proposed Testbed

A. The Hardware Description

This work proposes an Atheros SoC based 802.11 a/b/g/n hardware that runs on a 32-bit microprocessor without interlocked pipeline stages (MIPS) 24K processor core in the host device AR7161. The AR7161 [11] is a high performance wireless network processor running at 680MHz and makes a peripheral component interconnect express (PCIe) interface with the AR9280 WLAN device. The AR9280 supports full soft-MAC architecture, and it can be controlled by programming of memory-mapped registers by the host device. The AR9280 contains 10 prioritized queues to process the incoming data packets of different priorities, and an arbiter automatically grants the permission to the highest priority queue. Moreover, Atheros has released the complete source code for this device to the Linux community. Netgear WNDR3700 [11] wireless router, that is available off-the-shelf (costs US$200) and fully supported by OpenWrt, uses this advanced chipset to implement a dual band gigabit router. This hardware can also be trusted for its reliability and performance characteristics because it is a final customer product and would have gone through rigorous testing and compliance checks. Whereas, the research platforms discussed above are in the prototype stage and may not have gone through the extensive testing like a commercial product.

B. Testbed Design

A personal computer (PC) or laptop with a Linux operating system (OS) is required to develop an interface with the Netgear WNDR3700 hardware. For Linux installation, a virtual machine environment with Virtual Machine (VM) player was created in the PC, also, the Ubuntu 10.04 (desktop edition) Linux OS was installed from VM player.

A trivial file transfer protocol (TFTP) connection was established between the LAN port of the hardware and the PC to program the hardware. There are two other interfaces available on-board to debug the software: i) Universal Asynchronous Transmitter Receiver (UART), and ii) Joint Action Test Group (JTAG). This work used the UART interface, also commonly known as the serial interface, to monitor the software execution in a hyperterminal window of the PC, and to send bash commands from a command line interface. The UART link can be built by a modified Nokia CA-42 USB-to-Serial cable. This cable is available in the electronics stores and costs around US$10. Fig. 1 shows the testbed built from the WNDR3700 hardware. We can add N number of similar stations in the testbed without any limitation.

The firmware image of the WNDR3700 was compiled from the OpenWrt Linux distribution [12] using gcc (GNU Compiler Collection) commands. The steps to transfer the firmware image are as follows:

- Connect an ethernet cable between the PC and one of the LAN ports (any one of the four LAN ports but not the WAN port) of the targeted WNDR3700 hardware.
- Connect the custom build serial cable to the header "J1" on the WNDR3700 and set the local hyperterminal session (e.g. minicom) for 115200 bps 8N1, no software flow control and no hardware flow control.
- Set the PC ethernet port to use a static IP address of 192.168.1.2 and netmask 255.255.255.0.

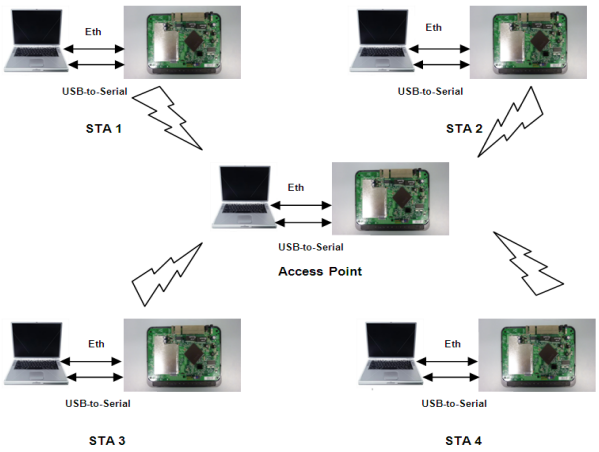

Figure 1. The proposed testbed with Netgear WNDR3700 wireless router

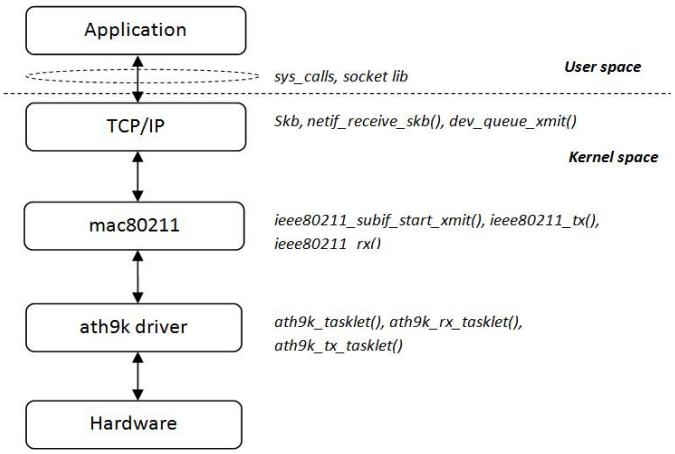

Figure 3. Top-level software architecture of OpenWrt

```
ar7100> setenv ipaddr 192.168.1.1
ar7100> setenv serverip 192.168.1.2
ar7100> setenv bootargs 'board=WNDR3700'
ar7100> tftpboot 80800000 image.out
```

Figure 2. TFTP commands for image transfer

- Turn-on the hardware, at the power on reset (POR), the hardware boots up with the uboot bootloader. A further boot-up of the kernel and the application from flash memory was aborted by hitting the 'EnterKey' in the hyperterminal. At this stage, the LAN port was up, and a default IP address 192.168.1.1 was assigned.
- Next, the commands shown in Fig. 2 were executed from the hyper-terminal to configure the TFTP parameters and transfer the "image.out" in the memory through the TFTP link.
- Once the image was downloaded into the memory, the $'bootm'$ command was used to run the new binary image.

