A Configuration Protocol for Embedded Devices on Secure Wireless Networks

by

Larry M. Sanders

B.S.Co.E., University of Kansas, Lawrence, Kansas, 2000

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

_____
Joseph Evans

_____
Victor Frost

_____
Perry Alexander

_____
Date thesis accepted

Acknowledgements

The Author wishes to acknowledge the following people that have improved his experience while at the University of Kansas.

To Benjamin Ewy for his guidance and continued support.

To Joseph Evans, Victor Frost, and Perry Alexander for time serving on my committee and advice throughout the years.

To Brett Becker, Mike Hulet, Dan Depardo, and the rest of the Faculty and Students at ITTC.

Abstract

Wireless networking has enjoyed overwhelming success and Wi-Fi network deployment continues to grow at an astounding rate. Unlike wired networks, the wireless medium of Wi-Fi networks cannot typically be physically secured. On traditional wired networks, authentication is implicitly provided by physical access when the network cable is plugged in and there are no layer two network parameters. Wi-Fi networks do not have that luxury and for that reason, 802.11 specifications include authentication and authorization standards, as well as an optional privacy service called Wired Equivalent Privacy (WEP). This can be an effective means of securing a wireless network, however it requires additional configuration parameters to be entered on the Wi-Fi client. Configuration of wireless embedded devices with limited or no input capability is time consuming, at best. Wi-Fi-Co is a protocol and suite of tools to facilitate the configuration of layer two link level network parameters on a host within a Wi-Fi network. This document describes how a host on the network can remotely configure a simple Wi-Fi enabled device with the proper link level parameters, allowing the host to use traditional configuration protocols such as DHCP to obtain IP level connectivity.

Contents

List of Figures

Chapter 1

Introduction

1.1 What is Wi-Fi-Co

Wi-Fi-Co is a protocol and suite of tools to facilitate the configuration of network parameters on a host within a Wi-Fi network. The Wi-Fi Alliance was formed in 1999 to certify interoperability of wireless Local Area Network products based on the IEEE 802.11 specification [1]. The Wi-Fi Alliance supports the term Wi-Fi, for "Wireless Fidelity", and defines [2] it to encompass all wireless networks based on the IEEE 802.11 specification. In the remainder of this document, Wi-Fi will refer to 802.11a, 802.11b, and 802.11g based wireless networks. Wi-Fi-Co enables a configuring host to send network parameters, such as encryption configuration, network names, and other required parameters, to an un-configured host within a Wi-Fi network in order to facilitate link layer connectivity.

1.2 Motivation

Traditional network configuration protocols, such as Dynamic Host Configuration Protocol (DHCP), only configure Internet Protocol (IP) level parameters, and cannot be utilized on a wireless network until the host has layer two, link level connectivity. Layer two connectivity in a wireless 802.11 infrastructure network involves being associated with a network Access Point (AP) [3]. This includes configuration of the

network name (Service Set Identification or SSID) and encryption parameters (Wired Equivalent Privacy or (WEP) keys), if encryption is enabled. In the typical case of configuring a laptop computer on a Wi-Fi network, a user must enter the network name and WEP information by hand and then request IP network address parameters via DHCP. Wi-Fi-Co was created out of necessity while developing an embedded wireless device. Configuring such a device, which has no display, keyboard, or other human interface means, proves to be quite difficult. The usefulness of Wi-Fi-Co can also be extended to devices that have limited input capabilities, such as a Personal Digital Assistant like a Palm Pilot or iPaq.

With the explosion of deployed wireless networks and the plummeting cost of wireless chipsets, we are likely to see a large increase in wireless embedded devices. Desktop and Laptop workstations with Wi-Fi Network Interface Cards (NICs) have become commonplace today. Embedded products such as 802.11b web cameras have already hit the market. Wireless embedded networked sensors are next – measuring everything from temperature and moisture levels to poisonous gases and environment pollutants [4]. These devices will not reach their full potential unless then can easily and cost effectively be configured for the installed wireless networks.

The configuration difficulty arises because the target wireless host is not yet associated with network – its layer-two connectivity is not yet established. For a Wi-Fi host to begin communicating on a Wi-Fi network in a normal operational mode, the host must associate with the Wi-Fi network by sending an association request

with the valid SSID. In addition, if WEP is enabled, datagrams destined for the target host will be encrypted with a key that is unknown to the target host.

1.3 Project Goals

This project aims to:

- Define a protocol that enables Wi-Fi embedded devices to be configured via the existing network.
- Create a suite of tools that users and network administrators can use to configure wireless devices on a Wi-Fi network.
- Show the validity of the implementation by presenting test results from the configuration of various wireless hosts.

1.4 List of Accomplishments

The following was accomplished while implementing this system:

- A new protocol for configuring Wi-Fi hosts was designed.
- Configuring tools were created that run on multiple platforms.

1.5 Layout

This document is organized into the following sections:

- Background – Background information useful for understanding this and related works.

• Architecture – An explanation of the protocol architecture and requirements that are needed to configure a Wi-Fi host on a wireless network.

• Implementation – A detailed description of the implementation of a software system that follows the requirements in the architecture section.

• Results – Description of the test environments and test results from using the implemented system.

• Conclusions and Future Work – Lessons learned and recommendations for future research on this topic.

Chapter 2

Background

2.1 Vision

This work describes an architecture and implementation that realizes the goal of enabling an embedded Wi-Fi device to easily be configured within an existing Wi-Fi network. Like their equivalent wired counterparts, wirelessly connected embedded devices will soon be commonplace in networks. Unlike wired embedded devices, wireless embedded devices are unable to immediately utilize existing higher-level configuration protocols like DHCP to achieve IP connectivity. A wired embedded device simply has to be plugged into an active Ethernet port and it implicitly has Link Layer connectivity. A Wi-Fi station has a much more complicated physical layer, that must be established, and must perform several steps that generally require the station to know several parameters first.

Many current embedded wireless devices such as Wi-Fi web cameras have a Universal Serial Bus (USB) or wired Ethernet interface to facilitate the initial wireless configuration. This solution adds cost and complexity and is not viable for

inexpensive embedded devices. Other embedded devices, such as PDAs, have clumsy input capabilities that make entering network configuration time consuming and tedious.

This work defines and implements a simple, easy to use solution for this problem.

2.2 Embedded Wi-Fi Market Segment

2.2.1 Wireless Network Deployment

The phenomenal growth in the Wi-Fi industry is nothing short of amazing. Wireless networking technology has existed for decades; however, with the standardization of 802.11b by IEEE in 1999 the wireless market has exploded. Laptop computers are obviously a wonderful fit with wireless networking and nearly all vendors are now offering a built-in Wi-Fi option. However, Wi-Fi is not limited just to laptop computers – many consumers find running wires for their home desktop workstations too messy and time consuming and prefer to simply purchase any one of the dozens of inexpensive Wi-Fi Nework Interface Cards (NICs) designed for the PC. Wi-Fi has also solved another problem – it can provide standardized network infrastructure where it was not previously possible, such as buildings deemed historical or that are too costly to run traditional wired solutions. In addition, nearly all PDAs support a Wi-Fi card and recent PDA's have Wi-Fi built-in.

Wi-Fi deployment has, in part, been fueled by the rapid growth of broadband Internet services. In the United States alone, an estimated 28 million households had a broadband Internet connection in 2002 [5]. Furthermore, In-Stat/MDR reports that at the end of 2001, 38 percent of those broadband home networks boasted Wi-Fi equipment [6]. A 2001 Campus Computing Survey found that of 600 public and private colleges and universities in the United States, 51 percent have wireless networks, an increase from 30 percent the year before [7].

The latest Wi-Fi buzzword is “hot spots”. Hot spots are Wi-Fi enabled areas in convenient public places. Hot spots are springing up in cafes, bookstores, hotels and airports in nearly every major city. Hot spots originally started showing up in various small areas like coffee shops and were primarily created by a tech savvy owner or employee for personal use. Hot spots are now becoming big business. Some are available to the public for free while others are subscription or pay-as-you-go based Internet access. In 2002 hot spots grew from under 2,000 locations to over 12,000 locations and continued rapid growth is expected for 2003 and 2004, according to In-Stat/MDR [8].

In December of 2002, Intel, AT&T and IBM created a new company called Cometa in a joint venture that is designed to sell more Intel chips, IBM consulting, and AT&T network services. Cometa plans to create hot spots and sell these services to telecommunications companies, Internet Service providers (ISPs), cable operators and wireless carriers, who can then offer their customers wireless Internet access. Cometa is hoping to create 20,000 hot spots by the end of 2003 [9].

Telecommunication leaders Verizon and Nextel are also very interested in Wi-Fi. While other communication companies are striving to create hot spots, Verizon and Nextel are targeting their existing corporate and small business customers [10,l1]. These moves come after competitor T-Mobile teamed up with Starbucks to offer Wi-Fi services to T-Mobile customers nationwide in any one of the popular nationwide coffee shops.

Wi-Fi "hot spots" could also save a slowly dying fixture – pay phones. With the saturation of cellular phones, pay phones are becoming more and more obsolete. Many phone companies are considering the idea of retrofitting existing pay phones with DSL and Wi-Fi public access points. Pay phones are ideal for instant Wi-Fi access points because of their existing infrastructure (power, right-of-way, cabling) and their typical high traffic locations like restaurants and airports [12].

2.2.2 Wi-Fi Chipset Projections and Price Declines

Even in the declining economy of 2001, Wi-Fi managed to thrive. In 2001 alone, In-Stat/MDR reports eight million Wi-Fi chipsets were sold worldwide [13]. In addition, In-Stat/MDR estimates annual sales of Wi-Fi chipsets to exceed 70 million units in 2006, which might be a conservative number with recent announcements from industry giants Intel, AMD, IBM, AT&T, and BroadCom.

In September of 2002, Intel announced its Banias line of chips for mobile computing [14]. The Banias chips come standard with a built-in Wi-Fi function and are to be installed in more than 20 million laptop computers shipped in 2003. Intel forecasts that some 60 million computers worldwide will contain Banias chips and its

Wi-Fi feature by the end of 2004. In October of 2002, Intel continued its Wi-Fi quest by announcing investments of $150 Million in companies developing high-speed wireless networking technology [15].

One of the greatest battles in the computer industry has been between hardware leaders Intel and AMD to continually bring the most powerful PC chip to the market each quarter. This legendary war kept the label “fastest pc processor” bouncing between the two every few months. AMD recently announced that it would no longer strive to compete with Intel on this issue and would instead focus its efforts on "customer-centric" technology and new 64-bit architectures [16]. Now it seems there is a new battle raging between the rivals with AMD announcing in a November 2002 press release its entry into the wireless market. AMD has developed its first Wi-Fi chipset (Alchemy Am1772) and plans to integrate the two-chip solution into its MIPS technology-based AMD Alchemy Solutions SOC processors [17]. AMD may beat Intel’s Banias line to the market and has future plans for many other Wi-Fi products.

All of these factors are driving down the price of Wi-Fi chipset even faster than the initially optimistic estimates. Wi-Fi Access Points that cost $1,000 in 2000 were available for $95 in late 2002. Wi-Fi PC NICs were available for as little $35 in late 2002 as well.

2.2.3 Embedded Systems

The microcontroller industry has enjoyed strong growth from when it was first introduced in 1980 [18]. Microcontrollers (MCUs) are virtually everywhere these days. There is likely more then a couple MCUs in your TV, DVD player, remote control, alarm clock, camera, and as many as five dozen in newer cars. MCUs contain an entire computer's primary functions on a single chip and 8-bit MCUs can be as cheap as $0.60 per unit. Total worldwide revenues of MCUs were nearly $11 billion in 2001 [19]. In-stat/MDR expects the MCU market to grow for at least another 5 years and estimates that some 5 billion units were shipped in 2001 [20]. Microcontrollers continue to be in such demand for many reasons; they are small, cheap, versatile, and powerful. They are easy to program and decrease time-to-market for the end-products they are integrated into.

The next big wave in Wi-Fi deployments is to move beyond traditional computers and support the over 5 billion embedded computers based on MCUs sold each year. Wi-Fi combined with inexpensive microcontrollers open up a whole new world of possibilities.

2.3 Supporting Technologies

In 1980, the Computer Society of the IEEE started Project 802, a project dedicated to creating standards to enable intercommunication between equipment from a variety of

manufacturers. The 802 Project is very modular approach and attempts to specify functions of the physical layer, the data link layer, and the network layer to support interconnectivity of major Local Area Networks. The 802 Project attempts to follow the Open Systems Interconnection (OSI) model [21] closely and has divided the OSI data link layer in two sublayers: the logical link control (LLC) and the Media Access Control (MAC).

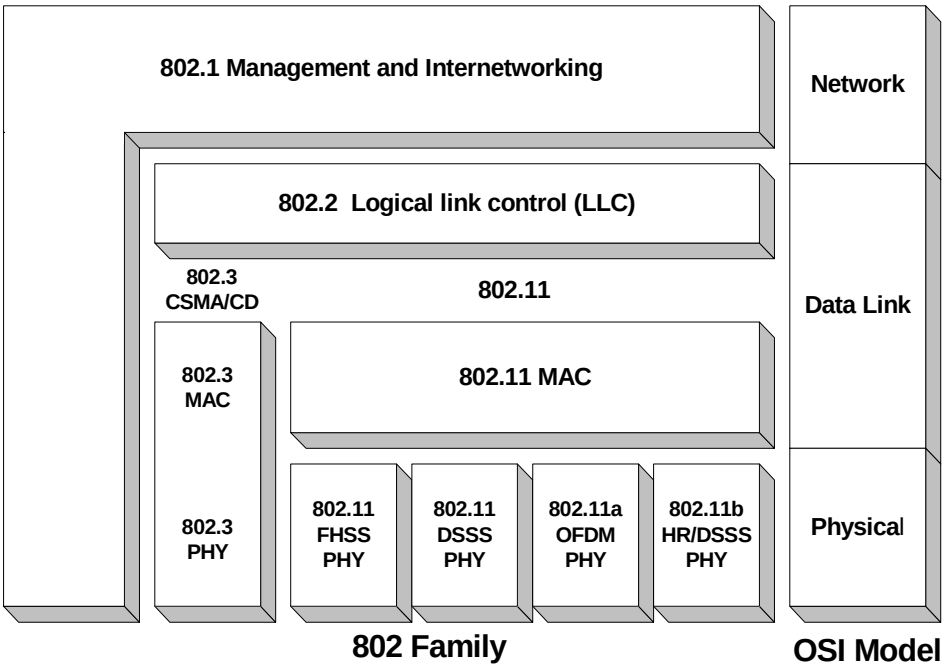


Figure 2.1 802 Family and OSI relation

2.3.1 802.11

In 1990 the Institute of Electrical and Electronics Engineers (IEEE) started working on an industry standard for wireless communication and in 1997 formed the 802.11 Working Group. The first standard, created in 1997, worked in the 2.4 GHz band and had speeds up to 2 Mbps but few real commercial products. In November 1999 IEEE ratified two new standards, 802.11a and 802.11b. 802.11b fueled the big Wi-Fi explosion and has speeds up to 11 Mbps operating in the 2.4 GHz frequency band. Since then, terms like Wireless LAN and Wi-Fi have become nearly commonplace.

802.11 is a member of the IEEE 802 family that specifies local area network (LAN) technologies. 802.11 focuses on the lower two layers of the OSI model, the data link and physical layer.

802.11 defines two media for communication – Infrared (IR) light and radio waves. Portable PCs have used IR communication between laptops and printers and other peripherals for years; however IR never really took off due to its many limitations, particularly its requirement of line-of-site. Nearly all 802.11 products on the market use the radio wave physical layer. The Federal Communications Commission (FCC) rigorously controls the radio spectrum and has designated three bands for *Industrial, Scientific, and Medical (ISM)* use. 802.11a operates in the C-Band ISM (5.725 - 5.875 GHz) and 802.11b operates in the S-Band ISM (2.4 – 2.5 GHz) [22]. 802.11 chose this range because the ISM bands are generally license-free

at low power so people who chose to setup a Wi-Fi network need not apply to the FCC.

802.11 is very similar to Ethernet which is a derivative of IEEE 802.3 – in fact the 48-bit physical Media Access Control (MAC) addresses for 802.11b NICs are assigned from the same pool.

