Supporting Real-Time Asynchronous Communication within Bluetooth Ad Hoc Networks

Steven Case
Department of Computer and Information Sciences
Minnesota State University, Mankato
steven.case@mnsu.edu
(507) 389-5310

Abstract

The Bluetooth protocols have been developed to support wireless transmission within personal area networks; primarily serving as a cable replacement technology for mobile devices. The standard protocol provides support for synchronous transmission of audio and asynchronous transmission of data. In general, the standard was developed to use synchronous connection-oriented (SCO) channels to transmit real-time audio communication and asynchronous connectionless (ACL) channels to transmit non-real-time data communication. No preexisting support exists for the transmission of real-time, asynchronous data communication.

Real-time systems that require asynchronous communication are characterized by the need to share data, alarm and event information between various sensors and processing units with the expectation that information is communicated within well-defined temporal constraints. In such an environment, both measurement and control can be achieved.

Unfortunately, the current Bluetooth specification provides no guarantees for timely delivery of asynchronous data transfers. However, the Bluetooth communication protocols are based on time-division multiplexing with all communication coordinated through a master radio. As such, the Bluetooth protocols are somewhat characteristic of token bus networks. Token bus networks have been well suited for many real-time distributed systems. Thus, there is the implication that Bluetooth should be capable of supporting asynchronous real-time communication.

This paper introduces research to model and adapt the Bluetooth technology so that it can provide for the deterministic transmission of real-time, asynchronous data within a Bluetooth piconet; hence supporting the transmission of real-time data within low-power, limited-range distributed systems. This approach is well suited to the transmission of real-time sensor data within various mobile and portable control systems. The paper documents the motivation, preliminary analysis, and initial results of an on-going research project supporting the viability of the use of Bluetooth within “hard” real-time and “soft” real-time control systems.

Introduction

Minnesota State University Mankato has taken significant steps to incorporate wireless networking into the educational experience. The campus has initiatives in place to bring a complimentary range of wireless technologies to the campus. Current infrastructure in place at MSU includes WAP, SMS, IEEE 802.11b, and LMDS technology. These technologies provide the campus community with complementary wireless networking solutions for the WAN (wide area), MAN (metropolitan area), and LAN (local area) domains.

A primary motivation for the research identified in this paper is to develop tools and support that can address wireless networking from the PAN (personal area) perspective and, thus, complete the spectrum of wireless solutions available to the University. As identified by Held, this collection of wireless protocols provides reasonably complete coverage for the leading wireless data communication standards emerging and evolving today [6]. In addition, expanding the wireless initiative to include personal area network allows for research into the "last meter" problem, which has been identified by the Defense Advanced Research Project Agency (DARPA) as one of the most compelling challenges of the next decade [8].

For the purpose of this research, Personal Area Networks (PANs) are networks focused on connectivity for the personal use of a single individual. Consequently, PAN solutions are focused more on interconnection of personal devices than on sharing resources, which has been the traditional motivation for other network technologies. Today, the most prominent standard in development that can address the unique requirements of Personal Area Networks is the Bluetooth standard.

The remainder of this paper provides an overview of the Bluetooth protocol stack; preliminary analysis of Bluetooth's ability to support real-time, asynchronous communication; and a brief discussion on the future direction and objectives of the research.

Overview of the Bluetooth Protocol Stack

Volume 1 of the *Specifications of the Bluetooth System* specifies the protocol stack of Bluetooth, which is shown in Figure 1 [3]. 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.

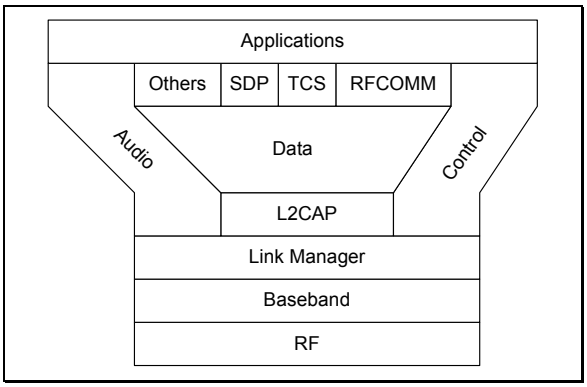

Figure 1. The Bluetooth Protocol Stack

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.

