國立交通大學

資訊科學與工程研究所

碩 士 論 文

即時嵌入式系統之混合模式軟體媒介層
延遲評估平台

Mixed Mode SoftMAC Latency Evaluation Platform
for Real-time Embedded System

研 究 生：盧昆鋒

指導教授：謝筱齡/許騰尹　教授

中 華 民 國 九 十 七 年 六 月

即時嵌入式系統之混合模式軟體媒介層延遲評估平台

Mixed Mode SoftMAC Latency Evaluation Platform for Real-time Embedded System

研 究 生：盧昆鋒　　Student：Kun-Fong Lu

指導教授：謝筱齡　　Advisor：Sheau-Ling Hsieh

許騰尹　　Terng-Yin Hsu

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

摘要

近年來，由於下一代無線通訊的需求，投入 SDR 的研究日益增加，SDR 是實體層中一項以軟體取代硬體功能的技術，而在 SDR 之上的協定也被認為必須能夠在不同媒體存取機制間適時地切換，SoftMAC 是一項可以滿足此需求的技術，其為一具有可透視性、彈性且低價的軟體元件，然而，IEEE 802.11 要求在一個傳輸中訊框間隔必須非常精確，這限制了此協定無法完全以軟體實現，為了解決這個問題，我們需要一個可以同時傳送硬體和軟體回應的平台，如此便能夠量測兩者之間的時間差，由於時間緊要的控制訊框通常是由硬體處理，而且只能在監聽模式下才可被軟體接收，因此，我們提出一個方法，在驅動程式中混合一般與監聽模式，再者，為了克服這個障礙，可利即時嵌入式系統回應時間短的優點，然而，即時嵌入式系統的架構不同於個人電腦，其處理器未配有硬體亂數產生器，而作業系統核心也過於老舊無法支援 mutex 銓鎖，我們發現硬體亂數產生器可由軟體取代，而 mutex 栓鎖可利用 spinlock 代替，儘管這些取代方式在個人電腦上是可行的，但驅動程式仍無法如預期般的在嵌入式系統上運作，所以全部的實驗皆實做在個人電腦上，實驗結果顯示軟體可在一般模式下接收控制訊框，我們鼓勵研究者投入即時 SoftMAC，並在 SoftMAC 上驗證新的協定而非使用模擬器，例如：ns simulator。

Abstract

In the past few years, researchers have devoted to developing software defined radio (SDR) significantly due to the demand of next generation wireless communications. The technique replaces hardware components with software in the physical layer. The protocol above SDR is also considered to be able to alter adaptively for different media access control (MAC) mechanisms. SoftMAC, a transparent, flexible and low cost software component for data link layer MAC protocol, fulfills this objective. However, IEEE 802.11 requires rigid interframe space during a transaction. It restrains the protocol unable to be implemented completely in software. In order to cope with the requirement, we need a platform that delivers the responses both hardware-wise and software-wise to evaluate the time difference between the two approaches. Since the responses for time-critical control frame are usually manipulated by hardware, it can only be received by software in monitor mode. Therefore, we proposed a mechanism that mixes station and monitor modes in Linux wireless driver. Furthermore, to overcome the barrier, real-time embedded system is demanded for developing since it generates responses rapidly. Nevertheless, the embedded system architecture is quite different from desktops. The processor is not capable of hardware random number generator, and the operating system kernel is outdated that mutex lock is not supported. We discovered that the hardware random number generator can be replaced by software; mutex lock can be substituted by spinlock. Notwithstanding these replacements have been proved feasible on desktops, the driver still cannot work completely as expected on target machines. Thus, we have done the experiments totally on desktops. The experiment results show that control frames can be received by software in station mode. We encourage the researchers to involve in real-time SoftMAC and verify their designs on SoftMAC but not on simulator, e.g. ns simulator.

致謝

兩年的時間，說長不長卻也不短，這兩年裡，有歡樂也有淚水，感謝我的指導教授謝筱齡與許騰尹老師的指導，讓我在這兩年內不管是在待人或者處事方面，都有著更成熟的想法，另外，還要感謝博通的 Eddie 和 Keven 資助這個計畫及提供實驗設備，還有實驗室同學這兩年的互相幫助，讓我的研究生活順遂許多，最後要感謝我的父母，讓我的生活不虞匱乏而能專心於學業上。

Table of Contents

List of Figures

Chapter 1 Introduction

SoftMAC makes a transit to the Media Access Control (MAC) layer from hardware to software. Traditional network communication is classified into several layers which are dedicated for specific tasks. Higher layers tend to be implemented in software. Conversely, lower layers are likely to be carried out by hardware. However, in order to gain more flexibility and reduce the cost induced by hardware. Lower layers are considered to be implemented with increased software portion. SoftMAC plays an important role for fulfilling this objective.

For instance, the next generation of wireless communication system may involve Software Define Radio (SDR) that can accommodate various protocols concurrently. SDR supports different protocols by replacing some hardware components with software. The idea of SDR is initially developed by US military but is eager to commercialize due to its excellence. Consequently, substantial effort has been dedicated to apply it onto the physical (PHY) layer of the SDR terminal.

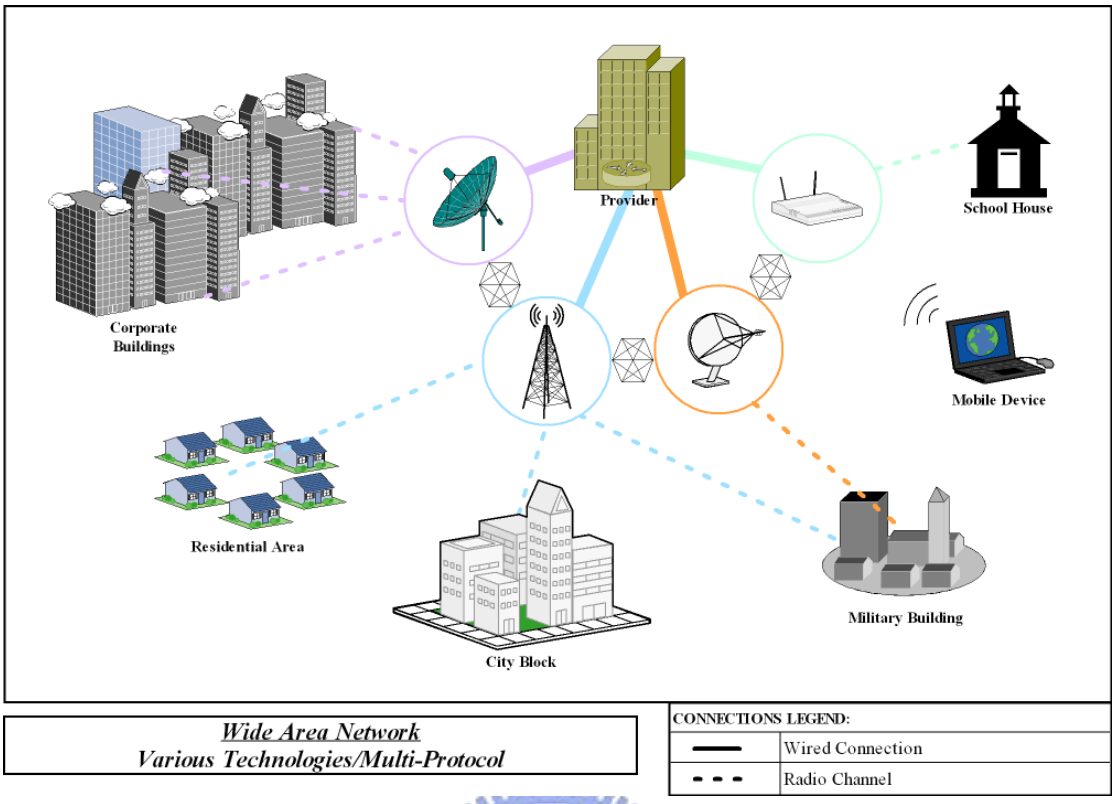


Figure 1-1: Next generation wireless communication scenario

Figure 1-1 depicts the next generation wireless communication scenario. In the future, the mobile devices will be required adaptively changing from one network to another. The modulation, coding and data handling mechanisms may vary within different radio protocols. SDR enables the mobile devices to roam through different environments without specific hardware requirement.

Besides, different devices also provide different channel access control mechanisms. Studies for SDR encourage us to extend the concept onto the data link layer. The data link layer is usually composed of software and hardware. The commercial Wi-Fi modules are equipped with a chip to cooperate with driver. A subset of the MAC protocol is implemented in microcode and executed by the MAC chip. It manipulates time critical frames, such as beacon frame, request to send (RTS)/clear to send (CTS), ACK and so forth. These frames are required to fit into a time period with a short interframe space (SIFS) in between them.

Because of the proprietary concern, the driver and microcode are rarely released in source form. The research has been confined due to the non-transparency.