2.3.1.1 802.11 Networks Configurations

The simplest 802.11 networks consist of a group of stations that communicate with each other by creating a *basic service set* (BSS). There are two possible configurations of the BSS, Independent (IBSS) and infrastructure (known simply as BSS). In an IBSS each station communicates directly with other Wi-Fi stations in the BSS. This of course means that each station must be within range of each other station. An IBBS network is sometimes referred to as Ad-Hoc and no additional equipment, other then the client NICs are needed. The smallest wireless network involves two Wi-Fi stations in IBBS mode. An IBBS network is typically created for temporary communication between a few stations. The more common and reliable configuration is called infrastructure. In this mode, each BSS has an Access Point (AP) and all communication is relayed through the AP. This can increase the coverage area since each station only needs to be within range of the AP. Access Points also provide the ability to buffer frames, which allows a mobile station that typically runs off battery power, to enter power save modes. In an infrastructure

network, client stations must first associate with an access point to communicate on the network.

BSS and IBBS are certainly useful networks, however these independent modes only facilitate communication between the Wi-Fi stations. The most useful type of 802.11 configurations is the *Extended Service Set (ESS)* that is created by linking BSSs together with a backbone network. The backbone network is referred to as the *distribution system medium* and is typically wired Ethernet. Most Wi-Fi Access Points operate as bridges to the wired Ethernet LAN. In this mode, each Wi-Fi station looks like a normal Ethernet station to the rest of the nodes on the wired Ethernet LAN. The AP is responsible for bridging traffic to and from the Wired LAN to the wireless BSS. It is important to note that 802.11 specifies that the *Distribution System* can be any network capable of 802.2 Logical Link Control encapsulation. Nearly all Wi-Fi networks use Ethernet as the Distribution system medium, however, an interesting fact is that 802.11 itself can be used as the Distribution System. In that configuration, the Access Points themselves form the backbone network. This configuration is uncommon and expensive because the APs require two Wi-Fi NICs and one of those typically has a higher power RF transceiver - but is becoming more common with dynamic mesh based networks, which have scalability and robustness advantages.

When several Access Points form an ESS, access points must inform other access points of associated stations. This is typically accomplished using *inter-access point protocol (IAPP)* over the backbone network. Currently, the IAPP protocol is

largely proprietary but the IEEE 802.11 working group has made IAPP standardization a priority and special Task Group F is responsible for delivering this IAPP standard.

2.3.1.2 802.11 Network services

802.11 defines nine services. Three of these services provide mechanisms used for moving data and the remaining six provide management operations.

Distribution

The distribution service is used in frame delivery to determine destination address in infrastructure networks. Any communication that is to or from a Wi-Fi station uses this service, even if it's from one Wi-Fi station to another Wi-Fi station in the same BSS.

Integration

The integration service provides a mechanism for frame delivery to an IEEE 802 LAN outside of the Wi-Fi network. 802.11 only specifies the services it must offer because its implementation is specific to the distribution service.

Association

Association is required before any Wi-Fi station can communicate on the network. The association service allows the distribution system to determine which Access Point is responsible for each Wi-Fi station

Reassociation

The reassociation service is charged with the responsibility of managing mobile Wi-Fi "handoffs" between Access Points. Reassociation is initiated by the Wi-Fi stations when signal levels indicate a switch to a different Access Point in the ESS would be beneficial. After reassociation, the distribution service updates its location records to reflect the availability of a Wi-Fi station.

Disassociation

The disassociation service is used to terminate an existing association. Disassociation is supposed to be initiated by the Wi-Fi station, but is not always done as it should. Therefore Access Point firmware normally accommodates stations that leave the network without disassociating.

Authentication

The authentication service establishes identity prior to association. In practice, however, most Wi-Fi implementation use "open-system" authentication. "Open-system" authentication essentially allows any station to authenticate. Although not specified in 802.11, many vendors have improved slightly over the open-system authentication by not broadcasting the SSID, or network name, needed to authenticate with a Wi-Fi network. This has been labeled "closed-system." Closed system, by its self, is not very affective and many attacks have been created to learn the SSID [32]. The 802.11 Working Group is currently working on standardization to incorporate IEEE's 802.1x authentication (see section 2.3.1.4).

Deauthentication

The deauthentication service terminates an authenticated relationship. Similarly to disassociation, deauthentication is a "polite" jester by Wi-Fi stations leaving a Wi-Fi network

Privacy

With a wired network such as Ethernet, an attacker generally has to have physical access to the computing resources. Unfortunately, Wi-Fi typically "leaks" into the surrounding areas, often outside the physical boundaries of a building, for example. 802.11 attempted to address this issue with an optional privacy service called *Wired Equivalent Privacy (WEP).* Many people confuse the acronym with "Wireless Encryption Protocol" because WEP uses a shared key encryption algorithm. (WEP is described further in the next section).

MSDU delivery

The *MAC Service Data Unit delivery* service is provided by the Wi-Fi stations and is responsible for delivering the data to the actual endpoint.

2.3.1.3 802.11 WEP

In any communications network, security is always a concern, but when the word "wireless" is added security is sure to be scrutinized. In order to access wired Ethernet networks you have to have physical access[1] to the medium (Ethernet wires),

[1] At least with today's known technology. The NSA recently declassified a project known as TEMPEST with the goal of circumventing this pesky problem [4 3].

which normally implies that you have to be in the building the network occupies, or even a specific room. Once you are "plugged in" you can do just about anything from sniffing traffic to hijacking the entire network. With wireless networks it is often impossible to have physical control of the medium (free-space). Short of encasing your building in metal mesh it cannot be guaranteed that "leakage" will not occur. Wired Equivalent Privacy (WEP) tries to address these issues.

WEP is a symmetric secret-key stream cipher that utilizes RC4, a Pseudo-Random Number Generator designed by RSA Data Security, Inc [23]. Wi-Fi networks utilizing WEP are far more secure then those that do not, however, it has been proven that breaking WEP is within the limits of any modern laptop.

In early 2001, the *Internet Security, Applications, Authentication and Cryptography (ISAAC)* group at the University of California, Berkeley published an initial report on their analysis of WEP that pointed out numerous implementation problems [24]. In August 2001 a paper entitled "Weaknesses in the Key Scheduling Algorithm of RC4" was published [25] proving a weakness that could be exploited in the Key Scheduling Algorithm for RC4 and a method for breaking WEP was outlined. Shortly after the paper was published several implementations were created that "crack" Wi-Fi WEP keys. With the best-known attacks currently, WEP can be broken after collecting an average of 5,000,000 encrypted packets on a Wi-Fi network[2]. RC4 is used in a variety of software applications, most notably secure Web

[2] Once a sufficient amount of network traffic is collected the actual time required for the algorithm to crack the key is on the order of seconds. This has caused some to erroneously cite that WEP can be broken in seconds.

pages and data via the SSL/TLS protocol. RSA security emphasizes that the WEP attack is a weakness in the implementation and not a weakness in RC4 itself and proposed a solution based on Fast Packet Keying that is under consideration by the 802.11 Working Group [26].

When debating whether or not WEP does its job, its important to know the correct acronym, because despite the ability to "crack" WEP, many argue that it does accomplish its goal of providing similar security inherent in wired Ethernet. Just as physical security keeps the general attacker off the network, WEP keeps the general attacker off of the Wi-Fi network.

The 802.11b specification allows for four WEP keys to be used at a time. Each packet that has been processed by WEP will have the WEP flag set in the 802.11 header and a modified data payload that contains a four-octet *IV* field, a variable length *Data* field, and a four-octet *ICV* field. The IV field contains a three-octet *Initialization Vector* that is concatenated with the WEP key during the RC4 cipher process. The idea behind the IV is to expand the useful lifetime of the WEP keys and is typically implemented as a counter in firmware that changes on each transmission[3]. The last octet of the IV field is used to select one of the four WEP keys to use. The use of four WEP keys was again intended to extend the useful lifetime of the WEP keys. The specification allows for each transition to rotate through the four WEP keys, but in practice only one WEP key is normally used. The

[3] Unfortunately a very small percentage of the 16,777,216 possible IVs have cryptographic properties that can be exploited as a result of WEP implementation in 802.11. These so-called weak IVs are what led to the initial cracking of WEP as mentioned above.

final field is an Integrity Check Value (ICV) derived by calculating the checksum before encryption.

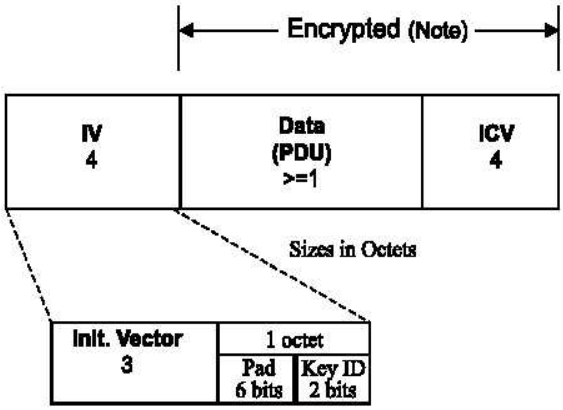


Figure 2.2 WEP data payload Source: IEEE 802.11 Section 8.2.5

Standard WEP key lengths are 40 bits (originally for export outside the United States) and 104 bits. Since the WEP key is concatenated with 24 bits from the Initialization Vector many vendors refer to 40 and 104 bit WEP keys as 64 and 128 bit keys. Some vendors allow the WEP keys to be derived from an ASCII password, which typically leads to less interoperability because some vendors take the hex value of the ASCII and some use a hash function to produce the hex key. If the ASCII values are used, the string lengths are five and thirteen for 40/64 bit and 104/128 bit encryption, respectfully. The 104/128 bit key was assumed to be secure but the specification allows for key lengths to be increased by the vendors. Unfortunately its been shown that due to the weakness in the IVs, increasing the key length only linearly increases the time required to recover the key [25].

2.3.1.4 802.11i and 802.1x

On May 30, 2001, the IEEE 802.11 Working Group created the special Task Group I formed to address the growing concerns of 802.11 security [28]. 802.11i is currently a work in progress and the various proposals are still being debated. It is clear that 802.11i will make use of Fast Packet Keying [29] and will use 802.1x to authenticate the stations and distribute the keys. 802.1x, " Port-Based Network Access Control," [27] was ratified by IEEE in October of 2001 and is an authentication framework based on the Internet Engineering Task Force's RFC2284 [30], "Extensible Authentication Protocol" (EAP). EAP is essentially an encapsulation that was designed to run over any link layer and use any number of authentication methods.

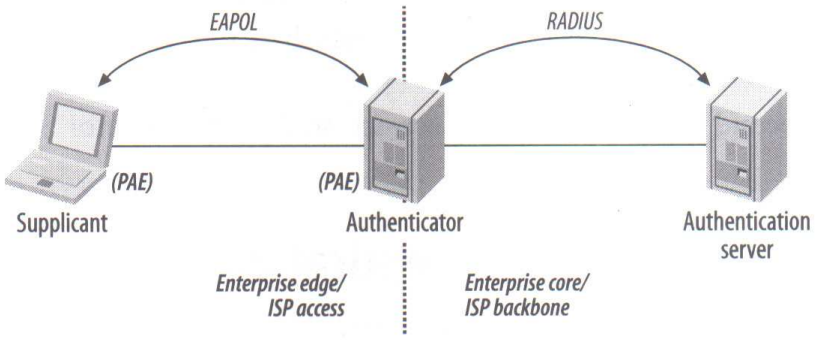


Figure 2.3 802.1x architecture Source: Gast, 802.11 Wireless Networks [3]

The 802.1x architecture has three components. The *supplicant* is the end user machine, the *authenticator* controls network access, and the *authentication server* processes the authentication request. The *authentication server* is typically a RADIUS server.

802.1x will bring many enhancements to the security of 802.11. Using 802.1x to distribute WEP keys regularly to implement the Fast Packet Keying solution will eliminate the known weak IV exploit. Moreover 802.1x allows for *mutual* authentication. The current 802.11 standards do not have any methods for a client Wi-Fi station to authenticate the Access Point, creating a rogue AP threat.

2.3.2 802.2 Link Layer Control

The IEEE 802 Project divided the OSI data layer into two sublayers. The lower layer, Media Access Control (MAC), deals with the problem of coordinating the access to the physical medium for the various 802 network technologies. The upper layer is called the Link Layer Control (LLC) layer. The LLC abstracts the details of the underlying MAC protocols and is common to all LAN protocols. The LLC also provides a means for exchanging frames between LANs that use different MAC protocols [31].

2.3.3 802.3 CSMA/CD

In 1972 researchers at Xerox Palo Alto Research Center (PARC) developed the first working example of Ethernet – which came to be easily the most successful Local Area Networking technology. In 1985, the IEEE 802.3 Working Group released the first version of 802.3 “Carrier Sense, Multiple Access with Collision Detect

(CSMA/CD)" standard. Modern day Ethernet is based on the 802.3 standard and is often erroneously called 802.3.

Ethernet is used in nearly all modern LANs and is typically the backbone network and a part of the *distribution* system for a Wi-Fi bridged network.

2.3.4 802.11 Details

2.3.4.1 Access Point Operation in Infrastructure mode

Wi-Fi network Access Points perform a variety of functions that facilitate communications on a Wi-Fi network. The most important of these functions is the bridging of the network traffic. This wireless-to-wired bridging function involves taking frames from one physical layer to another – typically Ethernet to 802.11.

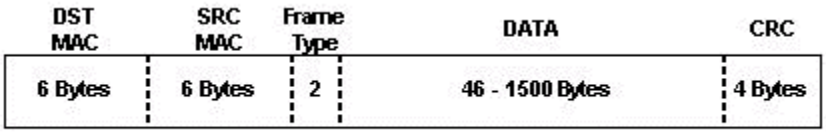


Figure 2.3 Ethernet II Encapsulation Source: Cisco [33]

Ethernet II is the most common of the four types of Ethernet encapsulations [33]. Figure 2.3 shows the frame format of an Ethernet II frame. The source and destination MAC addresses are the unique 48-bit machine addresses of the source and destination NICs. The Frame Type is a two-byte field that can be any value greater

then 0x05FF but is typically 0x0800 or 0x0806 denoting Internet Protocol (IP) or Address Resolution Protocol (ARP), respectfully [44]. The final field is a CRC integrity check.

802.11 utilizes 802.2 logical-link control (LLC) encapsulation to carry higher-level protocols. Two different methods can be used to encapsulate LLC data for transmission, 802.1h [34] and the more common RFC 1042 [35].

Figure 2.5 802.11 Frame Format Source: 802.11 Specification.

The 802.11 Data Frame format is shown in figure 2.5. The 30-octet MAC header includes four address fields and three control fields. The Frame Control field is a bitmapped field that conveys control information such as protocol version, type (Management, Control, or Data), power management, fragmentation, and WEP flags. The duration field has several uses and is primarily related to the physical time it takes to transmit the frame. The Sequence Control frame is used for both defragmentation and discarding duplicate frames.

This document is primarily concerned with the frames that are of type DATA and particularly interested in the four address fields in the 802.11 MAC header. Each of these address fields carry a 48-bit MAC addresses and have various uses

depending on the source/destination of the frame and the mode in which the Access Point is operating. Address 1 typically indicates the 48-bit MAC address for the next hop receiver of the frame. Address 2 typically indicates the transmitter address. Address 3 typically holds the original source address or the final destination address. Address 4 is only used in Wireless Distribution System bridged mode.

Table 2.1 Use of the address fields in the 802.11 MAC header.

Function	Address 1	Address 2	Address 3	Address 4
WDS	RA	TA	DA	SA
Independent BSS	DA	SA	BSSID	not used
To AP (infrastructure)	BSSID	SA	DA	not used
From AP (infrastructure)	DA	BSSID	SA	not used

Table 3.1 summarizes the use of the four address fields in the MAC header of 802.11 data frames. In infrastructure mode, when a Wi-Fi station sends a frame, Address 1 will contain the BSSID of the AP in which the Wi-Fi station is associated with. Address 2 and address 3 will contain the source and destination MAC addresses, respectfully. When a Wi-Fi station receives a frame from the AP, address 1 will contain the destination MAC address for the station and address 3 will contain the original source MAC address. Address 2 contains the BSSID. The Access Point firmware is responsible for repackaging frames from the wired Ethernet interface to the Wi-Fi interface and from the Wi-Fi interface to the wired Ethernet interface.

When WEP is enabled, the WEP flag in the 802.11 MAC is set and the data portion of the frame is encrypted, however, the 802.11 MAC headers are transmitted unencrypted.

2.3.4.2 Wi-Fi Broadcast Traffic

Wi-Fi broadcast traffic is similar to broadcast traffic on any shared network. However, there are some differences between normal Wi-Fi traffic and broadcast traffic. One major difference is that broadcast traffic is not possitivialy acknowledged like all other non-broadcast Wi-Fi frames.