Further constraining the RF layer, the radio must be able to operate with extremely low power. This requirement is driven by two factors. Government regulations restrict the amount of energy broadcast by a transmitter in the ISM band and users of Bluetooth will be energy limited. In other words, mobile users are inherently operating off of batteries. The more power used to transmit, the more energy required, and thus the shorter the lifetime of a battery.

Finally, the design of the RF layer must be usable worldwide. This implies that the ISM band selected for the RF layer must be globally available. Once all of these requirements are combined, it quickly becomes clear that the Bluetooth RF layer must operate in the 2.4 GHz spectrum.

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, 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 frequency hopping is used to combat interference and fading within the 2.4 GHz spectrum. Since the spectrum is unlicensed, such interference is to be expected. The 2.4 GHz spectrum is also used by the IEEE 802.11 wireless LAN standard, by the DECT and HomeRF standards, and even by emissions from microwave ovens! Due to the various government restrictions on the 2.4 GHz ISM band, the number of RF channels available varies from one country to another. In the United States, a total of 79 channels are available, whereas in Japan and some European countries only 23 channels are available.

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.

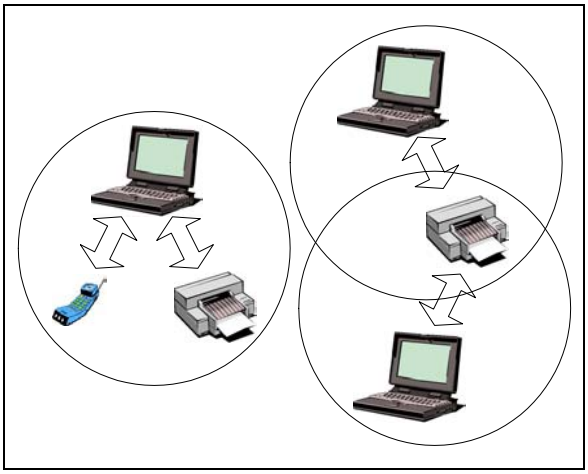

Figure 2. Bluetooth Piconet and Scatternet Configurations

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.

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.

As with many communication networks, the Bluetooth system provides two types of services, connection-oriented and connectionless. However, the connection-oriented and connectionless services at the baseband layer of Bluetooth are rather unique. All connection-oriented services are synchronous and unacknowledged. Therefore, connection-oriented services in Bluetooth do not necessary provide reliable data transfer. Rather, they result in the reservation of specific slots in the channel and thus provide a synchronous delivery of data and a guaranteed bandwidth. In order to ensure that SCO links due not reserve too much of the available bandwidth, a Bluetooth master is restricted to a total of three active SCO links. Bluetooth refers to this service as Synchronous Connection-Oriented (SCO) links.

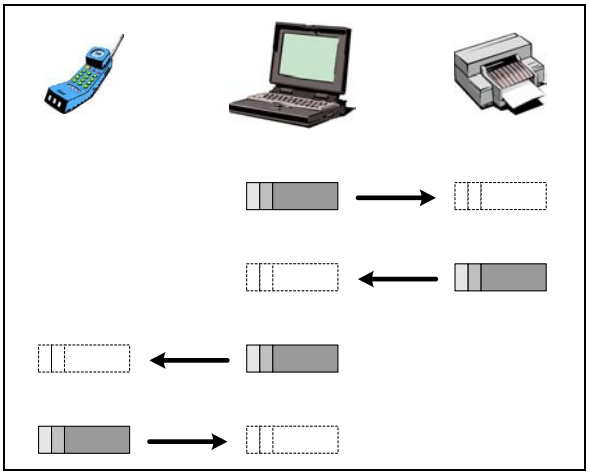

Figure 3. Bluetooth's use of Time Division Duplex and Its Timing

The alternative to SCO links in Bluetooth is the Asynchronous Connection-Less (ACL) link. ACL links do not provide guaranteed bandwidth but do provide for guaranteed delivery using error detection and retransmission. In addition, ACL links support multicast messages whereas SCO links are strictly point-to-point.

