

Building and Operating Robust and Reliable ZigBee Networks

Abstract

This paper is based on a presentation given by Daintree's CTO, Zachary Smith, at the 2008 Embedded Systems Conference (ESC) in San Jose.

There's more to ZigBee development than just porting your code to a ZigBee platform and turning it on. Key techniques and system components are covered.

In addition to application components and entities, a ZigBee network must contain infrastructure components that will help the network operate securely and reliably in the face of interference and changing conditions. This paper takes a cradle-to-grave view of the ZigBee device life-cycle and walks the prospective developer through the ins and outs of putting together ZigBee applications that do what they're expected to do for as long as they're expected to do it.

The 3,028m view

ZigBee is a broad-based standard that is intended to cover a range of applications and competing requirements. Figure 1 shows an example of the range of target markets typically discussed for ZigBee.

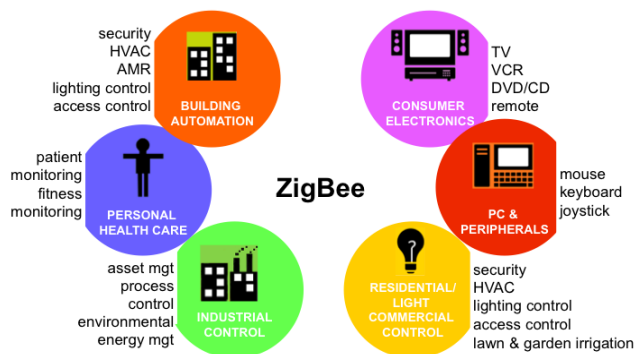


Figure 1 - ZigBee Market Segments

It is easy to imagine the conceptual gulf in requirements between, say, environmental monitoring and residential lighting control. But, even the rather more closely linked application areas for which the ZigBee Alliance has either published or is on the verge of publishing application profiles—Home

Automation (HA), Smart Energy (SE) and Commercial Building Automation (CBA)—have requirements that are sufficiently different as to discourage a "One Size Fits All" approach.

In view of this, the ZigBee specification contains a wide, not to say somewhat dizzying, array of options. These are focused into two feature sets known as ZigBee and ZigBee PRO, while a third option is available for application developers who want to think outside the box but aren't interested in building ZigBee certified products.

Furthermore, ZigBee is, first and foremost, a protocol standard governing interoperability and not a complete specification of device behavior. While certain key components such as the routing algorithms at the network layer are described in detail, there are quite a few other system components, especially application objects, for which the exact behavior is deliberately left as an exercise to the implementer. Two examples will suffice at this point:

- **The Trust Center:** The specification states that every ZigBee network must contain a Trust Center (TC) to control admittance to the network, manage security keys and so on. However, for reasons that will be described in more detail below, the TC application may have many possible flavors and may be called upon to implement policies that reflect specific application or deployment requirements.
- **The Network Manager:** Under certain conditions, ZigBee networks may wish to change channels, under the control of a Network Manager, in response to interference or other factors that degrade network performance. The specification provides protocols whereby these changes may be requested and information in support of network management may be collected, but the policies and procedures that control how and when the Network Manager makes channel-selection decisions and implements them are almost entirely unspecified beyond a discussion of "best practices."

In each case, stack vendors are unlikely to provide anything more than a simple TC or Network Manager implementation that allows developers to use and experiment with these features on the bench-top. The task of defining the exact variant for a particular application will fall on the shoulders of the application developer. The task of writing the relevant applications may fall there as well, although there are now 3rd-party middleware developers who are stepping in to provide these capabilities.

The main point here is that the application developer has choices to make both with respect to stack operation and the operation of other system components that reflect the capabilities of the system as a whole, the particulars of the deployment, the application profile, and the capabilities of the various users of the system.

What follows is roughly divided into two sections covering application and network design considerations and operational considerations respectively.

Design considerations

In deciding how an application is going to make use of ZigBee, it is worth considering, first of all, what the network is expected to look like. In embedded networks, device locations are often fixed by their application function. Lighting and HVAC devices in a home or office, for example, may be somewhat mobile but are most often fixed for reasons that have nothing to do with their ability to form a reliable network. Designers should consider the expected layout in terms of the expected distances between radios and the expected density of the network produced. If the network is expected to contain mobile devices, these should be considered carefully as well.