When an Access Point receives a broadcast packet from the wired Ethernet distribution network, the Access Point will copy the source and destination MAC address from the Ethernet headers. The 6-octet broadcast destination address will be copied into the Address 1 field of the 802.11 MAC header. The address 2 field will hold the MAC address of the Access Points wireless NIC. Address 3 will be filed in from the source address of the Ethernet header. If WEP is enabled, the WEP algorithm processes the data portion, the WEP flag in the MAC is set, and the encrypted data is copied to the data field in the Wi-Fi frame. Unlike many wired Ethernet networks that are configured with switches and hubs that relay the broadcast packet, the Wi-Fi broadcast frame is literally broadcast and received by all the Wi-Fi stations.

Chapter 3

Architecture

3.1 Overview

In order for a Wi-Fi station to start normal communication the station must join the network. This typically involves three steps in the infrastructure (BSS or ESS) configurations. First the Wi-Fi station must probe for existing networks. Second the station must authenticate with a specific Access Point in the ESS. Finally, the station must associate with that Access Point. In the least secure modes defined by the Wi-Fi specification, it's possible for the Wi-Fi station to do all three steps automatically. This would imply an "Open Wi-Fi network with no WEP enabled." In this network configuration, the Access Points will reply to all probe packets as well as broadcast its SSID (network name) in the beacon packets that are constantly sent. This wireless network configuration is very insecure and is sure to attract unwanted hackers from possibly many blocks away. For that reason, vendors have created some helpful firmware options in the APs they manufacture.

One of the most commonly implemented security features is not to broadcast the SSID. This is an added vendor security feature and involves removing the SSID

from beacon packets as well as not replying to probe request that do not specify the AP's SSID. This method is sometimes referred to as "closed-system" or "non-broadcast SSID." Since this option is very easy to enable and generally does not affect the system, most vendors are beginning to enable this option by default. This implies that a Wi-Fi station now must know, in advance, the SSID[4].

The other common way to boost up security on the Wi-Fi network is to enable WEP (see section 2.3.1.3). Again, the Wi-Fi station must also know either the WEP keys, or in the case of 802.1x, the certificate or other authentication credentials, in advance. While this is only mildly annoying for the user to enter on a laptop, it can be impossible on an embedded system with none or very limited input capabilities. In addition, since its been shows that WEP is breakable, it is a good idea to change the WEP keys on a regular basis, making it even more of a hassle for an embedded device.

The following architecture defines a protocol that enables a configuring client to program a Wi-Fi embedded station with the necessary information to join a Wi-Fi network. The configuring host can be on the Wi-Fi portion of the network or the backbone Ethernet network.

[4] There are known attacks that can "jam" Wi-Fi stations, forcing them to disassociate and reassociate, thus exposing the SSID [32]

3.2 Requirements

3.2.1 Configuring Host Requirements

The configuring host is the station responsible for sending out the configuration information via broadcast packets. The configuring host can be a computer in the Basic Service Set portion of the Wi-Fi network or a computer on the wired portion of the Ethernet that forms the backbone network for the ESS. The configuring host must be capable of sending broadcast packets that would reach the target Wi-Fi station if it were associated with the network.

3.2.2 Target Host Requirements

The target host is the Wi-Fi station that is being programmed with the proper Wi-Fi parameters to join the Wi-Fi network. The target host must be able to monitor the Wi-Fi channels and listen for broadcast packets. The target host needs to be able to sniff network traffic in promiscuous mode. The configuring code can execute within the Wi-Fi device driver or in a user-space program.

The target must be able to reassemble the fragmented packets, verify the integrity check, and set the Wi-Fi parameters of the interface card.

3.3 Architecture

3.3.1 General Overview

The following figure depicts the typical network topology for a Wi-Fi infrastructure network defining an Extended Service Set (ESS). There is a backbone Ethernet

network, a gateway to the Internet, and three Access Points that create the Extended Service Set.

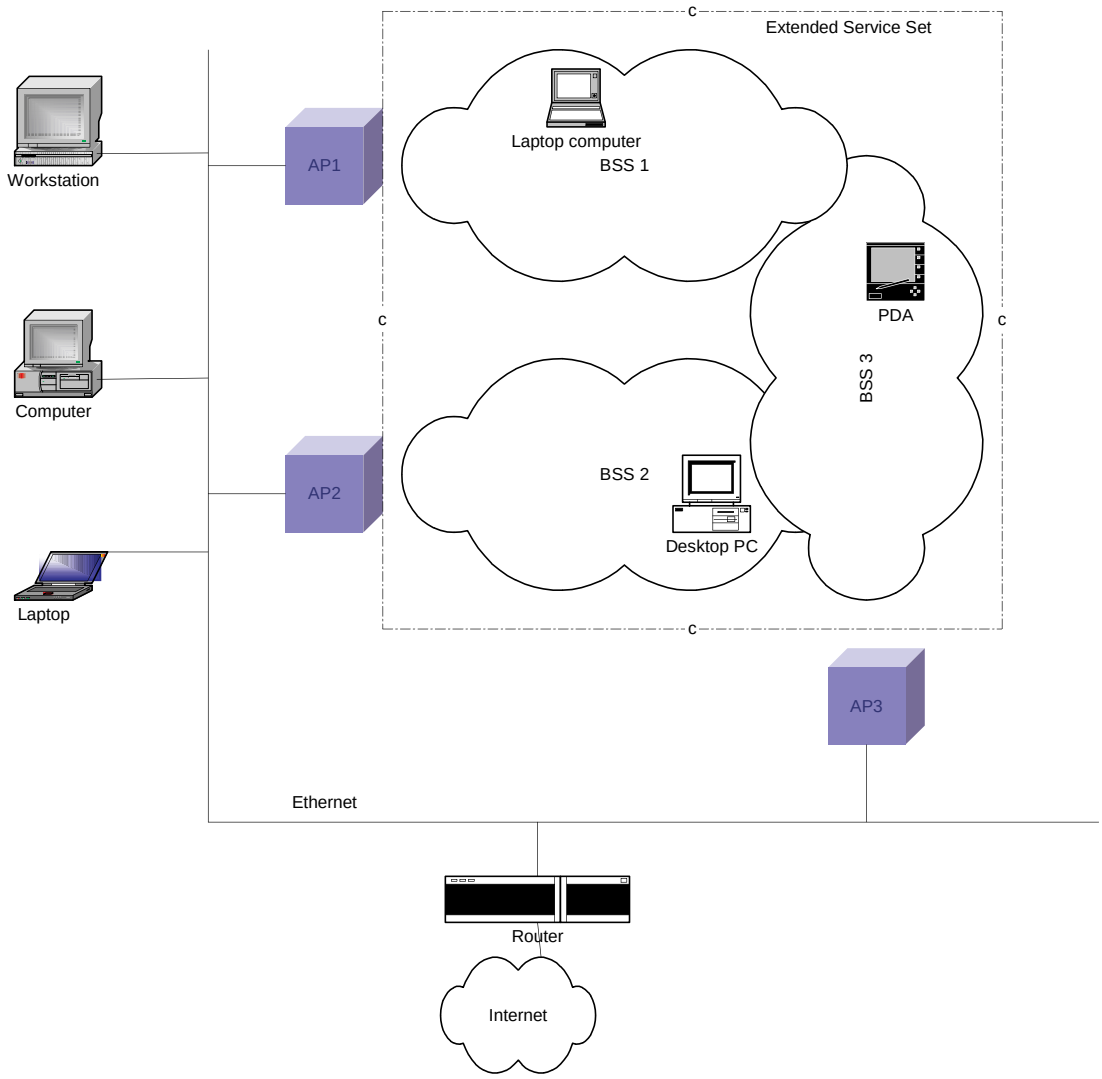


Figure 3.1 Typical Wi-Fi Extended Service Set network topology.

3.3.2 Wi-Fi-Co Configuration

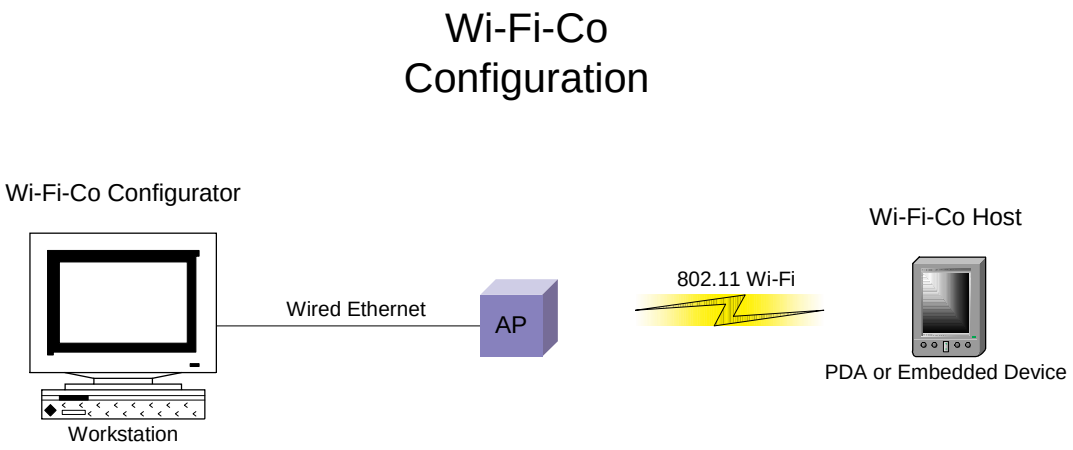


Figure 3.2 Wi-Fi-Co configuration system view

Wi-Fi-Co link level network configuration consists of two main systems – the configurator and the target. The configurator is the host that is responsible for creating and sending the broadcast packets that will configure the target. The target is the Wi-Fi host that is being sent the various network parameters. The configurator must be a station within the Extended Service Set (ESS) of the Wi-Fi network. Therefore, the configurator could be a computer on the wired portion of the bridged Ethernet network or a Wi-Fi station that is associated with an Access Point in the ESS. The target must be capable of capturing 802.11 frames in promiscuous mode and further must be within range of at least one AP with in the ESS of the configurator. The configurator sends the target host configurator information via broadcast packets.

3.3.4 Wi-Fi-Co Packet Format

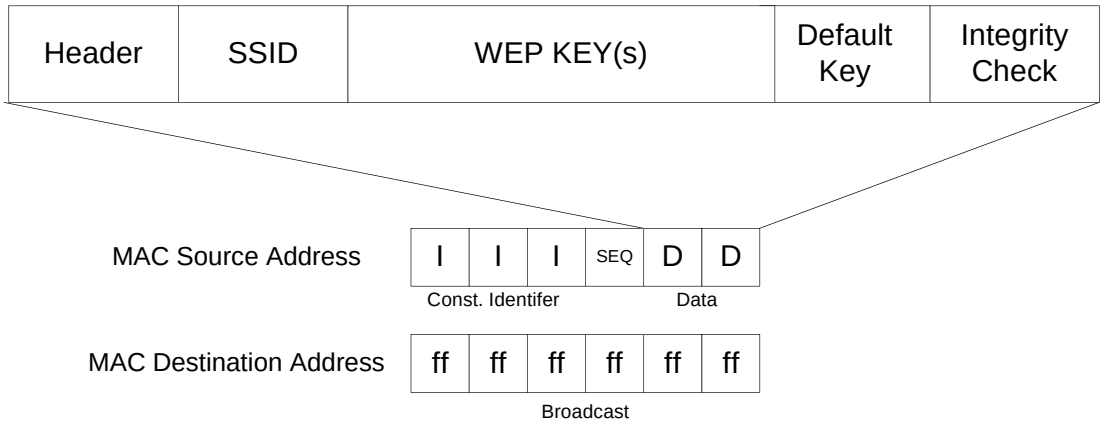


Figure 3.3 General Wi-Fi-Co Packet Format

The 6-octet source address field will contain fragmented packets that will contain the following fields:

- SSID
- WEP key[s]
- Default key
- Integrity check
- Feedback information (optional)

Each fragmented packet will also have the following fields:

- Identifier
- Sequence Number

Description of fields:

SSID

The Service Set Identifier is the network name of the network the Wi-Fi station will be joining. It is typically an ASCII word and has a length between 0 to 32 bytes.

WEP Key[s]

The WEP keys are the shared secret keys used in the encryption process of the Wired Equivalent Privacy protocol. The WEP keys can be any length but are typically 40 or 104 bits. The WEP specification allows up to four keys to be used.

Default WEP key identifier

The default WEP key identifier is a number between one and four that specifies which key the Wi-Fi station should use when sending out WEP data.

Integrity check

The integrity check is necessary to identify errors in the transmission of the frames.

Identifier

The identifier is a bit sequence that the target Wi-Fi station can use to identify Wi-Fi-Co configuration frames from normal broadcast traffic. The first three bytes of the MAC address typically identify the manufacture of the NIC. The identifier could be a block that has be designated PRIVATE (reserved) [36].

Sequence number

The sequence number is used to identify the particular fragment of the configuration packet.

Feedback field

The optional feedback field is used to notify the configurator that the target has been successfully configured. The field may contain an IP and port to which the target could connect to notify the configurator. Since the configurator is sending data in the source MAC address field, the target has no knowledge of the address (physical or IP level) of the configurator. The configurator can then stop sending configuration packets and notify the user that the operation was successful.

3.3.5 Wi-Fi-Co Privacy Protection

Since Wi-Fi-Co sends network parameters related to the security of the network, specifically the WEP keys, any implementation should address the threat of exposing the WEP keys. Since the configurator is broadcasting configuration information in link layer headers (which are plaintext) there is a reasonably high threat that an attacker within range of the ESS could be capturing the configuration packets. It would then be trivial to extract the WEP keys. One possible solution is a shared secret between the two – possibly a serial number from the embedded device. It is important to realize that input to the configurator is many times easier then input to the embedded target device.

Chapter 4

Implementation

4.1 Overview

A Wi-Fi-Co implementation consists of two main parts: the *Configurator* software and the *Target* software. The configurator software is responsible for sending the target the necessary parameters that allow the target to gain link level connectivity to the Wi-Fi network. The target is a wireless host on the Extended BSS network and is typically an embedded device. The configurator can be any host that is connected to the Extended BSS network.

The configurator determines the Wi-Fi network parameters, creates the Wi-Fi-Co configuration packet, and continuously broadcasts that packet. The configurator can determine the Wi-Fi parameters via user input or configuration files. If the configurator is located on the wireless portion of the network the software can discover the parameters by probing its own configuration. The Configurator can optionally wait for feedback from the Target, once the Target gains IP level network access.

4.2 Wi-Fi-Co Packet format

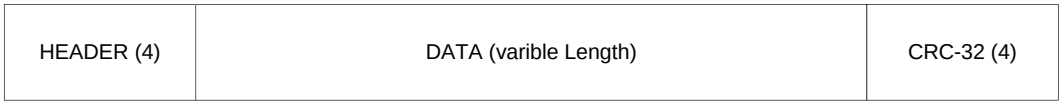


Figure 4.1 Wi-Fi-Co Packet Format

The Wi-Fi-Co packet consists of a four octet header, a variable length data portion, and a four octet CRC-32. The Wi-Fi-Co packet is converted to Network Byte Order before transmission. The sections are described in detail below.

4.2.1 Wi-Fi-Co Header

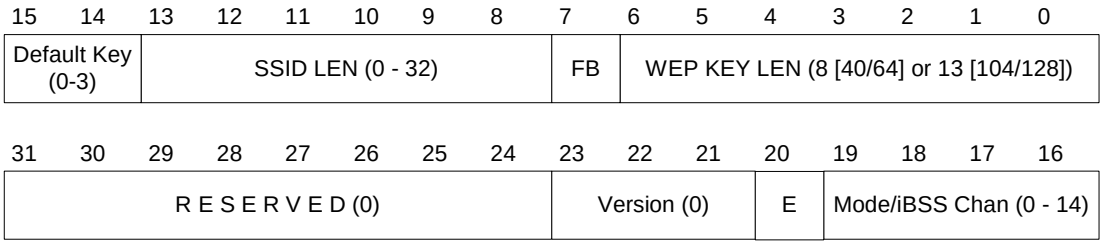


Figure 4.2 Wi-Fi-Co Header Format

The Wi-Fi-Co header consists of several fields:

WEP KEY LEN

This field contains the number of characters (ASCII or hex) in each of the WEP keys. Currently the standard WEP key lengths are 40/64 bit and 104/128 bit keys which correspond to eight and thirteen character lengths, respectfully.

FB

The Feedback Field is a 1 bit flag that specifies whether or not the target should attempt to contact the configurator after joining the network. In this implementation, if this flag is set, the length of the DATA field grows by 6 octets.

SSID Length

This field can have a value from 0 (specifies the special “any” ssid) to 32 and specifies the length of the SSID in octets.

Default key

This 2 bit field specifies which of the 4 WEP keys is the 802.11 default WEP key.

