Development of the User Interface for the MavBlue Development Environment

Phuong Anh Thi Loi
Department of Computer and Information Sciences
Minnesota State University Mankato
phuong-anh.loi@mnsu.edu

Steven V. Case
Department of Computer and Information Sciences
Minnesota State University Mankato
steven.case@mnsu.edu

Abstract

MavBlue is a Bluetooth development environment that has been created to facilitate undergraduate and graduate research and education activities. The MavBlue development environment includes a Bluetooth module, Bluetooth protocol stack, and a user interface to manage the environment. This development environment is intended to provide undergraduate and graduate students with a reasonably extensive set of hardware and software tools to explore research in personal area networking.

The MavBlue software has been developed in Java. Java has been used in order to facilitate portability of MavBlue to a broad range of hardware. The current implementation has been ported to common development platforms (such as Windows) as well as embedded platforms (such as TINI and SNAP).

The MavBlue Development Environment utilizes a multi-pane user interface to provide the developer with a thorough view of the Bluetooth system. A graphical view of the Bluetooth piconet is presented in order to provide the developer with a visual map of the devices within the Bluetooth piconet and the devices that have not yet joined the piconet. In addition to the graphical view, the user interface also provides a tree view of the devices. The tree view provides a hierarchical view of the devices and the services offered by the devices. Finally, tabbed panels are used to provide detailed profiles of all devices within the development environment. The tabbed panels allow the user to query and modify the Bluetooth devices within the system.

This paper details the current design and development of the user interface for the MavBlue Development Environment. The user interface has incorporated Swing for managing the graphical user interface and XML for managing the configuration database. Swing and XML are well supported within the Java community and they both facilitate portability of the user interface. Despite these benefits, Swing and XML also created unique obstacles during the development of the user interface.

Introduction

MavBlue is a Bluetooth development environment developed at Minnesota State University Mankato to facilitate undergraduate and graduate research and education activities. The current design and development of the user interface for the MavBlue user interface utilizes Java, Swing, and XML in order to facilitate portability of the user interface. This paper discusses the current design and implementation of the user interface. The paper begins with an overview of Bluetooth and then discusses the core technology components adopted by the user interface. The paper then provides an overview of the MavBlue software architecture and the design impacts of the architecture.

Bluetooth Protocol Stack Overview

Volume 1 of the *Specifications of the Bluetooth System* specifies the protocol stack of Bluetooth, which is shown in Figure 1 [1]. The layers of this protocol stack can be summarized as follows:

- The RF layer, specifying the radio parameters, most closely maps to the physical layers of the International Standards Organization (ISO) model and the IEEE 802 standards.

- The baseband layer, specifying the lower-level operations at the bit and packet levels, most closely maps to the medium access layer of the IEEE 802 standards or the lowest levels of the data link layer of the ISO model. The baseband layer is responsible for forward error correction operations, encryption, circular redundancy check (CRC) calculations, and the automatic-repeat-request (ARQ) protocol.

- The link manager layer most closely maps to the middle levels of the data link layer of the ISO model. The link manager also maps to the upper levels of the medium access layer and the lower levels of the logic link layer when compared to the IEEE 802 standards. The link manager is responsible for specifying connection establishment and release, authentication, connection and release of synchronous connection-oriented (SCO) and asynchronous connectionless (ACL) channels, traffic scheduling, link supervision, and power management tasks.

- The logical link control and adaptation protocol (L2CAP) layer maps to the logical link layer of the IEEE 802 standards. Similarly, the L2CAP corresponds to the service access points of the ISO model. This layer forms an interface between standard data transport protocols and the Bluetooth protocol.

Above the L2CAP layer are a variety of protocols that most closely map to presentation and application layer protocols in the ISO model. For example, the RFCOMM protocol is intended to provide emulation for standard RS-232 cabling whereas the service

discovery protocol (SDP) enables one Bluetooth unit to identify the capabilities of other Bluetooth devices within its transmission range.