The baseband layer of Bluetooth is a packet-based protocol with the standard packet structure consisting of a 72-bit access code, a 54-bit header, and a variable length payload. The actual payload can vary between 0 and 2745 bits in length.

The Bluetooth packet's access code consists of a preamble, synch word, and trailer. The critical element being the synch word as it is used to identify the master radio (and thus the piconet) associated with the packet. As a result, the access code is used to support signaling between radios.

The Bluetooth packet's header consists of a 3-bit active member address, a 4-bit type code, a 1-bit flag for flow control, a 1-bit acknowledgement indicator, a 1-bit sequence number, and an 8-bit header error check. These fields account for 18 bits of the header. However, the header is encoded using a forward error correction code that expands the 18 bits into a total of 54 bits.

The Bluetooth packet's payload consists of five fields. A 2-bit logical channel, a single bit for flow control on the logical channel, a length field, the actual payload data, and a 16-bit CRC checksum. The CRC checksum is calculated using a generator polynomial of 0100010000001000012.

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 [9,10].

A slave radio that is connected has several other operational states that it can use to reduce power consumption when the radio would otherwise be inactive. These operational modes of the connected state include the active mode, sniff mode, hold mode, and park mode.

The active mode is used when a master or slave is actively participating in the piconet. This mode does not provide significant opportunities for power management. However, it is not without some power management features. Specifically, an active mode slave radio can sleep during time slots when it knows it cannot be active. For example, if a slave does not receive a packet during the master-to-slave slot, then the slave can sleep during the associated slave-to-master slot.

The sniff mode allows slave radios to suspend during certain master-to-slave slots. During link set-up, the slave radio can negotiate time slots when it is expected to receive packets and, similarly, time slots when it is expected not to receive packets. Therefore, a slave radio that has performed such a negotiation can then suspend during time slots when it has negotiated to be idle. During these suspend states, the slave is said to be in sniff mode.

Hold mode is a mode that the slave radio can request to enter when it knows that it is going to be quiet for an extended period of time. The value of hold mode (compared to simply

disconnecting from the piconet) is that the slave radio retains its active mode address within the piconet. Since each piconet can have no more than seven addressable slave radios, there are just seven such active mode addresses – clearly a valuable commodity.

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.

Analysis of Bluetooth Protocol Performance

This section provides a brief analysis of the Bluetooth protocol's theoretical potential to carry real-time control messages. The analysis primarily focuses on the baseband layer as the baseband layer creates the most significant limitations as well as the most significant potential for real-time control. As such, the analysis has not yet addressed the effects of the link manager layer's power management support on real-time response, the link manager's support for dynamically establishing ad hoc connections, the L2CAP layer's support for message segmentation and reassembly, or the L2CAP layer's support for quality of service.

Bray [4] and Morrow [7] have written reasonably comprehensive books on the operation of the Bluetooth protocols and Bluetooth devices. Included in both books are discussion of Bluetooth operation and an explanation of the underlying radio and baseband technology. Both references also include a limited analysis of Bluetooth performance. The analysis, however, focuses on maximum throughput of asynchronous channels.

Real-time control systems rely on the deterministic behavior of the hardware and software, enabling the developer to determine if the system will be able to meet all of its real-time deadlines. Within the communication system, the ability to meet real-time deadlines is primarily determined by the ability to manage message latency.

