

# Multi Channel Wi-Fi Sniffer

Pradeep Reddy B, Hari Sharma and Dominic Paulraj

Arada Systems Pvt Ltd, #31, TBR Towers, JC Road, Bangalore, India

**Abstract** — Multi channel Wi-Fi sniffer is a system level solution that is scalable and modular, where the device can be used to sniff all the channels in 2.4GHz ISM (Industrial Scientific and Medical) radio band and multiple channels in 5GHz wireless spectrum simultaneously. The aim of the work is to provide intelligence support and detection of threats on open wireless networks. Since the device can sniff all the channels simultaneously, there is no chance of missing packets in a particular channel. Also in the suggested solution, there is no overhead of switching channels as in the case of existing single channel sniffer implementations.

**Index Terms** — Multi Channel Sniffer (MCS), Single Board Computer (SBC), Host-SBC Protocol (HSBC), Mini Peripheral Component Interconnect (Mini PCI), Wireless Local Area Network (WLAN), Packets per Second (PPS)

## I. INTRODUCTION

Multi Channel Sniffer (MCS) implementation is a system solution catering to multiple applications, which includes analyzing the wireless traffic in public hotspots to detect threats, to check on wireless environmental conditions and for any other applications that require complete Wi-Fi [1][4] spectrum information.

There are many standard implementations of single channel Wi-Fi sniffer solutions [7] available in the market. But to sniff multiple channels, the currently available solution allows sniffing each channel for a particular duration before going to the next channel. With this option, there is an overhead in switching channels. Also some of the packets in other channels will be missed, while it is sniffing on a particular channel. So capturing all the packets in all channels is not possible with existing solutions.

Another way of tackling this problem is to have multiple machines equipped with single channel sniffers, where each machine's sniffer is sniffing one channel. This solution is very expensive, cumbersome and is not very practical with respect to deployment.

The MCS addresses the short comings in existing sniffer solutions for multi channel sniffing. The work is planned as an embedded solution with four standalone boards, referred as Single Board Computer (SBC). Each SBC has provision for multiple Mini PCI form factor wireless cards using Atheros Communications wireless chipsets. Each SBC, from a third

party vendor, supports 4 Mini PCI Cards. Four such boards are stacked to have 16 Mini PCI cards available for sniffing. Each card is dedicated for one channel and the system all together can sniff sixteen different channels simultaneously.

The paper is organized as follows. Section 2 describes the complete MCS system architecture. Section 3 details the required hardware specifications of system. Section 4 describes software specifications. Section 5 explains the base implementation of MCS prototype. Section 6 presents results and analysis of prototype implementation. Section 7 describes the necessity for optimization, scope for optimization and optimized components. Section 8 presents the results of optimized system. Section 9 concludes the paper and section 10 describes the possible future enhancements and extensions.

## II. SYSTEM ARCHITECTURE

MCS is an embedded solution divided into two sub-systems. One subsystem consists of on the Personal Computer (PC), running either Linux or Windows, referred as "Host". The other subsystem consists of a set of SBCs, with wireless card support present, referred as "Target". The communication between the subsystems will be through the Ethernet interface. The applications running on the Host communicates with Target and controls behavior of each individual wireless card used for sniffing. The complete system setup is shown in Figure 1.

