GS-AN001

802.11 Packet Sniffer

Introduction

ONE OF the best ways to learn about a particular network protocol is to watch real live traffic. The best way to do this is with a protocol packet analyzer. This is software that captures all packets on a medium, parses them to determine the relevant protocols, dissects the packet into its constituent fields, and displays these fields in a meaningful way. IEEE 802.11 is no exception.

Components of an 802.11 Packet Sniffer

The three components required to make an 802.11 packet sniffer are:

1. Host computer
2. WLAN (IEEE 802.11) network interface
3. Packet analysis software

Wireshark

For a while, the most popular packet analyzer was a free tool named *Ethereal*. Originally released in 1998, it rapidly grew in popularity and scope. New protocols could be easily added, so over time many contributors developed new protocol dissectors, as needed. In 2006 the project left behind the *Ethereal* name and changed to *Wireshark*, but is still maintained by the same group of people.

Wireshark runs on both Microsoft Windows and Linux operating systems. For Linux, it relies on the *libpcap* library to perform packet capture. On Windows, the *WinPcap* utility is used. These programs rely on the network interface card (NIC) driver to provide raw packet data. The pcap utility (*libpcap* or *WinPcap*) then provides an API which *Wireshark* uses to extract packets. Any packet capture program that can write .pcap output files can also be used offline.

Wireshark works very well for viewing packets on Ethernet. You can see all TCP/IP packets, as well as lower level protocols, such as DHCP, ARP, etc. It also dissects packets of higher-level protocols, such as FTP or HTTP. Typically, by default, Ethernet network interface drivers only provide packets to the pcap utility that originate from or are addressed for the host's network interface (MAC address), or broadcast packets. In addition, most Ethernet NIC drivers provide a way to put the interface in ***promiscuous*** mode, thereby capturing every packet on the shared medium. Modern Ethernet networks are configured to use switches, which provide a high-bandwidth point-to-point connection between an end node (the host) and the switch. Since the router will only place directed or broadcast traffic onto the wire, promiscuous mode for Ethernet does not actually capture any additional packets. However, promiscuous mode is of great utility in the wireless medium, as we shall see below.

IEEE 802.11

IEEE 802.11, often referred to as Wi-Fi®, is far and away the most popular wireless LAN (WLAN) technology in the world. A Wi-Fi network is designed to look like an Ethernet LAN, but without the wires. Wi-Fi uses radio-frequency (RF) technology and therefore the air is used as the shared medium. Wi-Fi transmissions are always broadcast to whoever is in the area with an active receiver tuned to the correct channel.

Over the years, 802.11 has been amended to improve bandwidth, throughput, and performance. These different amendments are known as 802.11a, 802.11b, and 802.11g. (At the time of this writing, another amendment, 802.11n, is pending final ratification.) 802.11b offers the lowest maximum data rate (up to 11 Mbps), and operates at 2.4 GHz; it is backwards-compatible with the original 802.11 DSSS "classic" data rates of 1 and 2 Mbps. 802.11a allows up to 54 Mbps, but operates in the 5.2-5.8 GHz band, and therefore is not compatible with 802.11b or 802.11 "classic". 802.11g operates in the same frequency band as 802.11b (2.4 GHz), but at the higher 802.11a data rates (up to 54 Mbps). 802.11g has been carefully designed to coexist with older, slower 802.11b systems.

Both the 5GHz (802.11a) and 2.4GHz (802.11b/g) bands have been further subdivided into individual RF channels. In the USA, there are eleven 5-MHz channels in the 2.4 GHz band. However, 802.11b/g signals are actually wider than a single channel. In practice there are only three non-overlapping channels, often labeled 1, 6, and 11.

Most users of IEEE 802.11 operate in ***infrastructure mode***, where all ***stations*** (i.e., client computers) talk to – ***associate*** with – a single ***access point*** (AP) on a single radio channel. The access point is the gateway between the WLAN and a ***distribution system*** (DS), such as a corporate wired LAN. The DS forwards packets sent from associated stations to the LAN, and buffers LAN packets bound for the stations. Each AP has a name (the ***Service Set Identifier*** or SSID), which is used in all packets on its WLAN.