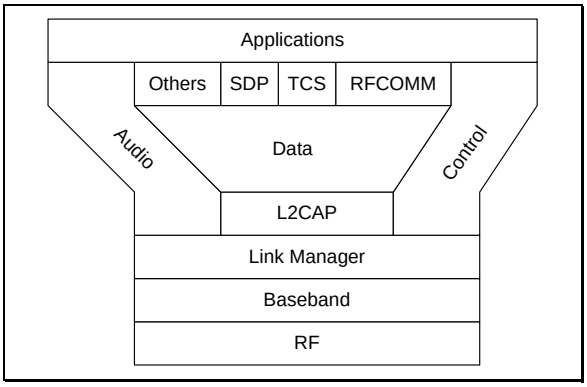

Figure 1. The Bluetooth Protocol Stack

The RF Layer

The RF layer of the Bluetooth protocol specification provides for the physical transmission of bits. This is the lowest layer of the protocol stack and, consequently, provides the least interest to the research detailed in this paper. Nevertheless, some details of the RF layer are worthy of discussion; particularly since the design of the RF layer imposes certain data rate limitations on the use of Bluetooth technology.

The goal of Bluetooth is to devise a ubiquitous, ad hoc radio system. Therefore, the choice of RF spectrum to use is constrained by government regulations on the use of the RF spectrum. In order for Bluetooth to become ubiquitous, it must incorporate a radio solution that does not require any special licensing. Therefore, the radio must operate in an unlicensed spectrum, typically referred to as the Industrial, Scientific, Medical (ISM) band. The Bluetooth RF layer operates in the ISM band located at 2.5 GHz.

The selection of the 2.4 GHz ISM band provides approximately 80 MHz of bandwidth for the RF layer. As with any data network, it is necessary to provide multiple users with concurrent access to this physical media. The Bluetooth solution is to divide the spectrum into separate RF channels, each with a 1-MHz allocation, and to use frequency hopping within the available channels.

The Baseband Layer

The baseband protocol is responsible for establishing the physical link between two Bluetooth radios. The Bluetooth standard partitions radios as masters and slaves. As such, the baseband layer provides a variety of services, including connection and connectionless services, error detection and correction, flow control, hop management,

address management, encryption, and authentication. Clearly, the baseband layer provides many of the same capabilities as the data link layer in the ISO protocol stack and the same services as the media access control (MAC) layer in the IEEE 802 standards.

The Bluetooth protocol at the baseband layer uses a combination of frequency modulation and time division modulation. To begin with, the radio spectrum is partitioned into 79 or 23 RF channels, depending on the national licensing issues. The system then uses a pseudo-random frequency hopping sequence to transmit data using the available physical channels. The hopping sequence allows logical channels to be constructed from the physical channels, with each logical channel using a unique hopping sequence. Each logical channel has one master radio and one or more slave radios. The master radio's clock and address are used to uniquely select a hopping sequence. Each logical channel is called a *piconet*. Multiple piconets with overlapping coverage areas form a *scatternet*. Figure 2 provides an illustration of possible example configurations of master and slave radios into a piconet and a scatternet. In the figure, the three notebook computers have master radios and the phone and printers have slave radios. Three piconets exist. However, two of the piconets have overlapping coverage areas. The printer that exists within the overlapping coverage enables two of the piconets to combine and form a scatternet.

It is important to realize that a radio cannot serve as the master in more than one piconet as there would be no way to create a unique hopping sequence for each of the piconets.

