A Software Framework for Application Development using ZigBee Protocol

Bastin Tony Roy Savarimuthu, Morgan Bruce and Maryam Purvis
Department of Information Science
University of Otago, Dunedin, New Zealand
{tonyr, tehrany}@infoscience.otago.ac.nz, brumo162@student.otago.ac.nz

Abstract

The problem with the uptake of new technologies such as ZigBee is the lack of development environments that help in faster application software development. This paper describes a software framework for application development using ZigBee wireless protocol. The architecture is based on defining XML based design interfaces that represent the profiles of ZigBee nodes that are used in the application.

1. Introduction

Wearable, portable computing devices have started to emerge in day-to-day interactions thanks to the convergence of networks that provide interaction capabilities between these devices. Some of the challenges with smaller computing devices are their battery life, the amount of data transferred and their reliability.

Bob Metcalfe, the inventor of Ethernet, which added a whole new dimension in the world of computers talking to each other said "*There are about eight billion microprocessors shipped every year, in everything from our cars to washing machines to industrial processes. ZigBee will network these devices*" [1]. Since ZigBee devices can form a mesh network of unlimited size, this will potentially be an important future technology for applications that require limited amounts of data to be transferred. The advantages of ZigBee protocol over the well-known protocols such as Bluetooth [2] and Wi-Fi [3] include the lower power consumption of ZigBee devices, their low cost and the support for relatively larger number of nodes in the network. Another important feature of this technology is its support for self-organising and adaptive networks. The current ZigBee application areas are in building embedded systems for industrial control, medical data collection, smoke and intruder warning and home automation.

One problem that is common to all new technologies is the unavailability of a suitable development environment that facilitates faster development of applications. ZigBee being a relatively new technology suffers from the same problem and this paper describes an approach to address this.

Section 2 provides an overview of the ZigBee protocol and compares it to other protocols. In section 3 we describe the software framework that we developed that helps easier and faster creation of ZigBee applications.

2. Overview of ZigBee protocol

2.1 ZigBee Protocol

ZigBee is designed to work on top of the IEEE 802.15.4 standard [4, 12] for low-rate wireless Personal Area Networks (PAN). The ZigBee standard is the result of collaborative design and development between a number of international companies, who together form a consortium known as the ZigBee Alliance [5]. This consortium is headed by some of the largest worldwide electronics companies such as Phillips, Siemens, Texas Instruments and Motorola.

ZigBee was designed particularly for applications that satisfy the criteria such as low rates of data exchange, low power consumption (hence, higher battery life), low cost, range exceeding 10 meters, possibility to include strong security measure and the support for open industry standard wireless protocol [6, 11, 12]. The other features of ZigBee protocol include support for a large number of nodes, fast and easy deployment, low latency, self-healing and interoperability.

2.2 ZigBee devices

There are three types of devices in a ZigBee network namely the coordinator, the router and the end device. The co-coordinator is responsible for managing (monitoring and controlling) the overall network. The router is a Full Function Device (FFD). It supports the full range of functions and features specified by the standard, and can also function as a network coordinator. A ZigBee end device is a Reduced Function Device (RFD), which can only transmit data to a router or a coordinator. A FFD can also be used as an end device. The end nodes have reduced

functionality in order to minimize the complexity and the cost.

Figure 1 shows the overall interactions between a coordinator and an end device in the context of a temperature sensing application. Once the coordinator and the end devices are up and running, the end devices scan the channel range to find the coordinator. The end device can join the network and an acknowledgement will be sent by the coordinator. The end device can then send the temperature data to the coordinator and the coordinator can display this information on the console.