Association is important because multiple APs can use the same RF channel in the physical vicinity at the same time. In fact, many APs and stations can be competing for RF medium access simultaneously. The IEEE 802.11 medium access algorithm (***Carrier Sense Multiple Access with Collision Avoidance***: CSMA/CA) deals with collisions between stations in a flexible and robust manner. Nevertheless, performance of any given station can be degraded when many stations are simultaneously contending for the same radio channel. Unencrypted traffic between a station and an AP is readily intercepted by any nearby receiver; the wireless medium is not intrinsically secure.

IEEE 802.11 MAC

The 802.11 ***Medium Access Control*** (MAC) function needs to manage association, authentication, medium contention resolution, and error handling. There are three main types of IEEE 802.11 MAC frames: control, management and data. Many different frame subtypes are used to perform these different tasks.

Control frames assist in data delivery. They are used to control access to the medium (e.g., ***request-to-send*** RTS, ***clear-to-send*** CTS, and ***acknowledgement*** of a packet, ACK). Management frames provide all the functions needed for stations to find and start communicating on the WLAN: finding a network, authenticating, association with an AP. Data frames carry useful data and can, in some cases, contain acknowledgements of previous data.

IEEE 802.11 for Microsoft Windows

IEEE 802.11 is supported by all major modern operating systems, including Microsoft Windows and the major distributions of Linux. There are over a hundred available Wi-Fi ***network interface cards*** (NICs) from many different vendors with a variety of interfaces: PC Card, USB, PCI, etc. Most importantly for this discussion, these cards are based on a small subset of IEEE 802.11 chipsets. These chipsets perform the lowest layers (MAC and PHY) of the 802.11 network stack. Each NIC requires its own driver, but NICs that use the same chipset can generally have very similar, if not identical, drivers.

Unfortunately, most of these Windows drivers do not allow promiscuous operation: that is, they supply only those packets addressed to the NIC itself to the application layer software. Thus, they cannot be used for packet sniffing. With such drivers, *WinPcap* and *Wireshark* treat the Wi-Fi NIC as an Ethernet NIC, and only show the contents of 802.11 data frames that are either originating from or bound for that NIC. In fact, if Ethernet promiscuous mode is enabled for these drivers, then *WinPcap* does not even report these packets.

Without a true promiscuous driver, it is impossible to see what is really happening on the WLAN. To analyze operation of the link, we need to captures all types of 802.11 frames (control, management and data). Since any given radio channel is a shared medium, all packets on that channel must be captured to evaluate true traffic load and interference levels.

IEEE 802.11 for Linux

IEEE 802.11 support on Linux has evolved over the years. Unlike for Windows, most commercial WLAN products do not come with native Linux drivers on a CD. However, there is a Linux program called *ndis-wrapper* that solves this problem. It permits use of a native Windows driver in Linux. This is great for connecting to and using a WLAN with Linux. However, this encapsulated driver comes with all the same restrictions as the original – no promiscuous mode, so no packet sniffing.

Instead, we must turn to native Linux drivers for WLAN hardware. Luckily, as noted earlier, there are far fewer WLAN chipsets than individual WLAN products. Over the years, people have written native drivers for any of these chipsets where the specifications are publicly available. Unfortunately, not all chipset vendors make these specs available for open source driver implementation. It turns out that most chipsets natively support a promiscuous mode and the Linux drivers expose this. In Linux this is often called "monitor" mode.

BUILDING A SNIFFER

Requirements

For our sniffer we chose:

- Laptop computer (IBM ThinkPad T30)
- Linux (Ubuntu 7.04, Feisty Fawn)
- US Robotics USR805423 Wireless USB Adapter
- *Wireshark*

The USR805423 is a very inexpensive IEEE802.11g (up to 54Mbps) wireless USB adapter. Like most wireless USB adapters, it is based on the Zydas ZD1211b chipset. This chipset has a native Linux driver, *zd1211rw*, that supports monitor (promiscuous) mode. In fact, this driver has been included in the kernel since Linux 2.6.18.

The particular laptop does not matter, *per se*, but it needs to meet the minimum requirements to run Ubuntu 7.04, *Wireshark*, and include a free USB port (preferably USB2.0).