Once the hardware is programmed with the newly compiled firmware image, the TFTP interface may be disconnected, allowing the hardware to work as a standalone wireless device.

C. Example of TI-MAC development with OpenWrt

To validate the TI-MAC, our previously published work [7], we used OpenWrt to implement the polling scheduler proposed in the TI-MAC. OpenWrt allows software development both in the user and kernel space. In the user space, it allows the control over the contention window (CW_{min}, CW_{max}), arbitrary inter-frame spacing (AIFS), short inter-frame spacing (SIFS), slot-time, channel data rate (Mbps), transmission frequency, ready-to-send (RTS)/clear-to-send (CTS) and acknowledgment (ACK). We have implemented the TI-MAC state machines in '*C*' programming language as two kernel space tasks in the '*ath9k*' device driver. The kernel space provides a direct control over the AR9280 and low real-time latency. First state machine named *ath_timac_baseSTA_work()*, runs in the AP, and generates the polling cycles and transmits the traffic for the associated stations. Second state machine named *ath_timac_STA_tx_work()*, runs in each remote station. The state machine running in the remote station waits for its polling cycles and starts the packet transmission once received.

D. OpenWrt Architecture

A top-level software architecture representing various layers in OpenWrt is shown in Fig. 3. The user space applications interact with kernel space through *sys_calls* and *socket* library. The device drivers for the peripheral hardware are built as loadable modules. WLAN device drivers are divided into two modules: i) hardware dependent module, and ii) protocol module (soft-MAC). The hardware dependent modules are different for each vendor and device. For WLAN, the soft-MAC framework is known as '*mac80211*'. The soft-MAC provides an interface for most of the MAC functionality defined in the IEEE 802.11 protocol. The WNDR3700 router uses a hardware dependent driver by Atheros, the '*ath9k*'. The '*mac80211*' uses the low-level functions of the '*ath9k*' device driver to transmit and receive the data packets. The functional flow between various kernel layers during the data packet transmission and reception is explained in the following two subsections. A developer may seek this functional flow to follow the packet buffer and its contents at various layers, which is typically required at the prototyping stage.

1) Packet Reception: A new packet arrival at the PHY is notified by an interrupt. This interrupt is handled by an interrupt service routine (ISR), *ath_isr()*, defined in the *'ath9k'* driver. The received packet is handled by *ath_rx_tasklet()*, which retrieves the packet bytes from the Rx queue buffer of the AR9280. The function named *ath_rx_tasklet()* also maps the packet into the socket buffer structure, '*skb*', and passes this '*skb*' to the function *ieee80211_rx()*. The *ieee80211_rx()* function is defined in '*mac80211*'. The function *ieee80211_rx()* passes this '*skb*' to the upper network layer by calling the function *netif_receive_skb()*. The network layer passes the received '*skb*' to the upper layer for further delivery to the application layer. It may also send the '*skb*' to *dev_queue_xmit()* for further routing of the received packets to another interface, e.g. WAN port. The sequence of various function calls during packet reception is shown in Fig. 4.

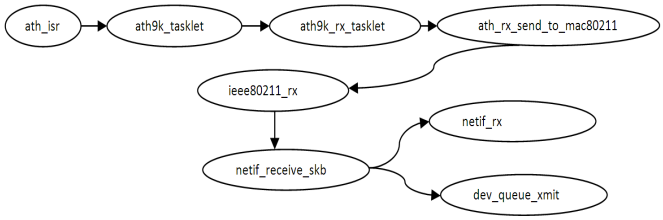

Figure 4. Packet reception functional flow

2) Packet Transmission: The packet arrival at the TCP/IP layer is handled by the function *dev_queue_xmit()*, which in turn calls the function *dev_hard_start_xmit()*. Both functions are defined in '*net*' driver of the kernel. This function in turn calls a function pointer '*ndo_start_xmit*', which is mapped to function *ieee80211_subif_start_xmit()* defined in '*mac80211*'. The *ieee80211_subif_start_xmit()* calls the *ieee80211_tx()* function, which finally calls *ath9k_tx()* from the '*ath9k*' device driver. The sequence of various function calls during packet transmission is shown in Fig. 5.