Therefore, the SoftMAC research platform has been developed by Michael Neufeld et al. in 2005 [1].The combination of various layer 2 protocols has been achieved by overriding the original MAC protocol. The study might have been far more interesting if they have included the original MAC protocol. There is a barrier in software implementation of the original protocol, since it requires the frames being sent in a timely fashion. Hyunseok Lee and Trevor Mudge point out the difficulty in their study [2]. In addition, a dual-processor platform has been proposed to meet the hard real-time constraint.

Recent researches in wireless networking have largely deployed the experiment environment by using commodity products. These products offer better cost-performance ratio. Furthermore there are abundant resources that can be acquired on the Internet, such as Linux wireless driver on linuxwireless [5]. The time consuming for developing the fundamental software is eliminated though. The researchers can therefore concentrate on the design of novel concepts. Besides, building up a real environment enables researchers to inspect the problems may occur as applying the design into reality.

Several attempts have been made to SoftMAC. Michael Neufeld et al. investigates a mechanism to ignore the time critical task, whilst Hyunseok Lee and Trevor Mudge exploit a supplemental processor to respond to the time critical frames. However, ignoring time critical task is not practical to infrastructural environment since IEEE 802.11 has been largely deployed to these places. For the compatibility, we have to adopt the standard IEEE 802.11 protocol. Although the supplemental processor proposed by Hyunseok Lee and Trevor Mudge can meet the time requirement, it cannot be carried out without specific hardware support. Recently, however, real-time techniques are evolving from time to time. Processing ability is also increasing as time passes. The barrier, time constraint, may be able to break up with these advances.

In addition, even if the IEEE 802.11 has been commercialized for a long period, numerous evidences have been presented by previous research that it fails to guarantee the security. John Bellardo and Stefan Savage have reported the vulnerabilities of IEEE 802.11 [3]. Deny-of-Service is able to carry out by deauthentication since the message is not authenticated itself. Any other station can generate a deauthentication to a victim. Moreover, network allocation vector (NAV) offers another way to force other stations unable to transmit. NAV is originally used by CTS/RTS mechanism to prevent collisions that induced by hidden terminal. Once the station receives RTS or CTS, further transmission can only be performed until the duration indicated by NAV expired. Nevertheless, they have argued that most of the commodity devices reset the NAV improperly. In general, the commodity products do not offer modification to microcode or driver. Therefore, their examination of NAV was done on ns simulator.

So far, however, there has been little discussion about how we can counter the time constraint without a specific hardware. The aim of this paper is to examine the mechanism that we can use for measuring the latency of the frames. It enables us to verify the software efficiency in timing perspective. To overcome the real-time constraint, a porting procedure will also be performed on a real-time embedded system. The system with real-time characteristic may be able to respond the message more rapidly.

The rest of this paper is structured as follows. In section 2, a comprehensive overview of IEEE 802.11 MAC layer will be given. Section 3 describes the concepts of the experimental platform. Section 4 explains how we achieved these concepts. Section 5 gives the result. Section 6 concludes this paper.

Chapter 2 Background

2.1 MAC Layer Details

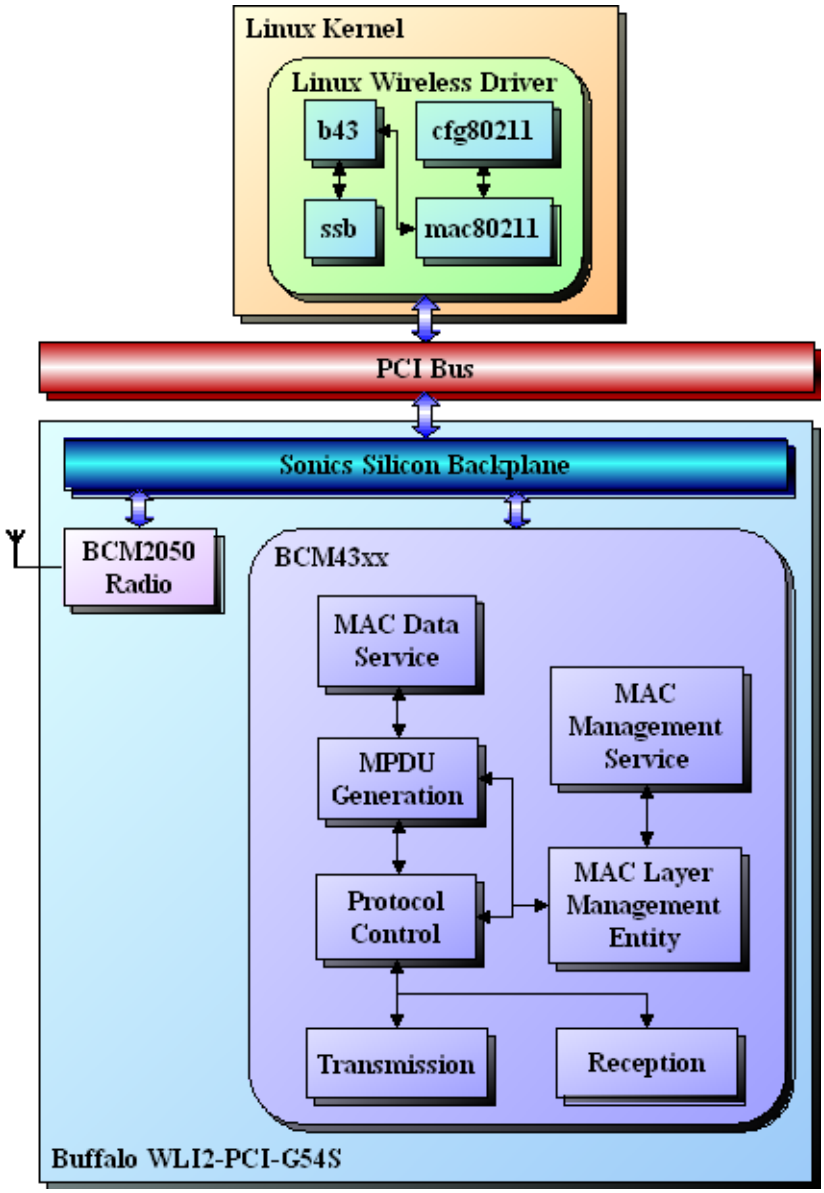


Figure 2-1: Hardware/Software components of wireless adapters

A network communication is accomplished through layer to layer. Packets received from PHY layer are manipulated by radio chip and then forwarded to MAC layer. As illustrated in figure 2-1, this transaction is performed on sonics silicon backplane (SSB). BCM43xx retrieves the packet from SSB and verifies whether the packet is valid or not. If it is valid,

BCM43xx will generate a response or pass it to Linux wireless driver. Here driver behaves as an interface between operating system and BCM43xx.

All the on-chip communications are take advantage of SSB. Whereas the ssb provides the APIs to other modules, the cfg80211 offers the configuration interface to the operating system that takes effect on b43 and mac80211.

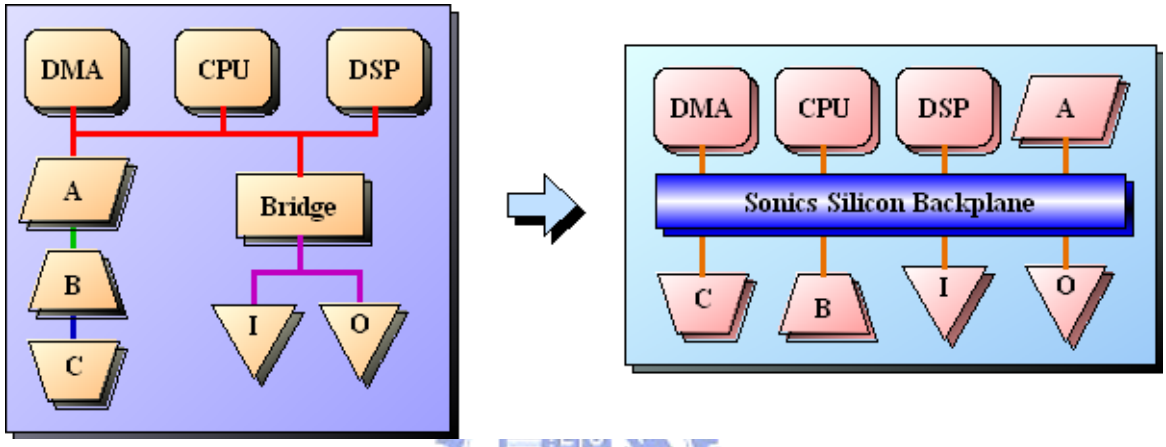


Figure 2-2: Transition from conventional bus system to SSB architecture

Figure 2-2 depicts the transition from conventional bus system to SSB architecture. In traditional system, each component has its own architecture. It complicates the implementation and lowers the performance of system bus. The SSB functions as a switch that allows on-chip communications to be done by accessing the corresponding memory address. Thus, the interfaces between devices can be unified. The design is therefore simplified and becomes more efficient.