Installing Ubuntu Linux

Ubuntu (http://www.ubuntu.com) is a distribution of Linux that is free, downloadable, and very easy to install. Ubuntu issues scheduled releases every six months, and support each release for 18 months from its initial release. Originally based on the Debian distribution, Ubuntu is a complete package in its own right, and includes, pre-installed, many useful, free, and open source applications. To install Ubuntu, go to their website, and click on Get Ubuntu, and then Download now. This will allow selection of the most recent version, which, as of July 2008, is Ubuntu 8.04. This should start a download of a CD image file (.iso). There are instructions on how to burn this image to a CD and how to use that CD to boot a computer and install Ubuntu.

Once everything is downloaded and the CD is burned, it is time to install Ubuntu. During installation, Ubuntu will figure out all installed hardware and install and configure all required compatible drivers. So, to expedite the wireless adapter installation, it is important to plug it in before starting Ubuntu installation.

Configuring the WLAN adapter

Before we can start capturing control and management 802.11 packets with *Wireshark*, we first need to enable monitor mode for the wireless adapter. We do this at the Linux command prompt. First, open a terminal window (Applications → Accessories → Terminal). To check for the installed WLAN device, use *iwconfig*. There should be a list of installed network devices, one or more of which provides WLAN services. Below is an example, showing a single IEEE 802.11 b/g device, `eth1`.

```
user@host:~$ iwconfig

eth0        no wireless extensions.

eth1        IEEE 802.11b/g  ESSID:"GAINSPAN"  Nickname:"zd1211"
            Mode:Managed  Frequency:2.462 GHz  Access Point: Invalid
            Bit Rate=1 Mb/s
            Link Quality:0  Signal level:0  Noise level:0
            Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
            Tx excessive retries:0  Invalid misc:0  Missed beacon:0
```

By default, the WLAN driver is in managed mode. For packet sniffing, we want to change to monitor (promiscuous) mode. Only the root account can change the device mode. The UNIX `sudo` command automatically executes what follows as root (but you must provide the root password).

```
user@host:~$ sudo iwconfig eth1 mode monitor
Password:
```

Similarly, before sniffing, the IEEE 802.11 channel should be set (e.g., to sniff on 802.11b/g channel 6):

```
user@host:~$ sudo iwconfig eth1 freq 6
```