Range and Reliability

For any layout of devices with a given range, one would naturally expect some devices to be able to communicate more clearly and reliably than others. Digging in a bit more deeply, however, it makes sense to ask what the effect of any particular arrangement will have on network performance. A number of papers including (De Couto, Aguayo, Chambers, & Morris, 2003) document that, for an arrangement

where separations are well distributed with respect to the rated RF range of the device, the performance can be diagrammed roughly as in Figure 2.

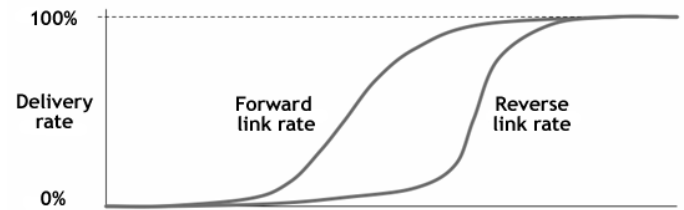


Figure 2 - Schematic View of Link Quality

In the figure, links for a hypothetical network are arranged from left to right by increasing packet delivery rate across the link. For a given pair of devices, say (A, B), the top line shows the reliability for transmission from A to B and the bottom line shows the reliability for transmission back from B to A. Each pair is assumed to appear only once.

What DeCouto and others have observed is that a large number of links in any given network turn out to be asymmetrical in nature and that these asymmetrical links show other bad behaviors such as intermittency, momentary fading and so on. Looking at the graph, it is easy to see that even links that appear quite good in one direction may be unreliable or almost useless in the other. It has also been observed that, to the extent that these effects correlate with distance, they become noticeable well short of the rated "ideal" range of the radio.

In addition, what we may call the reliable range of a device will depend not just on the performance of the radio but also on the hardware design—including packaging, antenna design and PC-board layout—as well as on the materials in the surrounding environment.

The lesson here is to take the rated range for a particular radio only as a starting point for an exploration of what the performance will be in the expected deployment environment of the final, production device, and to design the whole application around realistic expectations for the reliable range. Mesh networks depend on having a multiplicity of usable links between devices and if this link diversity is not available, it will be difficult to deliver a robust network.

Mitigation

If the expected distances between application devices is too great to provide link diversity, the best solutions are either to include inexpensive range extenders in the product line or else to increase the reliable range of the radios using a different antenna design or amplification either on the transmission side, to increase power output, or on the receiving side to increase sensitivity.

Density and reliability

Another factor affecting reliability is network density. Figure 3 shows a network in which the density, which may be thought of as the number of devices per unit area where the unit used is the square of their reliable range, is reasonable. In this case, every device has a few other devices within range but transmissions from a single device do not blanket the entire network and, as shown, devices out of range of each other may carry on independent communications, thereby promoting what it known as spatial reuse.

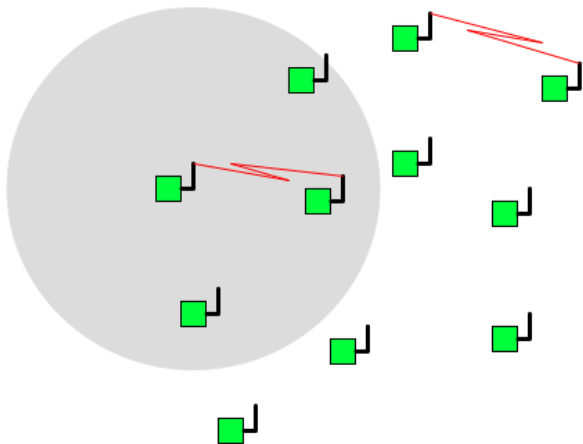


Figure 3 - Good Network Density

Figure 4 shows the same general device layout, but in this case, the reliable range of each device essentially covers the entire network. Here only one pair of devices may communicate at any given time.