The time division duplex, master-slave operation of the baseband layer results in a communication protocol that has significant similarities to the deterministic behavior of token-based networks. It is well understood how to apply token-based solutions to real-time, asynchronous communication [5]. It is the observation of these similarities that leads to the expectation that Bluetooth can be applied to real-time control systems.

The ideal Bluetooth scenario for real-time asynchronous communication within a piconet occurs when the piconet consists of exactly one master and one slave device. Additionally, no voice communication (i.e. no SCO channels) is utilized within the piconet. In this scenario, the entirety of the piconet's data throughput is available for real-time, asynchronous traffic.

Within this ideal scenario, assuming no overhead for higher-level protocols, control messages can be transmitted with a maximum latency of 1250 μsec. This maximum latency is due to the time division duplex operation of the baseband layer. Since each baseband packet is transmitted in a 625 μsec time slot and packets alternate between master-to-slave and slave-to-master communication, a packet can be delayed for up to two consecutive 625 μsec intervals.

Continuing to assume that the piconet carries no voice communication and no overhead for higher-level protocols, the worst-case scenario occurs when the piconet is fully populated and each master-to-slave or slave-to-master transmission utilizes asymmetric packets requiring five time slots (plus one to enforce full duplex operation). In this scenario, the maximum packet latency will be extended to 26,250 μsec.

However, these scenarios cannot be maintained in the presence of voice communication. Certain elements of the Bluetooth protocols inherently restrict the system from use for real-time activities other than voice communication. The synchronous channels within Bluetooth were intended for such purposes. As such, the channels are inherently restricted to voice-communication unless a system designer was to attempt to encode control data into a voice stream. Clearly a solution that is undesirable.

The situation is actually worse. Depending upon the specific SCO link utilized to carry the voice traffic, it may be impossible to carry any other data within a Bluetooth piconet. The baseband specification provides an HV1 packet that uses a 1/3 FEC code in order to allow for voice traffic to be transmitted on channels that must contend with significant noise. Unfortunately, when an HV1 SCO connection is present, all data transfer sessions must be suspended [1]. Thus, it is impossible to use Bluetooth to carry real-time control message concurrently with an HV1 SCO connection.

This does not imply that Bluetooth is incapable of carrying non-voice traffic concurrently with voice-traffic. In addition to the HV1 packet, the baseband protocol defines an HV2 and an HV3 packet. The fundamental difference between these packets is the voice payload carried within each baseband packet. An HV1 packet can carry 10 bytes of voice data per packet. An HV2 packet carries 20 bytes, and an HV3 packet carries 30 bytes. Basically,

these packet formats trade-off FEC data for voice data. The HV2 format uses a 2/3 FEC code and the HV3 format does not use FEC.

A typical voice channel using HV2 packets must reserve every fourth slot of the baseband communication. Since the master is limited to three concurrent SCO links, this suggests that three of every four packets could be reserved for HV2-based SCO links, leaving one of every four packets for asynchronous communication.

Similarly, a typical voice channel using HV3 packets will reserve one of every six packets. Assuming three concurrent SCO links, this will still leave 50% of the baseband throughput available for asynchronous communication.

Clearly, the presence of voice communication within the piconet will have an adverse effect on the latency of real-time, asynchronous communication. In the simplified case of a piconet with one master and one slave device, the worse case latency becomes 2,500 μsec when one SCO link exists. This latency occurs if the real-time packet becomes available during the time slot immediately preceding the reserved slot for the voice channel. Similarly, the latency becomes 3,750 μsec when two SCO links exist, and 5,000 μsec when all three SCO links are utilized. These worst-case scenarios are independent of whether the voice channel uses HV2 or HV3 packets. As already discussed, no support exists when HV1 packets are used for the voice communication.