Installing *Wireshark*

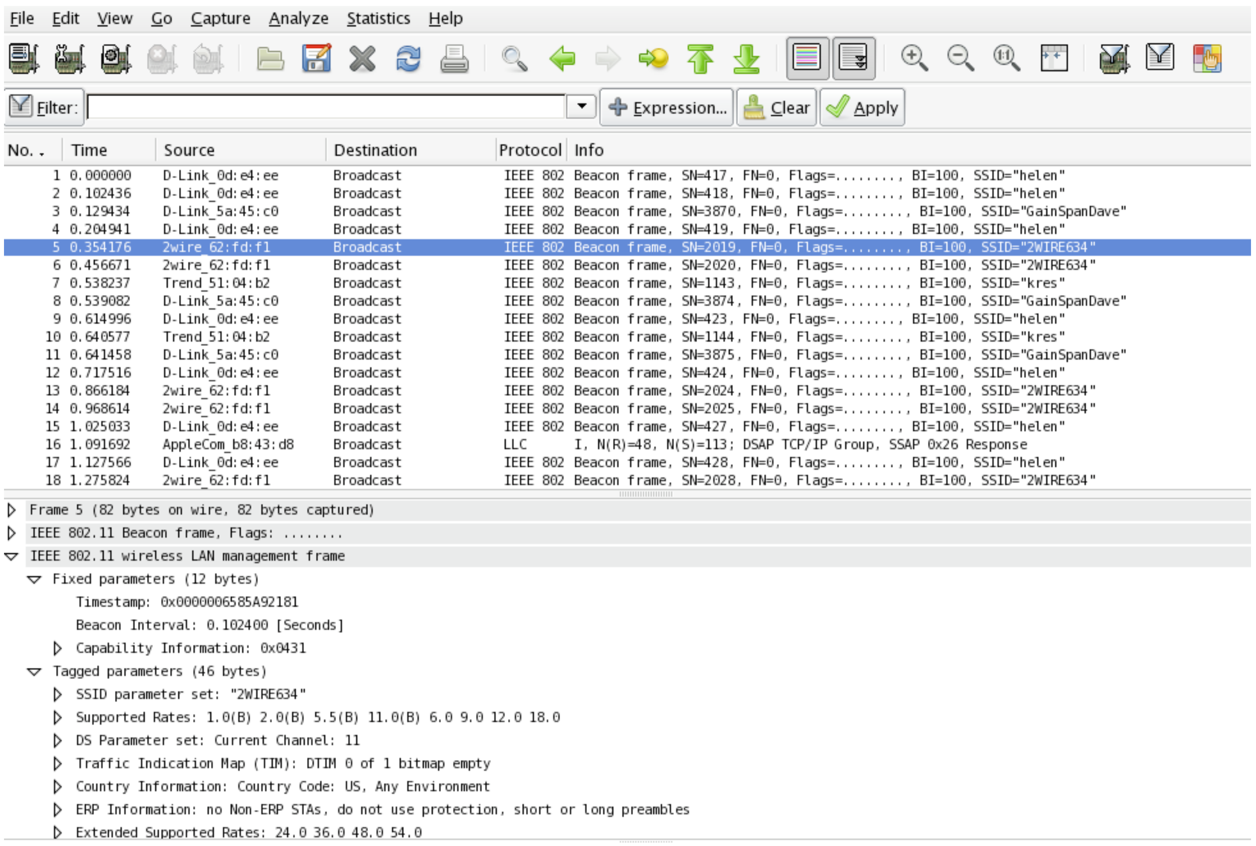
Once Ubuntu is up and running, it is very easy to install *Wireshark*. Ubuntu comes with an integrated software package installer. Just click on Applications → Add/Remove… This launches a handy application search and install utility. Search for *Wireshark*, select Wireshark (as root), and click Apply. The utility will download and install any required prerequisites, as well as the selected program, so a network connection is required. Alternatively, point your browser at http://www.wireshark.org. The "as root" version is required since *Wireshark* needs root privileges to capture packets using *libpcap*.

With *Wireshark* installed, our sniffer is ready to sniff!

SNIFFING WITH *WIRESHARK*

To sniff, *Wireshark* must be started with root privileges – launching the *Wireshark (as root)* script will do this automatically. Once started, the Capture → Interfaces… menu option opens a screen that shows all sniffable interfaces. The interface from *iwconfig* should appear. Clicking Options opens a window which allows selecting Capture packets in promiscuous mode and Update list of packets in real time. Lastly, clicking Start should display tons of traffic!

An example traffic stream is depicted in Figure 1. Each packet is timestamped using the local (host computer) clock. The source and destination MAC addresses, and summary info about each packet, are displayed in a table format. You can click on a row in the table to select that packet; detailed analysis of its contents is displayed in the frame below the table.