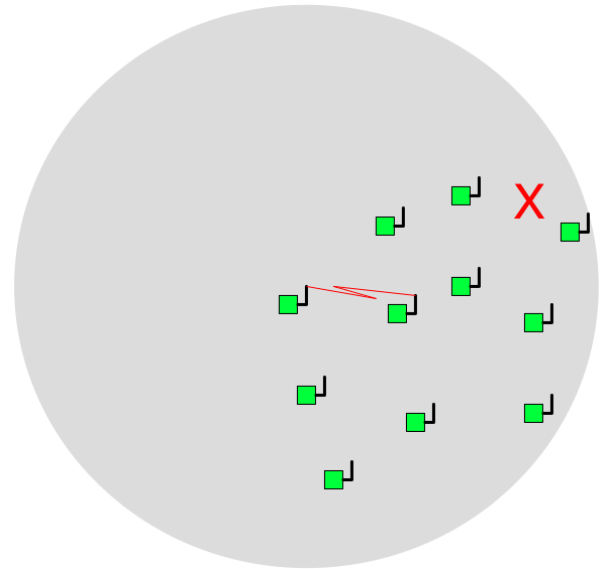


Figure 4 - Bad Network Density

Another way of saying this is that the Utilization Ratio

$$\frac{\text{Density}}{\text{Total Devices}}$$

is fractional in the first case and greater than or equal to unity in the second. The ideal case for networks of any sizes where $4 < \text{Density} < 16$ and the Utilization Ratio is a small fractional number, the smaller the better.

Mitigation: PAs considered harmful

It is often assumed that a sure-fire way to make wireless communication more reliable is to "pump up the volume" by adding a Power Amplifier (PA) to the radio in use. We can see now that, while in some cases this may be the only solution, it is by no means a guarantee of robust and reliable performance. In fact, for dense networks, turning down the power, thereby reducing the number of other devices "blanketed" by a single transmission may be a better policy.

Network utilization

In addition to the spatial layout of the network, the expected utilization is important as well. Utilization here includes throughput considerations as well as the communications patterns that will be in use.

Bit-rate and throughput

ZigBee is based on the IEEE802.15.4 Physical Layer (PHY) and according to (IEEE Computer Society, 2003) the "raw" data rate for a ZigBee radio operating at 2.4 GHz is 250kb/s. The effective data rate, however, will be quite a bit lower and application designers should be aware of the reasons for this.

First, and most obviously, there is the ZigBee and IEEE 802.15.4 protocol overhead. This is difficult to estimate precisely because it depends on the exact composition of the data frame, both in terms of protocol usage and the amount of data in the frame, but it is safe to say that, on average, only half of any message sent using ZigBee contains data.

Next, there is a slightly more subtle point about the operation of mesh networks. Figure 5 shows 4 time-steps in a hypothetical network where all the devices are arranged in a line or "relay chain". Traffic starts at the left side of the network and is relayed to the right at the maximum available rate. The figure shows that, after transmitting a frame in step 1, the device at the left end of the chain must wait until step 4 to transmit the next frame, since transmission is blocked in steps 2 and 3 by the activities of nearby devices.

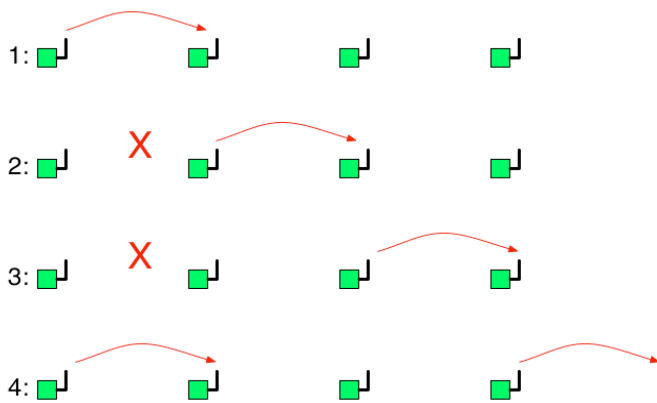


Figure 5 - Contention in a Multi-Hop Network

Taking these two factors together we can see that the effective data rate for a multi-hop network is at best 1/6th of the raw data rate. In practice other factors come into play as well, the most obvious of which is that most mesh networks are not simple relay chains with evenly spaced devices and a single source of traffic.