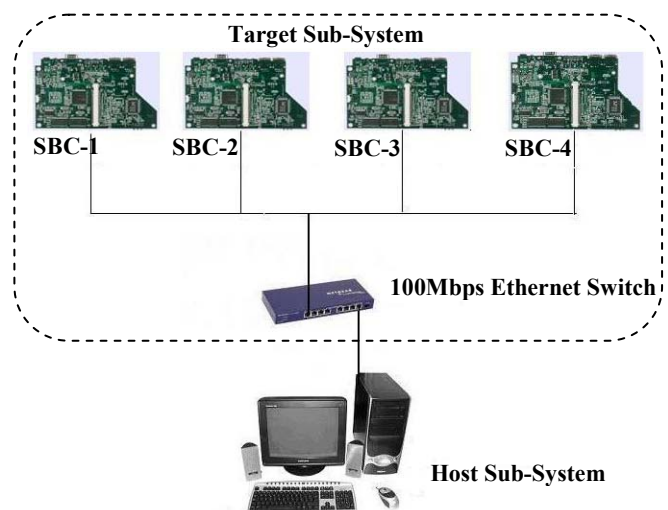


Figure 1. System Setup

### A. System Components

The hardware components in Target subsystem are: a) Third party vendor boards with multiple Mini PCI slot support b) Mini PCI cards using Atheros wireless LAN chipsets c) External Antennas d) 100Mbps Ethernet switch.

Software components on Target are: a) Board Support Package (BSP) for the SBC b) Wireless Local Area Network (WLAN) driver [5] in Linux for wireless Mini-PCI cards c) Server application, which acts as an interface between WLAN driver and host subsystem

The hardware components in Host subsystem are: a) PC running either Linux or Windows b) Ethernet Interface.

Software components are: a) Host API (Application Programming Interface) library, exposing set of APIs to provide complete control over target subsystem b) Socket-wrapper library, which can run in both Windows and Linux and used by Host API library to communicate with Target c) Sniffer application using Host API library.

## III. HARDWARE SPECIFICATIONS

The main criterion for selecting the SBC is that it should have support for maximum number of Mini PCI slots. This selection condition satisfies the requirement of having minimum number of SBCs to provide support for required number of wireless cards. Different embedded platforms are evaluated for the same requirement along with other factors such as processing power and memory support.

### A. SBC Specifications

A typical configuration required for the SBC is

- CPU, with processing speed of at least 533 MHz
- 4 Mini PCI Slots
- 128 MB memory (SDRAM) and 16 MB flash
- 100Mbps Ethernet Port

For SBC, power source required is 3.3V with 12A rating. To enable mobility of the product, a 3.3V source from a car battery can be used.

General Information about SBC is summarized in Table 1 below:

**Table 1. Typical SBC specifications**

Component	Description
Processor	Processor Operating 533MHz
Memory	128MB SDRAM
Flash	16MB
Physical Ports	4 Mini PCI slots 100 Base-T Ethernet Port
DC Supply	3.3V
Typical Operating Power (@ DC=3.3V)	3.6 W (Without Mini PCI) 6.1 W (With 4 Mini PCI cards only with receiving mode enabled)

### B. Mini PCI Specifications

Mini PCI cards should have dual band (2.4GHz and 5GHz) support. Each Mini PCI card should have antenna connector points to place external antenna.

### C. Antenna Specifications

A 3-in-1 integrated 4dBi omni directional antenna is used. The integrated antenna architecture reduces the physical count of antennas showing up on the device.

Mini PCI cards have very less coverage when used with internal antennas. External antennas are required for increasing the range of sniffing. Without external antenna, coverage is below 10m and with antenna it is around 30m. This data comes from the initial testing, which was conducted with the prototype. There were tests conducted with Access Point (AP) [4] placed at different distances for calculating the range.

### D. PC Specifications

Typical specifications of a PC, required for running MCS Host subsystem are Pentium IV dual-core processor operating at 2GHz, 2GB of RAM and a 100Mbps Ethernet port.

## IV. SOFTWARE SPECIFICATIONS

### A. Software Components

- 1) Boot Loader (RedBoot)
- 2) Operating System (SnapGear Embedded Linux)
- 3) Wireless LAN Driver
- 4) Host Single Board Computer (HSBC) API Library
- 5) Socket Wrapper Library
- 6) Sniffer Application

#### 1) Boot Loader

Boot Loader used is standard RedBoot [8] loader with support to network and serial interfaces. The loader runs from the target platform's flash boot sector, when the system is initially powered on.

Once RedBoot is run, the kernel image from flash gets loaded based on RedBoot script configuration. The RedBoot network/serial interface support is used for upgrading the firmware running on each SBC.

