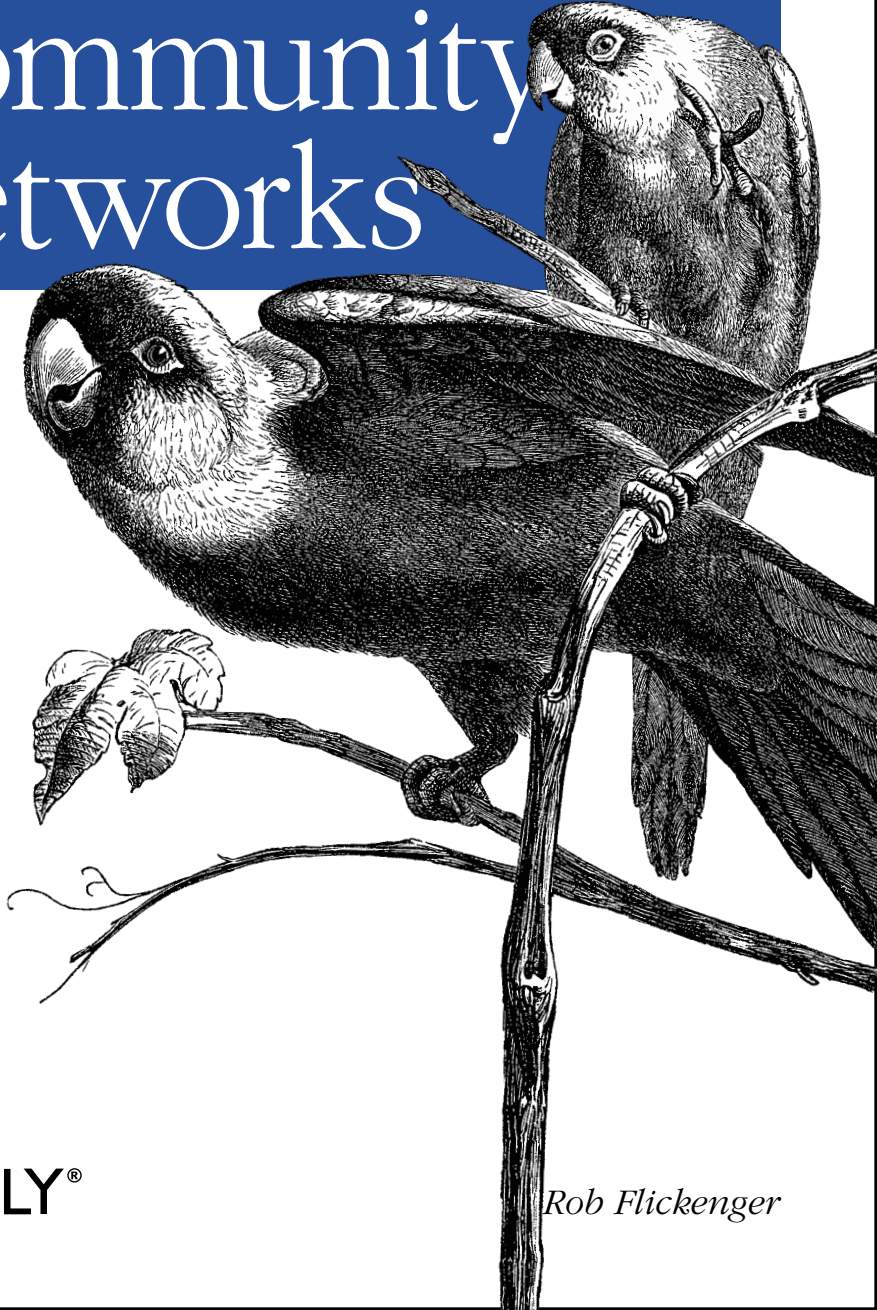


*Planning and Deploying
Local Wireless Networks*

2nd Edition
Expanded & Updated

Building

Wireless Community Networks



O'REILLY®

Rob Flickenger

Building Wireless Community Networks

SECOND EDITION

Building Wireless Community Networks

Rob Flickenger

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Network Layout

A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.

—Douglas Adams

As we saw in Chapter 2, there is an astounding variety of wireless networking equipment available on the consumer market today. While the champion technology of wireless community networks is still 802.11b, simply choosing equipment that is “Wi-Fi” compliant won’t necessarily guarantee a successful network project.

While equipment features, capabilities, and prices tend to change drastically in a short time, the essential network functions that they perform are still very straightforward. Let’s look at what your devices need to provide in order to fulfill your wireless networking dreams.

Layer 1 (Physical) Connectivity

Before any two components of your network can talk to each other, they must share a common physical medium through which they communicate. In the wired world, this is obvious; you would never try to connect a copper CAT5 cable to a piece of fiber and expect it all to “just work.”

In the wireless world, every device from your network to your cordless phone to your garage door opener must share the same physical medium: electromagnetic waves radiating through the air. It is possible for all of these devices to communicate without interfering with each other because they can be made sensitive to a particular portion of the vast electromagnetic spectrum. This is analogous to tuning channels on a radio or TV—many channels are broadcasting simultaneously, but they are well-coordinated and only use a portion of the available spectrum, to avoid interfering with each other.

So before any other considerations, devices that need to intercommunicate on your network must be able to send signals in the same frequency range. Obviously, an 802.11b card operating at 2.4GHz doesn't have a chance of carrying on a conversation with an 802.11a Access Point speaking at 5GHz. In addition to using a particular frequency range, each wireless protocol also defines a plan for using that range. For example, the original 802.11 specification defines two RF modulation schemes, FHSS and DSSS. Both operate at 2.4GHz, but use the spectrum differently. Frequency Hopping Spread Spectrum (FHSS) breaks the available spectrum into 77 channels, each 1MHz wide. It uses a time-based, pseudo-random algorithm to quickly skip between all of the available channels in an attempt to avoid noise from other 2.4GHz devices. As we saw in Chapter 2, Direct Sequence Spread Spectrum (DSSS) breaks the same frequency range into 11 overlapping channels, each 5MHz apart (but 22MHz wide). It uses one channel at a time and employs more sophisticated encoding techniques to avoid noise and increase the data rate. Although FHSS and DSSS devices both operate "at 2.4GHz," they have no hope of being able to communicate with each other.