As a practical matter, application designers should carefully test their assumptions and expectations with respect to throughput under conditions that, as closely as possible, approximate those of an actual deployment. The final design should take the results of these tests into account.

More detail about mesh network capacity is available in (Li, Blake, De Couto, Lee, & Morris, 2001).

Avoiding broadcasts

Broadcasts in ZigBee are reminiscent of the old joke—a whole class or sub-genre of jokes really—for which the punch line reads, in part, "...can't live with 'em, can't live without 'em." Needless to say, broadcast is an important mode of communication, and all stack layers make use of broadcasts. Both the network layer and the APS layer in ZigBee, for example, use broadcast in discovery operations—of routes and services respectively. However, broadcasts are not acknowledged at either the NWK or APS layers. The reasons for this should be obvious, but it means that the primary mechanism whereby the sender of a message is assured that the message was received as intended had been removed. In the absence of this mechanism, the network layer designers attempted to assure that every router in the network will receive the broadcast, essentially using retransmission.

Broadcasts in ZigBee and ZigBee PRO are transmitted 3 times—an initial transmission and 2 retries—with transmissions separated by around 1/2 second plus a modest jitter bounded by 40msec. Handled in this way, broadcast transmission has been shown empirically to be tremendously reliable in test after test, but a single broadcast can dominate the whole network for a matter of several seconds. ZigBee contains optional optimizations that, in certain cases, reduce the volume of traffic but it is best to assume the worst.

What this means is that broadcasts should be used sparingly and avoided wherever possible. This includes broadcasts employed by the lower stack on behalf of the application, for example discovery as discussed above.

A number of techniques exist for reducing or avoiding broadcast traffic:

- Aggregation routing:** The 2007 version of the ZigBee specification describes a feature known as aggregation routing. This is designed to address the extremely common communication pattern in which every device in the network must communicate on a regular basis with a single device, known as an aggregator. For many networks—most sensor networks are an example—this is the only traffic on the network, but even when there is other traffic, as in a building control network, there is often a central controller or monitor that acts, for all intents and purposes, as an aggregator. In order for aggregators to be generally accessible, every router in the network must have an entry in their routing table for the aggregator. For large networks this means that the same aggregator must be discovered many times over, and, if a number of routers try to discover the aggregator at the same time, the ensuing broadcast storm may cause route discoveries to fail or to produce sub-optimal routes. The solution proposed in the 2007 specification and adopted in the ZigBee PRO feature set is one in which the aggregator announces its presence using a special route discovery frame, which, as it transits the network, causes entries pointing to the aggregator to be set up in the routing tables of every router that relays it. Note that in the ZigBee feature set this same problem may be handled by using hierarchical routing, especially if the aggregator is the ZigBee Coordinator.
- Source routing:** A closely related problem is that of communicating from a centrally located aggregator with every other device in the network. Not only does this require that the aggregator discover and store a path to every device in the network, but, even if the aggregator is up to the task, nearby routers, which are assumed to be less expensive and less capable devices, may be overburdened. Figure 6 shows an example of a 17-device network in which, because of range considerations, all traffic to and from the aggregator must go through one of two smaller routers. If each of these routers is capable of managing 8 routing

table entries, a not unreasonable number and allowed as a minimum under the ZigBee feature set, then one of the remote devices will be inaccessible to the aggregator. As a solution, the 2007 specification proposes and the ZigBee PRO feature set adopts source routing, where complete routes to each device with which it is expected to communicate are stored in the aggregator and placed in the outgoing frame along with the data. Intermediate routers need store no information at all in order to route these frames. As an added benefit, source routes are discovered using unicast command frames not broadcast. Note again, that the solution for this problem under the ZigBee feature set is to use hierarchical routing.