Mode/iBSS channel

This field contains a 0 if the target BSS is infrastructure or a value from 1 to 14 that specifies the channel of the Independent BSS (Ad-Hoc mode).

Encryption

This flags specifies if the DATA portion of the packet is privacy protected (see section 2.4.5).

Version

This field specifies the Wi-Fi-Co version the configurator is using and is 0 in this initial implementation.

Reserved

This field is reserved for future use and should be set to zero.

4.2.2 Wi-Fi-Co Data Format

Figure 4.3 Wi-Fi-Co Data Format

The DATA field consists of the SSID, the four WEP keys and optional feedback address.

SSID

The SSID is the name of Wi-Fi BSS that the target should join. The length of the field is defined in the header and the value is not null terminated. The range is 0 (indicating the special case “any”) to 32.

KEY0 – KEY3

This field contains the Wi-Fi-Co WEP keys. The length of each field is defined in the header and the values are in hex.

Feedback Addr

This optional field contains the Ipv4 address and port that the target should attempt to connect to after gaining network access. The length of this field is six octets.

4.2.3 Wi-Fi-Co Integrity check

Wi-Fi-Co uses broadcast traffic to configure network parameters. Broadcast is an unreliable service and the failure rate is particularly high on a wireless medium. A

CRC-32 protects against corrupt data. CRC-32 will detect all single and double bit errors as well as burst errors up to 32 bits long.

4.2.4 Calculating the Wi-Fi-Co configuration packet length

The total packet length can be calculated from the four octet header.

*length = HEADER_SIZE + KEY_LEN * NUM_KEYS + SSID_LEN + FB * FB_LEN + CRC_SIZE*

In this implementation:

*length = 4 + KEY_LEN * 4 + SSID_LEN + (FB * 6) + 4,*

Where KEY_LEN, SSID_LEN, and FB are defined in the header.

4.2.5 Protecting the WEP key

A method for protecting the data in this protocol is important because of the broadcast nature of Wi-Fi-Co. Although Wi-Fi-Co is normally ran as a one time setup of an Embedded device, it would be trivial to implement a program that listened for Wi-Fi-Co configuration transmissions and recorded the sensitive parameters to a database.

This implementation utilizes a shared key crypto algorithm: RC4. Typically, the embedded device would ship with a certificate that would contain a serial number. This serial number could become the shared key, requiring the user to input the number when running the Configurator software. This is acceptable because user

input to the host the Configurator software is running on is much easier then input to the embedded device.

4.3 Wi-Fi-Co Configurator Software

The implementation of the configurator software has three main parts. First, the software must determine the network parameters to send. Second the configuration packet must be created. Finally, the configuration packet must be fragmented and transmitted (see figure 4.4). Optionally, the configurator may wait for confirmation of successful target configuration. A brief summary of the code follows.

In the current implementation, the network parameters can be specified as command line arguments or by prompting the user for input. The configuration software must fill out the following structure:

```
struct _wifico_pkt {

  struct wifico_header header;              /* Wi-Fi-Co header */
  __u16 ssid[MAX_SSID_SIZE];
  __u16 key[NUM_KEYS][MAX_WEP_KEY_LEN];  /* the wep keys */
  __u32 fb_ip;            /* ip to report back on if feedback = 1 */
  __u16 fb_port;          /* port for feedback */
  __u32 long crc;         /* populated by software */
} wifico_pkt;
```

Where wifico_header is defined as:

```
/* for little endian systems */
stuct _wifico_header {
      unsigned short key_len : 7 __attribute__ ((packed));       /* 8|13*/
      unsigned short feedback : 1 __attribute__ ((packed));      /* 0|1 */
      unsigned short ssid_len : 6 __attribute__ ((packed));       /* 0 – 32 */
      unsigned short default_key : 2 __attribute__ ((packed));  /* 0 – 3 */
      unsigned short mode : 4 __attribute__ ((packed));           /* 0 – 14 */
      unsigned short version : 3 __attribute__ ((packed));        /* 0 */
      unsigned short crypt : 1 __attribute__ ((packed));          /* 0|1 */
      unsigned short resrv : 8 __attribute__ ((packed));          /* should be 0 */
  } wifico_header;
```

Once this structure is populated a call to *fill_wifico_buff()* will create the Wi-Fi-Co configuration frame that can then be passed to *wifico_send_config()*.

The function *wifico_send_config()* is responsible for fragmenting and broadcasting the configuration packets. The packet is fragmented into two octet sections that are copied into the fifth and sixth octet of the Ethernet II source MAC address. The first three octets of the source address contain a delimiter that allows the target to distinguish Wi-Fi-Co packets from normal broadcast traffic; in the implementation, the reserved vendor address 10:00:00 is used. The fourth octet contains the sequence number. The open-source libnet project provides a portable and simplified interface for low-level network packet shaping, handling, and injection [38] and is used by this implementation to create and send the raw Ethernet II frames. *wifico_send_config()* fragments the wifico buffer and sends out the configuration data. Each of the Wi-Fi-Co configuration packets are simply Ethernet II frames with a type of 0x5555, destination MAC of broadcast, and source address as described above. The length of the Ethernet II data is zero and the payload of the frame is null The Wi-Fi-Co buffer is transmitted in this matter continuously for a specified number of times or until the target positively acknowledges successful configuration, if feedback is enabled.

4.4 Wi-Fi-Co Target Software

The implementation of the target has the following main steps: configure the Wi-Fi card to be able to receive the configuration packets, listen for Wi-Fi-Co configuration packets, assemble and verify received configuration, configure system network software, and if requested, notify the configurator of successful configuration. A brief summary of the code follows.

Before the Wi-Fi-Co target software can receive the configuration, the Wi-Fi card and system network must be configured to allow the application to capture Wi-Fi traffic. Initially this implementation created a simple application that placed the Wi-Fi card in the proper mode to allow the Wi-Fi card to capture packets in promiscuous mode. However, this early implementation was chipset-specific and not very efficient. The current implementation utilizes the kismit_hopper from the Wi-Fi sniffing and discovery package, kismet [40], released under the GNU General Public License. Kismit's implementation is efficient and supports nearly all Wi-Fi chipsets and firmware. The implementation puts the Wi-Fi card in promiscuous mode and calls kismet_hopper, which runs in the background and commands the Wi-Fi firmware to switch a specified number of channels every second. The hopper is needed because a Wi-Fi BSS could be on any one of fourteen channels.

The target software then calls *wifico_get_config()*. This function waits for Wi-Fi-Co configuration packets, identified as broadcast packets with a source MAC address beginning with *10:00:00* – which is a reserved MAC address area classified as experimental. The function assembles the fragmented buffer and verifies the

checksum once the complete buffer is received. The target software has to deal with corrupt packets for many reasons: the hopper moving between channels, normal Wi-Fi interference, and the unreliable broadcast delivery. If the *crc32*() function call returns a failed checksum, the Wi-Fi-Co buffer is discarded and the process is repeated.

After the target software has the verified Wi-Fi-Co data, the implementation utilizes a shell script called *wifico_set_config.sh,* passing the configuration data on the command line. This method allows for easy porting to various other Operating Systems, distributions, and network setup schemes. The user need only modify the script to configure their system. Scripts for various popular systems such as Debian, RedHat, Gentoo, and sharp zaurus have been included.

If the feedback flag in the Wi-Fi-Co header is set, the target software will also attempt to open an IP/TCP connection to the specified IP and port. The Target will relay to the Configurator the IP address assigned via DHCP as well as some statistical information about the configuration process.

4.5 Specific implementations and details on system ports

Configurator

The configurator software was written in C and designed and tested on a Linux Operating System. The configurator statically links to libnet functions and thus requires the libnet library for compilation. Currently there is only once version of the

configurator, but it could easily be ported to any operating system/architecture that libnet has been ported to.

Target

The target software was initially written in C and designed on a Linux Operating System. The Linux version uses the dynamic shared library, libpcap. Libpcap is a very common and popular packet capture library written in C. Libpcap abstracts the details of packet capturing and has been ported to a plethora of operating systems and architectures [39].

The target has also been ported to run on the Sharp Zaurus PDA. The PDA is an arm architecture and runs QT/Embedded Linux [42]. For the Zaurus version, libpcap did not handle the 802.11 headers properly and was abandoned and replaced with raw socket operations.

Wi-Fi-Co has also been implemented in the Smart Wireless Thermostat device driver designed for the rabbit microcontroller by Ambient Computing, Inc [41]. The implementation is written in embedded Dynamic C and handles the channel hopping, packet capturing, and network configuration directly within the custom device driver controlling the Wi-Fi card.

Figure 4.4 shows the timing diagram for the Wi-Fi-Co Protocol.

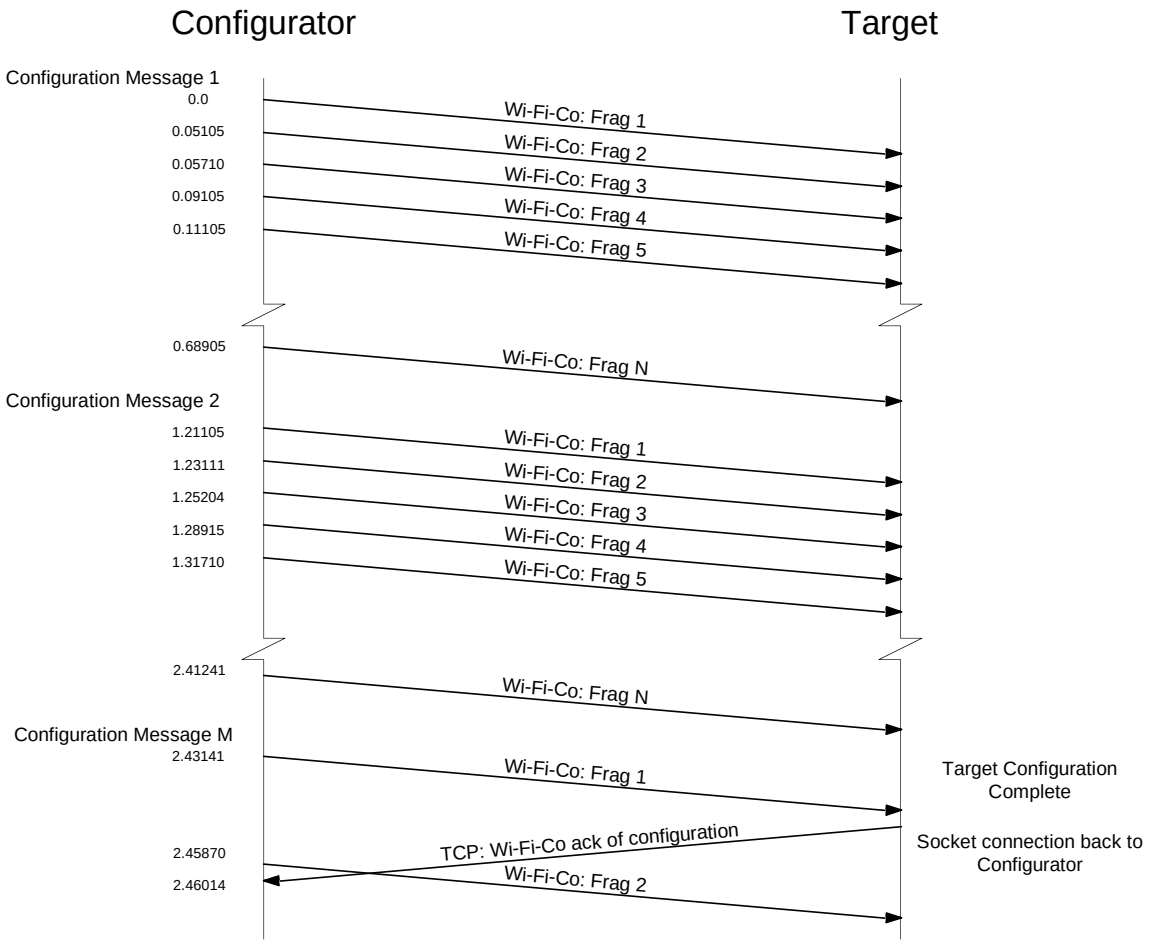


Figure 4.4 Timing Diagram for Wi-Fi-Co Protocol

Chapter 5

Results

5.1 Hardware and Software

This implementation was tested on several different and separated networks. This implementation was successful on all networks tested. The first set of results was performed on a small home network consisting of five computers and a Linksys Access Point on a 10/100 Mbit Ethernet switched network. The middle set of results presented were obtained from the largest, most active network tested – the Information Telecommunication and Technology Center's network located at the University of Kansas. The final section is devoted to the results obtained with experiments utilizing a Wi-Fi enabled embedded device as the *Target*.

As described in section 4.4, 802.11b Wi-Fi Access Points transmit on a specific channel and several APs on the same backbone network can be configured to increase the range of the Wi-Fi network by creating an Extended Service Set.

802.11b utilizes Direct Sequence Spread Spectrum to spread the power across a wider frequency band to minimize narrowband interference. This means that the 14 allocated 5MHz wide channels (11 in the U.S.) leak energy into adjacent channels. This requires APs within range of each other to be 5 channels apart to reduce interference. In the following test setups the term hopping appears. Hopping channels refers to requesting that the Wi-Fi firmware change the channel it's operating on. In promiscuous mode hopping channels will give traffic from any channel a chance to be received. Since the channels overlap, the hop sequence generally jumps five channels at a time (1,6,11,2,7…).

5.2 Test Results

All tests consist of a *Configurator Host* that runs the configuration software and transmits configuration data to a *Target Host* that runs the *Target* software that receives configuration information and allows the *Target* Host to configure its Layer two network parameters. Built into this implementation of the protocol are simple statistics that are presented in the following sections. The *Target* software, once successfully associated with the network, will send the *Configurator*:

- IPv4 Address assigned via traditional DHCP once associated with the Wi-Fi network
- Number of Wi-Fi-Co fragmented packets the *Target* Host processed
- Number of CRC failures after assembly of fragmented Wi-Fi-Co buffer

- The time it took to receive the Wi-Fi-Co configuration

See ***Appendix B*** for example output.

There are a few subtleties that should be noted. The *Target* host can only receive packets – the Access Point will in no way will forward any type of frames from the *Target* until successful association with the Wi-Fi network. This implies that only a positive acknowledgment can be implemented in this protocol. The ack is accomplished via traditional IPv4 TCP communication back to the Configuration Host.

This Wi-Fi-Co implementation consists of additional helper applications implemented in shell scripts, in order to provide cross platform portability. The time statistic returned by Wi-Fi-Co is *only* the time it took to receive and verify the Wi-Fi-Co configuration parameters. Additional time is required to perform the Operating System specific network configuration.

In each of the test setups below, the Wi-Fi-Co configuration parameters sent were constant. The number of bytes needed in the Wi-Fi-Co configuration buffer to hold this static configuration was 71, meaning that the minimum number of Wi-Fi-Co fragmented packets that the *Target* must process was 36 fragments (see section 4.3 for details).

5.2.1 Small Home Network Test Results

The results in section 5.3 were all performed on a small, low traffic home network. The network consisted of 5 computers connected via a 10/100 Mbit Ethernet switch. A Linksys WAP11 Access Point provided Wi-Fi Access.

5.2.1.1 Test Setup 1

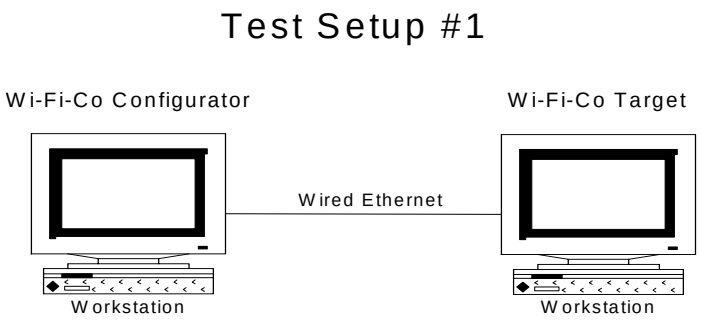


Figure 5.1 Test Setup 1

Test Setup 1 consisted of two workstations networked via switched 100 Mbps Wired Ethernet. While this test includes no wireless devices or wireless communications whatsoever, all functions of the Wi-Fi-Co protocol are exercised. This setup was included to contrast the retransmits that are a result of a less reliable wireless medium.

The *Configurator* was a RedHat 7.1 Linux workstation and the *Target* was a Debian Linux workstation.