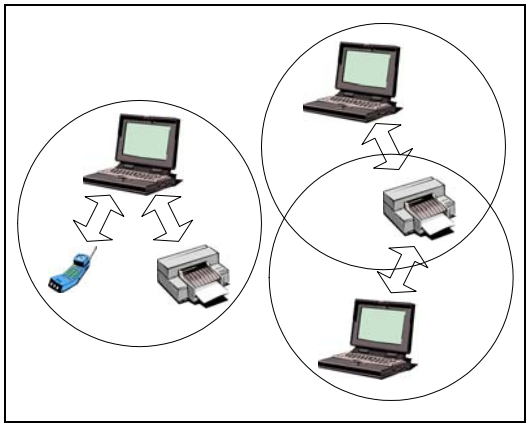

Figure 2. Bluetooth Piconet (left) and Scatternet (right) Configurations

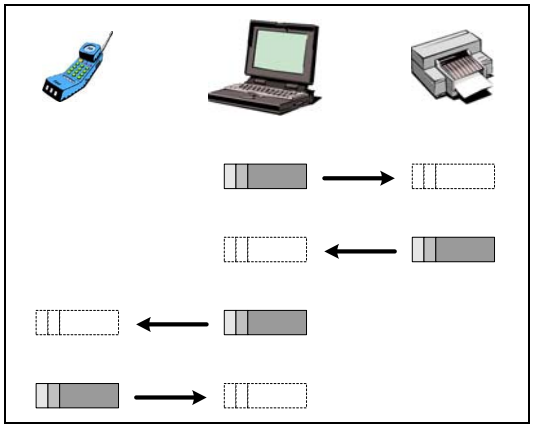

Figure 3. Bluetooth's use of Time Division Duplex and Its Timing

Within each logical channel, time division multiplexing is used as each channel is divided into time slots of 625-μs duration. Time slots are numbered using a 27-bit sequence number, thus providing a cycle length of 2^{27}. Finally, alternating transmission direction supports duplex operation. The master transmits to one (or more) slave radios on the even-numbered slots and then the slave addressed during that slot can transmit to the master on the next slot. Finally, transmission is packet-based with packet lengths of up to five time slots.

The concept of time division duplex and its timing is illustrated in Figure 3. In the illustration, a master radio (the notebook computer) communicates to two slave radios within its piconet. Four time slots are illustrated. The first two slots allow the master radio to communicate to the slave radio in the printer. In the third and fourth time slots, the master radio communicates with the slave radio in the wireless phone.

The Link Manager Layer

The link manager layer implements the Bluetooth link manager protocol (LMP), the next level up from the baseband protocol. The link manager layer is used for link set-up, security and control. Thus, the link manager creates connections between the master and slave. In addition, the link manager is responsible for negotiating parameters for data encryption. The Bluetooth protocols allow negotiation of encryption keys, key size, and the dynamic enabling and disabling of data encryption, polling intervals, and power management parameters.

Other than handling link set-up, perhaps the most critical responsibility of the link manager layer is to manage the various states in which each radio can operate. Many of the operational states are intended to provide extremely low-power modes of operation in order to maximize battery life for mobile devices.

The initial (or default) state of a Bluetooth radio on power-up is *standby*. This state operates the radio in low-power mode with only the native clock running. In the *connected* state, a connection has been established and packets may be exchanged between the master and the slave radios. To transition to a *connected* state from a *standby* state, the radio must follow certain procedures as defined in the substates *page*, *page scan*, *inquiry*, and *inquiry scan* [8][9].

The Logical Link Control and Adaptation (L2CAP) Layer

The L2CAP layer most closely resembles the data link layer of the OSI seven layer model. The L2CAP layer is responsible for protocol multiplexing, segmentation and reassembly of application level packets, and provides additional QoS capabilities.

The baseband protocol allows for the reliable delivery of up to 2,745 bits of user data; just 383 bytes of data. The L2CAP layer provides for reliable delivery of application packets up to 64KB in size. Clearly, in order to do so, the L2CAP layer must manage the segmentation and reassembly of these large packets into 383 byte packets.

In addition to segmentation and reassembly of application level packets, L2CAP provides support for another layer of logical channels within the piconet's logical channel. The L2CAP logical channels are analogous to sockets within the Internet's transport layer.

Finally, the L2CAP layer provides additional QoS support such as negotiation of flow specification (similar to that specified in RFC 1363), which is exchanged with the remote device during channel configuration.

MavBlue User Interface

This section of the paper focuses on the design and implementation of the MavBlue user interface. The section begins with a brief discussion of the key technologies utilized by the MavBlue implementation. The section then discusses the current design of the user interface implementation by providing an overview of the key object classes comprising the architecture of the user interface implementation.

The design of the MavBlue user interface is influenced by the decision to utilize three key technologies in the development of the user interface: Java, Swing, and XML. Java was selected as the programming language due to its support for software portability. Portability was also the primary consideration when selecting Swing was selected for the graphical user interface. Similarly, XML is used for persistent data storage in order to ensure that the data store is also portable to a wide variety of target devices.

Java and JSR-82

Our implementation of the MavBlue user interface and the Bluetooth protocol stack has been developed in Java. The decision to use Java was based on two primary considerations, portability and ease of learning. The introductory programming courses at Minnesota State University Mankato are Java based. Therefore, by implementing th MavBlue environment in Java, undergraduate majors are able to become involved in the research once they have completed their CS1 and CS2 courses. In addition, this approach allows networking of a collection of heterogeneous nodes without undue effort to port the software to those heterogeneous environments.

The implementation requires host platforms to support either J2ME or J2SE with javax.com. The primary development is done using J2SE on Windows-based workstations. However, to the extent possible, the software is also ported to the Imsys platform, which is based on J2ME. Details on J2ME are available in [7] and [5].