#### 2) Operating System (SnapGear Embedded Linux)

SnapGear's [6] version of embedded Linux, which is called as uCLinux, is used as Operating System (OS) on SBC. uCLinux has Linux kernel version 2.4.27. The OS resources used are low level PCI bus driver and network driver support.

#### 3) WLAN Driver

Base Linux wireless LAN driver [5] is modified for MCS implementation. The modifications are as follows:

- Support for multiple wireless cards
- Enumeration on the PC side for cards present in target subsystem

- New ioctls (Input/output controls) [5] for providing control over configuration of Mini PCI card like setting channel, mode of wireless operation, controlling the antenna diversity and selecting the antenna. Also support is added to provide information about all cards present in a particular SBC
- Support for application specific hardware information useful for sniffer applications
- Wireless cards are operated in promiscuous mode [5]

#### 4) Host Single Board Computer API Library

A communication protocol is defined between host and target subsystems. This protocol is termed as *Host Single Board Computer* protocol (HSBC). Host side HSBC APIs are used to control the target subsystem. This internally uses socket library APIs to communicate with target application using Ethernet interface. This set of APIs can be used by end-user application to initialize, configure and control wireless cards.

#### 5) Socket Library

Socket library is a wrapper library that can work on both Windows and Linux. The library contains all socket-level APIs [3] wrapped into one library independent of OS. The library is used in both host and target sub-systems for communication.

#### 6) Sniffer Application

This application runs on Host subsystem and uses HSBC APIs to control the Target subsystem. The application also provides configured output to the end-user.

### B. Layered Software Architecture

The layered software architecture for the MCS system is shown in Figure 2.

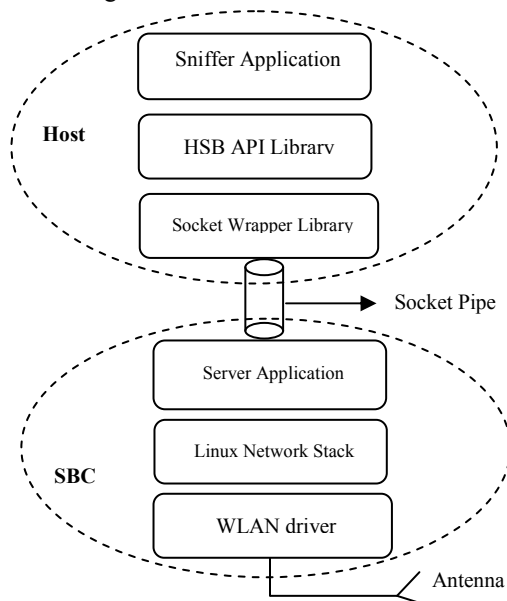


Figure 2. Layered Software Architecture

Figure 2 shows the various software components, explained in previous subsection, and how the communication happens between host and one SBC.

## V. BASE IMPLEMENTATION DETAILS

This section is divided into four sub sections A) Host B) Target C) WLAN Driver D) HSBC protocol

As soon as SBC is powered on, all 4 Mini PCI cards get detected and WLAN driver gets loaded. Each SBC has one target application, called *Server*, running on power up. The host application can connect to target servers for controlling the wireless cards, configuring them. Server application acts as an interface between host and driver on each SBC. The physical interface between host and target subsystem is Ethernet.

### A. Host

In enumeration phase, host connects to server applications running in four SBCs and collects wireless card's information like MAC address, wireless modes (IEEE 802.11 a/b/g) supported.

Based on above information, each wireless card can be put into one of the available channel, in a selected wireless mode. Then host sends a HSBC command to server to put the device in sniffing mode. The active devices capture wireless packets. These captured packets will be routed through socket interface between host application and target server application, created during initialization phase.

All the captured packets will be buffered at HSBC API layer and passed one packet at a time to end-user application on request through an API.

### B. Target

On power up of SBC, server application is initiated using startup scripts and it waits for incoming connection through a socket. During enumeration, server application collects information of wireless cards using ioctl interface support in WLAN driver [5].