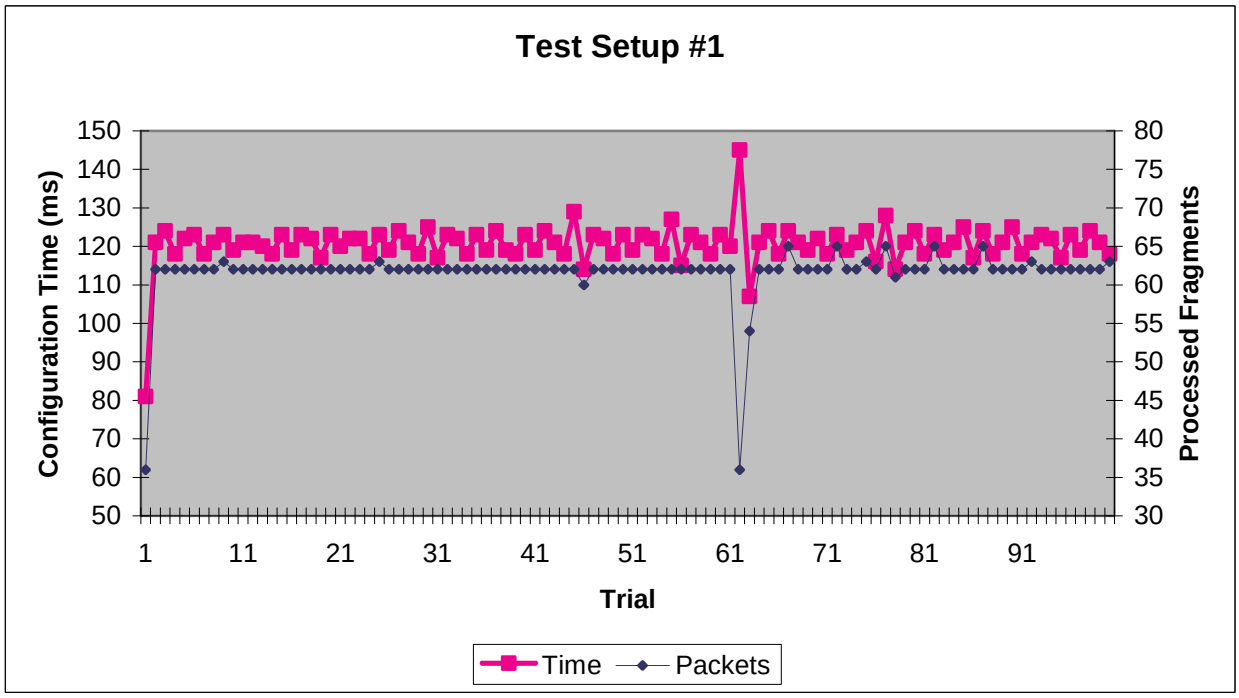


Figure 5.3 Results – Test Setup 1

Test Setup 1 produced expected results. The average time to configure was 120ms. The total number of fragmented Wi-Fi-Co packets for this test setup was 36 and the average number processed before successful configuration was 61. The minimum and maximum configuration times were 55ms and 142ms, respectfully. The minimum and maximum number of fragmented packets processed was 36 and 65, respectfully. There were 0 CRC failures.

5.2.1.2 Test Setup 2

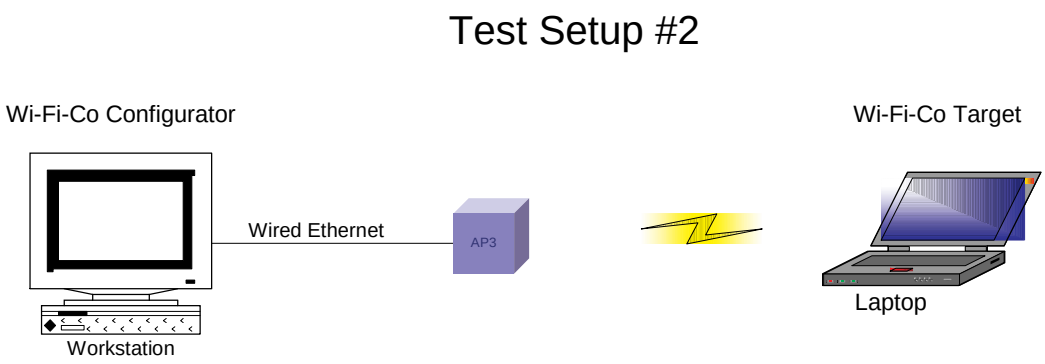


Figure 5.3 Test Setup 2

Test Setup 2 consisted of a Linux Configuration workstation that was on the Ethernet backbone portion of a Wi-Fi Extended Service Set. The *Target* host is a Wi-Fi Linux laptop. The test was performed with the *Target* host locked on a constant Wi-Fi channel. The channel selected was known to be within range of the *Target*. In the typical Wi-Fi-Co setup, the channel that each Basic Service Set Access Point was set on is unknown to the *Target*, forcing the *Target* Wi-Fi firmware to hop channels in order to guarantee it would always receive Wi-Fi-Co packets from an ESS within range. This test was intended to produce results that demonstrate functionality without the overhead of hoping Wi-Fi channels.

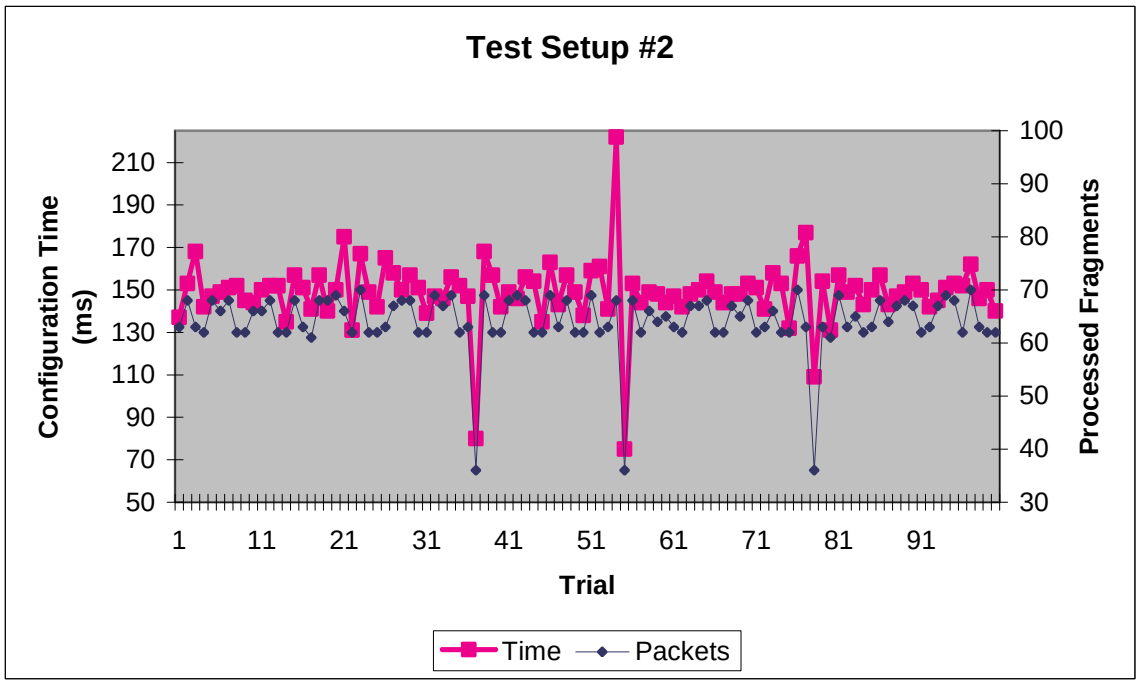


Figure 5.4 Results - Test Setup 2

The variation on configuration times and processed fragments is slightly greater then the Ethernet to Ethernet experiment in Test Setup 1. The average time to configure was 148ms with a minimum and maximum of 75ms and 222ms, respectfully. The average number of Wi-Fi-Co fragmented packets processed by the *Target* was 64 with a minimum and maximum of 36 and 70, respectfully. There were zero CRC failures.

5.2.1.3 Test Setup 3

Test Setup 3 was identical to Test Setup 2 with the exception of the amount of traffic on the Wi-Fi portion of the ESS. A large background ftp transfer was saturating the Wi-Fi link during the testing process.

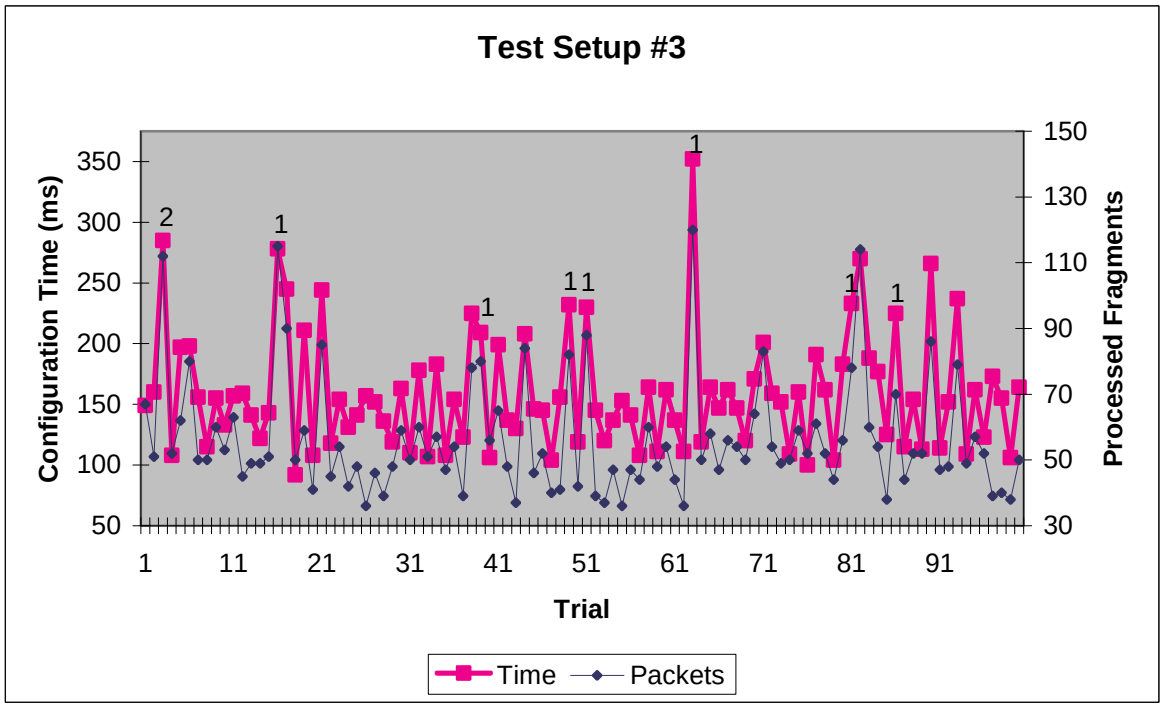


Figure 5.5 Results - Test Setup 3

As expected, there were an increased number of CRC failures: 9 compared to 0 in Test Setup 2. In figure 5.5, the number of CRC integrity failures is denoted above the trial number that the failures occurred on. The average configuration time was also slightly higher at 159ms. The minimum and maximum configuration times were 92ms and 352ms, respectfully. The average number of processed Wi-Fi fragmented packets was 56 with a minimum and maximum of 36 and 120, respectfully.

5.2.1.4 Test Setup 4

Test setup 4 is likely the most typical setup. It's identical to Test Setup #2 except that the Wi-Fi *Target* is not set on a specific Wi-Fi channel. Instead it is hoping three channels a second.

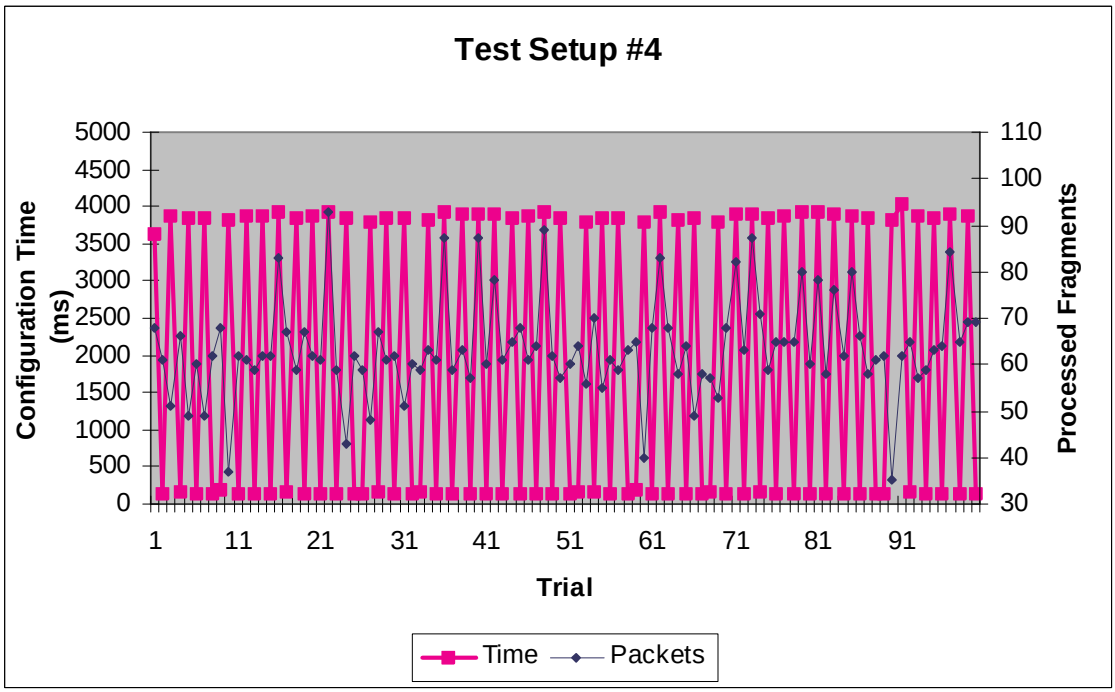


Figure 5.6 Results - Test Setup 4

As expected, the variation on configuration times is much greater then Test Setup 2. The Wi-Fi *Target* must hop across all eleven[5] of the Wi-Fi channels to guarantee that all portions of the Wi-Fi traffic within range of the *Target* have the opportunity to be

[5] For the United States. Fourteen in other portions of the world.

captured. On each hop, the Wi-Fi *Target* was able to capture traffic for about 333ms. Figure 5.6 clearly shows a consistent cycle – if the *target* was on the hop that the Wi-Fi network existed on, the configuration time was fast. Where the configuration times are higher, the *Target* was hopping on channels that didn't contain the Wi-Fi network. The average configuration time was 1,891ms with a minimum and maximum of 128ms and 4,031ms, respectfully. This result is very intuitive. At a hop rate of three channels/second the time between any one BSS on a specific channel is about 3.6 seconds.

11 channels / 3 channels / 1 second = 3.6666 seconds.

The average configuration time when not hopping channels was determined to be 148ms from Test Setup 2. Since 333ms/148ms is 2.25, we would expect to squeeze in 1 to 2 configurations while on a specific channel. We should then observe a long configuration time until the Wi-Fi firmware returned to the channel the BSS existed on. The cycle can clearly be observed from figure 5.6.

The average number of Wi-Fi fragmented packets processed was 64 with a minimum and maximum of 36 and 93, respectfully. Interestingly, there were zero CRC failures.

5.2.1.5 Test Setup 5

Test Setup 5 was identical to Test Setup 4 with the exception of the amount of traffic on the Wi-Fi portion of the ESS. A large background ftp transfer was saturating the Wi-Fi link during the testing process.