When the development of the MavBlue project was initiated, no standard Bluetooth APIs existed. The MavBlue project proceeded to create a project-unique API for each of the Bluetooth protocols. Each protocol was implemented as a package, all contained within the edu.mnsu.bluetooth domain. For example, the Bluetooth L2CAP protocol was implemented as the edu.mnsu.bluetooth.l2cap package.

Since the start of the MavBlue project, a standard Java API for Bluetooth has been developed. The standard was developed by the Java Community Process (JCP) and is available as JSR-82. The MavBlue project is now in the process of revising the project's implementations in order to be consistent with JSR-82. However, it is significant to note that JSR-82 was developed for J2ME, not for J2SE. As such, JSR-82 is heavily based on the generic communication framework (GCF) used by J2ME to perform input and output. Unfortunately, the J2SE environment does not support GCF. As a result, Bluetooth channels within the MavBlue environment are encapsulated as a MavBlue specific class rather than using the JSR-82 model. Details on JSR-82 are available in [4] and [3].

Swing

Development of the graphical user interface for MavBlue required a selection of available graphical components. The MavBlue project elected to use the Swing components. The primary reason for opting to use Swing was to maintain the portability of the MavBlue tools. Swing is written entirely in Java and, therefore, provides for greater portability since it does not rely on an underlying graphics subsystem. In addition, Swing makes more efficient use of memory, which is critical to MavBlue in order to support embedded platforms such as the TINI and SNAP modules. Details on Swing are available in [6] and [2].

XML

The MavBlue user interface requires access to persistent data storage. This storage is used to retain configuration parameters (such as the port number for the Bluetooth radio) as well as service discovery information. For example, the JSR-82 Bluetooth Control Center is required to retain information on well-known devices – those devices that are assumed to be within radio range of the local device without actually using the service discovery protocol to locate the devices.

The MavBlue user interface utilizes an XML document stored as a simple file to serve as persistent data storage. This decision was made as XML is supported in both J2SE and J2ME. The Java XML binding provides a means to search the XML document and, thus, allows the XML document to be used as a simple database.

MavBlue User Interface Design

The MavBlue user interface is illustrated in Figure 4. The user interface is divided into three primary regions; a graphical view, a tree view, and a tabbed view. Each region provides a differing perspective on the Bluetooth network. The graphical view provides a visual indication of the devices participating in the same piconet as the local device. In the example, the piconet is currently empty. The tree view provides a view of the local and remote devices. It is the intent of the tree view to list the devices along with the services provided by those devices. However, this functionality is not yet implemented. The tabbed view provides a detailed listing of the local device and all known remote

devices. In addition to providing the user with information pertaining to known devices and the current configuration of the piconet, the MavBlue user interface also provides the functionality of the JSR-82 Bluetooth Control Center.