The server application is the target side of HSBC communication protocol. It receives commands to initialize and configure wireless cards. The server application uses ioctl interface of driver for processing these commands. The response of requested command is sent back to the HSBC Host side protocol.

For getting captured packets from WLAN driver, server uses RAW Packet socket interface. New protocol family called, PF\_PACKET [2] is added in post 2.0 Linux kernel. SOCK\_RAW type of socket in this family allows application to directly get the packets from network driver. Then server passes these packets (one packet at a time) through Ethernet interface to host.

### C. WLAN Driver

The WLAN driver's role with respect to supporting sniffer functionality gets concentrated to optimized control of the receive path in the driver. Also additional support is added in the driver to control the receive characteristics and hardware features. The driver is enhanced to support multiple card architecture in terms of device resources as well as performance.

Once packet processing is completed at driver layer, captured packets are passed to Linux network stack, from where they will be transferred to server using RAW application socket interface. Physical layer information like channel in which the packet has been received, rate at which it has been received and signal level, will be populated in a proprietary header, attached to each packet.

### D. HSBC Protocol

As explained in section-IV, HSBC protocol is defined for communication between host and target subsystems. This protocol internally uses socket calls [3]. For all control commands (in both directions, host to target and vice versa) and data commands (only in one direction, target to host), TCP sockets are used for reliability.

## VI. EXPERIMENTATION RESULTS OF BASE SYSTEM

This section details the results and observations about performance of base system implementation. All the experiments are done multiple times under same conditions to make the results consistent. The experiments are performed in a RF shielded environment to avoid effect of any external interference.

One existing single card wireless sniffer is taken as reference to evaluate the performance of MCS. All results are obtained by running tests on complete system setup shown in Figure.1.

### A. Results

Three traffic generators have been deployed to assess the performance at various rates of traffic. One is packet injector tool, which is used to generate traffic almost at a constant rate. This tool generates at a rate of ~2200packets/sec (pps). Second test is transferring a huge file wirelessly between two clients connected to an Access Point (AP) in secure mode of authentication [4]. This generates the traffic at a rate of ~3700packets/sec. Third one is same as second one, but wireless connection is in open authentication [4] i.e. without any security. This test generates traffic at a rate of ~6000packets/sec.

**Table 2. Performance results, mentioning percentage of captured packets, calculated with a reference sniffer**

No. of Cards active \ Rate of Traffic	~2200 pps	~3700 pps	~6000 pps
1 Card / SBC	100%	100%	98%
2 Cards/ SBC	100%	92%	80%
3 Cards/ SBC	100%	78%	44%
4 Cards/ SBC	100%	54%	28%

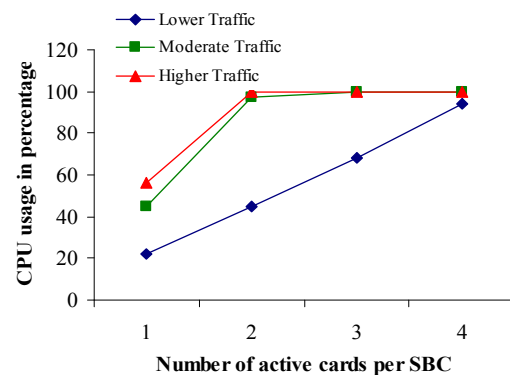
Table 2 shows the percentage of packets captured by MCS out of total packets captured by a reference single card sniffer. Each row mentions the statistics when certain numbers of cards are active per SBC in complete system. Each column specifies the specific rate of data traffic.

### B. Analysis and Observations

From above results, it is clear that each time adding one active wireless card to the system causes increase in packet loss at moderate and higher traffic rates. For lower rates (2200 pps) all four cards in SBC are able to capture 100% of the packets. Consider the extreme case, at higher rate (6000 pps) when all four cards per SBC are active the packets captured is around 28%, which is unacceptable packet loss of 72%.