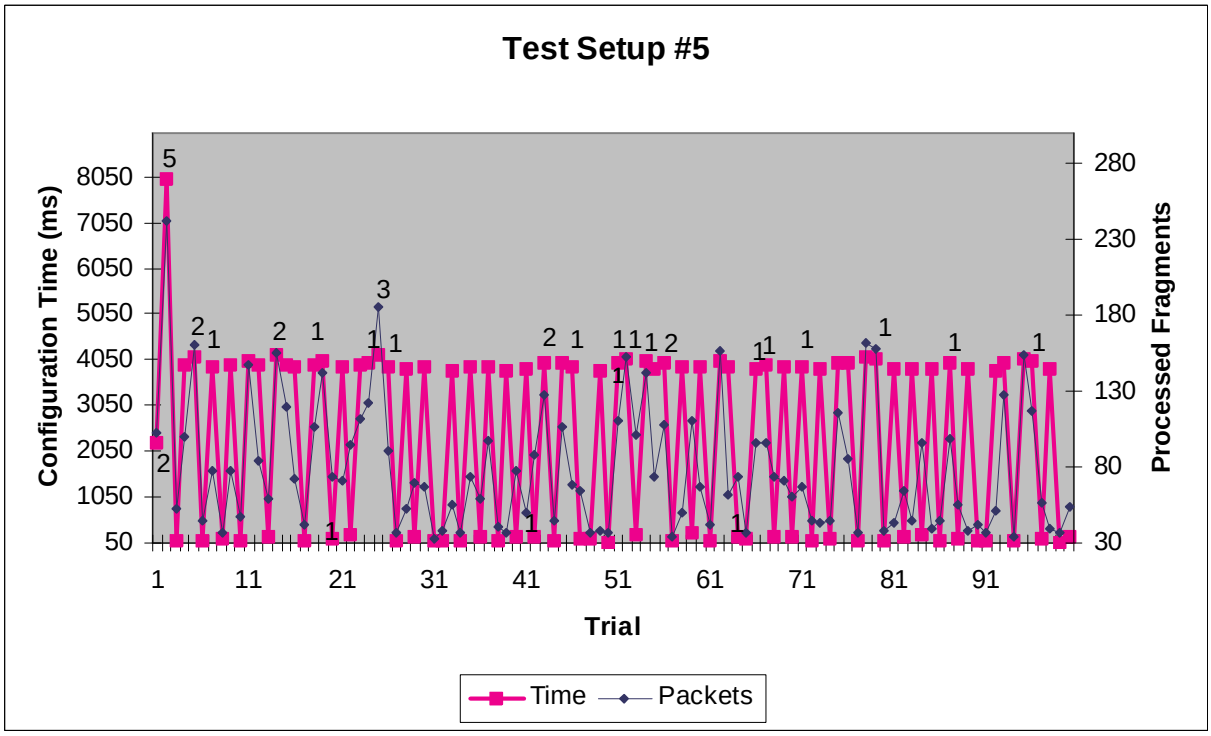


Figure 5.7 Results - Test Setup 5

The same correlation observed on the single channel low/high traffic tests was also observed here. With a saturated Wi-Fi link, a larger number of CRC failures occurred – 35 compared to zero in Test Setup #4. The configuration times and number of processed packets were also slightly higher. The average configuration time was 2,325ms with a minimum and maximum of 64ms and 7,987ms, respectfully. The average number of Wi-Fi-Co fragmented packets processed by the *Target* was 78 with a minimum and maximum of 36 and 252, respectfully.

5.2.1.6 Test Setup 6

Test Setup 6 consisted of a Linux Configuration and a Wi-Fi enabled Sharp Zaurus SL-5500 PDA as the *Target*. The Zaurus features a 206MHz Intel SA-1110 StrongARM processor and runs Embedix's Embedded Linux Operating System. The *Target* was hopping three channels a second.

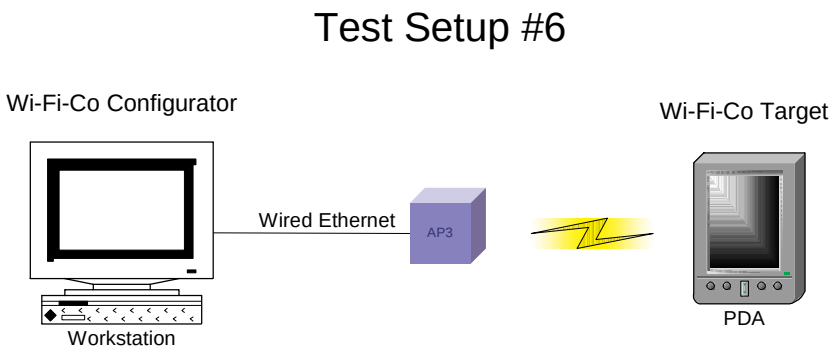


Figure 5.8 Test Setup 6

Figure 5.9 below shows the expected results. We can see that there are six CRC failures compared to zero with the laptop as the *Target* in the same setup. This is a result of processing power. The low speed PDA will occasionally drop packets, which causes increased CRC failures in the received configuration data. The average, minimum, and maximum configuration times was, 1,329ms, 45ms, and 4,236ms, respectfully. The average, minimum, and maximum number of Wi-Fi-Co packets processed was 48, 36, and 96, respectfully.

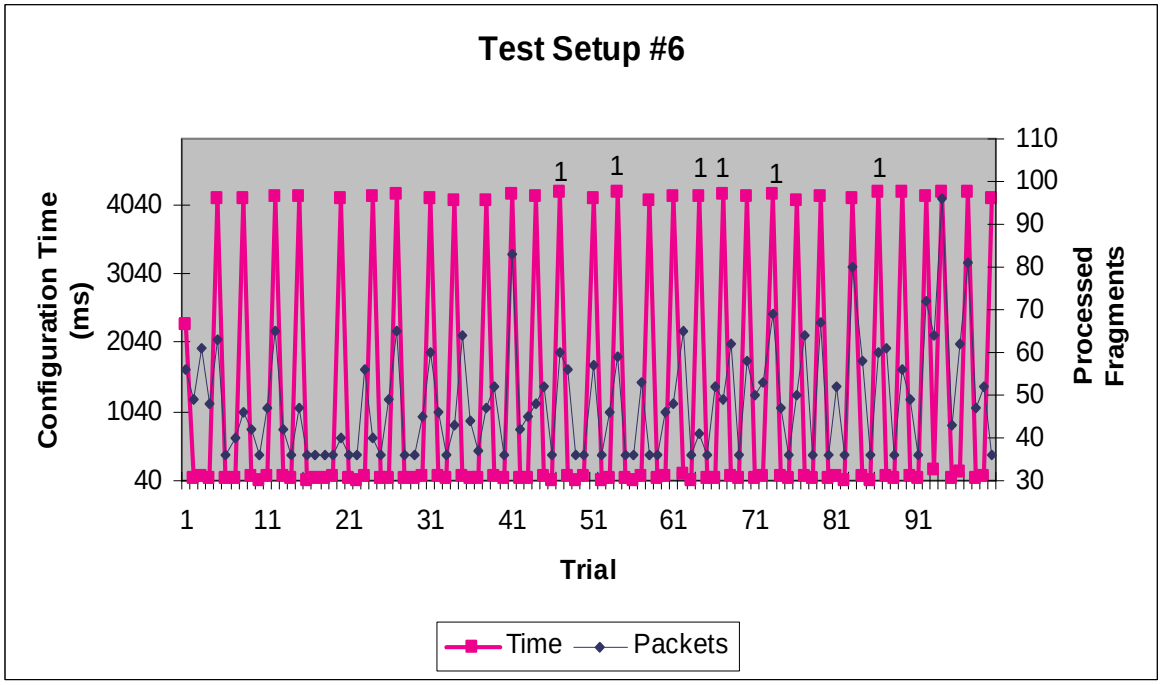


Figure 5.9 Results - Test Setup 6

5.2.2 Enterprise Network Test Results

The results presented in section 5.4 were performed on the network at the Information Telecommunication and Technology Center located at the University of Kansas. The network consists of approximately 160s host connected via 10/100 Mbit switched Wired Ethernet. Wireless connectivity is provided via ten Wi-Fi Access Points on 3 channels. There are four Orinoco AP-1000 Access Point and six Orinoco WavePoint IIs. The Wi-Fi network covers virtually the entire building.

5.2.2.1 Test Setup 7

Test Setup 7 consisted of a Linux Configuration host and a Wi-Fi enabled Linux Laptop. The *Target* was locked on a single channel known to be within range of the Extended Service Set.

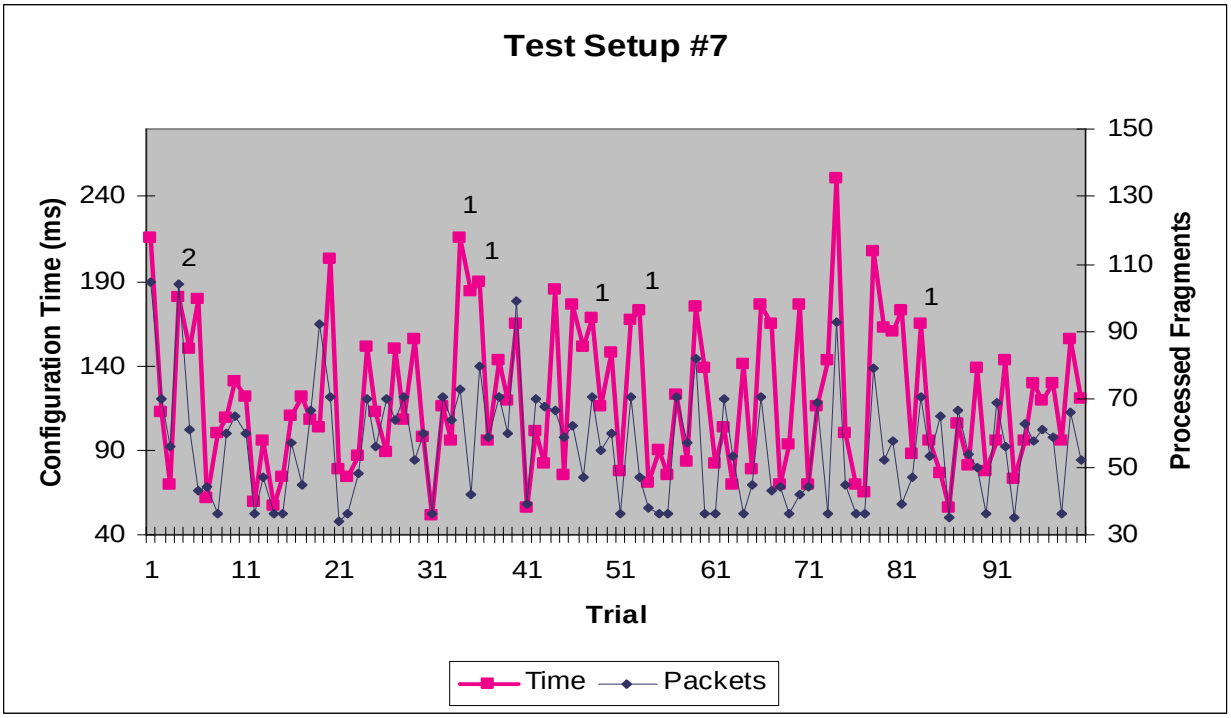


Figure 5.10 Results - Test Setup 7

The seven CRC failures are a result of a larger and more congested network. The average, minimum, and maximum configuration times were 120ms, 51ms, and 251ms, respectfully. The average, minimum, and maximum number of Wi-Fi-Co packets processed was 56, 34, and 105, respectfully.

5.2.2.2 Test Setup 8

Test Setup 8 was identical to Test Setup 7 with the exception of the amount of traffic on the Wi-Fi portion of the ESS. A large background ftp transfer was saturating the Wi-Fi link during the testing process.

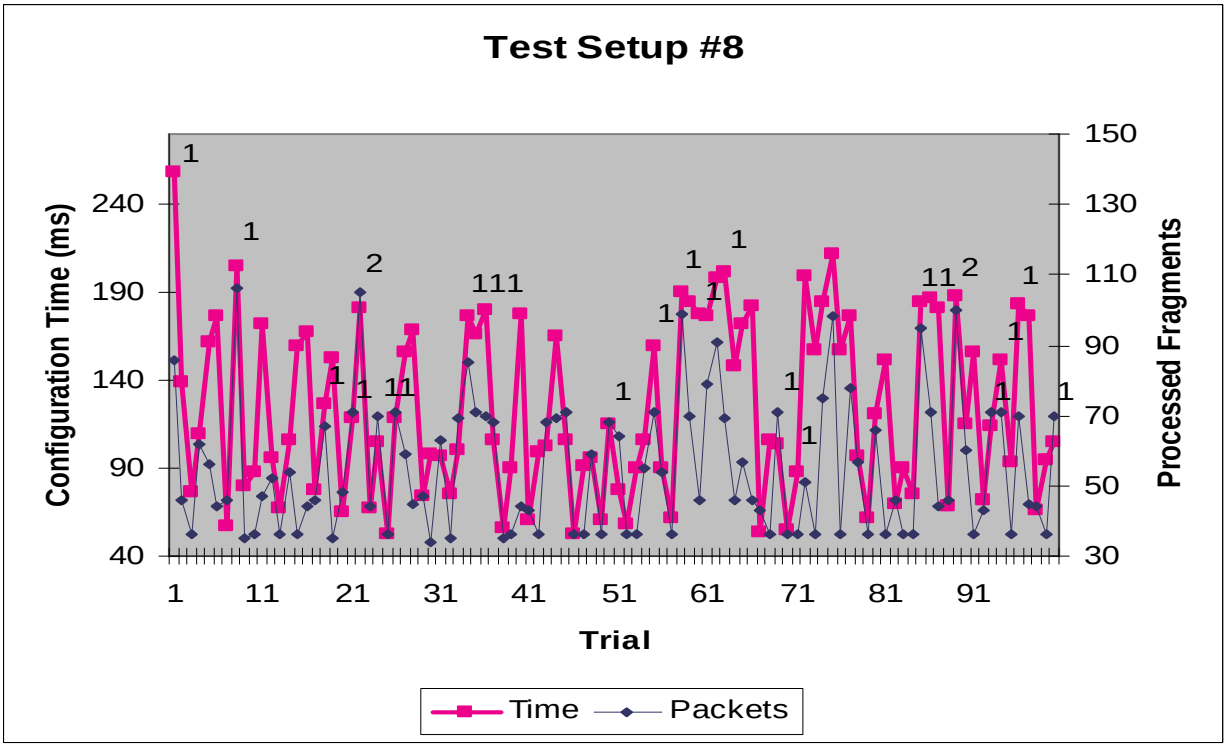


Figure 5.11 Results - Test Setup 8

Figure 5.11 shows a significant increase in the number of CRC failures: 25 compaired to 7. This setup resulted in a slightly higher configuration time. Interestingly, the number of processed Wi-Fi-Co configuration packets was about the same as Setup 7. The average, minimum, and maximum configuration times were 124ms, 53ms, and

258ms, respectfully. The average, minimum, and maximum number of Wi-Fi-Co packets processed were 55, 36, and 106, respectfully.

5.2.2.3 Test Setup 9

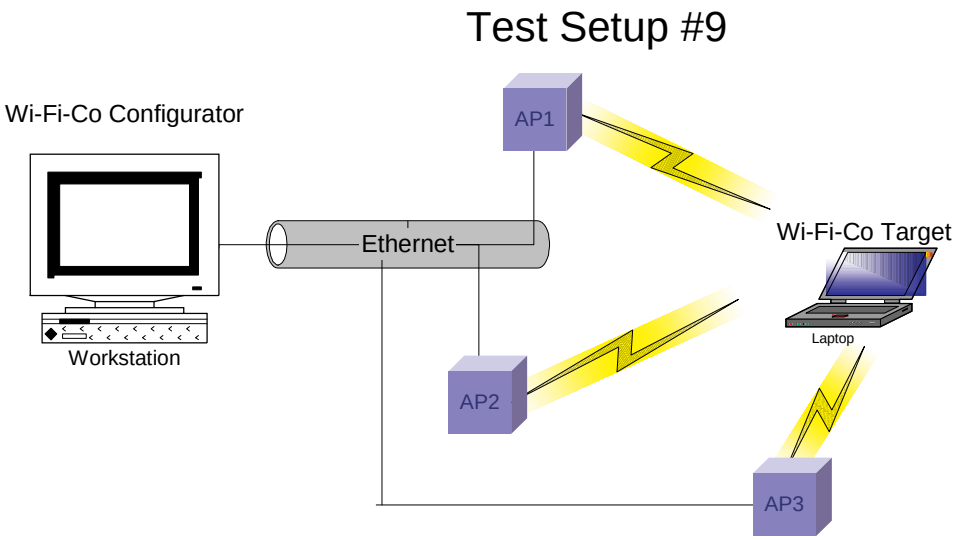


Figure 5.12 Test Setup 9

Test Setup 9 was identical to Setup 7 with the exception that the Wi-Fi Host's firmware was hopping three channels per second.