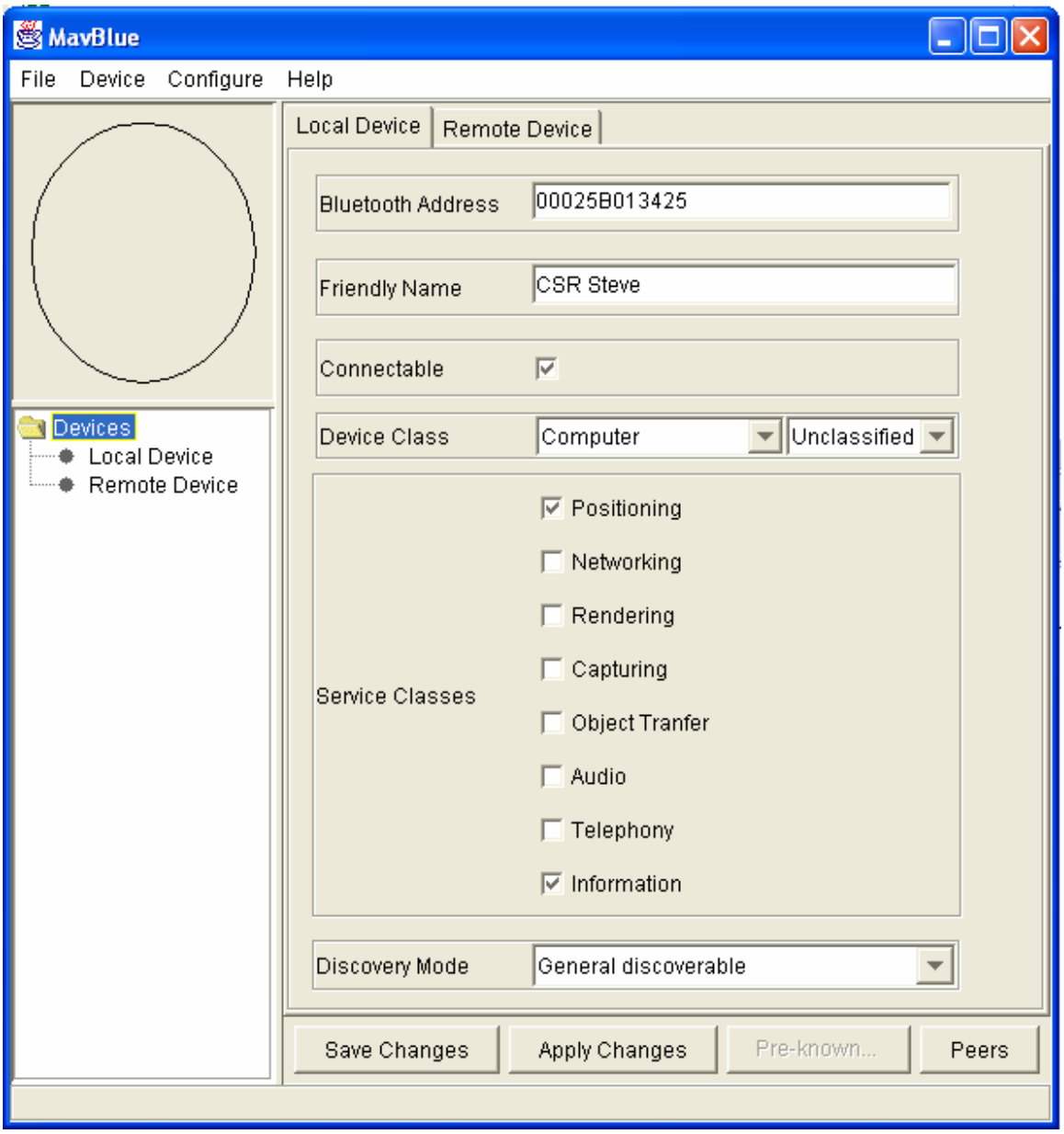

Figure 4. MavBlue Primary User Interface

A UML view of the top-level class architecture of the MavBlue application is illustrated in Figure 5. The MavBlue class integrates the MavBlue architecture into an executable program. The primary role of the MavBlue class is to initialize the Swing and Bluetooth environments. The majority of the remaining functionality is then encapsulated in the MavBlueFrame class.

