Wireless Protocol Stacks

ESC-325
Larry Mittag

Introduction

It used to be the case that the ability to communicate wirelessly was a unique specialty among computing systems. These days it is rapidly becoming an expected feature, something that is assumed. An engineer is more likely to be called upon to explain the reason for a lack of wireless communications rather than to justify adding it to a system under design.

This situation has been driven by the ubiquitous acceptance of wireless communications among laptop and handheld computers, as well as the massive presence of cellular communications. This works quite nicely if you happen to be working in the PC world or in a specialized area such as cellular handset design, but what if you are in the business of making specialized embedded systems doing anything else?

This paper concentrates on the details of wireless protocol stacks that are useful to engineers that need to dig deeper into the technology than the typical application programmer. The goal is to allow you to use these stacks to their fullest potential in the specialized systems you are designing.

Unique qualities of wireless stacks

In a sense, the thought that we would need to tweak wireless protocol stacks defeats the whole idea of a protocol stack in the first place. Isn't the concept that the lower layers of a protocol stack can be treated like a black box, something that just works and can be safely ignored?

There are two rebuttals to that point of view. The first one is that it is there and therefore it must be understood. This is the 'hacker' argument, using the original concept of hacking rather than the criminal sense that is finally beginning to fall out of favor with the press. This is the engineering mindset that drives children of all ages to disassemble their toys to see what makes them tick. It is only when all mysteries have been solved that we can truly be comfortable with any new technology, especially one as fascinating as wireless communications.

The second rebuttal is more practical. If you are building communicating systems that must interoperate with systems from other sources it is required that you use standards to do that communicating. If your system is a general-purpose computer like a laptop it is quite likely that the generic tuning that is used for these standards will be quite sufficient for your needs.

But what if you have control of both sides of the communications? In this case it still may very well make sense to use a standard for communications, but it may be undesirable to use that standard as it comes off-the-shelf. In this case it is quite possible to tune the standard to be optimized for your particular application.

This type of standards modification has a long history in embedded systems. Serial ports have been built that used TTL signal levels rather than the 12v signaling required by the RS-232 standard because the application did not need the line distance afforded by the higher voltages. TCP/IP protocol stacks have been gutted because all that was needed was raw Ethernet connectivity.

But what are the unique qualities of wireless communication that must be conserved, and what can be stripped out in the name of dedicated-system efficiency? The unique characteristics of wireless protocol stacks that must be understood boil down to a few ideas.

Data formatting for unreliable communications

Wireless communications is unreliable. This is not a value judgment, it is instead a simple fact of life. A wired interface can send digital data in a very simple mapping between signals and the data itself. This voltage is a zero, that voltage is a one. Wireless communications must deal with the fact that data sent is quite likely to be munged in the process. As a result, wireless systems will add redundancy in the data to allow error recovery. It is quite common for them to use specialized encoding to increase the Hamming distance between symbols (i.e. data values) within a data packet.

Resistance to interference

Another common feature in wireless data is the use of techniques that resist interference. The most common of these is some form of spectrum spreading. The concept is quite simple. Send data on a wide swath of spectrum. If one part of the spectrum is blocked, it will still get through on another part.

The simplest concept of interference is two signals broadcasting on the same frequency, but it turns out that this is just one of the possibilities for interference. There is a concept referred to as multipath that is often much more difficult to deal with. This is literally a signal interfering with itself. It is caused by a signal traveling many different ways to get from the source to the receiver. Each of these paths has a different length, so the same message gets received multiple times. This single problem is probably the biggest one to overcome in any wireless communications system.

Adaptation

They say that the one constant is change. Nowhere is this truer than in a wireless communication system. The parameters of signal reception can change completely in a fraction of a second as signal paths change and interference sources appear and disappear. Wireless protocol stacks must be able to adapt to these changes in real time and smooth the apparent communication path that is seen by applications software that is using that stack.

Solving these unique challenges has been what has allowed wireless data communications to break into the real world over the last few years. The following sections will examine some of the more popular protocol stacks that are currently being

used and start setting the groundwork to allow us to modify these stacks for specific applications.

802.11 family of specifications

Standard details

On the surface of it, the 802.11 family of specifications will be very familiar to anyone who has used TCP/IP. This familiarity is quite intentional, and is in fact one of the major reasons for the raging popularity of these standards over the last few years. It is quite possible to lift up the TCP/IP stack of practically any device and slide the 802.11 protocol stack underneath it quite successfully. Applications utilizing the stack will quite literally be unaware that they have just been transformed into wireless applications.