The key features of IEEE 802.11 MAC include distribute coordination function (DCF), point coordination function (PCF), backoff procedure, physical and virtual carrier sense, RTS/CTS exchange, fragmentation, defragmentation, retransmission, duplicate filtering, synchronization, power management, association and reassociation. Some of these features are included in the major components of BCM43xx. The major components of BCM43xx

including the following unit.

1) MAC Data Service
    a) Exchange data between logical link control and MAC sublayer.
    b) Validate request parameters and attach a basic MAC header before sending the MSDU to MAC sublayer.
    c) Extract the address and status info then remove the MAC header and indicates LLC to receive the MSDU.
2) MAC Management Service
    a) Exchange management frame with LLC.
3) MPDU Generation
    a) Fragment MSDU into MPDUs.
4) Protocol Control
    a) Generate RTS/CTS, ACK and announcement traffic indication message (ATIM).
    b) Route data frames to MAC Data Service and management frames to MAC layer management entity (MLME).
5) Transmission
    a) Send an MPDU to the PHY layer.
    b) Calculate the random backoff time.
6) Reception
    a) Receive an MPDU from the PHY layer.
    b) Validate received frames.
    c) Detect duplicated frames.
    d) Defragment the fragmented frames.
    e) Maintain channel state based on physical and virtual carrier sense.
7) MLME
    a) Record power save state.

b) Manage the station state such as scanning, join, beacon, active/passive, associate, reassociate, disassociate, authenticate, and deauthenticate.

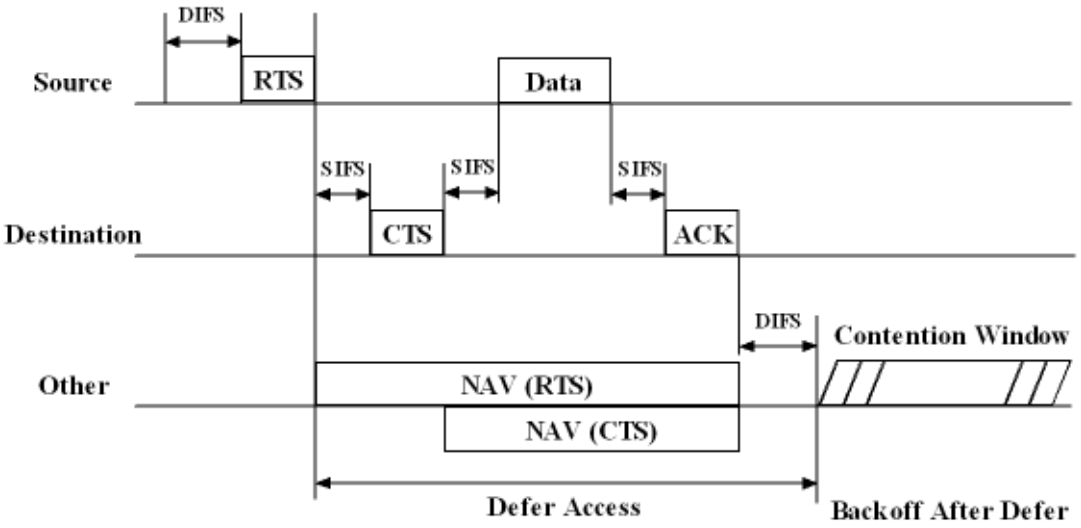


Figure 2-3: IEEE 802.11 RTS/CTS 4-way handshake

In order to share the medium with multiple devices, the contention between difference devices must be avoided by some means. IEEE 802.11 adopts the carrier sense multiple access/collision avoidance (CSMA/CA) to ensure that only one transmission can occur in the same time. As shown in figure 2-3, the RTS/CTS 4-way handshake prevents from other stations to interfere the transmission. In order to gain the throughput as much as possible, the interframe spaces during this transaction are required to be SIFS. According to IEEE 802.11 standard, the SIFS is defined as 28us or 10us depending on the PHY layer modulation which can be frequency-hopping spread spectrum (FHSS), direct sequence spread spectrum (DSSS) or infrared (IR). The latter two mechanisms require the 10us SIFS while the other one need SIFS to be 28us. This is why current researches are escaping from IEEE 802.11 or employing an FPGA to satisfy the time requirement.

2.2 Linux Wireless Driver

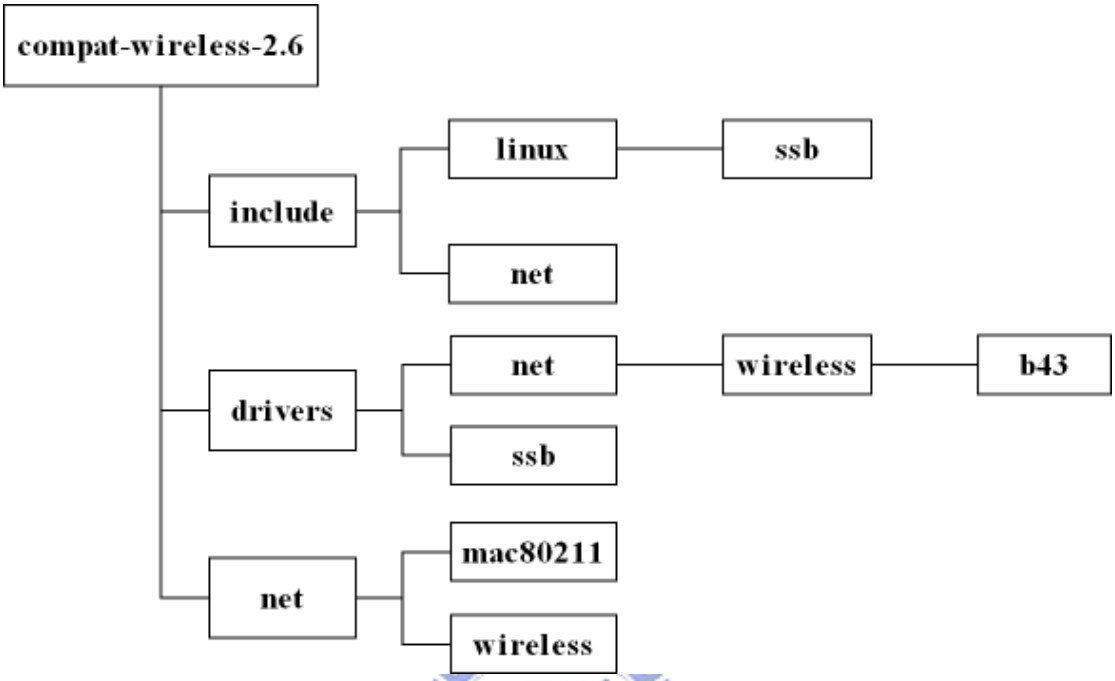


Figure 2-4: Linux wireless driver directory structure

The Linux wireless driver has been developing by Linux wireless group. The project is aimed on designing a standard IEEE 802.11 MAC protocol stack for Linux operating system. Besides, a large number of device drivers are also included to support various wireless adapters. Currently, mac80211 is responsible for high-level MLME operations, whereas low-level operations are handled by hardware or firmware. With mac80211, operating system invokes the APIs to write Linux wireless drivers. The mac80211 is currently resides in kernel space but it will be moved to user space ultimately.

The term SoftMAC is defined on Linux wireless website as a type of wireless card where the MLME is expected to be managed in software and the mac80211 serves as a driver API to these wireless cards. However, our definition to SoftMAC is the whole MAC protocol being implemented in software. The term SoftMAC will be used to refer to our definition throughout this paper.

To be able to operate the Broadcom chipset properly, four kernel modules are required: ssb, cfg80211, b43 and mac80211. The directory structure that has shown in figure 2-4 are the directories which correspond to the components needed by the experiment. All modules have the same name as the directory where they are located except cfg80211. The cfg80211 is located in wireless directory. All the files reside in the directories shown in figure 2-4 will be ported onto target machine. The header files will be put into an independent directory to avoid conflict with conservative header files. Section 4 will describe the procedure in more detail.