In the more complex case of a piconet with seven slave devices, the worst-case latency becomes 35,000 μsec when HV2 packets are used to carry the audio stream. This worst-case scenario is independent of whether one, two, or three SCO channels are in use. Regardless of the number of channels in use, the worse case scenario allows just one slave device to transmit a real-time, asynchronous packet during the interval between SCO reservations. Therefore, a slave device may have to wait four seven intervals of four duplex packets at 1,250 μsec per duplex pair. Thus, as the number of SCO channels increases, the maximum latency is not impacted but the maximum message length is reduced.

Analysis of Bluetooth Protocol States

The previous analysis of Bluetooth's protocol performance assumed the existence of an ACL connection between the master and slave devices within a Bluetooth piconet. When such a connection exists, the devices are considered to be in the "connected" state of the Bluetooth state machine. The Bluetooth protocol does not allow data to be transferred unless the device is in the connected state. Unfortunately, the master radio cannot add additional devices to the piconet without temporarily leaving the connected state.

A thorough analysis of the states defined within the Bluetooth protocol and their effect on real-time performance has not been completed. The current assumption is that any impediments to real-time performance that might be imposed by the state machine can be avoided by proper system design and analysis. After all, it would be possible to design a system to ensure that all dynamic configuration of the piconet is performed at system

initialization and further changes to the configuration are not allowed except in the event of catastrophic failures (i.e. a device within the configuration is no longer functioning). Unfortunately, such a limitation seems undesirable as it minimizes the ability to take advantage of the ad hoc networking capabilities supported by Bluetooth.

In lieu of a thorough analysis, some preliminary issues related to the protocol machine can be identified and discussed.

The fundamental problem related to configuring or reconfiguring the piconet appears to relate to hopping sequences. In the connected state, the piconet cycles through a hopping sequence that is intentionally intended to appear highly random. This random behavior makes it difficult for new devices to become part of the piconet as they are unlikely to be able to identify the correct hopping sequence. The Bluetooth protocol resolves this issue by identifying a set of states related to the *Inquiry* and the *Paging* processes.

The inquiry process allows a Bluetooth device to discover the existence of other Bluetooth devices. The paging process allows a Bluetooth device to invite another (already discovered) device to create a connection and, consequently, join the piconet. In order for these two processes to work, they each have their own well-defined hopping sequence that do not, my design, appear to be random.

The fundamental dilemma for real-time systems is that devices participating in an inquiry or paging activity cannot be in the connected state. As a result, they cannot actively transmit ACL packets. This in turn results in unpredictable delays in transmission, which will only serve to create additional latencies that must be accounted for when evaluating the ability of a Bluetooth piconet to support asynchronous, real-time communication.

Progress Towards Empirical Analysis

An evaluation environment is under development within the wireless lab at Minnesota State University Mankato which will allow for the empirical analysis of real-time, asynchronous traffic within a Bluetooth piconet. The test environment consists of a variety of embedded computer platforms and Bluetooth modules.

The embedded computer platforms used in the wireless lab include TINI modules from Dallas Semiconductor and Cjip and SNAP modules from Imsys. These particular platforms are used as they support development of embedded Java code. Java is gaining momentum as an embedded development language and offers certain efficiencies during the development process due to the ability to initially develop and test software on a typical workstation and then migrate the software to the embedded environment.

The Bluetooth modules include modules from Imsys that use the Ericsson Bluetooth chipset and modules from NSM Technology that use the CSR Bluetooth chipset. All of the Bluetooth modules implement the controller side of the HCI layer and implement the UART interface to the HCI layer.

In order to perform adequate modification and performance analysis of the Bluetooth protocols, full access to the complete Bluetooth stack is required. Commercial implementations of the Bluetooth stack are available but licensing fees for the source code are beyond the reach of the wireless lab. As a result, we have begun development of our own Bluetooth stack.