This feat is due in no small part to the work that was done in the original 802.11 specification, the one that defined the layer architecture of the rest of the standards. This layered architecture is as shown in Figure 0-1 below.

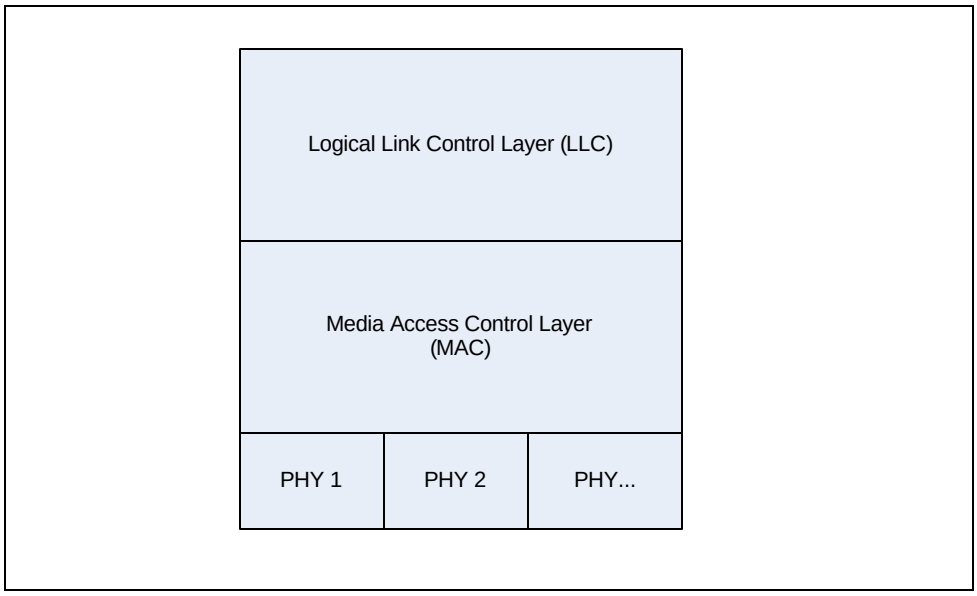

Figure 0-1 802.11 Protocol Stack

The LLC layer of this stack is designed to provide the functionality of the High-Level Data Link Control protocol of the ISO standard TCP/IP protocol stack. This allows an 802.11 protocol stack to be grafted rather painlessly underneath just about any TCP/IP implementation with little or no change to the upper layers.

One item to note is the number of PHY (Physical) layers that are listed underneath the MAC layer. The original 802.11 standard defined three specific physical network protocols. These were an infrared communications standard, a Direct-Sequence Spread Spectrum (DSSS) RF protocol, and a Frequency-Hopping Spread Spectrum (FHSS) RF protocol. As it turned out, the infrared protocol was basically ignored and the two RF protocols were implemented wi th very little customer education as to what they were or the fact that they did not interoperate. This was the first wave of 802.11 products, and it generally did not do well.

The next generation of the 802.11 standards was 802.11b. This is often thought of as a completely new standard, but in reality it utilized the LLC and MAC definitions that it inherited

from the original 802.11 and simply redefined the PHY layer. This version eliminated the IR and FHSS layers completely and defined a new DSSS standard based on a High-Rate (HR) version of the DSSS modulation techniques defined in 802.11. This simplification and the 11 Mbps raw data rate of this standard became an instant hit in the marketplace and is still the dominant standard today.

Similarly, the 802.11a and 802.11g standards are also redefinitions of the PHY layer of this protocol stack. These standards build upon the solid base that was defined in the original 802.11 standard and allow applications to be blissfully unaware of the nasty RF details that are handled at the lower layers. 802.11g operates within the same 2.4 GHz spectrum used by 802.11b and is in fact downwardly compatible with this standard, while 802.11a instead defines a new set of modulation capabilities within the 5 GHz spectrum. These standards are summarized in Table 1 below.

Standard	Maximum Data Rate	Frequency	Compatibility
802.11	2 Mbps	2.4 GHz	-
802.11b	11 Mbps	2.4 GHz	802.11
802.11g	54 Mbps	2.4 GHz	802.11b
802.11a	54 Mbps	5 GHz	-

Table 1 – 802.11 Standards

The documents describing all of these 802.11 standards are available online and at no cost from www.ieee.org. There are many other standards describing proposed enhancements to areas such as security, Quality of Service (QoS), and power management also available from the same source, although the more recent of these do require payment to receive. Check out that website and many others for the most recent information.