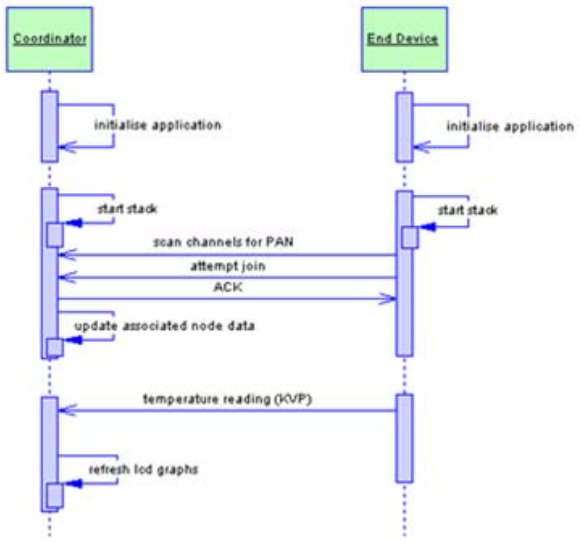

Figure 1. Interactions between a coordinator and an end device

2.3 ZigBee Topologies

Three types of topologies are supported by the ZigBee protocol: the star, mesh and cluster-tree. In the star topology, there are two types of devices - a coordinator and the end nodes. All end nodes communicate with the coordinator. In the mesh topology, the routers are connected to each other. The routers route messages from end devices to the central coordinator either directly or through other routers. This mechanism supports duplicate routers, which allows for traffic from the end devices to be re-routed through different paths. This allows the network to self-heal if a router fails. In the mesh topology if one of the routers fail, the end devices can still connect to another router (router) and send their information to the coordinator. The third type of network is the cluster-tree network which is a combination of both the star and mesh topologies. The advantage of the cluster tree is that it is able to extend the range of the network beyond that of a star and mesh topology. The mesh topology is reliable and robust as it can accommodate random failures of routers. But, the disadvantage of this topology is that the routers cannot go to sleep and hence would use large amount of power. In the cluster-tree topology, the routers can go to sleep, hence it is not as robust as the mesh topology in terms of router failures.

2.3 Comparison with other protocols

In this section we compare ZigBee protocol with the other well known and emerging wireless protocols. ZigBee focuses on large scale monitoring and controlling applications in which nodes transmit limited data and require very low power. ZigBee protocol can be used to connect the ZigBee nodes to be integrated with the IP networks. This approach enables remote sensing and controlling on the Internet.

Well known technologies such as Wi-Fi, target devices running applications that require larger bandwidths where large amounts of data need to be transferred. Wi-Fi enabled devices require large amounts of battery power. The use of Bluetooth technology has primarily been as a replacement for cabling between personal devices. The bandwidth it supports is lower than Wi-Fi. The network sizes supported by both Wi-Fi and Bluetooth are reasonably limited whilst ZigBee can support up to 65536 nodes in a network. ZigBee allows nodes to go to sleep which saves a considerable amount of power. The bandwidth supported by ZigBee is much smaller than that of Wi-Fi and Bluetooth and hence it is only suitable for applications that require small amounts of data to be transmitted. As the network size can be large, ZigBee is suitable for large scale applications (e.g. industrial control).

Wireless USB [7] serves the same purpose as Bluetooth but suffers from the same power problem similar to Wi-Fi and also the transmission range it offers is small. While Wi-Fi, Bluetooth and Wireless USB are not ideal for the same application domains as ZigBee (large scale monitoring and controlling applications), Wibree [8], Z-Wave [9] and EnOcean [10] technologies are aimed at similar market niches but they all suffer from the lack of standardization.

The Wibree [8] technology is a low-powered extension to Bluetooth, developed by Nokia. One of the advantages of this technology is that it can be implemented using existing Bluetooth devices, without

any additional hardware. While it has a higher data transmission rate than ZigBee, it has a very limited range, which makes it unattractive for large-scale networks.

Z-Wave [9] has been developed by a consortium of companies, including Intel, to meet requirements similar to ZigBee such as low-power, cost-effective and reliable wireless networking. It is aimed exclusively at home automation, however, it is not based on any recognized standard. Like ZigBee, Z-Wave can use a self-adaptive mesh topology to achieve wide-range and reliable networking. Unlike ZigBee, it does not use any central coordinator to help achieve this, and hence transmits data at a lower speed. This results in higher battery usage and an increased chance of collision between data packets. A Z-Wave network is also smaller than a ZigBee network which limits both its potential applications and future expansion.

EnOcean [10] solves the problem of battery life by not having batteries. Their wireless sensors are powered by 'energy harvesting' (by temperature fluctuations, solar power, piezo-electricity, vibrations or movement) in order to meet needs for home and building automation, medical, and logistics sensors. Like ZigBee, they form mesh networks that can interface with IEEE 802.11x and ZigBee networks. Unlike ZigBee and Z-Wave, it has been developed and patented by a single company, not a consortium. Table 1 shows the comparison of both well-known and emerging wireless technologies based on several criteria.