Even though the SBC has limited processing power, which is one reason for excess packet loss, this shortcoming had to be overcome by optimizing the software architecture and existing implementation. So the base implementation needs to be optimized for better performance for higher traffic rates.

Using the statistics counters placed at various levels, it could be interpreted that the loss is mainly between WLAN driver and target server. The CPU usage is measured in SBC with top (Table of Processes) utility of Linux. Under moderate & heavy traffic rates, when single card is active in a SBC, CPU usage is 56%. For two cards it exceeds 100% of CPU usage, which leads to huge packet loss. Figure 3 shows how CPU usage varies with various traffic rates, when particular number of cards are active per SBC.



**Figure 3. CPU Usage (in %) of base system for various traffic rates**

## VII. OPTIMIZED IMPLEMENTATION

### A. Scope for Optimization

In WLAN driver, the complete receive path lies in the interrupt context and processed in the post ISR (Interrupt Service Routine) context. So both the components need an optimized implementation to reduce the load on the processor. Involvement of kernel and network stack in passing up the captured packets to application creates an overhead, which can be avoided for optimization.

Using TCP socket [3] for HSBC protocol communication creates a bottleneck for throughput of data packets communication from target to host. And the act of passing bunch of captured packets (“aggregate”) from target to host rather than transferring single packet at a time further improves the performance.

### B. Optimized Components

This section explains about optimized components of base system implementation to achieve better performance. Based on scope of optimization explained in previous sub-section, the optimized components are:

#### 1) Intermediate Character Driver

Instead of involving Linux network stack to hand over the packets to application, a new intermediate character driver [5][5] has been introduced into the base software architecture. This is termed as “mcsdev”.

Main functionality of *mcsdev* is intermediate buffer management. WLAN driver supplies the sniffed packets to *mcsdev* and it manages these packet buffers effectively before passing them to application.

#### 2) Using UDP for HSBC protocol data packets

For sending data packets from target to host, we need better throughput rather than reliability, without an overhead of acknowledgement for each and every packet. Using UDP for data communication reduces the load on both host target subsystems greatly. For HSBC commands, we still need reliability rather than throughput. So TCP sockets [3] are used for control commands and UDP sockets [3] for data packets.

#### 3) Aggregation

Instead of sending single packet at a time from target server application to host, sending bunch of captured data packets from target server to host, improved performance of HSBC protocol. Target server waits for either certain length of buffer to get filled or certain number of packets has been received and sends aggregated packet to host. The format of aggregated packet is shown in Figure 3.

Number of packets	Total length	Pkt-1 length	Pkt-1	Pkt-2 length	Pkt-2	.....
-------------------	--------------	--------------	-------	--------------	-------	-------

Figure 4. Aggregated frame format

In Figure 4, first field indicates total number of packets in

the aggregate. Second field is the length of the aggregate. Rest of the frame contains each packet’s length followed by actual packet.

#### 4) Zero Copy feature

With optimizations, described in section VII, SBC’s CPU utilization is reduced drastically. But still the copy of captured packet’s data from WLAN driver to intermediate character driver was consuming major portion of processing time of single packet. By using kernel timers, it could be concluded that this copy is an overhead and consumes a good amount of scheduler time.

So the “zero copy” enhancement is added to avoid a copy of data from WLAN driver to character driver. A chain of buffers will be given to wireless card’s DMA (Direct Memory Access) engine [5] at initialization and once the packet is completely received, buffer will be detached from buffer chain. The de-linked buffer will be passed to *mcsdev*’s queue manager directly without any copy. The buffered data then will be copied to user space buffer on application request. Only one copy of data, received at wireless adapter, is required for passing packet to application layer.

## VIII. EXPERIMENTATION RESULTS OF OPTIMIZED SYSTEM

This section details the results and observations about performance of optimized system implementation. All the experiments have been carried out under same conditions as that of base system.

### A. Results

Traffic generators used for these experiments are same as that of base system’s experiments. The performance results are tabulated in Table 3.