Whatever wireless equipment you choose, be sure that both ends are capable of speaking the same protocol at the same frequency range, whether that's 802.11b speaking DSSS at 2.4GHz, 802.11a speaking OFDM at 5GHz, 802.11g speaking OFDM at 2.4GHz, or something altogether different, new, and wonderful. If two pieces of equipment claim compatibility with the same IEEE standard (such as 802.11b), they should theoretically be able to interoperate. Be sure to check the fine print on any device that only claims compatibility with an umbrella term (such as Wi-Fi), because the definition of the term can change at the whims of marketing moguls.*

As 802.11b is by far the most common technology used in the wireless community effort, we will focus on its particulars for the rest of this chapter.

Layer "1.5" Connectivity

Simply using equipment that adheres to the same standard on the same channel doesn't quite fulfill the requirements of Layer 1 (physical) connectivity. There are a few more protocol requirements that must be met before we can move on to Layer 2.

802.11b defines two possible (and mutually exclusive) radio modes that stations can use to intercommunicate. Those modes are *BSS* and *IBSS*.

* At the time of this writing, the term Wi-Fi can refer to either 802.11b or 802.11a gear, which are *not* interoperable. *Wi-Fi5* was supposed to refer to 5GHz gear, but evidently that didn't fly. For a good laugh, see <http://news.com.com/2100-1033-960880.html>.

BSS stands for Basic Service Set. In this operating mode, one station (the *BSS master*, usually a piece of hardware called an access point, or AP) provides wireless-to-Ethernet bridging. Before gaining access to the wired network, wireless clients (also called *BSS clients*) must first establish communications with an access point within range, as shown in Figure 3-1. Once the AP has authenticated the wireless client, it allows packets to flow between the client and the attached wired network, either routing traffic at Layer 3, or acting as a true Layer 2 bridge. A related term, Extended Service Set (ESS), refers to a physical subnet that contains more than one AP. In this sort of arrangement, the APs can communicate with each other to allow authenticated clients to “roam” between them, handing off IP information as the clients move about. Note that (as of this writing) there are no APs that allow roaming across networks separated by a router.*

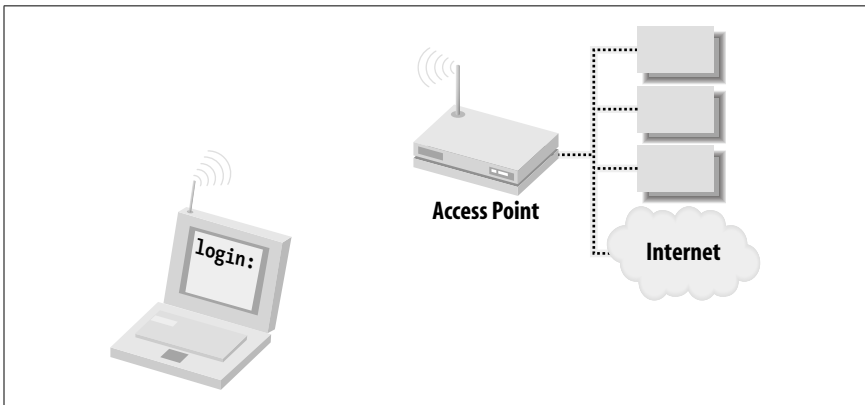


Figure 3-1. In BSS (or ESS) mode, clients must associate with an AP before accessing the wired network

IBSS stands for Independent Basic Service Set, and is frequently referred to as ad-hoc or peer-to-peer mode. In this mode, no hardware AP is required. Any network node that is within range of any other can communicate if both nodes agree on a few basic parameters. If one of those peers also has a wired connection to another network, it can provide access to that network. Figure 3-2 shows a model of an IBSS network.

Note that an 802.11b radio must be set to work in either of these modes, but cannot work in both simultaneously. Both modes support shared-key WEP encryption (more on that later).

* This is a difficult problem that experimental technologies such as Mobile IP attempt to solve.

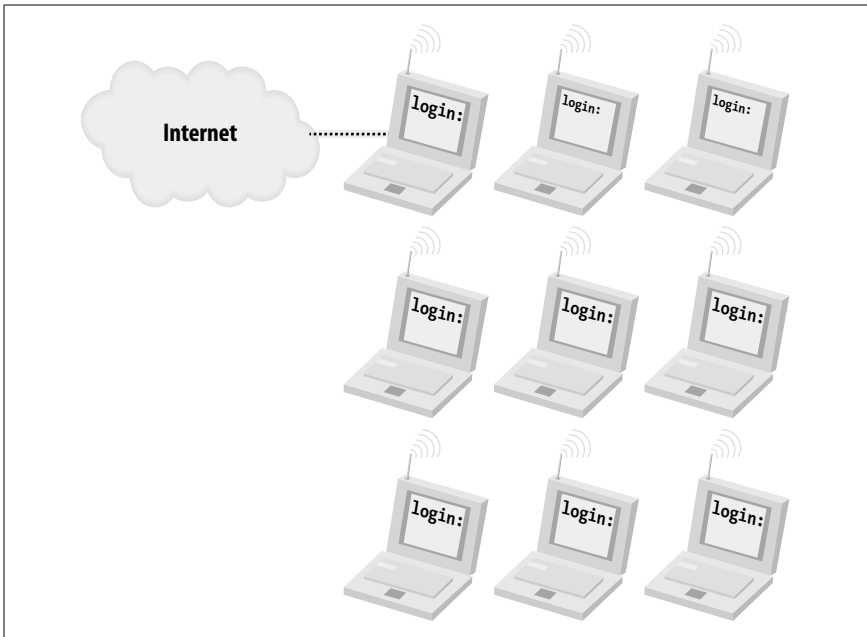


Figure 3-2. In IBSS mode, nodes can talk to any other node within range; a node with another network connection can provide gateway services

I give specific examples of how to set up BSS and IBSS networks in Chapters 4 and 5. Once you have two or more IBSS mode stations, or a BSS master and one or more BSS clients all within range, you are ready to move on to actual networking.