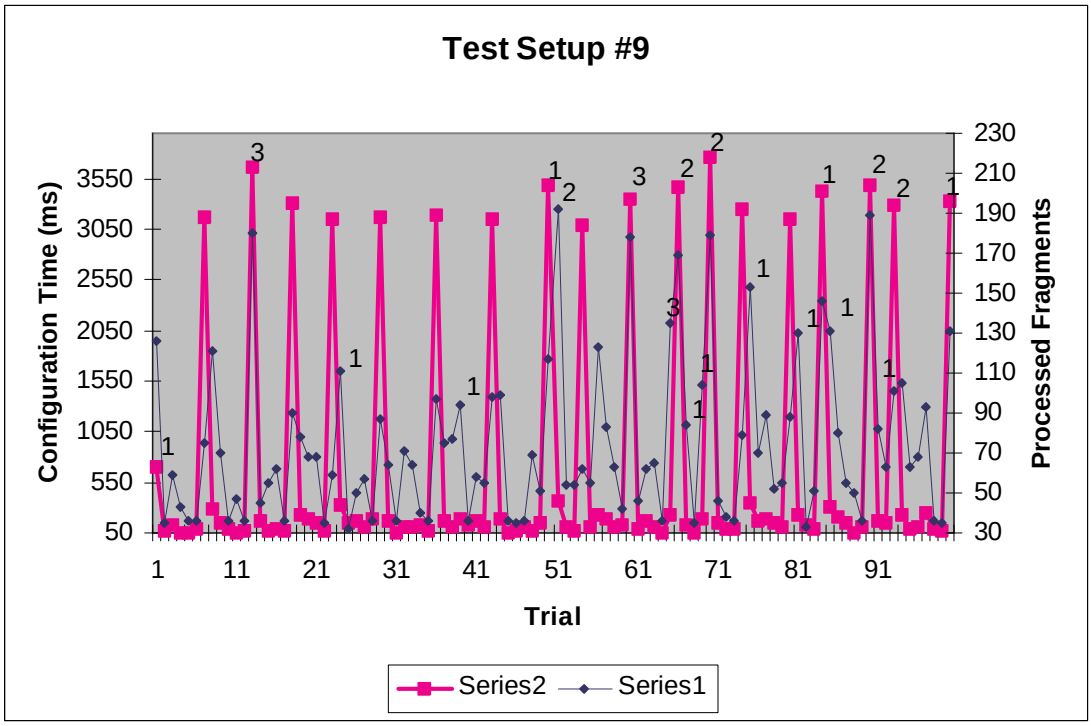


Figure 5.13 Results - Test Setup 9

The results for Setup 9 were very interesting. This experiment was designed to complement the identical experiment of Test Setup 4, but on a larger network. We would expect to see the same cycle as in figure 5.9. However, we observed a smaller number of peak configuration times and a higher number of "fast" configuration times. Careful examination of the signal strengths at the location revealed the reasons behind the numbers. There was three Access Points creating three Basic Service Set Networks (all belonging to the same Extended Service Set) within range of the *Target*. The Access Points were on channels 1,6, and 11. With a standard hopping sequence of 1,6,11,2… we would expect to be able to receive Wi-Fi-Co configurations packet for one full second before hopping to channels that did not

contain a BSS. This would imply 4-5 “fast” configurations times then a peek configuration time, as observed.

The average, min, and max configuration times were 718ms, 50ms, and 3,770ms, respectfully. The average, minimum, and maximum number of process Wi-Fi-Co packets processed was 74, 32, and 192, respectfully. There were 31 CRC failures.

5.2.2.4 Test Setup 10

The previous test results in Setup 9 inspired this experiment. The experiment was identical with the exception of channel hop sequence. The default hop sequence is 1,6,11,2,7,3,8,4,9,5,10,1. There were Access Points on channels 1,6, and 11. The hop sequence was changed to 1,2,3,4,5,6,7,8,9,1011.

As shown in figure 5.14, with a sequential hop sequence the expected result of a lower variance on the configuration times was observed. The average configuration time was also slightly smaller at 552ms. The minimum and maximum configuration times were 51ms and 2,323ms. The average, minimum, and maximum number of Wi-Fi-Co packets processed by the *Target* was 67, 34, and 224 packets, respectfully.

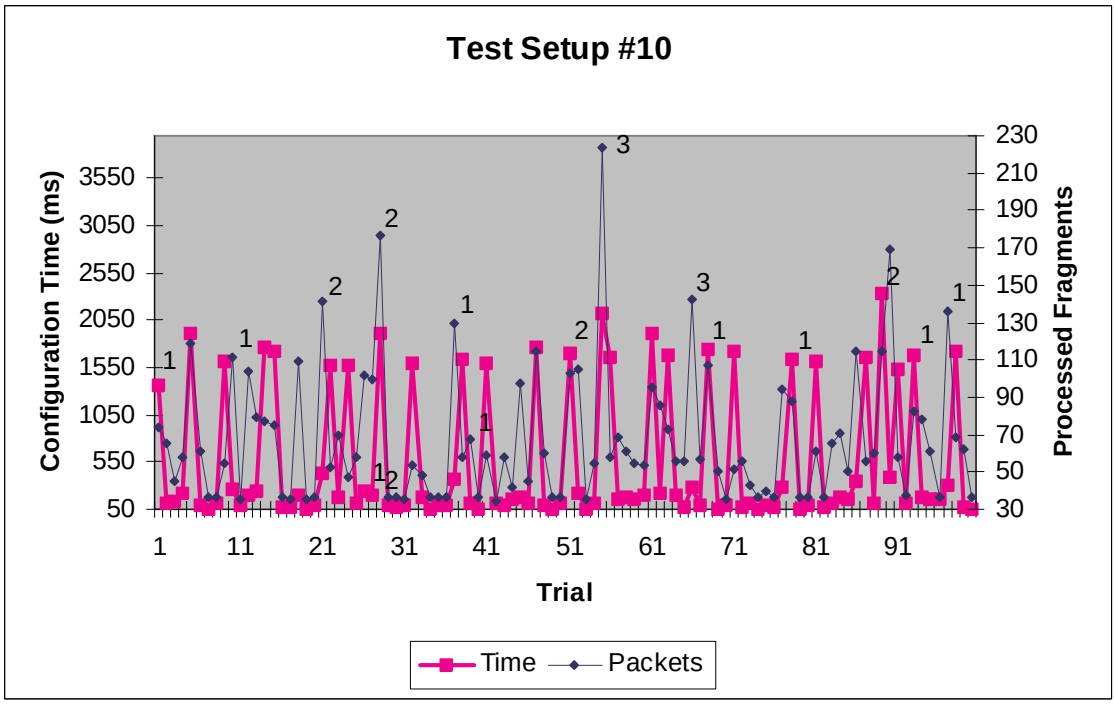


Figure 5.14 Result - Test Setup 10

5.3 Embedded Device Test Results

The inspiration and motivation for this project was the ability to configure an embedded Wi-Fi enabled device with the necessary parameters to gain link level configuration on a WEP enabled 802.11b network.

The embedded device was the Smart Wireless Thermostat developed by Ambient Computing, Inc [41]. The Thermostat contains an 8-bit microcontroller and a Wi-Fi Prism2.5 Network Interface Card. The Author designed and implemented the USB system driver and Prism2 interface drivers while working at Ambient

Computing, Inc. This experience allowed Wi-Fi-Co to be implemented directly into the network driver for the embedded device.

The Embedded device includes three buttons on the unit that support traditional thermostat operations such as adjusting and setting the temperature. The unit also boasts a built in Web server that enables many additional high level features.

The three physical buttons on the unit could theoretically facility some type of user input to receive the network configuration. The user experience would suffer significantly if the user were required to enter all the parameters by using two buttons to scroll through the alphabet and the third to select. The Thermostat's primary user interface is through the Web server that, unfortunately, is unusable until network configuration is complete. Until this implementation was created the only way to configure the Thermostat was to pre-program the Thermostats firmware with the network parameters.

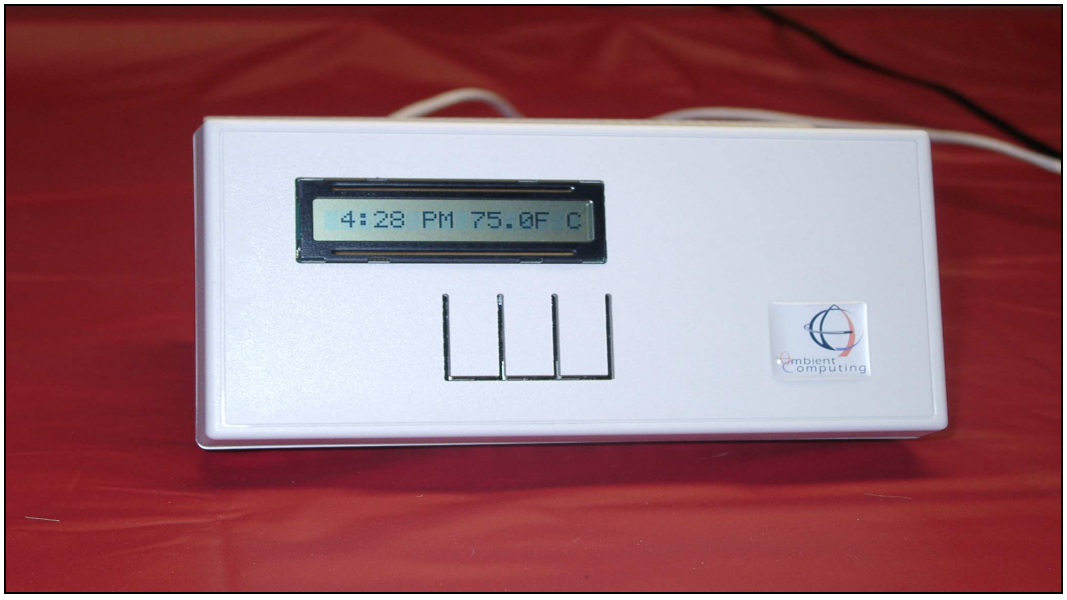


Figure 5.15 Smart Wireless Thermostat Embedded Device

5.3.1 Network Performance of Embedded Device

Before the results for the Thermostat are presented it should be noted that the device is a prototype and the USB and Wi-Fi drivers are initial experimental drivers. The Thermostat is powered by an 8-bit microcontroller running at 30 MHz. The un-optimized drivers and low speed of the processor result in much higher configuration times. Simple TTCP bandwidth results are presented below in table 5.1.

Table 5.1 TTCP Results

Buffer Size	Ethernet	Wired Thermo	Wireless Thermo
.5 MB	921 KB/s	123 KB/s	12.3 KB/s
1MB	1577 KB/s	140 KB/s	12.56 KB/s
5 MB	1924 KB/s	132 KB/s	12.52 KB/s

5.3.1.1 Test Setup 11

This experiment was performed on the Enterprise network in section 5.4. The *Configurator* was a Linux workstation. The *Target* was the Smart Wireless Thermostat.

The results from Setup 11 were slightly discouraging. Although the Configuration was successful, the maximum time observed was 45 seconds, compared to 145ms, from the Ethernet to Ethernet experiments. The slow times were not a direct result of the Wi-Fi-Co implementation. From table 5.1, it is apparent that the network performance of this embedded device is over 100 times worse then the

100Mbs Ethernet tests. The average configuration time for Test Setup 11 was 19,414ms that is about 130 times longer then Test Setup 2. The number of CRC errors are not denoted on the graph because there were 251 total CRC errors. The embedded device simply could not keep up with the configuration data. The *Configurator* was broadcasting out packets as fast as it could – about 500 packets a second. Test Setup 12 addresses this issue.

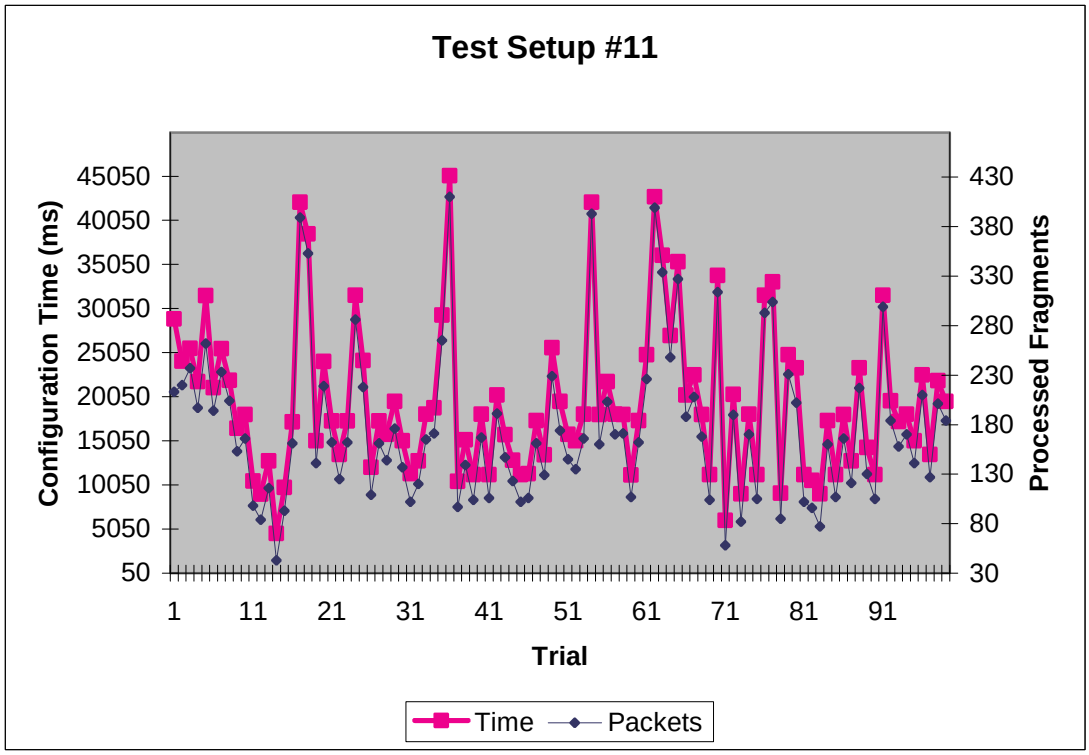
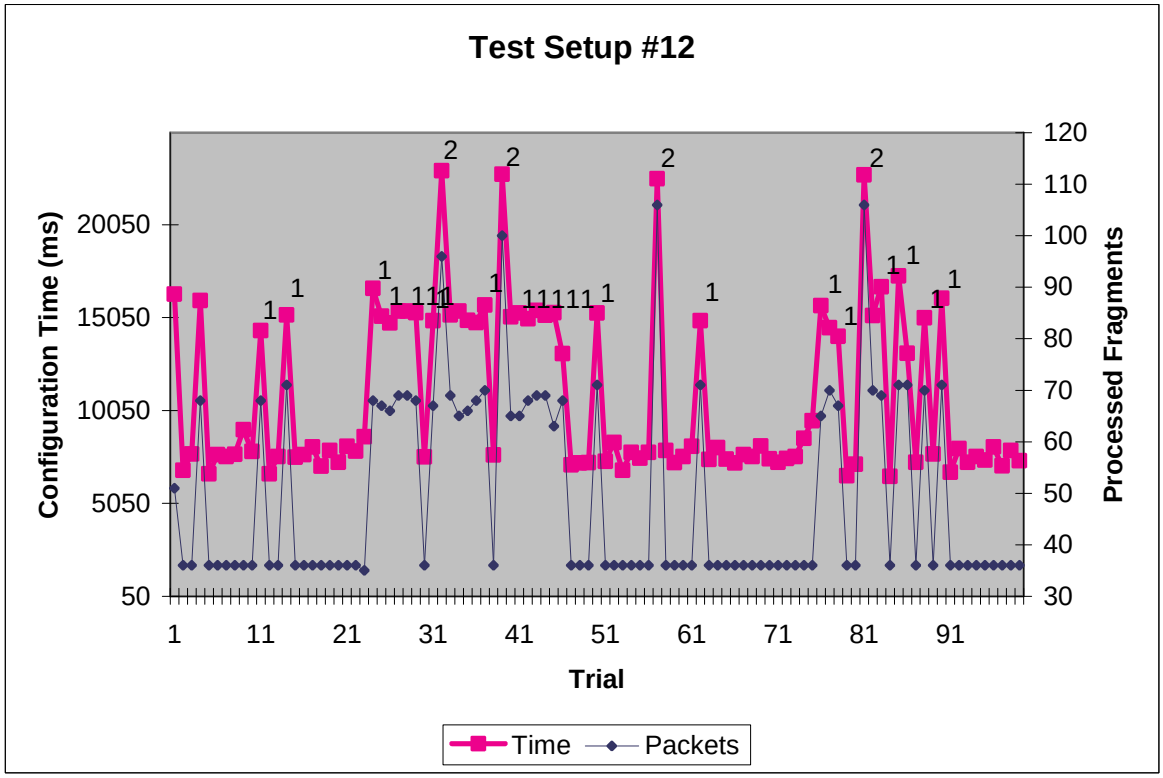


Figure 5.16 Results - Test Setup 11

5.3.1.2 Test Setup 12

Test Setup 12 was identical to Test Setup 11 with the exception of the rate at which the *Configurator* was sending the fragmented packets. The *Configurator* was limited to sending five packets a second.



The results from Test Setup 12 were much better then Test Setup 11. The average configuration time was cut in half to 10,815ms. That maximum configuration time was 22 seconds, instead of 45 seconds. There were 29 CRC failures compared to the 251 failures from Test Setup 11.