Our implementation of the Bluetooth stack and all test software is being developed in Java. In addition to the benefits outlined above, 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. Ideally, we would also like access to the Real-Time Specification for Java, but have found limited support at this time for the Java real-time extensions. This real-time specification was selected (rather than more traditional C or C++ run-time environments) in order to maximize portability while still maintaining support for asynchronous event handling, asynchronous control transfer, asynchronous thread termination, and access to physical memory [2].

The intention of the test environment is primarily to validate the predicted analytical results. Once the analytical results are validated, the model will then be used to extend the performance model to account for deterministic delays caused by the run-time environment, protocol stack, and the application software.

Conclusion and Future Work

The preliminary analysis indicates that the Bluetooth protocols have significant potential to support deterministic behavior, thus lending themselves to real-time, asynchronous communication. However, it is clear from the analysis that using the synchronous communication facilities of Bluetooth in order to carry real-time audio will adversely affect the ability to concurrently maintain real-time deadlines for asynchronous events. In addition, it will likely be necessary for real-time, asynchronous systems to turn-off or otherwise not utilize certain features of the Bluetooth standard (such as periodic inquiry or page scans).

An Java implementation of the Bluetooth protocol stack is in development and has been tested in whole or in part within the Windows (2000, XP, and CE), Linux, Imsys, and TINI operating environments. The ease of transition across these operating systems has been a testament to the use of Java and the portability goals that motivated the selection of Java. It is important to note that real-time performance has been limited to the Imsys and TINI implementations.

The integration of this Bluetooth protocol stack has proven most effective at gaining hands-on understanding of the Bluetooth protocols, from the baseband layer through L2CAP. It is expected that the implementation will evolve to provide similar benefits to other educational and research activities relating to personal area networking. Specifically, the following activities are planned:

- Use the implementation to gather empirical evidence to predict performance of both Bluetooth piconets and scatternets.

- Use the implementation to create a real-time implementation of the service discovery protocol and evaluate the effects of piconet and scatternet management on real-time, asynchronous communication.

- Use the implementation to evaluate the relative effects of segmentation and reassembly on real-time performance.

- Use the implementation to evaluate the relative effects of quality of service on real-time performance and propose additional quality of service parameters to enhance support for real-time communication.

- Allow students to make protocol modifications by implementing their own version the class-implementation of a protocol layer.

The present implementation is used to support research in mobile data communications by allowing faculty, graduate students, and undergraduate students to experiment with the Bluetooth protocols within an embedded, real-time environment. Specific applications in which the environment is either currently used or planned for use include weighing systems (aircraft and vehicle), data collection systems (primarily environmental sensors), and medical systems (primarily for patient monitoring).

References

[1] Bhagwat, P. (2001). "Bluetooth: Technology for Short-Range Wireless Apps". IEEE Internet Computing. May-June 2001. pp. 96-103.

[2] Bollella, G., & Gosling, J. (2000). "The Real-Time Specification for Java". Computer. June 2000. pp. 47-54.

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

[4] Bray, J., & Sturman, C. (2002). *Bluetooth 1.1: Connect without Cables*. Upper Saddle River, NJ: Prentice Hall PTR.

[5] Halsall, F. (1996). Data Communications, Computer Networks and Open Systems. Fourth Edition. Addison Wesley. Harlow, England. 1996.

[6] Held, G. (2001). "Data Over Wireless Networks". McGraw-Hill. New York, NY, 2001

[7] Morrow, R. (2002). *Bluetooth Operation and Use*. New York, NY: The McGraw-Hill Companies.

[8] Rabaey, J., Ammer, M., da Silva, J., Patel, D., & Roundy, S. (2000). "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking". Computer. July 2000. pp. 42-48.

[9] Wilson, J., & Kronz, J. (2000). "Inside Bluetooth Part I". Dr. Dobb's Journal. 25(3). pp. 62-70, 2000.

[10] Wilson, J, & Kronz, J. (2000). "Inside Bluetooth Part II". Dr. Dobb's Journal. 25(4). pp. 58-64, 2000.