Layer 2 and Up

Once the physical layer is established, a wireless network is very much like a traditional Ethernet network. Assuming that you want to connect your wireless clients to the Internet, you'll want to provide all of the usual TCP/IP services that make networking so much fun (such as DNS and DHCP). To the rest of your network, wireless clients look like just another Ethernet interface, and are treated no differently than the wired printer down the hall. You can route, rewrite, tunnel, fold, spindle, and/or mutilate packets from your wireless clients just as you can with any other network device. Once wireless packets hit the wire, you use the same hubs, switches, and routers that make up the majority of traditional wired networks.

Wired Wireless

Presumably, no matter how many wireless clients you intend to support, you will eventually need to “hit the wire” in order to access other networks (such as the Internet). There are a number of different kinds of physical devices you can use to jump from wireless back to your wired infrastructure.

Access Point Hardware

APs are widely considered ideal for “campus” coverage. They provide a point of entry to the wired infrastructure that can be configured by a central authority. They typically allow for one or two radios per AP, theoretically supporting hundreds of simultaneous wireless users at a time. They must be configured with an ESSID (Extended Service Set ID, also known as the Network Name or WLAN Service Area ID, depending on who you talk to); it’s a simple string that identifies the wireless network. Many APs use a client program for configuration and a simple password to protect their network settings. All hardware access points provide BSS master services.

Most APs also provide a number of enhanced features. External antennas (or antenna connectors), advanced link status monitoring, and extensive logging and statistics are now common on many APs. In addition, most access points provide two additional security measures: MAC address filtering and closed networks. With MAC filtering enabled, a client radio attempting access must have its MAC address listed on an internal table before it can associate with the AP. In a closed network, the AP doesn’t beacon its ESSID at regular intervals. This means that each client must know the ESSID ahead of time, which makes it more difficult for people using programs such as *NetStumbler* to detect the network.

Other enhanced modes include dynamic WEP key management, public encryption key exchange, channel bonding, and other fun toys. Unfortunately, these extended modes are entirely manufacturer- (and model-) specific, are not covered by any established standard, and do not interoperate with other manufacturers’ equipment.

In addition to dedicated AP hardware, certain radio cards (in particular, those based on the Prism 2 chipset) can be made to operate as a BSS master and act as if it were a regular AP. In Chapter 5, I will show you how to “roll your own” AP using the Host AP driver for Linux.

APs are by far the most widely used devices for providing wireless services, particularly in corporate networks. They provide a high degree of control over who can access the wire, but they are not cheap (the average AP at the time of this writing costs between \$500 and \$1000).

Another class of AP is occasionally referred to as a *residential gateway* (RG). The Apple Airport, Orinoco RG series, and Linksys WAP11 are popular examples of RGs. They are typically much less expensive than their “commercial” counterparts, costing between \$100 and \$300. Many have built-in modems, allowing for wireless-to-dialup access (which can be very handy, if Ethernet access isn’t available). Most even provide Network Address Translation (NAT), DHCP, and bridging services for wireless clients. While they may not support as many simultaneous clients as a high-end AP, they can provide cheap, simple access for many applications. When configuring an inexpensive AP for bridged Ethernet mode, you can still have a high degree of control over what individual clients can access on the wired network by controlling communications at a higher level. See the “Captive Portal” discussion in Chapter 7 for more details.

Note that APs (that is, BSS masters) do *not* talk to each other over the air. In order to have 802.11b BSS mode communications, one device (e.g., an access point) must be a master, and the other must be a client.

BSS Client Hardware

While the typical BSS client is a PCMCIA or other plug-in radio card, there are also other hardware devices that will serve as a BSS client that connect directly to Ethernet. The Linksys WET11, 3Com Wireless Workgroup Bridge, and Orinoco Ethernet Converter are examples of this type of hardware. Some RGs (such as the Linksys WAP11) can even be made to operate as a BSS client. The typical *wireless client bridge* is a small box that provides one or more Ethernet ports and bridges them (at Layer 2) directly to a wireless network. The radio is configured via Ethernet (or a USB port) to act as a client to an existing wireless network. After initial configuration, no further interaction with the bridge is necessary. As far as the wired device is concerned, it is directly attached to an Ethernet network and requires no special drivers or other preparation to use the wireless network.

These devices are very handy in some circumstances, especially when you would like to get an Ethernet-equipped device onto the wireless network, but can’t install a wireless card. One typical use is to connect an Ethernet printer to a wireless network, so you can install it somewhere that doesn’t have CAT5 available. Another popular use for the WET11 is to bridge a console game (such as Sony PlayStation 2) to your wireless network, thereby avoiding the need to run CAT5 to your television. They are also handy for connecting remote access points back to a central wireless infrastructure. I’ll provide an example of how to do that in Chapter 7.

The two big drawbacks to most BSS client hardware are price and performance. Since they aren't as popular as client cards, they are typically a bit more expensive. They also tend to offer poor performance compared to client cards (2 to 4Mbps throughput is typical, compared to 5 to 6Mbps with client cards). Despite these issues, Ethernet bridges are an ideal solution to some networking problems.

Peer-to-Peer (IBSS) Networking

Radios that are operating in IBSS mode can communicate with each other without a hardware access point if they have the same ESSID and WEP settings. This is particularly handy for setting up temporary wireless workgroups without an AP, or for building point-to-point wireless connections. As stated earlier, any computer with an 802.11b card and another network connection (usually Ethernet, dialup, or even another wireless connection) can serve as a gateway between the two networks.

There is one important constraint on using IBSS mode: although it is defined by the 802.11b standard, few client cards actually interoperate well in the real world with others using IBSS. While two radios of the same manufacturer (and of the same firmware revision) generally work just fine, trying to get a Cisco card to talk to a Proxim card in IBSS mode (for example) is usually futile.