Table 3. Performance results of optimized system, mentioning percentage of captured packets, calculated with a reference sniffer

Rate of Traffic No. of Cards active	~2200 pps	~3700 pps	~6000 pps
1 Card / SBC	100%	100%	100%
2 Cards/ SBC	100%	100%	99%
3 Cards/ SBC	100%	99%	99%
4 Cards/ SBC	100%	98%	95%

Table 3 shows the percentage of packets captured by MCS out of total packets captured by a reference single card sniffer. Each row mentions the statistics when certain numbers of cards are active per SBC in complete system. And each column specifies the rate of traffic.

### B. Analysis and Observations

By observing the results mentioned in Table 3, it can be concluded that optimized system implementation has achieved

major performance improvement with available hardware resources. With this optimized implementation the load on both SBC and host processor has been reduced greatly, even when all four cards are active in a SBC. Figure 5 shows SBC's CPU usage in percentage for all kinds of data traffic rates.

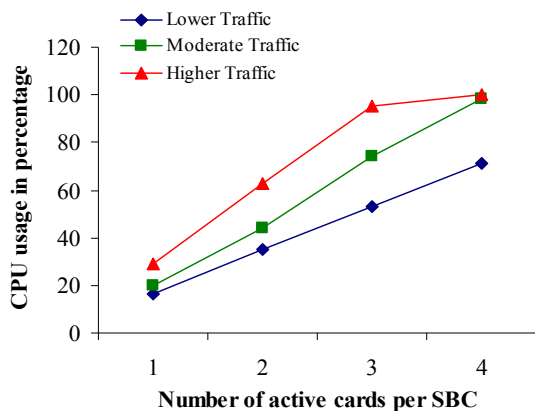


Figure 5. CPU Usage (in %) after optimization for various traffic rates

## IX. CONCLUSION

There is a definite need of a solution recommended in the paper for various reasons. The wireless LAN is fast penetrating into embedded and consumer space from the PC space. As the number of devices increases, there is lot of requirements for optimized wireless network setup. Also the security aspect proves to be a major requirement, since the communication using the WLAN would be taking a heavy upturn. Enhanced application using the APIs exposed by MCS would cater to additional support for surveillance, site survey and other security applications.

## X. FUTURE SCOPE

The solution has scope to be extended to support the most recent enhancements in the IEEE 802.11 [1][4] series of standards. The most promising and challenging scope for extension is in providing support for high throughput WLAN (IEEE 802.11n) sniffing. The High Throughput (HT) extension supports a data rate of 150 to 200Mbps and would require some hardware updates for supporting higher data rates. This includes faster dual core processor and support to dual Gigabit Ethernet interface to the host (PC) and definitely more flash and ram requirements.

## REFERENCES

- [1] IEEE Std. 802.11-2003, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11-1999 (R2003) edition, 2003.
- [2] The Linux Socket Filter: Sniffing Bytes over the Network. <http://www.linuxjournal.com/article/4659>.
- [3] W. Richard Stevens, "UNIX Network Programming," Volume 1, Second Edition: Networking APIs: Sockets and XTI, Prentice Hall, 1998, ISBN 0-13-490012-X.
- [4] Bob O'Hara, Al Petrick, "IEEE 802.11 Handbook" Published by: IEEE Press, ISBN 0-7381-1855-9.

- [5] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman "Linux Device Drivers," Third Edition. <http://lwn.net/Kernel/LDD3/>
- [6] SnapGear Embedded Linux Distribution, <http://www.snapgear.org/>
- [7] Mingzhe Li, Mark Claypool, and Bob Kinicki, "Wireless sniffing by example - how to build and use an IEEE 802.11 wireless network sniffer," Tech. Rep. WPI-CS-TR-05-19, Department of Computer Science at Worcester Polytechnic Institute, Nov. 2005, and Online: <ftp://ftp.cs.wpi.edu/pub/techreports/pdf/05-19.pdf>.
- [8] RedBoot User's Guide, <http://ecos.sourceware.org/docs-latest/redboot/redboot-guide.html>