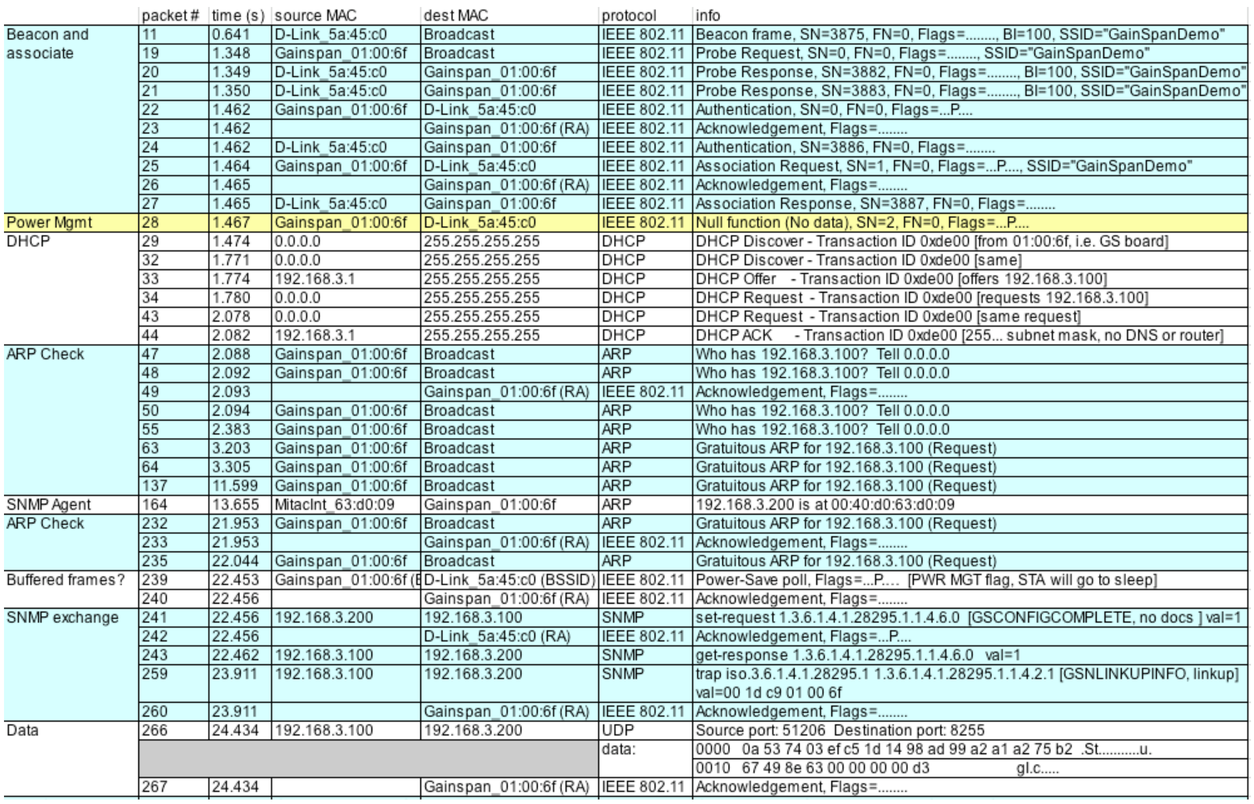
```
No. .  Time      Source             Destination   Protocol  Info
     1 0.000000  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=417, FN=0, Flags=........, BI=100, SSID="helen"
     2 0.102436  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=418, FN=0, Flags=........, BI=100, SSID="helen"
     3 0.129434  D-Link_5a:45:c0    Broadcast     IEEE 802  Beacon frame, SN=3870, FN=0, Flags=........, BI=100, SSID="GainSpanDave"
     4 0.204941  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=419, FN=0, Flags=........, BI=100, SSID="helen"
     5 0.354176  2wire_62:fd:f1     Broadcast     IEEE 802  Beacon frame, SN=2019, FN=0, Flags=........, BI=100, SSID="2WIRE634"
     6 0.456671  2wire_62:fd:f1     Broadcast     IEEE 802  Beacon frame, SN=2020, FN=0, Flags=........, BI=100, SSID="2WIRE634"
     7 0.538237  Trend_51:04:b2     Broadcast     IEEE 802  Beacon frame, SN=1143, FN=0, Flags=........, BI=100, SSID="kres"
     8 0.539082  D-Link_5a:45:c0    Broadcast     IEEE 802  Beacon frame, SN=3874, FN=0, Flags=........, BI=100, SSID="GainSpanDave"
     9 0.614996  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=423, FN=0, Flags=........, BI=100, SSID="helen"
    10 0.640577  Trend_51:04:b2     Broadcast     IEEE 802  Beacon frame, SN=1144, FN=0, Flags=........, BI=100, SSID="kres"
    11 0.641458  D-Link_5a:45:c0    Broadcast     IEEE 802  Beacon frame, SN=3875, FN=0, Flags=........, BI=100, SSID="GainSpanDave"