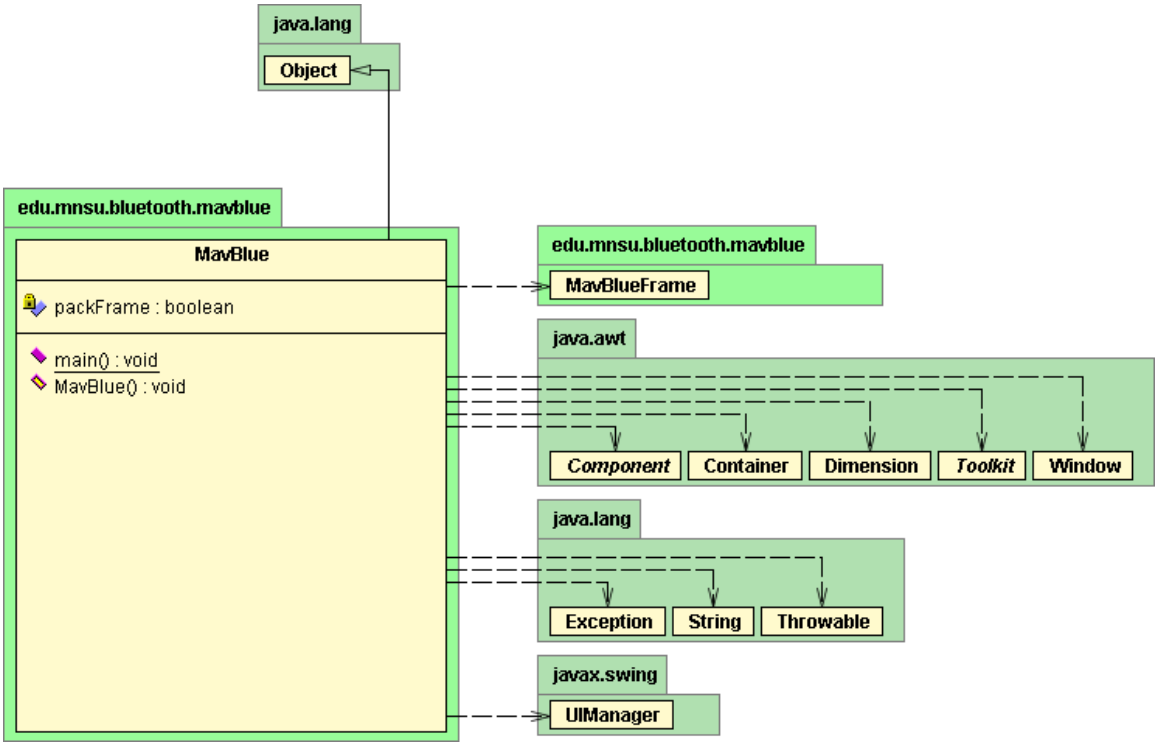

Figure 5. MavBlue Top-Level UML Diagram

The design of the MavBlueFrame class is illustrated in Figure 6. This UML diagram illustrates that the MavBlueFrame is organized into three primary regions that are, in turn, implemented by the three classes of PiconetPanel, DeviceTreePanel, and DevicesPanel. These three classes encapsulate the user interface regions discussed earlier.

The MavBlueFrame class also contains the initial bindings to the Bluetooth stack by creating an L2CAP channel object. The L2CAP channel object is actually an artifact of our early implementations of the stack. Originally, the Bluetooth stack implementation required all upper-level services to pass through the L2CAP layer. This implementation decision provided a centralized API for application developers but resulted in too many static methods. The static methods existed solely for the purpose of serving as a pass-through to lower-level services, such as obtaining the Bluetooth device address of the local device. This portion of the design is currently in transition as we migrate the architecture to be consistent with the JSR-82 specification. The majority of these static methods become methods of the LocalDevice class of JSR-82.