With this in mind, why would you choose to use IBSS mode rather than use an AP or the Host AP driver? There are a couple of reasons. If you happen to have two cards of the same manufacturer and a couple of old computers, IBSS mode is ideal if you want to create a fixed point-to-point connection. Also, Host AP supports only a limited set of wireless cards—if you already own a card that isn't supported, you're out of luck. Finally, if you're using a laptop and need to exchange data with another wireless user, IBSS is your only option if you're out of range of an AP and can't run Host AP.

In Chapter 5, I'll build a Linux-based wireless gateway from scratch, using both IBSS mode and the Host AP driver. In Chapter 7, I'll examine one method of extending the gateway to provide different classes of service, depending on who connects to it.

Vital Services

A network can be as simple as a PPP dialup to an ISP, or as grandiose and baroque as a multinational corporate MegaNet. But every node on a multi-million dollar network in Silicon Valley needs to address the same fundamental questions that a dialup computer must answer: *who am I, where am I*

going, and how do I get there from here? In order for wireless clients to easily access a network, the following basic services must be provided.

DHCP

The days of static IP addresses and user-specified network parameters are thankfully far behind us. Using DHCP (Dynamic Host Configuration Protocol),* it is possible (and even trivial) to set up a server that responds to client requests for network information. Typically, a DHCP server provides all of the information that a client needs to begin routing packets on the network, including the client's own IP address, the default Internet gateway, and the IP addresses of the local DNS servers. The client configuration is ridiculously easy and is, in fact, configured out of the box for DHCP in all modern operating systems.

While a thorough dissection of DHCP is beyond the scope of this book, a typical DHCP session goes something like this: a client boots up, knowing nothing about the network it is attached to except its own hardware MAC address. It broadcasts a packet saying effectively, "I am here, and this is my MAC address. What is my IP address?" A DHCP server on the same network segment is listening for these requests, and responds with "Hello MAC address, here is your IP address, and by the way here is the IP address to route outgoing packets to, and some DNS servers are over there. Come back in a little while and I'll give you more information." The client, now armed with a little bit of knowledge, goes about its merry way. Figure 3-3 shows how this conversation takes place.

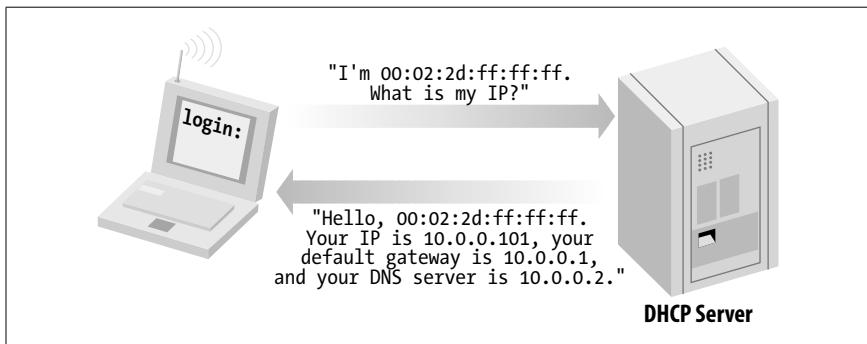


Figure 3-3. DHCP lets a node get its network settings dynamically and easily

* See <http://rfc.net/rfc1533.html> for an overview.

In a wireless environment, DHCP is an absolute necessity. There isn't much point in being able to wander around without a cable if you need to manually set the network parameters for whatever network you happen to be in range of. It's much more convenient to let the computers work it out on their own (and let you get back to more important things, such as IRC or "Quake III Arena"). Since DHCP lets a node discover information about its network, it's easy to get "online" without any prior knowledge about that particular network's layout. This service demonstrates a condition that network administrators have known for years: users just want to get online without knowing (or even caring) about the underlying network. From their perspective, it should just work. DHCP makes this kind of magic possible.

From a network administrator's perspective, the magic isn't terribly difficult to bring about. As long as you have exactly one DHCP server running on your network segment, your clients can all pull from a pool of available IP addresses. The DHCP server will manage the pool on its own, reclaiming addresses that are no longer in use and reassigning them to new clients.

In many cases, a wired network's existing DHCP server serves wireless users with no trouble. It will see the wireless node's DHCP request just as it would any other and responds accordingly. If your wired network isn't already providing DHCP, or if your access point isn't capable of Layer 2 bridging, then the access point itself will likely provide DHCP. I'll cover setting up DHCP services on a homebrew wireless gateway in more detail in Chapter 5.

DNS

My, how different the online world would be if we talked about sending mail to *rob@208.201.239.36*, or got excited about having just been *64.28.67.150*'d. DNS is the dynamic telephone directory of the Internet, mapping human friendly names (such as *oreillynet.com* or *slashdot.org*) to computer friendly numbers (such as the dotted quads mentioned previously). The Internet without DNS is about as much fun and convenient as referring to people by their Social Security Numbers.

Much like DHCP, your network's existing DNS servers should be more than adequate to provide name resolution services to your wireless clients. However, depending on your particular wireless application, you may want to get creative with providing additional DNS services. A caching DNS server might be appropriate, to reduce the load on your primary DNS servers (especially if you have a large number of wireless clients). You might even want to run dynamic DNS for your wireless hosts, so that wireless nodes can easily provide services for each other.

One handy use for DNS is to provide local top-level domains (TLDs) that don't normally resolve on the Internet, but direct people to local services. For example, the ad-hoc TLD of the NoCat network in Sebastopol is *.cat*, and the TLD for SeattleWireless is *.swn*. This allows for nifty names such as *gateway.rob.cat* or *music.nodeone.swn*. These addresses are not reachable by the Internet, but will resolve for anyone connected to the wireless network. I'll look at how various community network groups are extending TLD name service (and even connecting their networking projects via Internet tunnels) in Chapter 7.