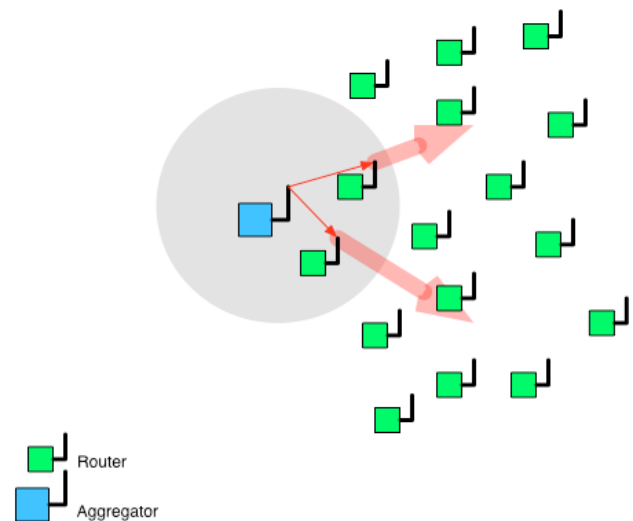


Figure 6—Constrained Routing from Aggregator

- Avoiding General-purpose Discovery:** ZigBee offers a rich set of protocols for discovering devices and their various attributes, including the features and services they support. These are provided to address the general case and operate in the absence of any other information. Thus, a device wishing to discover whether a particular basket of services is offered by any device on the network may broadcast a command requesting that any such devices report in. As has already been discussed, this sort of activity can lead to a great deal of traffic, both broadcast and unicast, and may cause frames and the information contained in them to be lost. Fortunately, application designers often know a great deal

more about the kinds of devices and services that will typically be on their networks and may use this information to avoid laborious and costly discovery.

- **Throttling:** If a particular actuator, say a switch, has the capacity to cause broadcast traffic, it makes sense to throttle that traffic so that a careless user is not able to overload the network by sending out continuous broadcasts.

Polling also considered harmful

Another typical communications pattern that should be used with care is polling, especially polling across a multi-hop mesh.

Polling, is a natural model for wired, bus-oriented networks and is enshrined in such protocols as Modbus. Polling makes perfect sense for such networks. The medium in a Modbus network does no extra "work", such as routing, on the message after it has been transmitted. Therefore, from the systems perspective, use of the medium is relatively low-cost. By contrast, while the actual medium in a ZigBee network (air) is also low-cost, use of the medium in a ZigBee network typically involves other devices, which are called upon to participate in routing and other related activities. So, in ZigBee the "wire", that is the RF medium and multi-hop network, is more costly than in a simple bus-oriented network and this fact should be taken into account when designing (or redesigning) the applications that run on it.

Furthermore, in a network where end devices may be battery-powered, a polling model requires that end devices stay in synch with the network master and wake up on a schedule dictated by the master. Certain classes of sensors, such as door and window closure sensors, can achieve much better duty cycles by waking up on their own schedules in an event-driven manner.

Finally, in a strict polling model, each data request takes up network resources for both the request and the response and does so regardless of whether the device being polled has any new data to report or not.

In ZigBee, and in other wireless, mesh networks, it makes sense to allow end devices to report in only

when they have something new to report. This may take two forms:

- **Change-driven reporting:** For data-collection devices, it makes sense for the device to only report a result when the result it reports would be sufficiently different for the master to care. What this means and what thresholds are used depends largely on the application, but the general principal should be clear and, in any case, is well known.
- **Event-driven reporting:** There is a large class of devices, such as door-closure sensors, whose behavior is largely driven by external events. There are only two reasons for a door closure sensor to report: 1) when the door is opened or closed, and 2) to inform the master that it is still operational, which may happen relatively infrequently or, perhaps, not at all.

ZigBee supports both of these models directly.

Interoperability

Interoperability is one of the primary drivers for ZigBee. It is worth noting that this falls directly out of the fact that many of the promoters of ZigBee in the early days of its development were semiconductor companies. If the goal for semiconductor companies is to move as many chips as they possibly can, then a stack, application support framework and certification programs that promote interoperability—thereby keeping the barriers to entry for prospective application developers as low as possible—are ideal.

Application developers, on the other hand, may not be so sure.

When embarking on the development of a ZigBee-based application, it makes sense for application designers to ask themselves how important interoperability and a multi-vendor environment are for the success of their applications. It may also make sense to ask this question both from the point of view of the company providing the devices and of their customers. The answers may differ and, in this case at least, the customer is probably right, since they will be the ones making the procurement decisions.