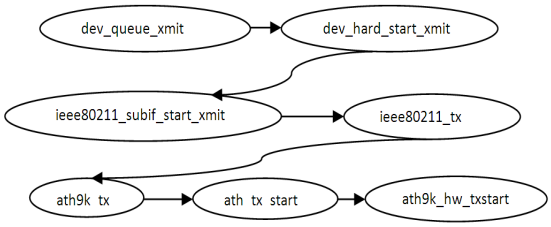

Figure 5. Packet transmission functional flow

IV. Experimental Analysis With Proposed Testbed

We have used the proposed testbed as shown in Fig. 1 to implement and evaluate the TI-MAC [7] against the 802.11e EDCA. We have generated a video IP telephony scenario, where a constant bit rate data was injected at 956 Kbps (please refer [7] for other details). Figs. 6, 7 show the throughput received in both upstream and downstream directions with four stations and one access point. The trend in throughput, in general, resembles the throughput observed in the NS2 simulation [13] with some additional accuracy achieved through the use of TI-MAC and the proposed hardware testbed.

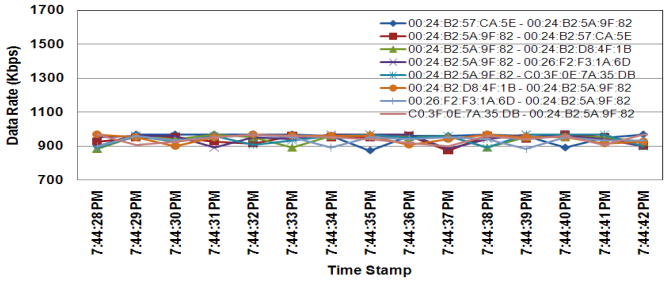

Figure 6. Downstream and upstream data rates with TI-MAC at 9 Mbps

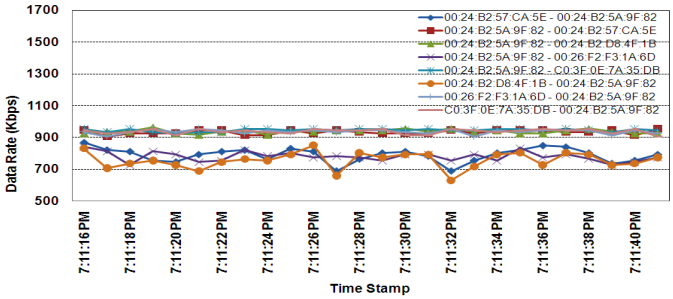

Figure 7. Downstream and upstream data rates with EDCA at 9 Mbps

V. Conclusion

For implementation and commercialization of new algorithms in WLAN, a proof-of-concept conducted in a hardware platform is crucial, if not mandatory. Currently, there is no cost-effective hardware platform available for WLAN researchers. In this letter, we have presented a hardware platform embedded with OpenWrt firmware that can be used for implementing and testing of MAC protocols in a practical system. We have used an off-the-shelf Netgear wireless router as the practical system, and implemented the TI-MAC as a test case. The hardware platform has generated results that are consistent with other similar studies (e.g. using NS2) and as such, the effectiveness of the hardware testbed has been demonstrated.

References

[1] N. Seitz. Itu-t qos standards for ip-based networks. *Communications Magazine, IEEE, 2003*, 41(6):82–89, 2003.

[2] C.J.A. Bastos-Filho, J.D. Andrade, M.R.S. Pita, and A.D. Ramos. Impact of the quality of random numbers generators on the performance of particle swarm optimization. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 4988 –4993, oct. 2009.

[3] Minyong Qi and Jinxin Dong. Research and application of entropy in the sequence randomness test. In *Computational Intelligence and Industrial Applications, 2009. PACIIA 2009. Asia-Pacific Conference on*, volume 2, pages 224 –227, nov. 2009.

[4] WARPnet: http://warp.rice.edu/trac/wiki/CSMAMAC. last cited on 20th december 2011.

[5] Riccardo Manfrin, Andrea Zanella, and Michele Zorzi. Functional and performance analysis of calradio 1 platform. *2009 Eighth IEEE International Symposium on Network Computing and Applications*, 2009.

[6] ath9k device driver: http://linuxwireless.org/en/users/Drivers/ath9k. last cited on 20th december 2011.

[7] Sushil Dutt, Iftekhar Ahmad, and Daryoush Habibi. A novel optimized scheduler to provide qos for video ip telephony over wireless networks. *13th IEEE International Conference on High Performance Computing and Communications, HPCC 2011*, 2011.

[8] OpenWRT : https://openwrt.org/. last cited on 20th december 2011.

[9] C.E. Palazzi, M. Brunati, and M. Roccetti. An openwrt solution for future wireless homes. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1701 –1706, july 2010.

[10] A.P. Ortega, X.E. Marcos, L.D. Chiang, and C.L. Abad. Preventing arp cache poisoning attacks: A proof of concept using openwrt. In *Network Operations and Management Symposium, 2009. LANOMS 2009. Latin American*, pages 1 –9, oct. 2009.

[11] OpenWRT : http://wiki.openwrt.org/toh/netgear/wndr3700. last cited on 20th december 2011.

[12] OpenWRT : http://downloads.openwrt.org/backfire/. last cited on 20th december 2011.

[13] J. Wyatt, D. Habibi, I. Ahmad, and H. Zen. Providing qos for symmetrical voice/video traffic in wireless networks. In *15th IEEE International Conference on Networks, ICON*, pages 312–317.