2.3 IEEE 802.11 Operation Modes

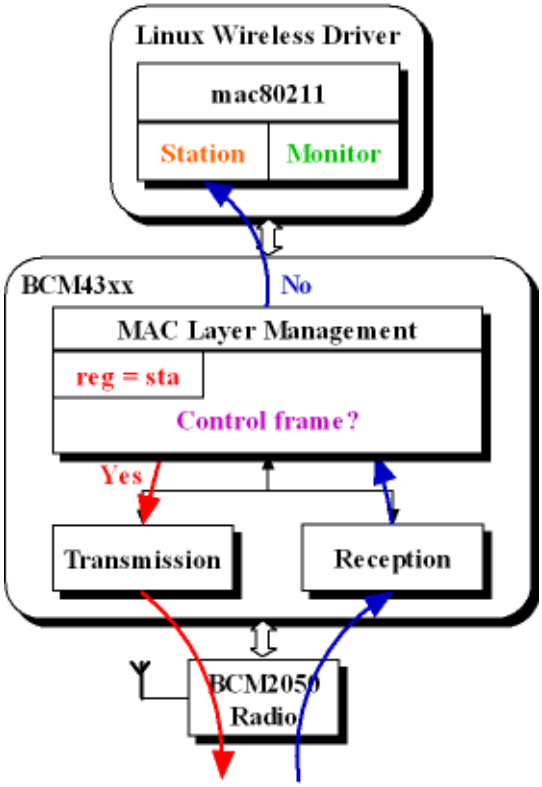


Figure 2-5: Frame receiving path in station mode

In IEEE 802.11 wireless adapter, there are several operation modes that can be set. However, only station and monitor modes will be described in this paper. As a station, it

operates as a finite state machine that defined in microcode, which replies the control frame by itself and passes the data frames to upper layer. Figure 2-5 illustrates the procedure of a frame being received. The BCM43xx has been configured as a station by setting the corresponding bit to the MAC control register. There are also different receiving path in Linux wireless driver. In station mode, the basic service set identifier (BSSID) and Ethernet address of the frame need to be compared to confirm the validation of the frame. Figure 2-6 illustrates the time sequence for frame receiving in station mode.

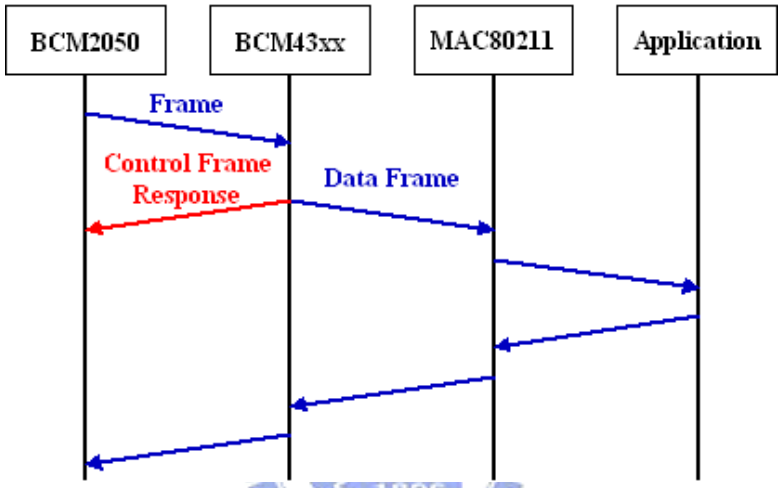


Figure 2-6: Timing diagram of frame receiving in station mode

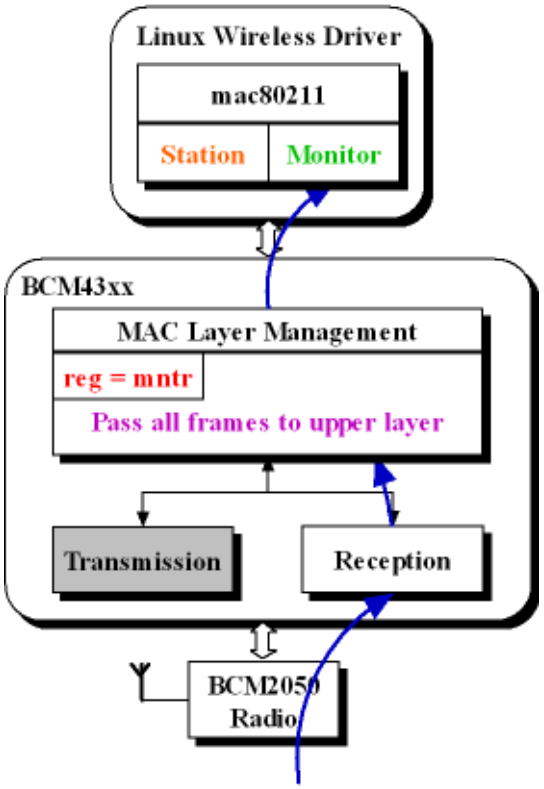


Figure 2-7: Frame receiving path in monitor mode

Figure 2-7 depicts the receiving flow in monitor mode. The MAC control register is configured with the bits that required by monitor. Under the monitor mode, all the frames that percept by the radio will be passed to Linux wireless driver by the MAC without extra manipulation. The driver receives the frames without regarding to what BSSID they belong to or which Ethernet address they bound for. Additionally, since the main function of monitor mode is to intercept all the frames which are transmitting on the air, the transmission component has become inactive.

2.4 Override MAC protocol

The SoftMAC proposed by Michael Neufeld et al. is oriented to accommodate different protocols concurrently. To accomplish this goal, there are six primary tasks to be done.

1) Override 802.11 MPDU frame format

2) Eliminate ACK and retransmission

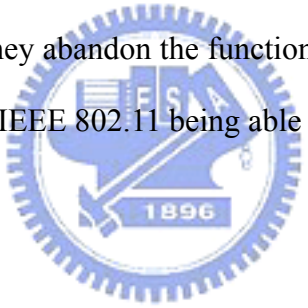
3) Eliminate RTS/CTS exchange

4) Eliminate virtual carrier sense

5) Control PHY clear channel assessment (CCA)

6) Control transmission backoff.

The first three tasks were achieved by configuring the adapter into monitor mode. They take the advantage of Atheros chipset to be able to transmit the frames in monitor mode by marking them as retry frame. Eliminating NAV can be done by changing the MAC address since it is only applied when the destination address matches. The CCA function is also provided by the Atheros chipset. Eventually, transmission backoff can be controlled by contention register settings. Although Michael Neufeld et al. have achieved multi protocols by overriding the original behavior, they abandon the functionality to accommodate IEEE 802.11. Our goal will focus on facilitating IEEE 802.11 being able to integrate into SoftMAC.



Chapter 3 Platform Design

3.1 Measurement Scenario

In principle, a generic approach would be slower than a specific approach and so as software implementation to hardware implementation. Thus, the software MAC must be slower than a hardware MAC since the hardware MAC is a specific hardware approach with particular microcode that is dedicated to the IEEE 802.11 MAC protocol. As described in previous section, SIFS has been defined as 10 us in IEEE 802.11 standard. This can be the major concern because the software implementation will spent much more time than hardware implementation does.

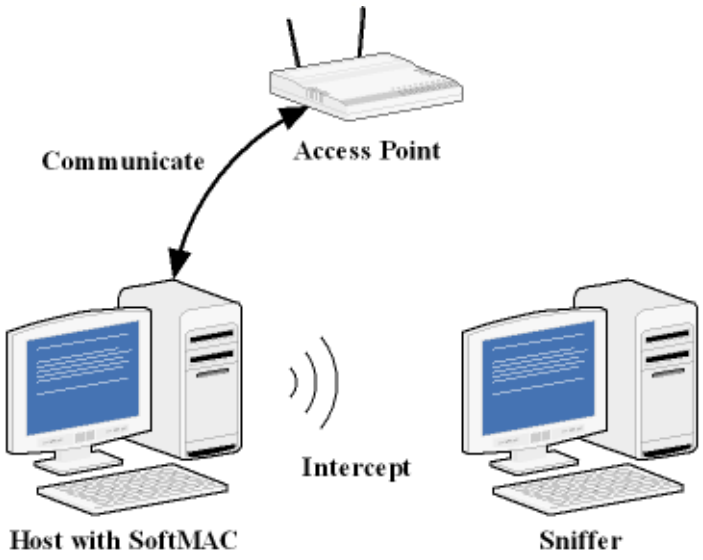


Figure 3-1: Scenario of latency measurement platform

In order to estimate the possibility of software implementation for strict time constraint protocol, we need a third party that can observe the time difference between software packet and hardware packet. Figure 3-1 depicts the scenario of the measurement platform. A usual communication is performing between the host and the AP. The sniffer measures the difference by intercepting the packets sent by the host.

Except the time measurement platform, a real-time system can facilitate the process to shorten the frame generating duration. As real-time application has been widely deployed, the real-time requirement will be the basic functionality to embedded system. The response time on these systems will be shorter than normal system. Thus, we will also port the Linux wireless driver onto a target machine with real-time characteristic.