If it is decided that designing for a multi-vendor environment makes sense, then there are a few things that the application designer can do to promote interoperability:

Obviously, the most important thing to do is to **adopt a ZigBee standard profile**. ZigBee has a growing list of application areas for which it has published profiles and where the intended application overlaps with one of these areas it makes sense to adopt the standard. It also probably makes sense to **avoid vendor-specific extension** to the standard profile where possible. Even in cases where a standard profile isn't available, ZigBee offers a large library of standard clusters, the ZigBee Cluster Library (ZCL), as grist for the application-designer's mill. A decision to **use standard clusters**, even when they are not grouped into a standard profile, will help interoperability in the long run.

As the experience of other standards that have sought to build ecosystems of interoperating products has shown, standard profiles are only half the battle. As an example take binding. Any time a device is called upon to remember a destination for a particular kind of application traffic this is, in essence, a binding. ZigBee offers standard mechanism for binding and, if application developers **use standard binding**, it provides a separation of concerns between the activities proper to the application and the task of remembering where application-related frames are supposed to be sent. In effect, the application can be written in a destination-independent manner and separate tools and commissioning steps can be used to hook up the "plumbing."

Generally, developers who care about interoperability should **use standard commissioning and maintenance procedures**, as defined in the ZCL and as directly supported in the ZigBee specification itself, since this allows for a multi-vendor deployment in which a tool from Vendor A may be called upon to commission or maintain a device from Vendor B.

Conclusions

In the forgoing, we have reviewed some of the important design decisions that go into building a robust, stable ZigBee network that delivers interoperability and ease of use. Next we turn to the

components of an operating ZigBee network and, in particular, those components that the application developer may be called upon to supply or, at least, to support.

Operational considerations

The ZigBee Alliance's slogan is "Wireless Control that Simply Works," and to a remarkable extent this is the case. Under a wide variety of operational conditions, ZigBee networks can be expected to configure themselves, discover how to route traffic, how to heal themselves in case of device failure, and so on. However, there are a number of engineering problems for which the ideal solution is so dependent on the particulars of the application, or, in some cases, of the deployment, that it is impractical to solve them in what is primarily a network layer specification. What this means is that an operating ZigBee network must contain a number of application components that provide solution to these problems. In some cases these are well described in the specification and in some cases they are not.

Stack providers may provide the building blocks with which to put these application components together but, for exactly the same reasons stated above, they will probably not be in a position to provide a single solution for all possible scenarios.

Managing sleeping devices

End devices in ZigBee only have one communications path into the network and that is through their parent device. As a result of this, they are not expected to participate in network activities such as routing, and they may be allowed to sleep for indefinite periods in order to conserve power. This means that an end device might not be able to receive frames directed to it until it wakes up.

ZigBee, or more precisely IEEE 802.15.4, provides a mechanism whereby traffic for a sleeping end device is buffered temporarily on its behalf by its parent router. However, this mechanism only guarantees that the MAC will buffer a frame for a time governed by the parameter *nwkTransactionPersistenceTime*, which is normally defined to be around 7.68 seconds. Clearly, this mechanism will be adequate in some cases but in the vast majority of cases some other

mechanism will be required to assure that messages reach a particular end device.

Noting that an end device's parent is chosen by the NWK layer in a complete absence of any application information, and that only a peer application entity can be expected to know what a particular end device's operational parameters are, it makes sense for designers to define an application-specific manager for sleeping devices.

In order for a manager device, that is not the sleeping end device's parent, to communicate effectively with a sleeping end device, that end device should implement 2 levels of sleep. Figure 7 shows the typical sequence, which may be elaborated, as the application requires.