NAT

In order for any machine to be reachable via the Internet, it must be possible to route traffic to it. A central authority, the IANA (Internet Assigned Numbers Authority, <http://www.iana.org>), holds the keys to the Internet. This international body controls how IP addresses are partitioned out to the various parts of the world, in an effort to keep every part of the Internet (theoretically) reachable from every other and to prevent the accidental reuse of IP addresses in different parts of the world. Unfortunately, due to the unexpected popularity of the Net, what was thought to be plenty of address space at design time has proven to be woefully inadequate in the real world. With thousands of new users coming online for the first time every day (and some large corporate users simply refusing to give up huge chunks of unused address space), the general consensus is that there simply aren't enough IP addresses to go around anymore. Most ISPs are increasingly paranoid about the shortage of homesteading space, and are loath to give out more than one per customer (and in many cases, they won't even do that anymore, thanks to the wonders of DHCP).

Now we see the inevitable problem: suppose you have a single IP address allocated to you by your ISP, but you want to allow Internet access to a bunch of machines, including your wireless nodes. You certainly don't want to pay exorbitant fees for more address space just to let your nephew get online when he brings his wireless laptop over once a month.

This is where NAT can help you. Truly a mixed blessing, NAT (referred to in some circles as "masquerading") provides a two-way forwarding service between the Internet and another network of computers. A computer providing NAT typically has two network interfaces. One interface is connected to the Internet (where it uses a real live IP address), and the other is attached to an internal network. Machines on the internal network use any of IANA's reserved IP addresses and route all of their outgoing traffic through the NAT box. When the NAT box receives a packet bound for the

Internet, it makes a note of where the packet came from. It then rewrites the packet using its “real” IP address, and sends the modified packet out to your ISP (where it winds its way through the rest of the Internet, hopefully arriving at the requested destination). When the response (if any) comes back, the NAT box looks up who made the original request, rewrites the inbound packet, and returns it to the original sender. As far as the rest of the Net is concerned, only the NAT machine is visible. And as far as the internal clients can tell, they’re directly connected to the Internet. Figure 3-4 shows a NAT configuration.

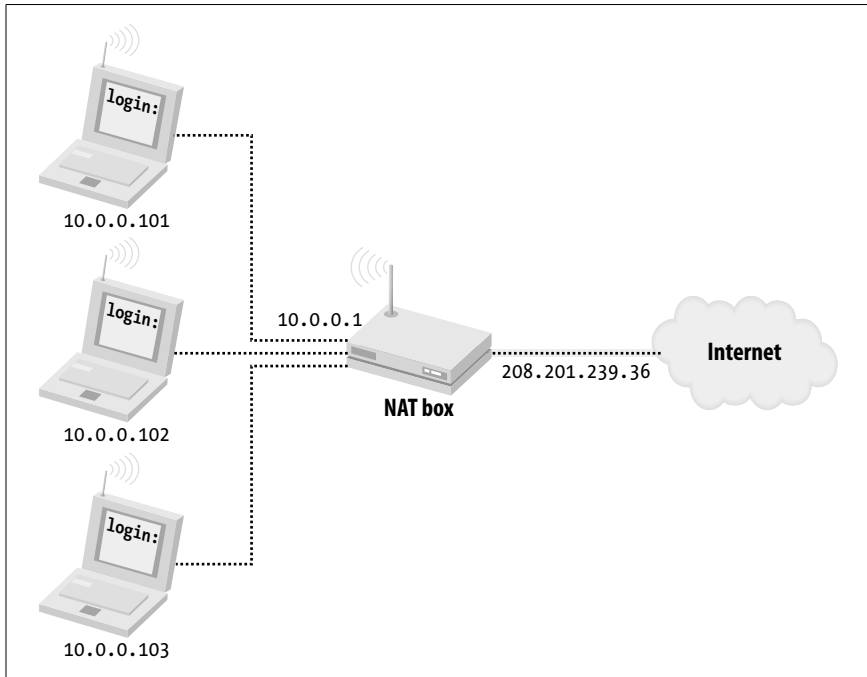


Figure 3-4. Using NAT, several computers can share a single “real” IP address

The IANA has reserved the following sets of IP addresses for private use (as outlined in RFC 1918, <http://rfc.net/rfc1918.html>):

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.0 to 192.168.255.255

These addresses will never be used on the Internet. As long as your internal machines use IP addresses in any of these three ranges, your traffic will not interfere with any other host on the Net.

NAT is handy, but isn't without its drawbacks. For example, some services may not work properly with some implementations of NAT. Most notably, active FTP sessions and some online games fail when running behind many NAT boxes. Another big disadvantage to NAT is that it effectively makes the Internet a read-only medium, much like television. If you can have only out-bound traffic (to web servers, for example) but traffic from the Internet can't reach your machine directly, then you have no way of serving data and contributing back to the Net! This doesn't prevent you from using two-way services such as IRC and email, but it does preclude you from easily running on-demand services where Internet users connect to you directly (for example, running your own web server from behind a NAT isn't trivial, unless you're the one who controls the NAT).

Despite these drawbacks, NAT is an invaluable tool for allowing throngs of people to access Internet resources. In Chapter 5, I'll build a Linux gateway that will do NAT for you and handle almost every popular form of Internet traffic you care to throw at it (including active FTP). In Chapter 7, I will extend it even further, to try to get around some of the potentially antisocial aspects of NAT.

Of course, if you're lucky enough to have a ton of live IP address space, feel free to flaunt it and assign live IPs to your wireless clients! Naturally, most people (and, indeed, their laptops) are unprepared for the unbridled adrenal rush of using a live IP address without a firewall. But if you have that many real IPs to throw around, you must be used to living large. Just don't worry when you find your clients spontaneously rebooting or suddenly serving 0-dAy W@r3z. It's all part of the beautiful online experience.

Security Considerations

Although the differences between tethered and untethered are few, they are significant. For example, everyone has heard of the archetypal "black-hat packet sniffer," a giggling sociopath sitting on your physical Ethernet segment, surreptitiously logging packets for his own nefarious ends. This could be a disgruntled worker, a consultant with a bad attitude, or even (in one legendary case) a competitor with a laptop, time on his hands, and a lot of nerve.* Although switched networks, a reasonable working environment, and conscientious reception staff can go a long way to minimize exposure to the physical wiretapper, the stakes are raised with wireless. Suddenly, one

* As the story goes, a major computer hardware manufacturer once found a new "employee" sitting in a previously unoccupied cube. He had evidently been there for three weeks, plugged into the corporate network and happily logging data before HR got around to asking who he was.

no longer needs physical presence to log data: why bother trying to smuggle equipment onsite when you can crack from your own home or office two blocks away with a high-gain antenna?