3.2 Mixed Mode Latency Evaluation

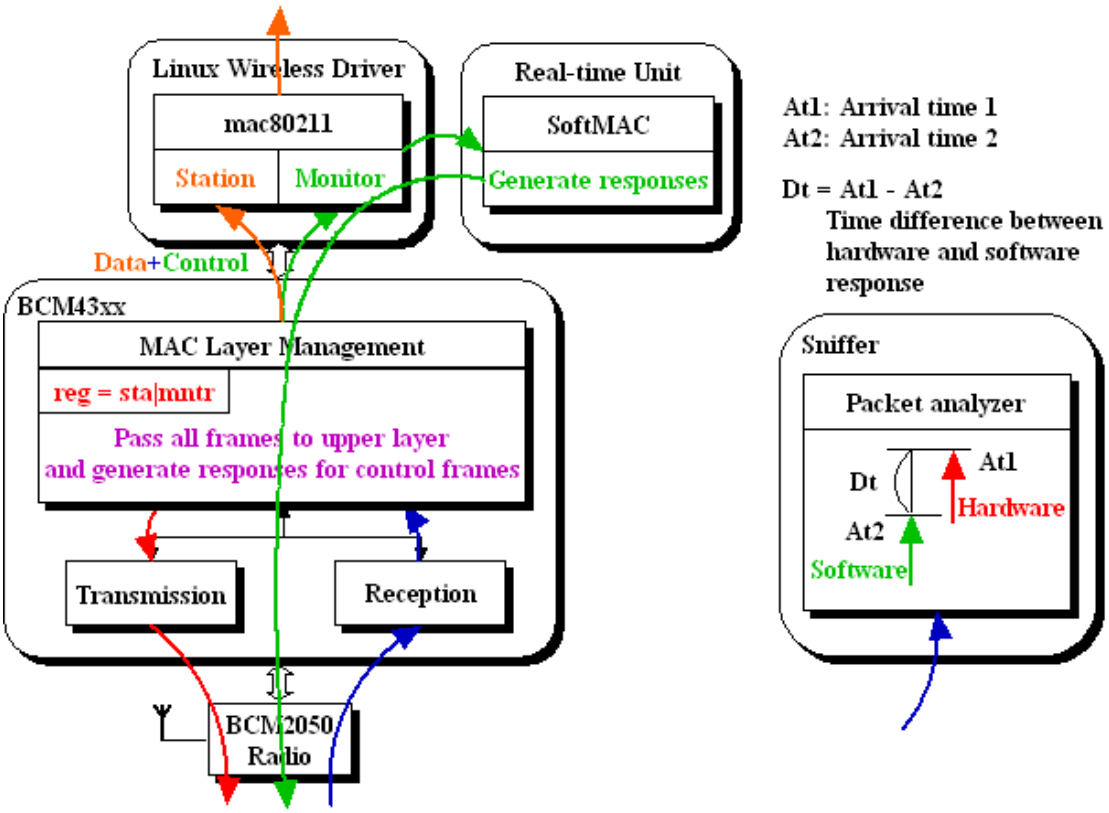


Figure 3-2: Mixed mode for latency measurement

The difficulty of the SoftMAC implementation is the timing limitation. Therefore, we need a platform to evaluate the time difference between software and hardware. A sniffer with packet analyzer can be employed for this task. In order to compare the arrival time for both responses, the host must be able to generate both responses in hardware and software

concurrently. The sniffer can thus calculate the time difference between different frames. Figure 3-2 depicts the inner architecture of the host and how the sniffer calculates the difference. The host is capable of mixing station and monitor mode so that control frames can also be received by the upper layer but not filtered by BCM43xx. Whereas the BCM43xx reply to the control frames as usual. The sniffer intercepts the responses and records the arrival time for each frame. The Dt can be calculated by subtracting At2 from At1. Figure 3-3 illustrates the time sequence of frame traversal between different components.

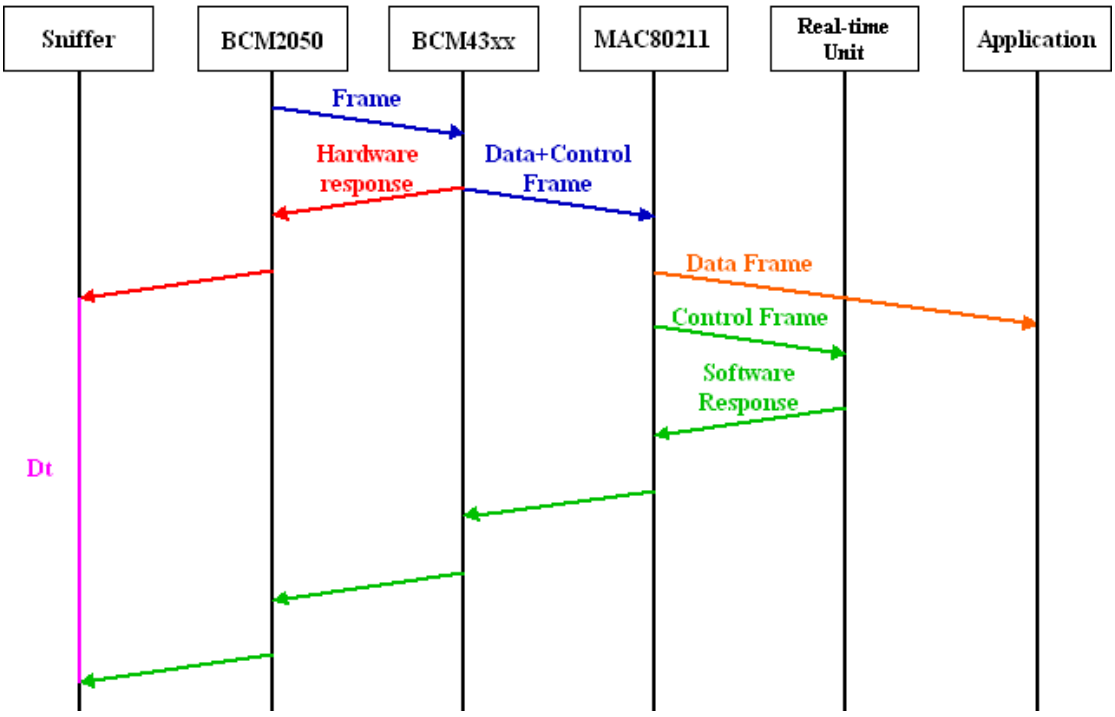


Figure 3-3: Timing diagram of packet transmission in mixed mode

However, the real-time unit for generating responses is not discussed in this paper. It can be in any form that is helpful to accelerate the frame generating task. Such as real-time task supported by some real-time operating systems.

Chapter 4 Implementation

4.1 Setting Up Environment

To set up the experiment environment, we need an AP and two hosts with wireless adapter. The WHR-HP-G54 is used as the AP and wireless adapter WLI2-PCI-G54S is equipped to the hosts running Linux 2.6.23.9 kernel. One of the hosts is employed as our SoftMAC platform so the Linux wireless driver has been installed onto the host. Besides, the code modification will be performed on this host. Another host became the sniffer that intercepts the packets sent by SoftMAC platform, thus Wireshark has been installed onto the sniffer. Wireshark is the tool that we used for capturing the packets.

4.2 Mixing the Modes

As discussed in previous section, there are different receiving path in protocol stack as well as on BCM43xx chip. A code modification is needed for mixing the receiving path. The microcode executed on the chip manipulates the frames according to the register value. We need to set the B43_MACCTL_PROMISC bit which enforces the adapter to enter promiscuous mode. In promiscuous mode, the host can receive any packets even if the packet is not destined to the host. Moreover, the B43_MACCTL_KEEP_CTL bit must be set for preserving the control frames instead of filtering and responding them in the chip.