Tweaking opportunities

Now we get to the fun part. On the face of it there appears to be very little that can be done with the 802.11 stack. When you dig a little deeper, though, you find any number of possibilities for optimization. These opportunities come in a number of areas.

Security

One of the few problems that have shown up to slow the distribution of wireless LANs has been security – or, rather, the lack thereof. It is quite true that the original WEP standard for encryption was inadequate. This is not tremendously surprising, since it was defined as something of a stopgap measure until a better security model could be designed and approved. This situation was exacerbated by some very poor implementations of the security in some very high-profile 802.11b products. The result was quite a bit of extremely poor press regarding the security of these networks.

As it turns out, it really is not that hard to make wireless networks much more secure. The list of steps is relatively straightforward.

Don't broadcast your SSID – The default setup for a base station is to be very friendly and tell people their name. if you want to be secure don't do that. Make the clients figure it out.

Use encryption – A significant number of wireless LANs don't use the encryption that is available. Yes, WEP is weak but it is better than nothing (by the way, 64 – bit

WEP is just about as good as 128 – bit and a lot less computationally intensive). If you have WPA or AES available they are much better, but not if you don't use them!

Change the access passwords on your base stations. The default ones for popular models are very well known, and not just by the bad guys.

It is quite possible to add your own security to that which is available within 802.11. If nothing else, data can be encrypted at the application level before it ever gets into the TCP/IP stack. This can be done either at the file level through PGP or in a data stream through SSH, depending on the nature of your application. Yes, it is more work, but if your application demands it then it certainly can be done.

Service sets

There is one of two classes of service present in any particular 802.11 network. The first of these is an Independent Basic Service Set (IBSS), which is commonly referred to as the ad hoc mode. This mode is very useful for setting up small groups of wireless devices that do not require a centralized base station. In this mode the stations communicate in a peer-to-peer mode directly with each other.

It is worth noting that this mode is the most efficient way to communicate between wireless leaf nodes. If you need to stream data between two nodes this is definitely worth considering, since the alternative requires that data first be sent to a base station and then forwarded to the other node.

The second service option is an Extended Service Set (ESS). This is the usual setup with a wireless LAN built around one or more base stations. It is sometimes referred to as Infrastructure mode.

Roaming between cells of an ESS network is supported under the 802.11 standard, but there are a few possible problems. Theoretically base stations are capable of communicating between themselves on a wired backbone when a mobile device moves out of range of one station and into the range of another. This handoff rarely goes smoothly, though, and if the base stations are made by different manufacturers it is quite likely that the transition will be much less smooth. Part of the reason for this is that most 802.11 implementations do not support soft handoffs, where the transition occurs as the signal from the old station is weakening but before it is lost. It is much more common for this not to happen until the old signal completely goes away.

Nonstandard PHYs

The previous categories of tweaks all stayed within the boundaries of the standards. Now it is time to take a walk on the wild side.

As is evident from the number of different PHY implementations that are part of the standards set that is 802.11 there are a number of choices that are possible for that layer of the stack. There is absolutely no reason that you cannot come up with a customized PHY layer of your own that is optimized for your application. As a matter of fact, there are nonstandard PHYs available from many of the vendors that provide the 802.11 chipsets. For example, Atheros has specialized modes of the 802.11a PHY that crank up the speed of the network to 72 or even over 100 Mbps. Of course these modes

only work if both sides of the communication are using the same chipsets, but that can happen in many embedded system implementations. If raw speed is what you need, this is definitely worth considering.

Keep in mind the tradeoffs here, though. Generally these speeds are achieved at the expense of the number of users that can be supported on the network. If you need to have a lot of client devices on the network it may be a very bad idea to crank up the throughput in this fashion.

There are other considerations that might cause you to create a custom PHY. If you need to communicate in other parts of the spectrum than the ISM bands typically used by 802.11, then a custom PHY may be just the ticket. This can be done either by shifting the spectrum outside of the standard chip setup or by changing the RF circuitry of the 802.11 chipset itself. This is definitely on the extreme end of modifications, but it is quite possible.

These changes definitely move you outside the protection of standards, so you are definitely on your own for these changes. The advantage is that changing the PHY in this way can be done without impacting the application software at all (assuming your implementation works, of course). This is the advantage of doing this type of hacking based on a solid framework rather than starting from scratch.

Bluetooth

The Bluetooth protocol was created by a consortium of companies outside the purview of the IEEE. The thinking behind this was that the Personal-Area Networks (PANs) that were envisioned were going to happen too quickly to allow the ponderous process of consensus that IEEE utilizes. It was felt that a focused group of interested companies could significantly shorten the time required and get to market much quicker.