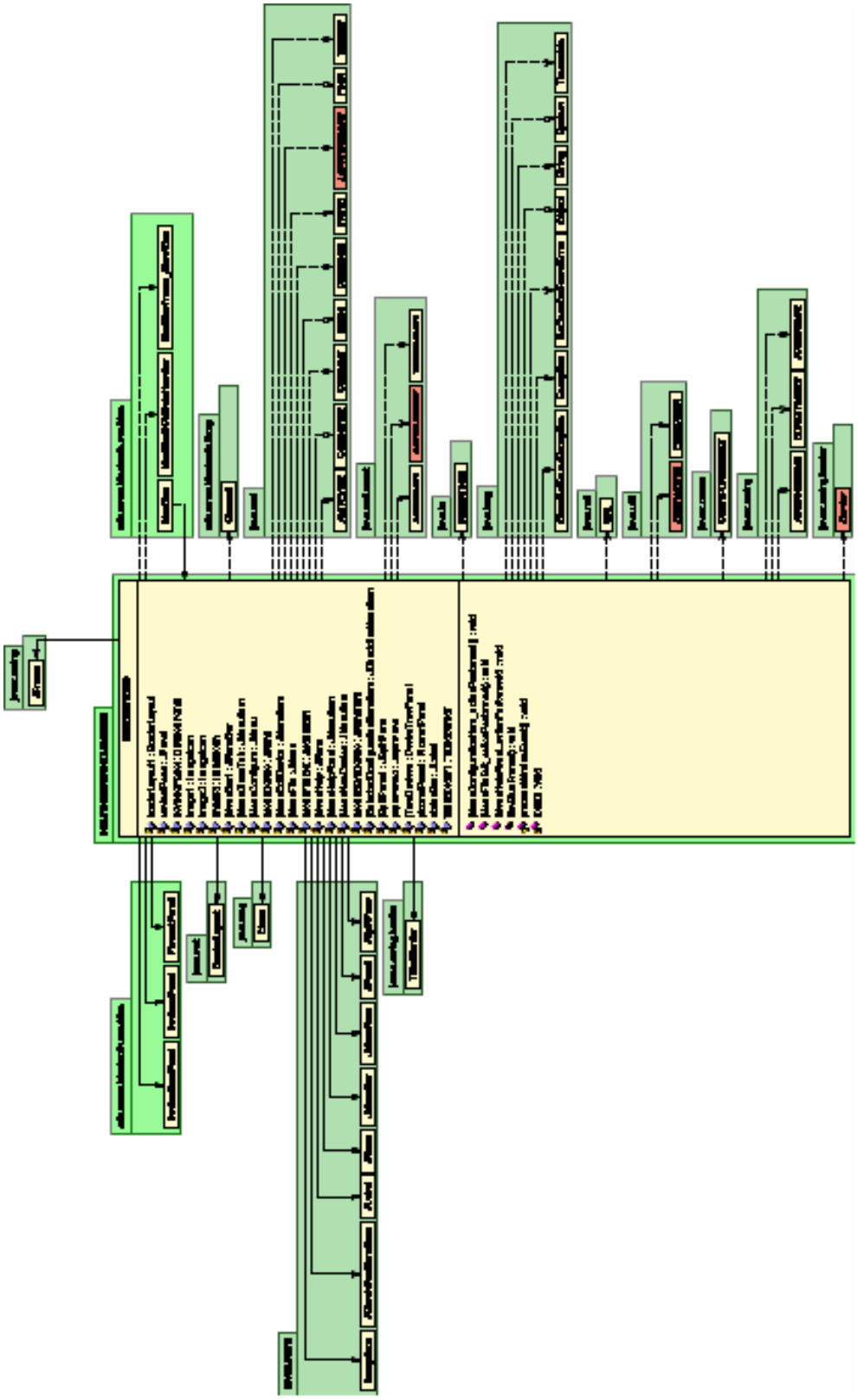

Figure 6. MavBlueFrame UML Diagram

The design of the DevicesPanel class is illustrated in Figure 7. The DevicesPanel provides the user with details on known devices, as well as the services available from those devices. As such, the DevicesPanel interfaces with the Bluetooth service discovery protocol in order to discover devices and services. The DevicesPanel would also interface with Swing in order to render the device attributes in the tabbed panel of the user interface. The actual design of the MavBlue user interface decomposes this functionality into a DevicePanel object, which provides the capability to render the device's attributes as a frame within the user interface.

The DevicesPanel class contains a DicoveryListener class. The DiscoveryListener class serves as the interface between the MavBlue user interface and the service discovery protocol of the Bluetooth protocol stack. The key elements of this class are the interface methods of deviceDiscovered(), inquiryCompleted(), servicesDiscovered(), and serviceSearchCompleted(). These methods serve as callbacks from the service discovery protocol in order to notify the application of the discovery of new devices, the completion of a device search, the discovery of new services, and the completion of a service search, respectively.