Visions of cigarette smoking, pale-skinned über-crackers in darkened rooms aside, there is a point that many admins tend to overlook when designing networks: the whole reason that the network exists is to connect people to each other! Services that are difficult for people to use will simply go unused. You may very well have the most cryptographically sound method on the planet for authenticating a user to the system. You may even have the latest in biometric identification, full winnow and chaff capability, and independently verified and digitally signed content assurance for every individual packet. But if the average user can't simply check their email, it's all for naught. If the road to hell is paved with good intentions, the customs check-point must certainly be run by the Overzealous Security Consultant.

The two primary concerns when dealing with wireless clients are these:

- Who is allowed to access network services?
- What services can authorized users access?

As it turns out, with a little planning, these problems can be addressed (or neatly sidestepped) in most real-world cases. In this section, we'll look at some tools that can help keep your data flowing to where it belongs, as quickly and efficiently as possible.

WEP

The 802.11b specification outlines a form of encryption called wired equivalent privacy, or WEP. By encrypting packets at the MAC layer, only clients who know the "secret key" can associate with an AP or peer-to-peer group. Anyone without the key may be able to see network traffic, but every packet is encrypted.

The specification employs a 40-bit shared-key RC4 PRNG* algorithm from RSA Data Security. Most cards that use 802.11b (Proxim Orinoco, Cisco Aironet, Apple Airport, and Linksys WPC11, to name a few) support this encryption standard.

Although hardware encryption sounds like a good idea, the implementation in 802.11b is far from perfect. First of all, the encryption happens at the link layer, not at the application layer. This means your communications are protected up to the gateway, but no further. Once it hits the wire, your packets

* Pseudo-Random Number Generator. It could be worse, but entropy takes time.

are sent in the clear. Worse than that, every other legitimate wireless client who has the key can read your packets with impunity, since the key is shared across all clients. You can try it for yourself; simply run *tcpdump* on your laptop and watch your neighbor's packets just fly by, even with WEP enabled.

Many manufacturers have implemented their own proprietary extensions to WEP, including 104-bit keys and dynamic key management. Unfortunately, because they are not defined by the 802.11b standard, there is no guarantee that cards from different manufacturers that use these extensions will interoperate.

40 vs. 64 vs. 104 vs. 128-bit WEP

Why are so many different key lengths quoted by various card manufacturers? The original 802.11b spec defined a 40-bit user-specified key. This key is combined with a 24-bit initialization vector (the IV), a random number that is part of the WEP algorithm. Together, this yields 64 bits of "key," although the IV is actually sent in the clear!

Likewise, 104-bit WEP is used with the IV to yield 128 bits of "key." This is why user-defined keys are 5 characters long ($5 \times 8 = 40$) or 13 characters long ($13 \times 8 = 104$). The user doesn't define the IV; it is part of the WEP algorithm (and is generally implemented as 24 random bits).

More bits sounds more secure to the consumer, so some manufacturers choose to list the larger number as the "key length." Unfortunately, for WEP, having more bits does not guarantee significantly greater security. Read on.

To throw more kerosene on the burning WEP tire mound, a team of cryptographers at the University of California at Berkeley and other experts have identified weaknesses in the way WEP is implemented, and effectively these vulnerabilities have made the strength of encryption irrelevant. With all of its problems, why is WEP still supported by manufacturers? And what good is it for building public-access networks?

WEP was not designed to be the ultimate "killer" security tool (nor can anything seriously claim to be). Its acronym makes the intention clear: wired equivalent protection. In other words, the aim behind WEP was to provide no greater protection than you would have when you physically plug into your Ethernet network. (Keep in mind that in a wired Ethernet setting, there is no encryption provided by the protocol at all. That is what application layer security is for; see the tunneling discussion later in this chapter.)

What WEP does provide is an easy, generally effective, interoperable deterrent to unauthorized access. While it is technically feasible for a determined intruder to gain access, it is not only beyond the ability of most users, but usually not worth the time and effort, particularly if you are already giving away public network access!

As you'll see in Chapter 7, one area where WEP is particularly useful is at either end of a long point-to-point backbone link. In this application, unwanted clients could potentially degrade network performance for a large group of people, and WEP can not only help discourage would-be link thieves, but encourage them to set up more public-access gateways.

802.1x

802.1x is a fairly new IEEE specification. The full title of 802.1x is “802.1x: Port Based Network Access Control.” Interestingly enough, 802.1x wasn't originally designed for use in wireless networks; it is a generic solution to the problem of port security. Imagine a college campus with thousands of CAT5 jacks scattered throughout libraries, classrooms, and computer labs. At any time, someone could bring their laptop on campus, sit down at an unoccupied jack, plug in, and instantly gain unlimited access to the campus network. If network abuse by the general public were common, it might be desirable to enforce a policy of port access control that permitted only students and faculty to use the network.

This is where 802.1x fits in. Before any network access (to Layer 2 or above) is permitted, the client (the *supplicant*, in 802.1x parlance) must authenticate itself. When first connected, the supplicant can exchange data only with a component called the *authenticator*. This in turn checks credentials with a central data source (the *authentication server*), typically a RADIUS server or other existing user database. If all goes well, the authenticator notifies the supplicant that access is granted (along with other optional data) and the client can go about its merry way. The various encryption methods employed are not defined, but an extensible framework for encryption (*EAP*, or *Extensible Authentication Protocol*) is provided.

802.1x has been widely regarded by the popular press as the fix for the problems of authentication in wireless networks. For example, the optional data that is sent back to the supplicant might contain WEP keys that are dynamically assigned per session. These keys could be automatically renewed every so often, making most data collection attacks against WEP futile. Unfortunately, 802.1x is susceptible to certain session hijacking methods, denial-of-service attempts, and man-in-the-middle attacks when used with wireless

networks, making the use of 802.1x as the ultimate security tool a questionable proposition.