As it turned out, this was not necessarily the case. Bluetooth certainly got into the public eye much more quickly, but by the time a workable standard was actually delivered most of the interest of the wireless world was on the 802.11b networking standard. This is particularly troubling, since applications that are relevant to WLANs are often not particularly well-suited to PANs, and vice versa. Unfortunately, Bluetooth had also lost its focus on the mission of replacing the tangle of specialty cables that hinder any number of small devices from cell phones to PDAs to iPods. As a result, many engineers overlook Bluetooth for applications for which it would be a very good solution.

Technical details

The Bluetooth Consortium took a very different approach than the 802.11 Committee did. Rather than build Bluetooth as an adjunct to TCP/IP, Bluetooth was built as a standalone protocol stack that included all layers required by an application. This meant that it included not only wireless communications but also service advertisement, addressing, routing, and a number of application-level interfaces referred to as profiles. The latest Bluetooth Core Specification (Version 1.2) weighs in at a hefty 1200 pages! Needless to say, I will not be covering it in such detail here.

Bluetooth is based on a FHSS modulation technique that hops at a very fast rate (1600 hops per second) and has a symbol rate of about 1 Mbps. This translates to about

700 Kbps of actual useful data transfer. Bluetooth networks are organized in piconets, which contain one master and a number of slaves. Bluetooth nodes can simultaneously be members of different piconets, but they cannot be masters in more than one at a time. Overlaid piconets that include common members are referred to as scatternets.

The primary elements of the Bluetooth stack are shown in Figure 0 - 2 below. As with a typical diagram of the TCP/IP stack, there are a number of details that are hidden by the apparent simplicity of the stack. Specifically, there are a number of profiles that sit roughly on top of the L2CAP layer that provide much of the power (and also the complexity) of the Bluetooth protocols.

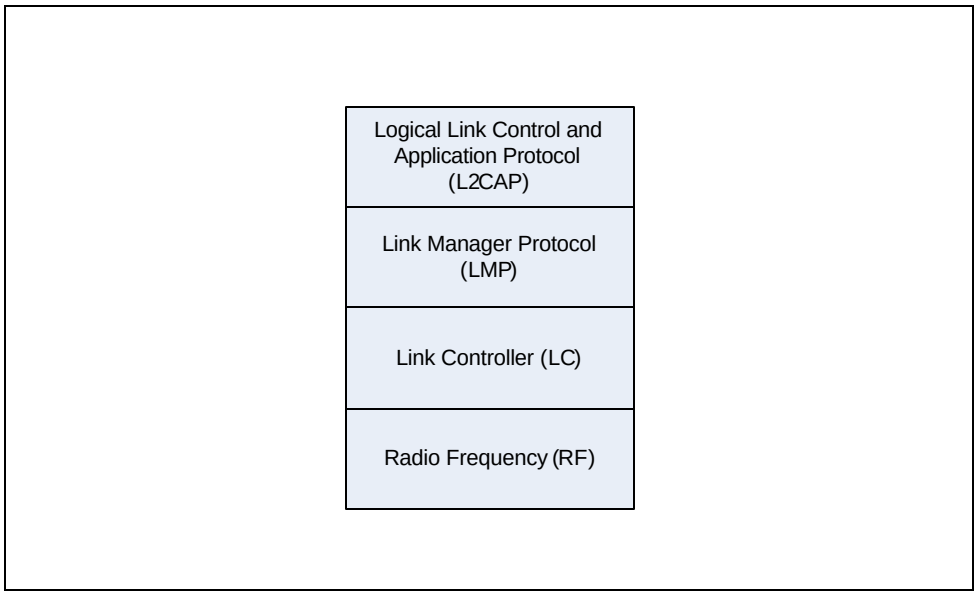

Figure 0-2 Bluetooth Protocol Stack

These profiles are the primary entry into the stack for an application. Essentially, they define the set of services that are available from a connection. Currently there are a total of 28 different profiles defined or in the process of being defined, as well as four higher-level protocols for specialized data and four more specialized transports for communications interfaces. By the way, the documentation for these items is separate from that 1200-page tome I mentioned earlier. Bluetooth definitely requires some heavy reading.

Tweaking opportunities

The good news is that there are any number of choices for selection and optimization within the Bluetooth stack. The profiles are roughly equivalent to the numerous application protocols present in TCP/IP for things such as FTP, Dial-Up Networking (DUN), Serial Port Profile (SPP), etc. these allow a Bluetooth connection to morph into a large number of familiar forms.