    12 0.717516  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=424, FN=0, Flags=........, BI=100, SSID="helen"
    13 0.866184  2wire_62:fd:f1     Broadcast     IEEE 802  Beacon frame, SN=2024, FN=0, Flags=........, BI=100, SSID="2WIRE634"
    14 0.968614  2wire_62:fd:f1     Broadcast     IEEE 802  Beacon frame, SN=2025, FN=0, Flags=........, BI=100, SSID="2WIRE634"
    15 1.025033  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=427, FN=0, Flags=........, BI=100, SSID="helen"
    16 1.091692  AppleCom_b8:43:d8  Broadcast     LLC       I, N(R)=48, N(S)=113; DSAP TCP/IP Group, SSAP 0x26 Response
    17 1.127566  D-Link_0d:e4:ee    Broadcast     IEEE 802  Beacon frame, SN=428, FN=0, Flags=........, BI=100, SSID="helen"
    18 1.275824  2wire_62:fd:f1     Broadcast     IEEE 802  Beacon frame, SN=2028, FN=0, Flags=........, BI=100, SSID="2WIRE634"
```

```
▷ Frame 5 (82 bytes on wire, 82 bytes captured)
▷ IEEE 802.11 Beacon frame, Flags: ........
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (12 bytes)
      Timestamp: 0x0000006585A92181
      Beacon Interval: 0.102400 [Seconds]
    ▷ Capability Information: 0x0431
  ▽ Tagged parameters (46 bytes)
    ▷ SSID parameter set: "2WIRE634"
    ▷ Supported Rates: 1.0(B) 2.0(B) 5.5(B) 11.0(B) 6.0 9.0 12.0 18.0
    ▷ DS Parameter set: Current Channel: 11
    ▷ Traffic Indication Map (TIM): DTIM 0 of 1 bitmap empty
    ▷ Country Information: Country Code: US, Any Environment
    ▷ ERP Information: no Non-ERP STAs, do not use protection, short or long preambles
    ▷ Extended Supported Rates: 24.0 36.0 48.0 54.0
```