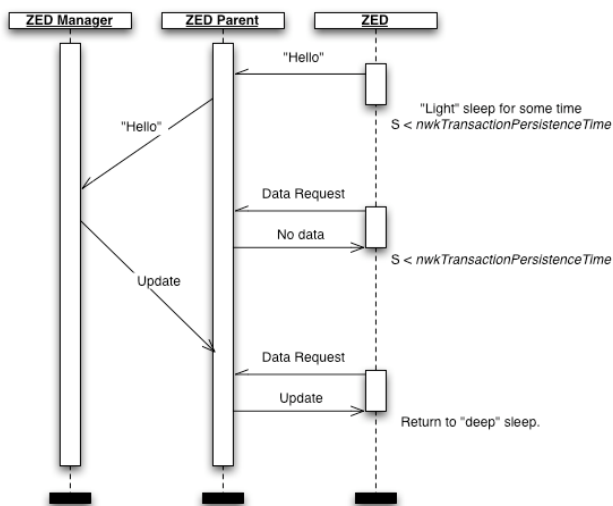


Figure 7 - End Device Management

In the figure, the ZED wakes up, either as a result of an external event or because of its application-specific duty cycle. It sends its manager a "Hello" message asking if there are any updates or other data available. Once it has sent this message, it may immediately go back into sleep mode but should wake again after a sleep interval $S < nwkTransactionPersistenceTime$ to check for a response from its manager. This "light sleep" state should persist until an update is received from the manager. After an update is received, whether it contains data or not, the ZED may return to its normal state. The ZED may also close the loop by sending some form of acknowledgement back to its manager.

Dealing with interference

One of the most frequently asked questions about ZigBee, and in general about wireless systems operating in the various ISM bands, is how prone these networks are to interference.

There is a growing body of evidence that ZigBee and the other wireless protocols in the IEEE 802 family do a good job of coexisting and of dealing with interference in general. For an Alliance whitepaper that discusses interference-avoidance measures in ZigBee in a general context, see (ZigBee Alliance, 2007).

Studies show that, while it is always possible to jam an RF signal with particularly aggressive interference, under operating conditions that are not fabricated in the laboratory, ZigBee networks perform well and degrade gracefully in the face of powerful or nearby interferers. Nonetheless, it must be acknowledged that even this "graceful" degradation may cause unacceptable application performance if it is sufficiently severe.

One of the mitigation measures that the Alliance has included in the 2007 specification and adopted in both the ZigBee and ZigBee PRO feature sets is the ability for the entire network to change channels if a problem of this sort is discovered. Essentially, the solution defines a Network Manager entity, which is able to use any router on the network as a kind of "RF probe" to both report communications failures in the normal course of operations, and to sample the RF environment on different channels and report the results. Once the Network Manager has made a decision to change channels, it informs the rest of the network. Devices that have "fallen off" due to interferences have the ability, inherent in 802.15.4, to scan other channels in search of their network and rejoin it.

What the specification explicitly declines to spell out is the decision-making procedures whereby the Network Manager makes a decision to change. To show just one of the ways in which decisions of this kind can be difficult to make reliably, imagine the network shown in Figure 8. The network is roughly divided into 2 "lobes". The left lobe maintains a fairly constant high level of activity, which is nonetheless

sustainable, while the right lobe has a more intermittent activity profile.

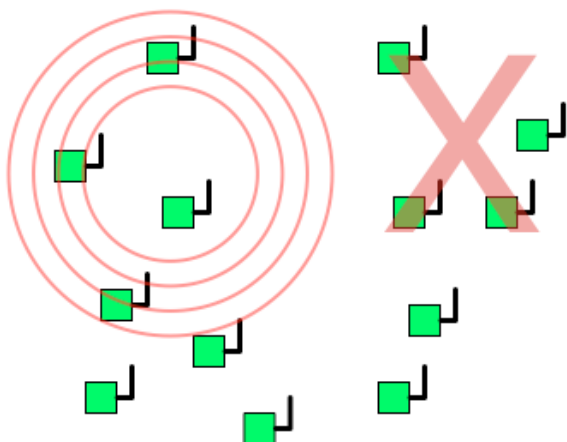


Figure 8 - Auto-interference

In this situation, attempts to exchange frames between devices in the right hand lobe may be severely hampered by the ongoing activity in the left hand lobe. Furthermore, if the devices in the right hand lobe are asked to look for a better channel they will, in all likelihood, be able to find a channel that has less in-band activity but switching to that channel will obviously be futile because, after the network resumes normal activity, because the same level of auto-interference will be present as before. In this case, it may make sense to integrate the Network Manager into the application as a whole so that it can attempt a range of solutions to network performance problems, some of which may be implemented at higher layers.