That is, of course, part of the bad news as well. There will be a significant learning curve if you want to dig deeply into the Bluetooth protocol stack. It should be possible for most applications to use it relatively easily once it is set up, but that setup can be significant.

Most of the customization in a Bluetooth stack will probably take place in these profiles. It is quite possible to bypass them and communicate directly at lower layers. This essentially allows you to create your own application profiles for specialized applications. This could be a very good choice for an application that wants to build a private dedicated network based on Bluetooth protocols. This allows the use of addressing and RF capabilities on top of the basic communications channel but strips away the other overhead.

Cellular data

Cellular data is becoming very interesting. There were very early uses for cellular communications to send data in the form of paging networks, and there were even some applications that used voice cellular to send modem signals, but in general these were either too unreliable, too slow, or too expensive (or maybe all three) for most mainstream applications.

More modern cellular transports are becoming available, however. GPRS/EDGE and CDMA 1xEVDO are being deployed now by the major carriers, and these have full packet-switched data communications capabilities at reasonable data rates. The carriers are even beginning to be more reasonable as far as charges for data communications. It might be a good idea to take another look at cellular data communications.

Technical details

The alphabet soup of cellular acronyms is settling these days into two separate camps. First, there is the GSM camp, which is practically ubiquitous in Europe and also has a strong presence in the US. On this side the current state of the art is Enhanced Data rates for GSM Evolution (EDGE), which provides up to 400 Kbps capability. This is being deployed by carriers such as AT&T, Cingular, and T-Mobile.

The other side of the coin is based on the Qualcomm CDMA technology. The current state of the art there is 1xEVDO (Evolution Data Optimized), which promises up to 2.4 Mbps. This service is being deployed in the US primarily by Verizon Wireless.

The important thing to realize about the data rates described in the above paragraphs is the qualifier “up to”. Wireless data rates in general should be taken with a grain of salt, and these in particular rate a whole shaker full of it. Most applications should count on at best half of that rated throughput for their applications.

One thing that is much improved in these networks is latency. Early data forms such as SMS and other packet technologies had tremendous variation in latencies, causing huge problems for some early attempts at applications. Current networks do a much better job at providing smoother packet flows.

These issues have been problematic, but by far the biggest barrier to use of cellular for data transmission for embedded applications has been the need to qualify devices for transmission on the cellular networks. Cell phones are very specialized devices that are built by a very few companies in the world, and generally those companies have not been interested in supporting devices that wanted to get on their airwaves. Once when we contacted Qualcomm for technical information to use their chipsets in an embedded application we got a call back within the hour from a lawyer

threatening to sue us for patent infringement! This is not exactly what I would consider good customer support procedure.

Tweaking opportunities

Given this closed environment, what possible opportunities could exist for tweaking in cellular data? As it turns out, that problem is exactly the biggest opportunity.

One answer is to use Bluetooth or even wired communications as a data path into the cellular networks. The real advantage of this approach is that it insulates you from the changes that are taking place in the cellular networks. Rather than building GPRS/EDGE or 1xEVDO directly into your system, instead you build a Bluetooth link and use whatever network is available via an external cellular handset.

In fact, the handset itself can be worked around as well. A few companies now sell modules that are qualified for cellular communications and present serial or direct memory interfaces to allow easy integration into embedded systems. I am working with one of those right now for GPRS from Wavecom, and I know of a CDMA module from Kyocera as well. These can provide very easy integration into embedded systems.

The data transport itself is worth considering as well. There have been many applications using SMS as a data transport, but there are real limitations to that approach. SMS was originally included in cellular networks as a maintenance channel, and it was almost accidentally exposed as a service to the public to send short text messages. In Europe many phone bills (especially for teenagers) have higher billing for SMS than they do for voice service.

The problem with using SMS for data applications is that it is a very limited packet service. The packet size is small (typically about 160 characters) and delivery is fairly uncertain. More modern TCP/IP-based systems are rapidly becoming available and are certainly the choice for new applications.

Conclusion

Wireless data is coming of age. The first stage of this was a hobbyist phase, where the tools were very immature and not suited for serious applications. This gave way to serious mass-market applications in the form of WLANs for laptop computers and cellular handsets, which built the network infrastructure.

The current phase of this evolution is the use of this infrastructure for any number of smart devices. None of these by themselves would justify the creation of the networks, but just like the Internet there are a huge number of applications that can use the infrastructure once it has been built.