Figure 1: Example of 802.11 traffic captured with Wireshark.

Most modern office environments have numerous 802.11 WLANs operating. *Wireshark* may capture a lot of packets, many of which may be of little interest in analyzing a specific link. You can use *Wireshark*'s Filter capability to look at only a certain type of packet, or only packets from a given source or targeted to a specific Destination. Filtering is very helpful in reducing the voluminous traffic record to useful information.

Filtered frames can be extracted and interpreted in detail. An example summary of the exchange between an access point and a GainSpan sensor node is shown in Figure 2. We can see that the node sends a probe packet to find the appropriate SSID (GainSpanDemo here), and a handshake process follows leading to association with the AP. The node then interacts with the DHCP server to get an IP address, sends an SNMP Linkup Trap message to maintain the association, and sends a data packet. The 802.11 Power Save mechanism is used by the node to get buffered frames from the AP, so that it can spend much of its time in the low-power Standby state without missing messages.



	packet #	time (s)	source MAC	dest MAC	protocol	info
Beacon and associate	11	0.641	D-Link_5a:45:c0	Broadcast	IEEE 802.11	Beacon frame, SN=3875, FN=0, Flags=........, BI=100, SSID="GainSpanDemo"
	19	1.348	Gainspan_01:00:6f	Broadcast	IEEE 802.11	Probe Request, SN=0, FN=0, Flags=........, SSID="GainSpanDemo"
	20	1.349	D-Link_5a:45:c0	Gainspan_01:00:6f	IEEE 802.11	Probe Response, SN=3882, FN=0, Flags=........, BI=100, SSID="GainSpanDemo"
	21	1.350	D-Link_5a:45:c0	Gainspan_01:00:6f	IEEE 802.11	Probe Response, SN=3883, FN=0, Flags=........, BI=100, SSID="GainSpanDemo"
	22	1.462	Gainspan_01:00:6f	D-Link_5a:45:c0	IEEE 802.11	Authentication, SN=0, FN=0, Flags=...P....
	23	1.462		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........
	24	1.462	D-Link_5a:45:c0	Gainspan_01:00:6f	IEEE 802.11	Authentication, SN=3886, FN=0, Flags=........
	25	1.464	Gainspan_01:00:6f	D-Link_5a:45:c0	IEEE 802.11	Association Request, SN=1, FN=0, Flags=...P...., SSID="GainSpanDemo"
	26	1.465		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........
	27	1.465	D-Link_5a:45:c0	Gainspan_01:00:6f	IEEE 802.11	Association Response, SN=3887, FN=0, Flags=........
Power Mgmt	28	1.467	Gainspan_01:00:6f	D-Link_5a:45:c0	IEEE 802.11	Null function (No data), SN=2, FN=0, Flags=...P....
DHCP	29	1.474	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xde00 [from 01:00:6f, i.e. GS board]
	32	1.771	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0xde00 [same]
	33	1.774	192.168.3.1	255.255.255.255	DHCP	DHCP Offer - Transaction ID 0xde00 [offers 192.168.3.100]
	34	1.780	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0xde00 [requests 192.168.3.100]
	43	2.078	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0xde00 [same request]
	44	2.082	192.168.3.1	255.255.255.255	DHCP	DHCP ACK - Transaction ID 0xde00 [255... subnet mask, no DNS or router]
ARP Check	47	2.088	Gainspan_01:00:6f	Broadcast	ARP	Who has 192.168.3.100? Tell 0.0.0.0
	48	2.092	Gainspan_01:00:6f	Broadcast	ARP	Who has 192.168.3.100? Tell 0.0.0.0
	49	2.093		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........
	50	2.094	Gainspan_01:00:6f	Broadcast	ARP	Who has 192.168.3.100? Tell 0.0.0.0
	55	2.383	Gainspan_01:00:6f	Broadcast	ARP	Who has 192.168.3.100? Tell 0.0.0.0
	63	3.203	Gainspan_01:00:6f	Broadcast	ARP	Gratuitous ARP for 192.168.3.100 (Request)
	64	3.305	Gainspan_01:00:6f	Broadcast	ARP	Gratuitous ARP for 192.168.3.100 (Request)
	137	11.599	Gainspan_01:00:6f	Broadcast	ARP	Gratuitous ARP for 192.168.3.100 (Request)
SNMP Agent	164	13.655	MitacInt_63:d0:09	Gainspan_01:00:6f	ARP	192.168.3.200 is at 00:40:d0:63:d0:09
ARP Check	232	21.953	Gainspan_01:00:6f	Broadcast	ARP	Gratuitous ARP for 192.168.3.100 (Request)
	233	21.953		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........
	235	22.044	Gainspan_01:00:6f	Broadcast	ARP	Gratuitous ARP for 192.168.3.100 (Request)
Buffered frames?	239	22.453	Gainspan_01:00:6f (E	D-Link_5a:45:c0 (BSSID)	IEEE 802.11	Power-Save poll, Flags=...P.... [PWR MGT flag, STA will go to sleep]
	240	22.456		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........
SNMP exchange	241	22.456	192.168.3.200	192.168.3.100	SNMP	set-request 1.3.6.1.4.1.28295.1.1.4.6.0 [GSCONFIGCOMPLETE, no docs] val=1
	242	22.456		D-Link_5a:45:c0 (RA)	IEEE 802.11	Acknowledgement, Flags=...P....
	243	22.462	192.168.3.100	192.168.3.200	SNMP	get-response 1.3.6.1.4.1.28295.1.1.4.6.0 val=1
	259	23.911	192.168.3.100	192.168.3.200	SNMP	trap iso.3.6.1.4.1.28295.1 1.3.6.1.4.1.28295.1.1.4.2.1 [GSNLINKUPINFO, linkup] val=00 1d c9 01 00 6f
	260	23.911		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........
Data	266	24.434	192.168.3.100	192.168.3.200	UDP	Source port: 51206 Destination port: 8255
					data:	0000 0a 53 74 03 ef c5 1d 14 98 ad 99 a2 a1 a2 75 b2 .St..........u.
						0010 67 49 8e 63 00 00 00 00 d3 gI.c.....
	267	24.434		Gainspan_01:00:6f (RA)	IEEE 802.11	Acknowledgement, Flags=........

Figure 2: Detailed interpretation of the packets exchanged between a GainSpan sensor node and AP when the node is powered up.

Packet sniffing using 802.11 promiscuous mode and *Wireshark* is easy to set up, and provides invaluable capabilities for analyzing and optimizing the operation of GainSpan-node-based sensor networks. We recommend that every developer and implementer include this hardware and software, or equivalents, in their toolkit.

Version	Date	Remarks
1.0	10 July 2008	Initial release.
2.0	4 October 2009	Update GainSpan HQ address.

SP-2.0