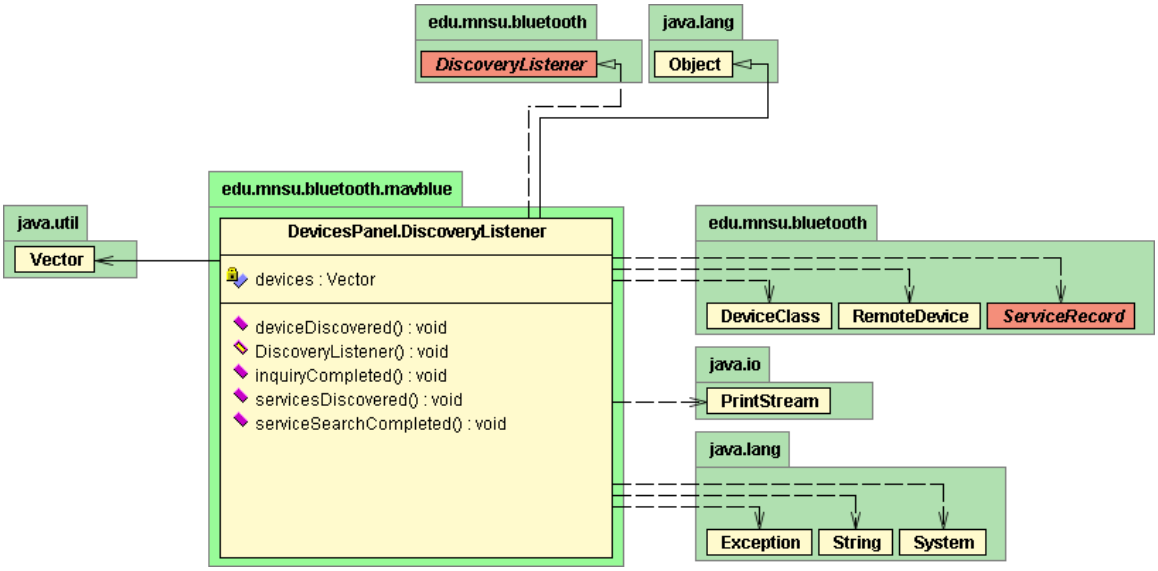

Figure 7. DevicesPanel UML Diagram

Future Enhancements

The MavBlue user interface provides students with a functional interface to explore a Bluetooth environment. In order to be of greater benefit to future researchers, the MavBlue user interface should be extended with the following capabilities:

- Dynamic invocation of applications from within the MavBlue environment. The MavBlue environment should allow the developer to run Bluetooth applications from within the MavBlue environment. The applications would be invoked on the local host computer and becomes services available from the local device.

- Simulated radios using multicast sockets. In many research activities, it is necessary to simulate the existence of additional devices. Copies of the MavBlue environment could be initiated on remote workstations that are networked on a standard TCP/IP network. The Bluetooth protocol stack can simulate Bluetooth packet communication by transmitting baseband packets as UDP/IP multicast messages.

- The current MavBlue user interface and MavBlue environment are limited to the HCI, L2CAP, and SDP protocols. The MavBlue environment should be extended to support all of the Bluetooth protocols. This will enable the user interface to automatically configure the local device to a specific Bluetooth profile.

- The current MavBlue implementation uses the standard Java XML bindings to access the persistent storage. The Java XML binding is specific to J2SE and J2EE, but not supported for J2ME as the binding is too memory intensive for J2ME devices. The MavBlue user interface should be modified to use a less resource intensive XML binding such as MinML.

Conclusion

MavBlue is a Bluetooth development environment developed to facilitate undergraduate and graduate research and education activities. The MavBlue user interface has been developed in Java in order to facilitate portability of MavBlue to a broad range of hardware. The current implementation has been ported to common development platforms (such as Windows) as well as embedded platforms (such as TINI and SNAP).

The MavBlue user interface utilizes a multi-pane user interface to provide the developer with a thorough view of the Bluetooth system. A graphical view of the Bluetooth piconet is presented in order to provide the developer with a visual map of the devices within the Bluetooth piconet and the devices that have not yet joined the piconet. The user interface also provides a tree view of the known devices and the services offered by the devices. Tabbed panels are used to provide detailed profiles of all devices within the development environment.

The current design and development of the user interface for the MavBlue user interface utilizes Java, Swing, and XML in order to facilitate portability of the user interface. Despite these benefits, Swing and XML also created unique obstacles during the development of the user interface. The Swing and XML bindings are relatively memory intensive and may not be suitable for embedded devices. Consequently, it may be necessary for a future implementation of the MavBlue user interface to adopt alternative solutions for the graphical user interface and persistent data storage.

References

[1] Bluetooth SIG. (2001) “Specification of the Bluetooth System – Version 1.1”, Volumes 1 & 2. February 2001.

[2] D. Geary. (1999) “Graphic Java 2, Volume 2: Swing (3rd edition).” Prentice Hall PTR, 1999.

[3] B. Hopkins, & R. Antony. (2003) “Bluetooth for Java.” New York: Springer-Verlag, 2003.

[4] C. Kumar, P. Kline, & T. Thompson. (2004) “Bluetooth Application Programming with the Java APIs.” San Francisco: Morgan Kaufmann Publishers, 2004.

[5] R. Riggs, A. Taivalsaari, & M. VandenBrink. (2001). “Programming Wireless Devices with the Java 2 Platform, Micro Edition.” Boston: Addison-Wesley, 2001.

[6] M. Robinson, & P. Vorobiev. (2003) “Swing, second edition.” Greenwich, CT: Manning Publications Co., 2003.

[7] K. Topley. (2002) “J2ME in a Nutshell.” Sebastopol: O’Reilly, 2002.

[8] J. Y. Wilson, J. A. Kronz. (2000) “Inside Bluetooth Part I”. Dr. Dobb’s Journal. 25(3). pp. 62-70, 2000.

[9] J. Y. Wilson, J. A. Kronz. “Inside Bluetooth Part II”. Dr. Dobb’s Journal. 25(4). pp. 58-64, 2000.