As of this writing, 802.1x drivers are available for Windows XP and 2000 and many access points (notably Cisco and Proxim) support some flavor of 802.1x. There is also an open source 802.1x implementation project available at <http://www.open1x.org/>.

How relevant is 802.1x to community wireless projects? It definitely depends on your goals. The vast majority of community projects incorporate open access points, with no authentication or encryption enabled. 802.1x could help provide a degree of security to a private wireless network (probably even better than WEP alone), although it shouldn't be considered a magic bullet. Combining 802.1x with a strong encrypted tunnel or VPN (see the upcoming section) will likely keep out the most tenacious of system crackers, but will make participation in your network much more difficult for casual users.

For a good discussion of 802.1x security methods and problems online, take a look at <http://www.sans.org/rr/wireless/80211.php>. Researchers at the University of Maryland have also published a paper on 802.11 security; it's available at <http://www.cs.umd.edu/~waa/wireless.html>.

Routing and Firewalls

The primary security consideration for wireless network access is where to fit it into your existing network. You need to consider what services you want your wireless users to be able to access, both on the Internet and on your internal network. Since the primary goal of this book is to describe methods for providing public access to network services (including access to the Internet), I strongly recommend setting up your wireless gateways in the same place you would any public resource: in your network's DMZ, or outside of your firewall altogether (as in Figure 3-5). This will give you the most flexibility in defining an internal security policy. Even in the absolute worst case of a complete breakdown of security precautions, the most that any social deviant will end up with is Internet access, and not unrestricted access to your private internal network.

This configuration leaves virtually no incentive for anyone to bother trying to compromise your gateway, as the only thing to be gained would be greater Internet access. Attacks coming from the wireless interface can easily log MAC address and signal strength information. With wireless, this can be a fair deterrent: because the would-be attacker needs to transmit to carry out an attack, they necessarily give away not only a unique identifier (their MAC address), but also their physical location!

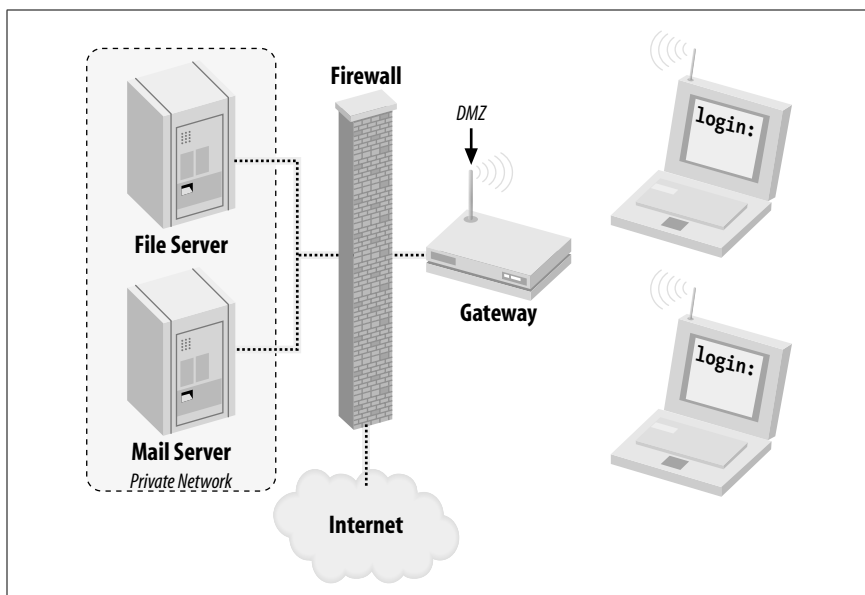


Figure 3-5. Place your wireless gateways outside of your private network!

Assuming that all wireless connectivity takes place outside of your private network, what happens when you or your friends want to connect from the wireless back into the inside? Won't other wireless users be able to just monitor your traffic and grab passwords and other sensitive information? Not if you use strong application-layer encryption.

Encrypted Tunnels

Application layer encryption is a critical technology when dealing with untrusted networks (such as public-access wireless links, for example). This is obvious when looking at a network diagram, as in Figure 3-6. When using an encrypted tunnel, you can secure your communications from eavesdroppers all the way to the other end of the tunnel.

If you're using a tunnel from your laptop to another server, would-be black hats listening to your conversation will have the insurmountable task of cracking strong cryptography. Until someone finds a cheap way to build a quantum computer (and perhaps a cold fusion cell to power it), this activity is generally considered a waste of time. In the previous example, a web server providing 128-bit SSL connections provides plenty of protection, all the way to your wireless laptop. SSL provides application layer encryption.

SSL is great for securing web traffic, but what about other network services? Take this typical scenario: you're at work or at home, merrily typing away

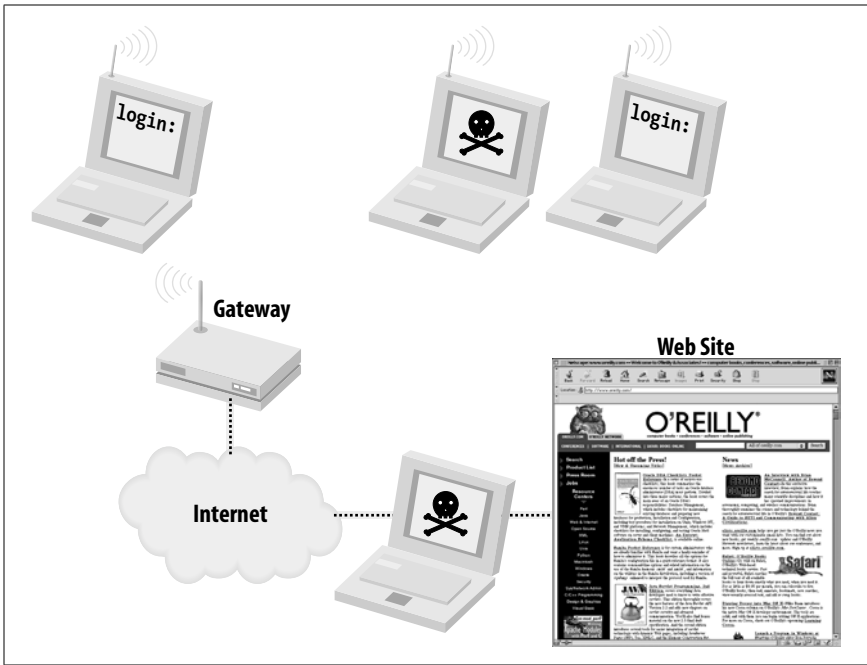


Figure 3-6. WEP encrypts only to the gateway, exposing your traffic to other wireless users and anything after the wire; tunnels protect your traffic from end to end