5.4 Summary of Results

The following table summarizes the configuration times and CRC failures from the previous sections.

Table 5.2 Summary of Configuration Times and CRC failures

	Test Setup Number					
	1	**2**	**3**	**4**	**5**	**6**
Average	121.02	148.79	157.27	1,869.91	2,216.89	1,378.34
Minimum	107.00	75.00	100.00	128.00	64.00	45.00
Maximum	145.00	222.00	352.00	4,031.00	4,141.00	4,236.00
Std.Deviation	4.28	16.87	46.35	1,858.60	1,889.14	1,900.59
CRC failures	0	0	6	0	22	6
	Test Setup Number					
	7	**8**	**9**	**10**	**11**	**12**
Average	119.72	124.23	718.32	552.27	19,413.69	10,815.08
Minimum	51.00	53.00	50.00	51.00	4,559.00	6,510.00
Maximum	251.00	258.00	3,770.00	2,323.00	45,101.00	22,948.00
Std. Deviation	43.75	48.77	1,228.20	700.71	8,488.02	4,393.61
CRC failures	7	25	31	23	251	29

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In the process of compiling the results section of this document, the underlying protocol created by this implementation was executed thousands of times without a single failure in the process. There were CRC failures that resulted in a slower configuration, but the process itself never failed. The implementation has successfully run on several networks, using many different Access Points and Wi-Fi network interface cards. The implementation was ported to Linux, Embedix Embedded Linux for the Sharp Zaurus PDA, and Dynamic C for an embedded device created by Ambient Computing, Inc. All goals of the project were successfully accomplished.

6.2 Challenges

In the early stages of this project, the goal was to create a protocol that would be coded directly into the setup routines for an embedded device. Eventually it was realized that this protocol was not just for embedded devices, but also is very helpful for link-level configuration of any device with limited input capabilities. Entering in lengthy WEP keys into a Personal Digital Assistant (PDAs) or a Web Tablets proved to be time consuming and annoying.

One of the most challenging problems was creating an implementation that facilitated porting to various other platforms and Operating Systems. For example, the Wi-Fi configuration data for RedHat, Debian, and Gentoo Linux distributions are all stored in different locations and have different file names. Furthermore, there are several Wi-Fi interface drivers to support the plethora of Wi-Fi Network Interface Cards available, each with a slightly different command interface. These requirements led to a solution that separates out the reception of the data and the configuration of the system. This was accomplished through the use of shell scripts. When the main executable receives the complete configuration data, it passes the data to a script via a system call. This allows the user to easily manipulate the data and create the necessary script for their distribution. Of course, scripts for many common Linux distributions are included in this implementation.

6.3 Lessons Learned

The protocol described in this document uses broadcast packets to accomplish remote configuration of simple Wi-Fi enabled devices. Due to the flooding nature of multicast communication, great care was taken to ensure that other hosts would simply ignore the packets. MAC addresses were chosen from the predefined experimental ranges and an unassigned Ethernet II Frame Type value was used. However, in the initial versions, the Author failed to limit the rate at which the packets were being sent. There were no noticeable effects on the 100Mbit portion of the network. However, it was discovered later that the packets were nearly saturating a long haul, low speed Wi-Fi link to a remote lab. Wi-Fi-Co was modified to send packets and a more network friendly rate.

6.3 Future Work

The project was successfully implement, however improvements can always be made. The following is a list of tasks that could improve the initial version of Wi-Fi-Co.

- Graphical version for the *Configurator*
- Create Microsoft Windows versions
- Research a possible replacement for the RC4 cryptography algorithm used to protect the WEP key. Perhaps there's a solution that will result in smaller user input on the Configuration side (sec section 3.3.5 for details).

Appendix A

AiroPeek Captured Configuration Packets

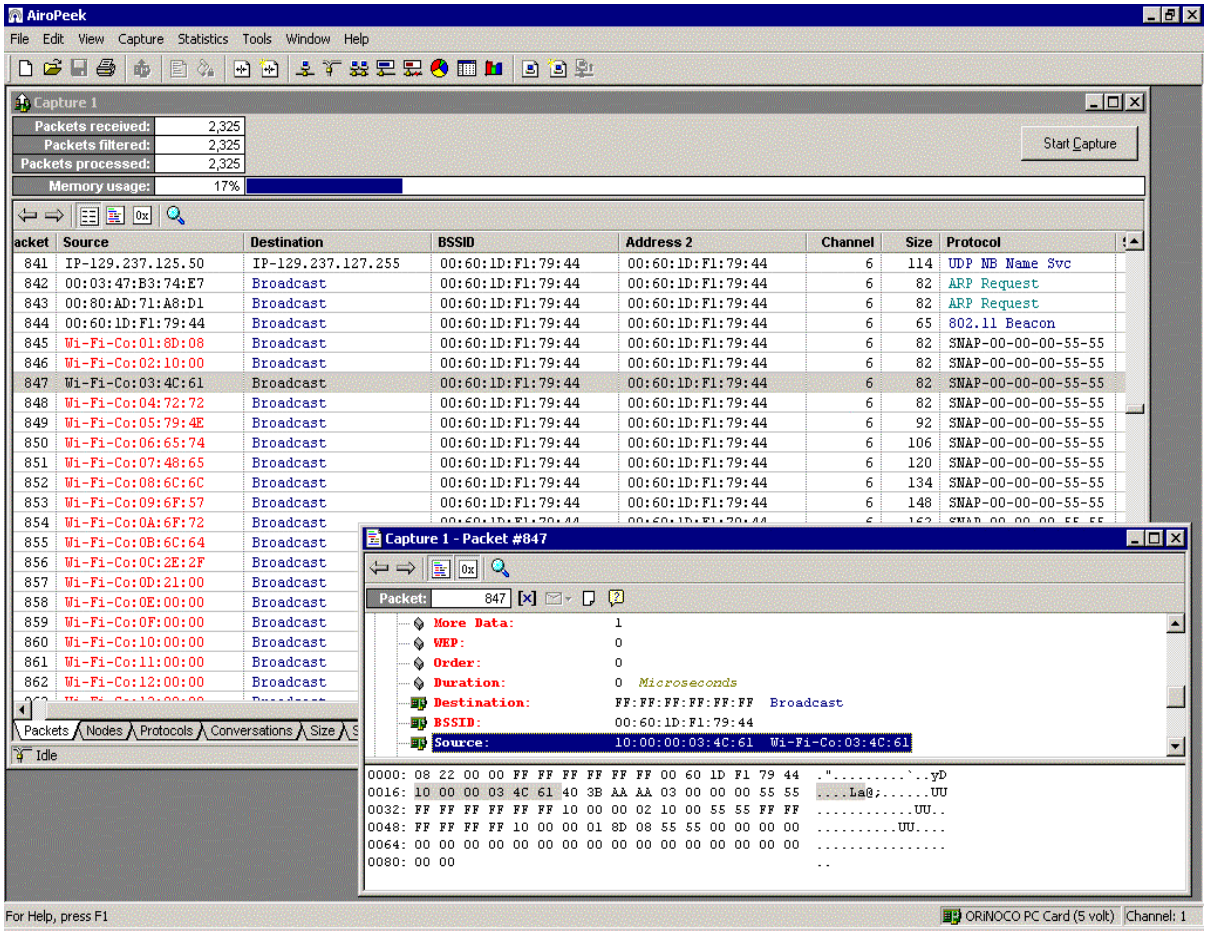


Figure A1 AiroPeek Captured Wi-Fi-Co Configuration Packets

Appendix B

Sample Configurator Output

[lsanders@essence wifico]# ***./wifico_config_linux –h***

Usage: ./wifico_config_linux [OPTIONS]
If no options - Run in interactive mode

Options:

-i, --interface <interface>	Interface (eth0 default)
-s, --ssid <ssid>	SSID (network name)
-[abcd], --wep-key[0-3] <key>	Wep key ASCII or xx:xx:xx...
-f, --feedback <ip:port>	Enable feedback to ip:port
-w, --default-wep-key <0-3>	The default WEP key
-n, --run-times <num>	Number of times to run (0 forever)
-v, --verbose	Verbose output
-h, --help	What you are reading

[lsanders@essence linux]# ***./wifico_config_linux --verbose***
Running in interactive mode

Interface to Broadcast configuration packets [eth0]: ***eth0***

SSID: ***wifico_test***

WEP KEY0: ***HopeThisWorks***

WEP KEY1 [HopeThisWorks]:

WEP KEY2 [HopeThisWorks]:

WEP KEY3 [HopeThisWorks]:

Default WEP key ID [0]: ***0***

wifico_pkt is:

SSID: "wifico_test" [len 11]
Mode: 0 [Infrastructure]

WEP key length: 13 [128/104 bits]
feedback: 1 [YES]

Default key: 0
WEP keys:
KEY0: 48 6f 70 65 54 68 69 73 57 6f 72 6b 73 [HopeThisWorks]
KEY1: 48 6f 70 65 54 68 69 73 57 6f 72 6b 73 [HopeThisWorks]
KEY2: 48 6f 70 65 54 68 69 73 57 6f 72 6b 73 [HopeThisWorks]
KEY3: 48 6f 70 65 54 68 69 73 57 6f 72 6b 73 [HopeThisWorks]
Feedback IP: 129.237.127.147
Feedback port: 7778

Sending packets:
Sending Packet #0 .....
Sending Packet #1 .....
Sending Packet #2 .....

Target assigned IP address: 129.237.127.147
Total number of Wi-Fi-Co Packets seen: 67
Length of the Wi-Fi-Co Buffer: 77 (39 packets)
Number of CRC Failures: 0

Bibliography

[1] The Wi-Fi Alliance. Wi-Fi backgrounder. c. 1999 <http://www.wi-fi.org/OpenSection/backgrounder.asp>.

[2] Information technology -- Telecommunications and information exchange between systems -- Local and metropolitan area networks – Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11, 1999.

[3] Gast, Matthew S. 802.11 Wireless Networks, The Definitive Guide. Sebastopol: O'Reilly, 2002.

[4] Fulford, Benjamin. "Sensors gone wild." Forbes 28 Oct. 2002. <http://www.forbes.com/global/2002/1028/076.html>.

[5] Charny, Ben. "Survey: DSL growth hits record high." MSNBC 10 Dec. 2002. <http://stacks.msnbc.com/news/845913.asp?cp1=1>.

[6] "WiFi and Ethernet Champion Home Networking, Report Says." Electronic News 9 Oct. 2002. <http://www.e-insite.net/electronicnews/index.asp?layout=article&articleid=CA251212&spacedesc=news>.

[7] The Campus Computing Project. "Campus Portals Make Progress; Technology Budgets Suffer Significant Cuts." Oct. 2002. <http://www.campuscomputing.net/summaries/2002/>.

[8] "With Footprint Expanding Rapidly, Hotspot Market Has Stellar Year in 2002." Electronic News 11 Dec. 2002. < http://www.instat.com/rh/en

/newmk.asp?id=453&SourceID=00000436000000000000>.

[9] "Cometa Networks Formed To Provide National Wireless Internet Access." Intel 5 Dec. 2002. <http://www.intel.com/pressroom/archive/releases/20021205corp.htm>

[10] Associated Press. "Wi-Fi wireless technology goes mainstream with Verizon service." USA Today 22 Nov. 2002 < http://www.usatoday.com/tech/news/2002-11-22-wifi-verizon_x.htm>.

[11] Charny, Ben. "Verizon takes Wi-Fi to the office." Business Week 15 Jan 2003 < http://www.businessweek.com/technology/cnet/stories/971023.htm>.

[12] Mehta, Stephanie N. "Making Pay Phones Pay." Fortune 7 Jan. 2003 <http://www.fortune.com/fortune/technology/articles/0,15114,405456,00.html>.

[13] "WLAN Chips to Embark on Incredible Journey." Electronic News 3 Apr. 2002 < http://www.instat.com/rh/en/newmk.asp?id=168&SourceID=00000167000000000000>.

[14] "Intel Outlines Future Banias Mobile PC Platform." Intel 10 Sept. 2002 <http://www.intel.com/pressroom/archive/releases/20020910comp.htm>.

[15] "Intel Invest $150M in 802.11" Electronic News 21 Oct. 2002 <http://www.e-insite.net/electronicnews/index.asp?layout=article&articleid=CA253839&spacedesc=news>.

[16] Stevenson, Reed. "AMD to move beyond PC, faster chips no longer key." Forbes 19 Nov. 2002 < http://www.forbes.com/newswire/2002/11/19/rtr799607.html>.

[17] "AMD Launches WLAN Devices." Electronic News 4 Nov. 2002 <http://www.e-insite.net/electronicnews/index.asp?layout=article&articleid=CA256995&spacedesc=news>.

[18] Dierdorff, Jack. "Where the Winners in Chips Are." Business Week 6 Dec. 2002 <http://www.businessweek.com/technology/content/dec2002/tc2002126_1241.htm>.

[19] Lawton, Stephen. "Eternally yours at 8 bits." Electronic Business 1 Oct. 2002 < http://www.e-insite.net/eb-mag/index.asp?layout=article&stt=000&articleid=CA245020&pubdate=10/1/2002>.

[20] "MCU Market to Experience Unit Growth Through 2006 While Prices Erode." In-Stat/MDR 10 Sept. 2002 < http://www.instat.com/newmk.asp?ID=325>.

[21] Open Systems Interconnection (OSI) page. International Organization for Standardization. < http://www.iso.org/iso/en/CatalogueListPage.CatalogueList?ICS1=35&ICS2=100>.

[22] FCC Radio Spectrum Home Page. Federal Communication Commission. <http://www.fcc.gov/oet/spectrum/>.

[23] RSA Security Company. Bedford, Mass. <http://www.rsasecurity.com>.

[24] Internet Security, Applications, Authentication and Cryptography. U. of California, Berkeley. c. July, 2001. < http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html>.

[25] Fluhrer, Scott, Itsik Mantin, and Adi Shamir. "Weaknesses in the Key

Scheduling Algorithm of RC4." c. Aug. 2001.

[26] WEP Fix using RC4 Fast Packet Keying. RSA Security. c. Sept. 2001

[27] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Draft 802.1X/D11, March 2001.

[28] Amendment to STANDARD [for] Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Medium Access Method (MAC) Security Enhancements, IEEE Draft 802.11i. 30 May 2001.

[29] Andersson, Hakan. "Wireless LAN upper layer authentication and key negotiation." RSA Security. 17 Jan. 2002.

[30] Blunk, L., J. Vollbrecht. "RFC 2284: PPP Extensible Authentication Protocol (EAP)." Internet Engineering Task Force. Mar. 1998. < http://www.ietf.org/rfc/rfc2284.txt>.

[31] Leon-Gracia, Alberto and Indra Widjaja. Communication Networks, Fundamental Concepts and Key Architectures. Boston: McGraw-Hill, 2000.

[32] Moskowitz, Robert. "The Myth of Hiding SSIDs." The Internet Security Conference. Vo. 6., Issue 16. 25 Oct. 2002.

[33] Ethernet Encapsulation Page. Cisco. 30 Oct. 2002. <http:// www.cisco.com/warp/public/105/encheat.html>.

[34] Information Technology – Telecommunications and information exchange

between systems – Local and metropolitan area networks – Technical reports and guidelines – Part 5: Media Access Control (MAC) Bridging of Ethernet V2.0 in Local Area Networks, IEEE Std. 802.1h, 1997.

[35] Postel, J., J. Reynolds. "A Standard for the Transmission of IP Datagrams over IEEE 802 Networks." Internet Engineering Task Force. Feb. 1988. <http://www.ietf.org/rfc/rfc1042.txt>.

[36] IEEE MAC allocation page. IEEE. < http://standards.ieee.org/ regauth/oui/oui.txt>.

[37] IDI's Ethernet Adapter Identification. International Development Institue. <http://www.playerslounge.com/scripts3/ether.idc?teamkey=HACKING2>

[38] Libnet Packet Assembly. The Packet Factory. <http://www.packetfactory.net/projects/libnet/>

[39] Tcpdump/libpcap. Lawrence Berkeley National Laboratory. http://www.tcpdump.org/

[40] Kismet Wireless network sniffer. http://www.kismetwireless.net/

[41] Ambient Computing, Inc. "Smart Wireless Thermostat Proof of Concept Detailed Report." June, 2002.

[42] Qt/Embedded OVERVIEW. Trolltech. <http://www.trolltech.com/products/embedded/>

[43] The Complete, Unofficial TEMPEST Information Page. Joel McNamar. <http://www.eskimo.com/~joelm/tempest.html#start>

[44] Ethertypes. < http://www.sinclair.org.au/keith/networking/type.html>