	ZigBee	**Wi-Fi**	**Bluetooth**	**Wireless USB**	**Wibree**	**Z-Wave**	**EnOcean**
Standard	802.15.4	802.11x	802.15.1	USB	N/A	N/A	N/A
Application Focus	Monitoring & Control	Wireless LAN	Short range cable replacement	USB cable replacement	Low-power Bluetooth (eg. sensors)	Monitoring & Control	Monitoring & Control
Bandwidth	20 – 250 Kbps	54 Mbps	1 Mbps	110 – 480 Mbps	1 Mbps	40 Kbps	120 Kbps
Network Size	65536	32	7	N/A	*	232	*
Transmission Range	10 – 100m	50 – 100m	10m	10m	5 - 10m	30m	300m
Power Consumption	Very low	High	Medium	High	Low	Very Low	Extremely Low
Typical Applications	Home & Building automation, industrial controls, sensors	Wireless LAN connectivity	Wireless connectivity between devices (e.g. laptops & phones)	Computer peripherals	Low power connectivity, e.g. watches, sports sensors, toys	Home automation & sensor networks	Home & Building Automation, Sensors, Medical

Table 1 – Comparison of different wireless technologies.
(* refers to details unavailable, N/A refers to details not applicable)

3. Design and Implementation of a framework to develop ZigBee applications

To realize our objective of developing a ZigBee based application we chose Jennic's hardware-software implementation of ZigBee protocol [13]. The toolkit had one coordinator, two routers and two end devices. Soon after developing some sample applications it became clear that the developer had to write large amounts of code (~ 300 lines for each device) to create a simple system like the light-switch application. So, we focused on designing and developing a framework that provides a development environment that helps faster development of ZigBee applications.

We identified three options that can help reduce the development effort required. The first approach was to provide a higher level API on top of Jennic's ZigBee API (e.g. API for node joining and leaving). The limitation of this approach was that it can only be used only with Jennic's implementation. The second option was to generate application specific templates. These templates would have all the code required for each node except the application logic. For example, in the light switch context, all the code except the one that turns the light on or off will be provided in the template. This approach is limited because, it will again work only with the Jennic's implementation and also the designers of the templates cannot foresee all possible applications.

The third approach to reduce the development effort was to create an XML based interface for the devices involved in the application. This XML based interface is a generic interface which is based on the ZigBee Protocol specification provided by the ZigBee alliance. This XML based interface can then potentially be converted into implementation specific code by other ZigBee vendors. As this approach assumes XML to be the common denominator, we chose this option.

The specification provided by ZigBee alliance defines a concept called *application profiles*. An application profile represents the profile of the messages and message formats exchanged by devices participating in an application. The application profile enables individual nodes that are part of an application to send commands, request data, process commands and requests between devices. For example, in the context of light-switch application, the application profile describes, the messages and message formats that should be exchanged between these two devices.

Figure 2 shows the various entities that make up the ZigBee *Profile*. A profile consists of several *nodes* (one or more) depending upon the application that is being built. A node usually corresponds to one physical object. A node in ZigBee network can have 240 devices attached to it. For example, an end node can have three sensors namely thermal sensor, humidity sensor and the motion sensor. Each of these devices (or sensors), have an endpoint reference. Each of these devices consists of *feature sets* and *cluster* information. Feature sets encapsulate the set of mandatory and optional features supported by the device. A cluster is a related collection of *commands* and *attributes*. Clusters store information about the devices that are participating (bound to a cluster) by using the unique identifier for each cluster. The attributes are the data that is stored which is a physical quantity (e.g. temperature) or a state (e.g. on or off). The commands allow devices to manipulate the attributes (e.g. *set* and *get* commands). In the light-switch example, one node acts as a client (switch) which typically manipulates the attributes held by the server (light). Information about some commonly used clusters is available in the ZigBee Cluster Library document provided by the ZigBee alliance [14].

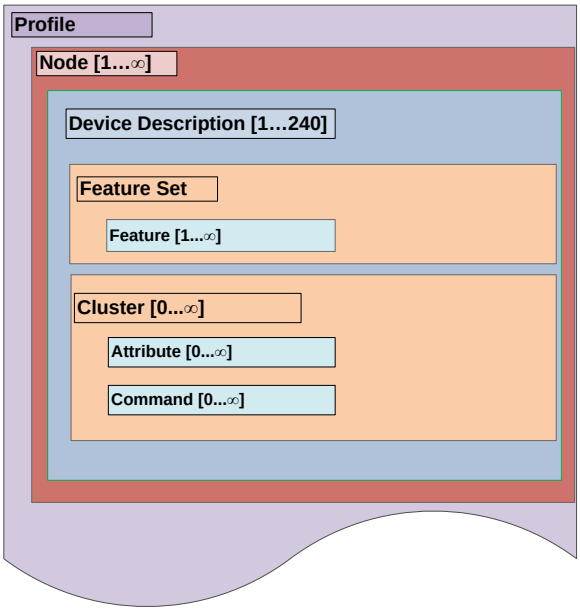

Figure 2. Structure of the ZigBee profile

The concept of application profiles is used to model a XML based interface. The software framework that we have implemented follows a three step sequence to generate application specific code from a given

application profile document described in XML. Figure 3 shows the architecture of our conversion framework.