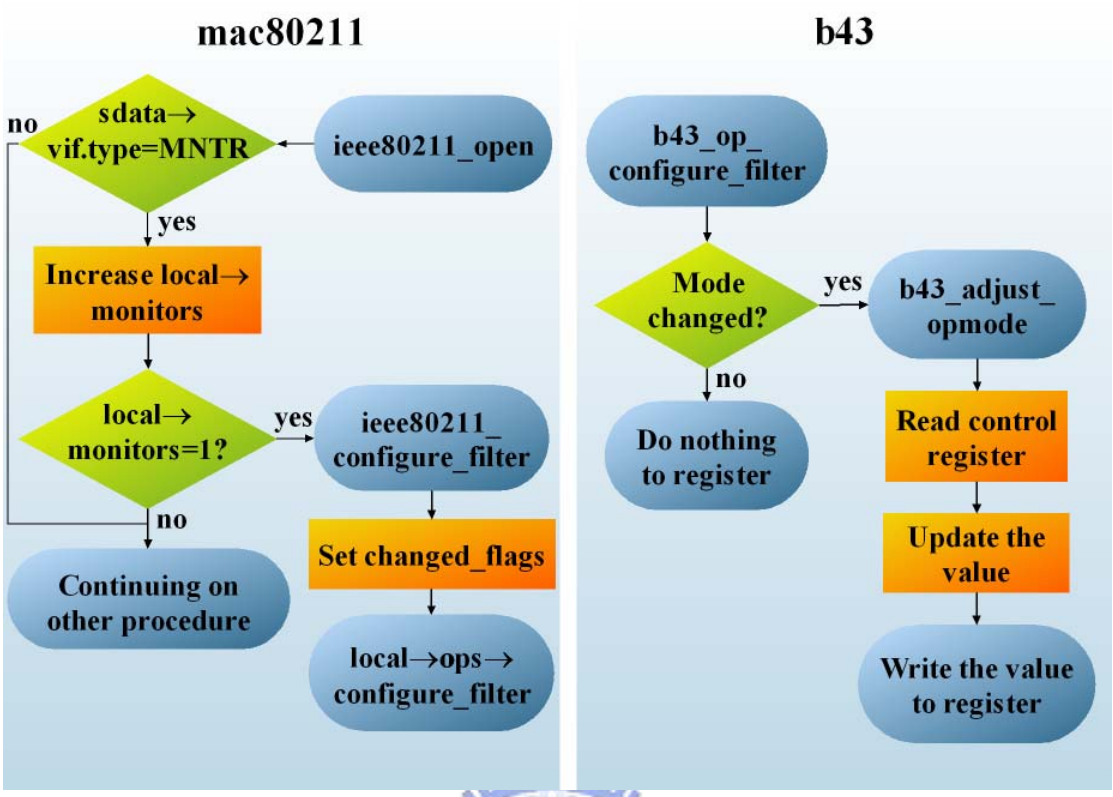


Figure 4-1: Filter configuring procedure

As shown in figure 4-1, the MAC control register is set by the function b43_adjust_opmode in module b43, whereas the b43_adjust_opmode is called by b43_op_configure_filter. If the mode has been changed, b43_op_configure_filter will invoke b43_adjust_opmode to change to the requested mode. However, b43_op_configure_filter is invoked as a callback function by local→ops→ configure_filter in module mac80211. The mac80211 invokes local→ops→ configure_filter by ieee80211_configure_filter which sets the changed_flags prior to local→ops→configure_filter. The ieee80211_configure_filter invokes the callback function according to the local→monitors which is set in ieee80211_open. The ieee80211_open has different treatments for different modes. If the interface type is MNTR, local→monitors will be increased and proper configuration will be set. Therefore, we can set local→monitors to 1. The driver b43 will configure the register automatically if the flag was set.

Furthermore, the main receive path is the function prepare_for_handlers. The path for

station mode filters the frames which contain different BSSID. Even if the BSSID matches, the Ethernet address must also be compared or multicast flag must be examined. The packets can only be received when BSSID matched and Ethernet address matches or it is a multicast packet. In order to mix the receive path, we have to remove these filtering procedures for station receive path.

4.3 Porting to Embedded System

BCM6358 is our target machine for verifying the design. It is a MIPS based embedded system and equipped with BCM4318 wireless adapter. It runs on Linux 2.6.8.1 operating system while Linux wireless driver requires kernel 2.6.22 or above. In general, kernel modules are highly dependent on contemporary kernel. Without corresponding linkage to a kernel object, the module cannot be compiled correctly. The linked kernel object offers the functions that required by the module.

In addition, the hardware architecture between target and desktop is also different. Since x86 processor provides hardware random number generator, the driver utilizes the component to generate random number. Nevertheless, MIPS do not support this function so that we have to use software random number generator instead. Furthermore, Linux wireless driver employ mutex lock to serve the critical sections, whereas kernel 2.6.8.1 supports spinlock but not mutex lock. Thus, the mutex lock must be replaced by spinlock.

Moreover, the Linux wireless driver we used is an independent package that is not integrated into kernel release. Although the Linux 2.6.24 includes the driver, it was not released during this project. Therefore, we integrated it into the Linux kernel that we used. In order to integrate the modules into kernel, the Makefile must be modified and Kconfig must be added at first. Secondly, the header files must be compared to see the differences between them. There are different approaches for header file replacement which depends on how

different they are. a) add/modify the statement in the header file. b) replace the header file with new one. c) header files coexist in different directories. Finally, add or modify the kernel files for the differences by referring to kernel 2.6.23.9.

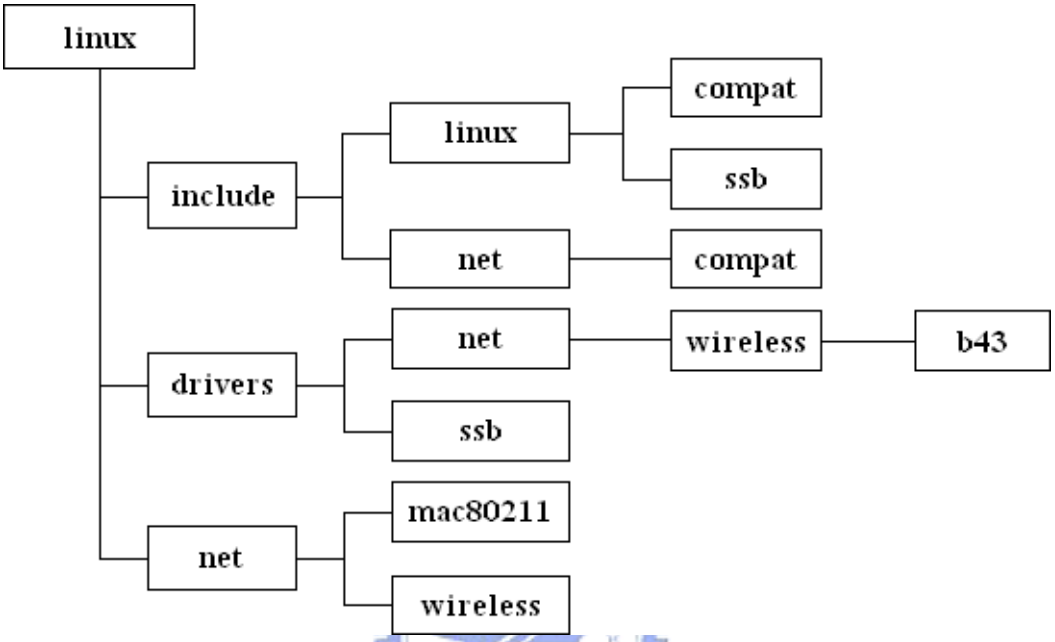


Figure 4-2: Integrating Linux wireless driver into kernel

As shown in figure 4-2, the Linux wireless driver is distributed into two directories, drivers and net. The Makefile in these directories must be modified to add the modules path so that the options can be seen in menuconfig. Furthermore, some header files have to exist in unique directory other than original directory.