on your wireless laptop. You want to retrieve your email from a mail server with a POP client (Netscape Mail, Eudora, fetchmail, etc.). If you connect to the machine directly, your email client sends your login and password “in the clear.” This means that a nefarious individual somewhere between you and your mail server (either elsewhere on your wireless network, or even “on the wire” if you are separated by another network) could be listening and could grab a copy of your information en route. This login could then be used not only to gain unauthorized access to your email, but in many cases will also grant a shell account on your mail server!

To prevent this, you can use the tunneling capabilities of SSH. An SSH tunnel works like this: rather than connecting to the mail server directly, we first establish an SSH connection to the internal network that the mail server lives in (in this case, the wireless gateway). Your SSH client software sets up a port-forwarding mechanism, so that traffic that goes to your laptop’s POP port magically gets forwarded over the encrypted tunnel and ends up at the mail server’s POP port. You then point your email client to your local POP port, and it thinks it is talking to the remote end (only this time, the entire session is encrypted). Figure 3-7 shows an SSH tunnel in a wireless network.

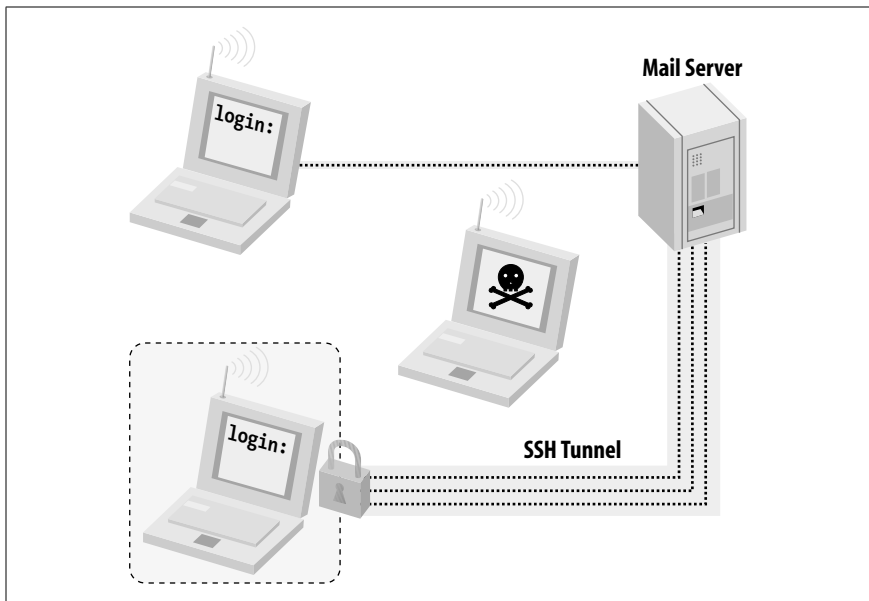


Figure 3-7. With an SSH tunnel in place, your otherwise-insecure conversation stays private

With the tunnel in place, anyone who tries to monitor the conversation between your laptop and the mail server will get something resembling line noise. It's a good idea to get in the habit of tunneling anything that you want to keep private, even over wired networks. SSH tunneling doesn't have to stop at POP connections either. Any TCP port (SMTP, for example) can easily be set up to tunnel to another machine running SSH, almost anywhere on the Internet. We'll see an example of how to do that in Chapter 7. For a full discussion of the ins and outs of this very flexible (and freely available) tool, I highly recommend *SSH, The Secure Shell: The Definitive Guide* (O'Reilly).

VPN

Using a virtual private network (VPN) is another popular method for dealing with wireless security shortcomings. Most VPN software uses powerful encryption and strong authentication to protect not only traffic to an individual port, but to all network traffic in general. If a wireless client is using good VPN software, all traffic from it can be well-protected, regardless of the security shortcomings of the underlying network. As with encrypted tunnels, sniffing the wireless traffic of a client associated with a public access point is possible, but will yield only strongly encrypted packets. While the tunnel server's IP address and amount of traffic being sent is still revealed,

the actual data and ultimate destination of the user's traffic is still well-protected. Likewise, authentication credentials to otherwise unprotected services (such as unencrypted web and email passwords) are also protected.

Examples of popular VPN software include PPTP, *vtun*, IPSec tunnels, and even PPP over SSH. I highly recommend running strong VPN software as the only gateway back into your internal network, as this greatly simplifies access and sidesteps many security issues. Unfortunately, setting up VPN software is beyond the scope of this book, but there are many resources available online to assist you with this task. In general, VPN software is network agnostic, and will usually work with your wireless network without any additional configuration.

Other Potential Threats

If you are paranoid about security (as well you should be), there are a number of additional issues to consider when running open access points. The rules change considerably when people who have access to your physical infrastructure can't be trusted, particularly when Layer 2 network traffic can be easily and anonymously molested. I will discuss wireless security in more detail in Chapter 7, and offer some examples of common attacks, as well as describe the tools you can use to defend against them. For even more details, consult the excellent discussion of potential problems (and solutions) in *802.11 Security* (O'Reilly).

Summary

In order to maintain maximum compatibility with available 802.11b client hardware and yet still provide responsible access to the Internet, you can apply a combination of inexpensive hardware and freely available software to strike an acceptable balance between access and security.

In the following chapters, I'll show you how to set up basic wireless access to augment your existing wired network. I'll describe a workable method for providing wireless services to your local community, for minimal cost, while promoting community participation and individual responsibility.