Step 1: The ZigBee application profile is represented as a XML document using the schema which defines the data types used by the application. This XML document encapsulates all the necessary details required to create the application profile (i.e. the elements shown in figure 2).

Step 2: Once a XML based model is created, it can be converted into implementation specific code by our conversion framework. The XML document is parsed by the ZigBeeProfile Manager (ZBP Manager) class. Dom4j and XPath expressions have been used for parsing purposes. ZBPManager class creates several Java classes encapsulating certain aspects of the XML document. These generated Java classes include ZigBee application profile class, consisting of nodes, device descriptions, their clusters and attributes as well as other implementation details such as network ID and channel. The nodes, device descriptions, clusters and attributes were created as separate classes. For a light-switch application profile, there will be classes corresponding to three node types (coordinator, router and end device) and their corresponding device descriptions. Each node type will have reference to the corresponding cluster types that they are using (typically these nodes will refer to the same cluster type instances; just the role attribute will be different which specifies whether a node is a server or a client).

Step 3: The ZigBeeFactory class uses the appropriate Java classes that were generated along with the C templates that are predefined by the framework and generates the ZigBee application code with annotations that indicate where the application logic should be added. The C template classes were Java classes which helped to produce C code. For example, there was a separate class (CTemplateFunction.java) that was used to construct the functions in C and another class called CTemplateImports.java constructed the necessary #include statements of the C program. The resultant files are *coordinator.c*, *router.c* and *end_device.c*. The application programmer can then insert the application logic to the C programs appropriately.

In step 3, an application developer can choose to generate code for all the nodes or select one or more node types and also optionally indicate which clusters should be associated with the chosen node. This gives developers the flexibility to concentrate on one node. This also helps in the partition of tasks between developers. While one developer focuses on developing a router program the other can work on the end-device program.

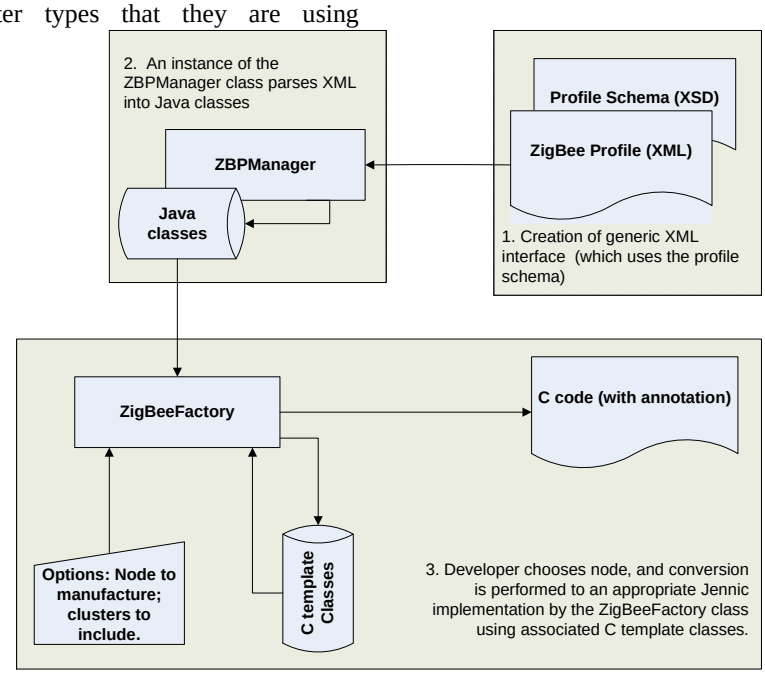

Figure 3. Architectural diagram of the software framework for developing ZigBee applications

For a simple application involving a light and a switch, table 2 shows the Source Lines of Code (SLOC). Some basic code is provided by Jennic for each of the devices (349 lines for each device). It can be noted that significant amount of the code is automatically generated by the framework which otherwise had to be written by the developer. For the router (switch), the application developer had to write 51 lines and for the end-device (light) 21 lines were needed. No extra lines were needed for the coordinator because there was no application logic specific to the coordinator. The coordinator served as a link between these two devices through which data was transferred. In other applications the coordinator also has the potential to act as an end-device if need arises. Usually, the coordinator is used as a point of contact for monitoring and controlling applications such as connecting to a computer which displays the information from all end-devices (e.g. sea-level monitoring application).