Chapter 5 Results

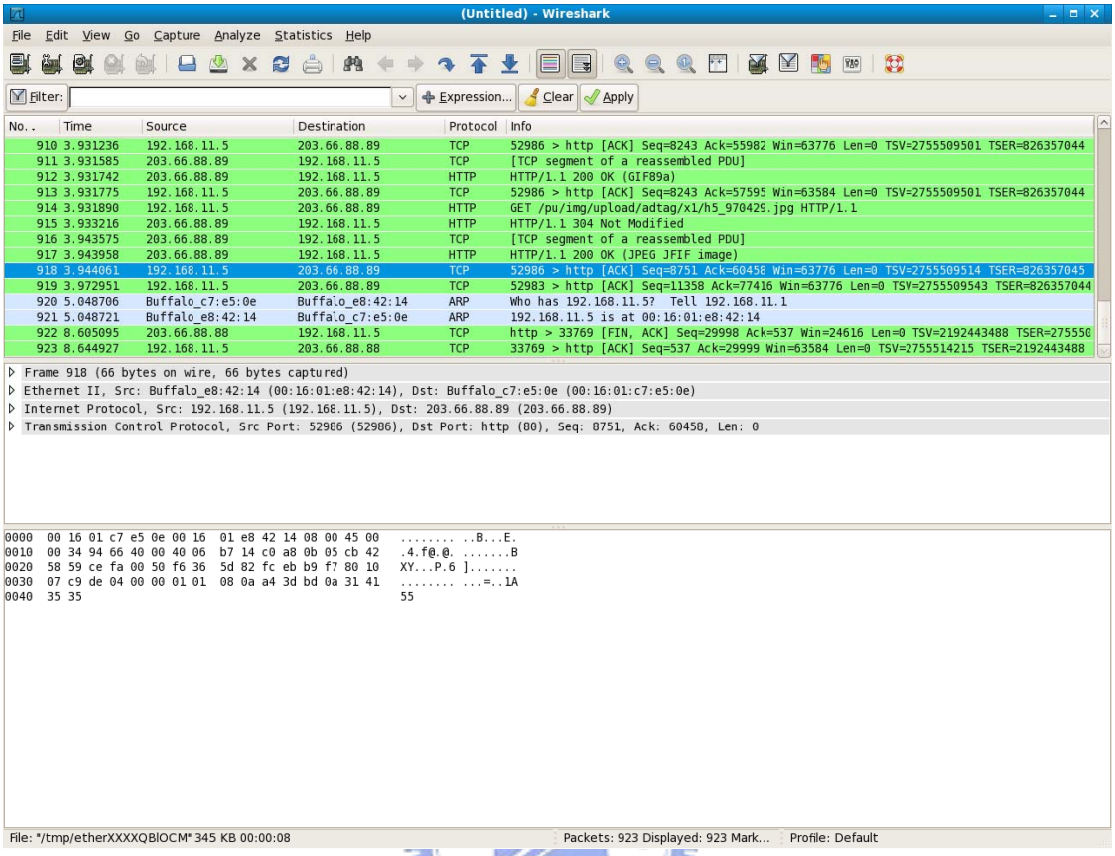


Figure 5-1: Packets captured in station mode

Figure 5-1 shows the Wireshark captured results for station mode operation. The adapter operates as a station and the received packets are decoded as the Ethernet frames. As shown in figure 5-2, the adapter captures all packets that appeared within the transmission range while we configured the adapter into monitor mode. The radiotap header is retained if the packets were received in monitor mode, and Wireshark decodes the packets as IEEE 802.11 packets. After modified the code to mix the station and monitor modes, the adapter still operates well as it would be in station mode. Nonetheless, the control frames are also received by the adapter. Since Wireshark decodes the frames as Ethernet frame, the improper decoding mess up the frame information as we can see in figure 5-3. However, we can still recognize the control frame from contents.