In many, perhaps most, cases the best option is to leave channel-changing decisions to human operators who may have a clearer idea of who the possible interferers are and what may be done about them.

Secure deployment and maintenance

As stated above, every ZigBee network is required to have a Trust Center (TC), which manages security for the network. The specific capabilities of the TC will depend on the application profile or profiles being deployed, the underlying stack profile or feature set and developer decisions with regard to optional security features.

A partial list of options follows:

- Two security modes, known as Standard Security and High Security, are called out in the ZigBee specification. Devices conforming to the ZigBee feature set must use Standard Security while devices conforming to the ZigBee PRO feature set may use either Standard Security or High Security.
- Some features, including the use of Master Keys, which are shared between each device and the TC, are optional in some cases and mandatory in others.
- Policy decisions, such as whether to admit devices that support lower levels of security than the one implemented in the TC, which are left to the implementer.
- The option now exists to include a key establishment cluster based on Public Key cryptography in the TC application.
- Even if the key establishment cluster is not used, there are a number of options for placing keys in devices, which may affect usability and degree of security. Keys may be distributed in the clear at network join time—the least secure option—or they may be preconfigured at manufacturing time or in some pre-installation configuration step using in-band or out-of-band methods.
- In the case of preconfigured keys, the entire network, including the TC, may be preconfigured, thereby leading to a closed network, or the TC may be informed out-of-band of the key being used by a joining device.

With respect to maintenance, it should be clear that any tools required to maintain the network should, first of all, only be allowed to do so by permission of the TC and, second of all should only be admitted to the network if they can be authenticated.

Conclusion

This document has attempted to lay out and explicate some of the choices presented to application developers who wish to use ZigBee. The hope is that it will help them quickly navigate to a solution that works the way it's supposed to from its first deployment to the day, some time in the far future, when the network is decommissioned to make way for new applications – presumably based on molecular computing and faster-than-light (FTL) quantum networking. Until that day, there's ZigBee.

Bibliography

De Couto, D. S., Aguayo, D., Chambers, B. A., & Morris, R. (2003). *Performance of Multihop Wireless Networks: Shortest Path is Not Enough*. <http://pdos.csail.mit.edu/papers/grid/hotnets02/paper.pdf>.

IEEE Computer Society. (2003). *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE.

Li, J., Blake, C., De Couto, D. S., Lee, H. I., & Morris, R. (2001). *Capacity of Ad Hoc Wireless Networks*. MIT.

ZigBee Alliance. (2007). *ZigBee and Wireless Radio Frequency Coexistence*. ZigBee Alliance.

About Daintree Networks

Based in Mountain View, California, Daintree Networks is a clean technology company that provides wireless control solutions for commercial buildings. Daintree has a strong background in wireless sensor and control mesh networking, with extensive knowledge and experience gained through its industry-standard design verification and operational support tool, the Sensor Network Analyzer (SNA). In addition to wireless embedded expertise, Daintree has put together a team of seasoned professionals from the lighting, telecommunications and networking worlds. Daintree's expertise and knowledge is now being focused on the development of cost-effective building automation systems. These provide benefits including reduced energy consumption, costs and carbon footprint, compliance with new "green" building regulations, and cost savings available through government rebates and the ability to take advantage of demand response programs.

Daintree's Wireless Lighting Control Solution (WLCS) allows lighting manufacturers to speed their time to market, and enables them to deliver powerful, comprehensive, flexible, and reliable wireless lighting control systems for commercial buildings. For more information, visit www.daintree.net or email sales@daintree.net

Copyright © Daintree Networks, 2004–2010
August 2008

ZigBee is a registered trademark of the ZigBee Alliance.

802.15.4 is a trademark of the Institute of Electrical and Electronics Engineers (IEEE)

Daintree Networks Inc
1503 Grant Road, Suite 202
Mountain View, CA 94040 U.S.A

(w) www.daintree.net
(e) sales@daintree.net
(p) +1 (650) 965-3454