Device	SLOC			
	Provided by Jennic	**Generated code**	**Application code**	**Total**
Coordinator	349	19	0	368
Router (Switch)	349	316	51	716
End Device (Light)	349	394	21	764

Table 2 – Generated and application specific SLOC

5. Discussion

We note that using our framework would reduce the effort required to develop larger and complex ZigBee applications. The XML based interface would be an important design artifact to model the system and can serve as a common standard to facilitate interoperability between different vendors. The XML document will also serve as a common reference point in a firm, if multiple developers work on the same application at the same time. We expect that the ZigBee standard would continuously evolve to accommodate expanding needs of different types of applications that should be supported. Hence, the modular XML structure that we have proposed needs to be changed.

There have been some issues with the development of applications tailored to run on Jennic's implementation. For example, the Jennic ZigBee implementation software does not support explicit representation of commands (e.g. set and get commands that manipulate a given attribute). Even though we have extracted out the commands in the XML profile, this was not put to use at the code level. To overcome this problem, the developer had to write explicit methods in C once the application code was generated. Another problem that we encountered was the lack of support for many-to-one and many-to-many bindings between devices in the Jennic's software implementation. Currently, multiple one-to-one bindings are used to deal with this scenario.

One limitation of the framework is that we use three separate programs that are used for the following purposes: 1) create XML interface, 2) generate C code and 3) writing and compiling of application specific code. An Integrated Development Environment (IDE) that can perform all these tasks under a single window would be desirable. Eclipse's plug-in development environment can be used for this purpose in the future. Another improvement that is required is to use an XML editor customized for creating ZigBee profiles. This editor would have built-in components that will facilitate easier creation of ZigBee related elements such as clusters.

An important lesson learnt from developing applications for emerging technologies such as ZigBee is to know the distinction between the standard and the implementation. This distinction helps the developer to identify what is possible with the current implementation. Also, the developer should keep abreast of the changes that are made to both the standards and the implementation.

6. Conclusions

In this paper we have provided background information on the ZigBee protocol and have compared this with well known and emerging wireless technologies. We have described the software framework that we have developed for rapid application development using ZigBee protocol. We have used a XML based interface to represent application profile information. This platform independent approach will not only reduce the development time but also increase the interoperability between vendors that develop ZigBee application.

We are planning to communicate our findings to both ZigBee Alliance as well as Jennic towards establishing standards that can facilitate interoperability of applications that are developed using ZigBee protocol.

7. Acknowledgements

We thank Abdulla Aljawder and Hayden Kane who helped in the literature survey of the project. This project was supported through the University of Otago Research Grant in 2007.

8. References

[1] Article on ZigBee's future, http://www.techworld.com/mobility/news/index.cfm?NewsID=2406, accessed on 30th September 2007.

[2] Bluetooth specifications, http://www.thewirelessdirectory.com/Bluetooth-Overview/Bluetooth-Specification.htm, accessed on 15th September 2007.

[3] Wireless Fidelity (WIFI), Specifications from http://www.irit.fr/~Ralph.Sobek/wifi/, accessed on 15th September 2007.

[4] IEEE Computer Society. (2003). IEEE 802.15.4 Standard Specification, accessed on 15th September 2007.

[5] Zigbee Alliance. (2006). ZigBee specification, http://www.zigbee.org/en/about/, accessed on 15th September 2007.

[6] Frank, A. R., ZigBee's Buzz - A New Low-Cost Wireless Control Technology is Spreading its Residential Systems, accessed on 15th September 2007.

[7] Wireless USB, http://en.wikipedia.org/wiki/Wireless_USB, Accessed on 1st July, 2007

[8] Wibree, http://www.wibree.com, Accessed on 1st July, 2007

[9] Z-Wave, www.z-wave.com, Accessed on 1st July, 2007

[10] EnOcean, http://www.enocean.com, Accessed on 1st July, 2007

[11] Frank, R. (2006, May). Five New Zigbee-Based Wireless Systems. Design News , pp. 85-90.

[12] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989. Baronti, P., Pillai, P., Chook, V., Chessa, S., Gotta, A., & Fun Hu, Y. (2006). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. Computer Communications , 1656-1690.

[13] Jennic's JN5139 ZigBee Evaluation Kit, http://www.jennic.com/files/product_briefs/JN5139-EK010-PBv1.01.pdf, accessed on 15th June, 2008

[14] ZigBee Alliance's technical documents download, http://www.zigbee.org/en/spec_download/zigbee_downloads.asp, accessed on 15th June, 2008