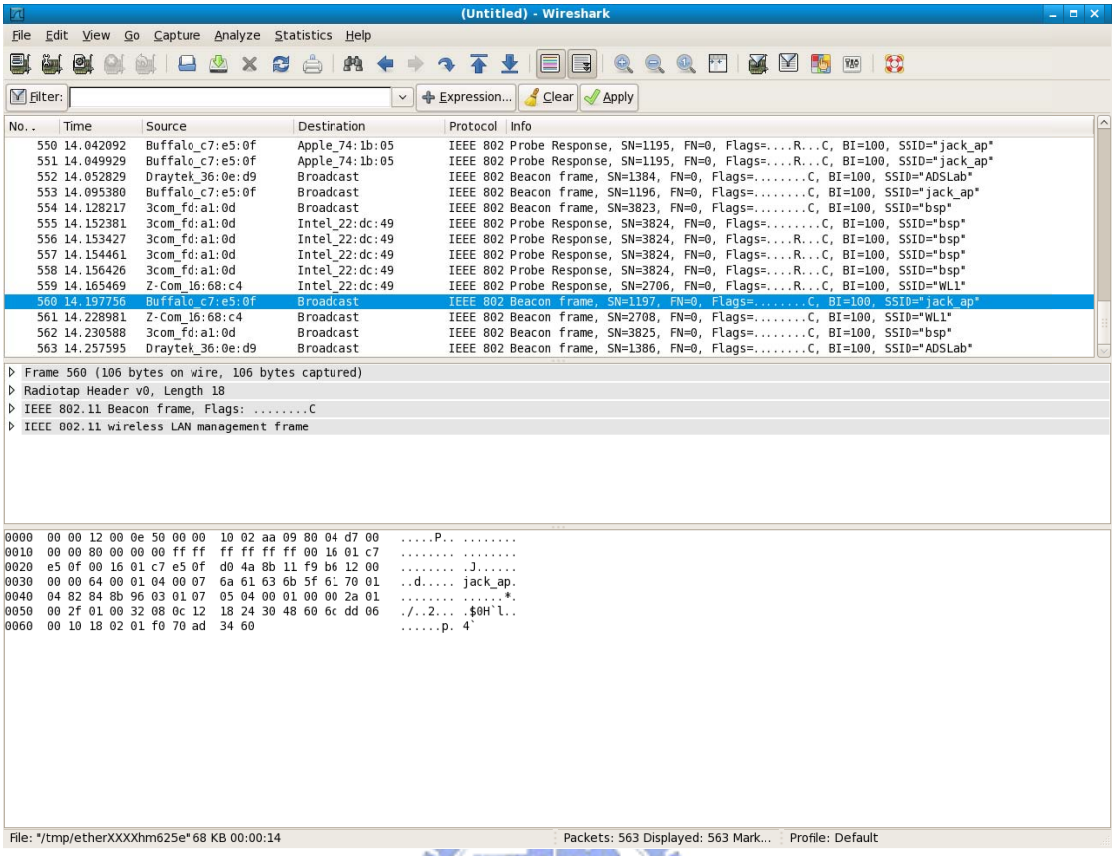


Figure 5-2: Packets captured in monitor mode

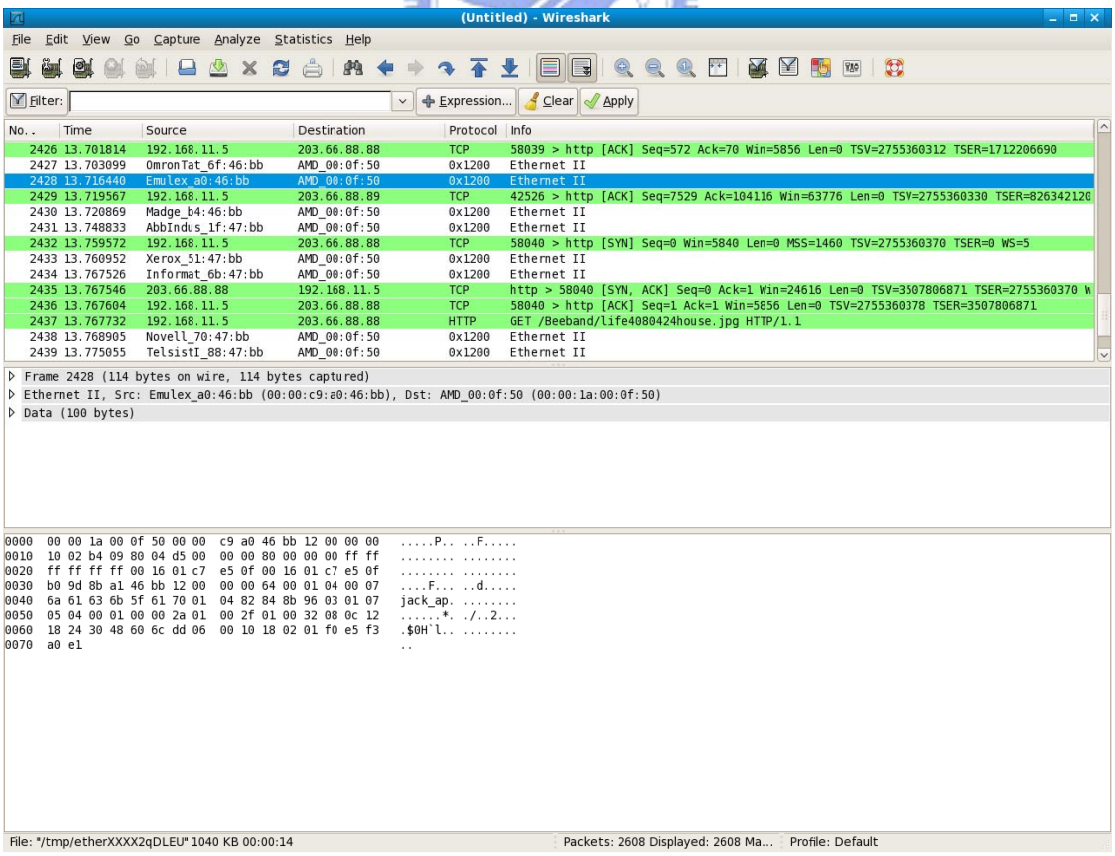


Figure 5-3: Packets captured in mixed mode

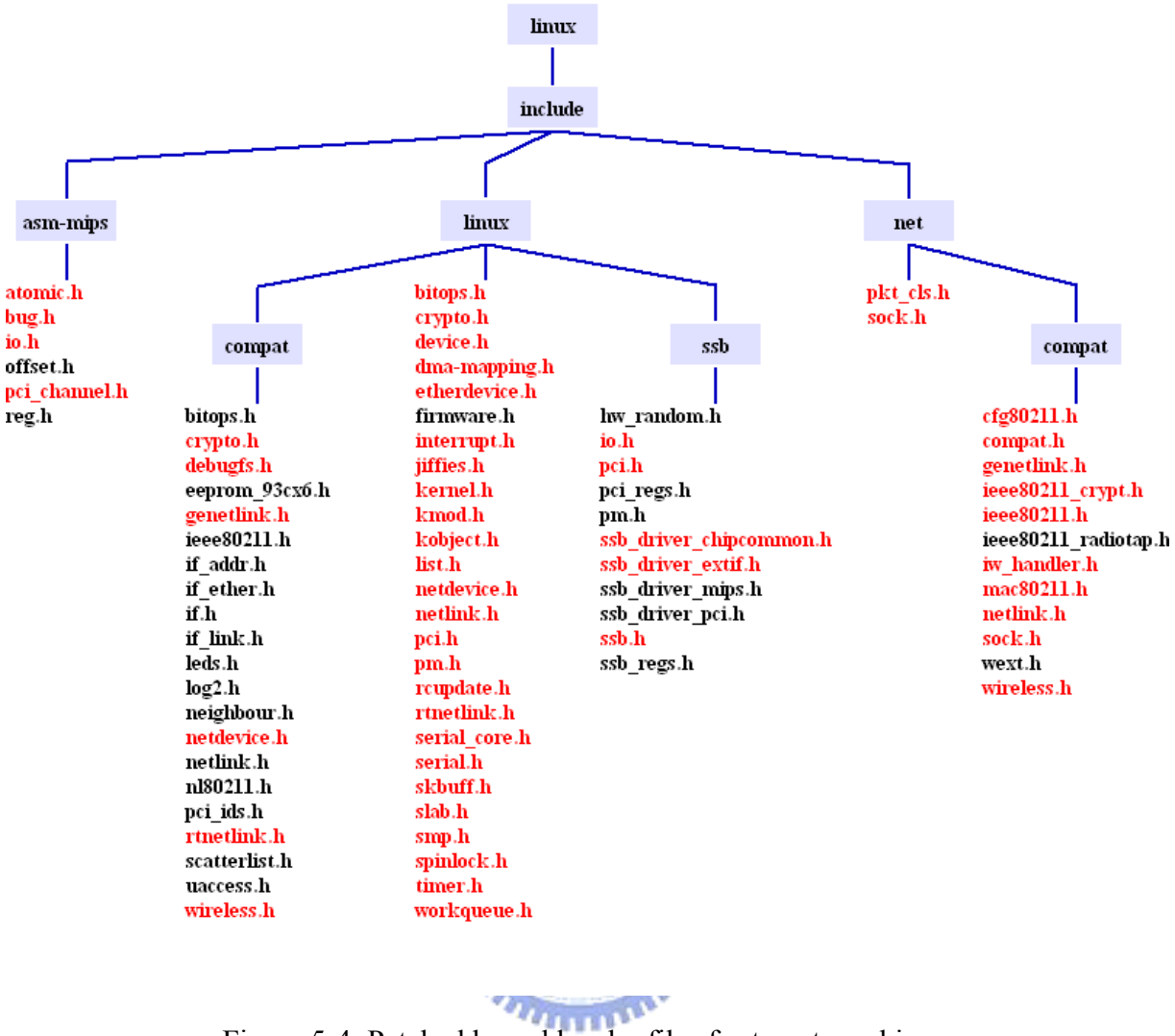


Figure 5-4: Patched kernel header files for target machines

For the porting procedure, the wireless modules cannot be compiled correctly at the beginning. By modifying the kernel module linked with the wireless modules, the wireless modules can be compiled correctly. However, there are still some routines needed by the wireless modules that do not exist. We patched the kernel by referring to kernel 2.6.23.9. After all the procedures being added or modified, the wireless modules can be inserted correctly. Figure 5-4 lists the header files that have been modified. The kernel source files have been listed in figure 5-5. The modified files are in red.

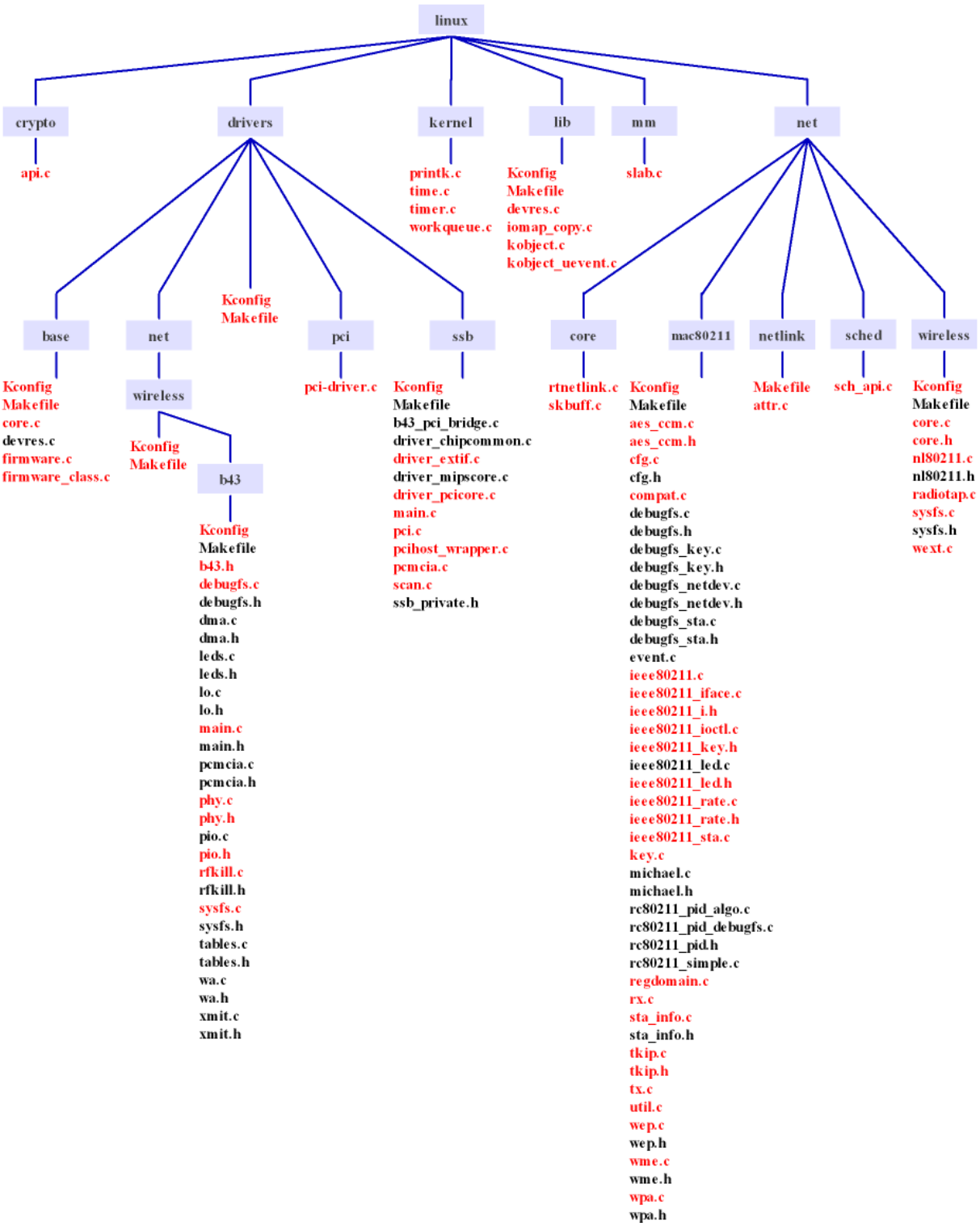
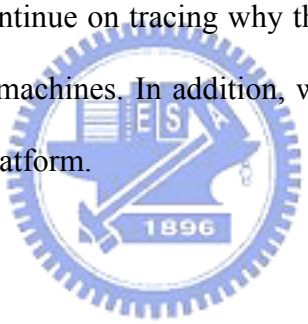


Figure 5-5: Patched kernel source files for target machines

Chapter 6 Conclusion and Future Work

This paper has investigated the mechanism for facilitating SoftMAC latency measurements. In order to compare the time difference between hardware and software, we proposed the mixed mode transmission. Both the hardware-wise and software-wise MAC responses must be transmitted concurrently. The sniffer can therefore intercept these frames and calculate the time difference between them. Our experimental results show that the driver can be modified to receive control frames so that further study can be done to the real-time SoftMAC. Although the ported wireless modules cannot operate as expected on target machines, we have done the same modification on desktops. We proved that hardware random number generator can be replaced with software. Furthermore, mutex lock can be substituted by spinlock. In the future, we will continue on tracing why the wireless modules cannot operate completely as expected on target machines. In addition, we will move towards the real-time SoftMAC that is suitable for the platform.



References

[1] Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth, and Dirk Grunwald, "SoftMAC – Flexible Wireless Research Platform, " Proc. HotNets-IV, College Park, Maryland, USA, Nov. 2005.

[2] Hyunseok Lee, and Trevor Mudge, "A Dual-Processor Solution for the MAC Layer of a Software Defined Radio Terminal," CASES 2005, San Francisco, California, USA, Sep. 2005.

[3] John Bellardo, and Stefan Savage, “802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions.” In Proceedings of USENIX Security Symposium, August 2003.

[4] IEEE 802.11-1999, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, " Jun. 2003

[5] http://linuxwireless.org

[6] http://bcm-v4.sipsolutions.net