Designing User-specific Plug-n-Play into Body Area Networks

Ryan A. Danas*
Douglas T. Lally* Nathaniel W. Miller* John S. Synott* Craig A. Shue
Krishna K. Venkatasubramanian†
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA, 01609
{ryandanas, nwmiller, dlally, john, cshue, kven}@wpi.edu

ABSTRACT

A Body Area Network (BAN) consists of a set of sensing devices deployed on a person (user) typically for health monitoring purposes. The BAN continuously monitors various physiological and environmental parameters and typically transfers this information to a base station for processing and storage in a back-end medical cloud. Despite the incredible potential that these systems offer, their utilization is largely limited to lab settings. One of the requirements for adoption in the real-world is the ease of deployment and configuration of such systems for the users. Much work has been done in developing middleware-based solutions that enable easy application development for BANs by abstracting out the details of the devices and sensors. However, none of the current approaches extend this capability to the users of the system. What is required is the ability to provides a means to dynamically add diverse devices in to the system without requiring substantial reprogramming of the device and the base station.

In this paper, we present **BAN-PnP**, a communication protocol for enabling devices and the base station (or middleware) to communicate effectively with minimal user involvement. The key idea of the protocol is to allow the devices in the BAN to "teach" the base station about their capabilities. By adding a few extra control messages, we are able to transform a traditional BAN into a plug-n-play BAN that is easy for the usually non-tech-savvy users of such systems to deploy. The performance analysis of the BAN-PnP protocol demonstrates that the protocol enables plug-n-play operation of BANs with an affordable increase in overhead.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Applications*; C.2.4 [**Computer-Communication Networks**]: Network Operations—*Network management*

*Undergraduate co-student leads listed in alphabetical order; each participated equally
†Point of Contact

1. INTRODUCTION

Emerging Body Area Networks (BANs) have demonstrated great potential in a broad range of applications in healthcare and wellbeing. A BAN consists of a set of monitoring devices deployed on a user, that continuously monitor them and provids this information to a sink entity called the base station for processing, visualization as well as transport to a back-end personal health monitoring system for long-term storage and retrieval. BANs intend to improve health outcomes, decrease isolation, reduce health disparities, and substantially reduce costs. Studies show that BANs can improve patient care and patient safety and result in annual cost savings of up to $81 billion [8].

Despite the incredible potential that these systems offer, their utilization is largely limited to lab settings [2,6,10,13,16] and they require considerable configuration and technical sophistication. This raises the question of barriers to deployment of these platforms in real-life. One of the requirements for real-world adoption is the ease of deployment and configuration of such systems. It is important to note that BANs, when deployed, will be largely used by users who are not necessarily tech-savvy. Therefore, the deployment of the BANs should largely be automatic requiring minimal user involvement. They should be *plug-n-play (much like USB devices of today) from a management and configuration standpoint for the user*. We envision a future where users can go to any store and buy a sensing device and add it to BAN without any additional configuration. Enabling BANs to be plug-n-play will allow the users of such systems to focus on the more important health matters they are trying to monitor or address, rather than on the technical details of the system.

Researchers have tried to address this issue from a system's standpoint by developing a middleware for abstracting out the details of the individual BAN devices from the application layer. This allows application developers to be oblivious of the devices within the network. The middleware manages the low-level details of the devices and their sensors, providing a clean API for the application programmers [17, 18]. This does enable plug-n-play of devices from an application development stand-point. However, the middleware still has to be configured for each type of device that is part of the network, which does not necessarily make the task of the user of the BAN any easier.

In this paper, we present **BAN-PnP**, a communication protocol for enabling devices and the base station (or middleware) to communicate with minimal user involvement in terms of configuration. The key idea of the protocol is to allow the devices being added to the BAN to "teach" the base station about its capabilities. Therefore, when a new device is added to the BAN, the base station can query it for information including the number of sensors it has, the type of values the they return, the units of measurement, and the various commands the devices accept. Once this information is known, the base station can configure the device automatically without the user involvement. The proposed protocol is simple in its implementation and adds very little overhead to the operation of the BAN in terms of throughput, latency and device battery life costs compared to the traditional BANs.

The rest of the paper is organized as follows. Section 2 presents the system model and problem statement. Section 3 presents the BAN-PnP protocol, followed by Section 4, which presents the performance results of the protocol. Section 5 presents the related work and finally, Section 6 concludes the paper.

2. SYSTEM MODEL

A BAN is comprised of a number of wearable devices capturing and collaboratively processing physiological signals on humans. The devices collect health and contextual data at regular intervals and forward it over a *single-hop* network to a highly capable base station node for further processing. The base station may transmit final results to a back-end server for clinical decision making and interventions. A typical device consists of one or more sensing elements, an analog-to-digital converter, a wireless communication stack, processor, and memory. We assume that the devices communicate wirelessly, as wires running between devices in a BAN will make it obtrusive. We refer to the person on whom the BAN is deployed as the *user*. Typically, it is the users who are responsible for managing the BAN and these users might not be tech-savvy.

Given the model of the BANs, we now describe how BANs are assembled and disassembled, i.e., the typical workflow utilized by BANs, followed by a requirements of the problem we are trying to solve.

2.1 BAN Workflow

Our model is a fairly high-level one and primarily deals with (1) device connecting and disconnecting to the base station, (2) devices sending data to the base station, and (3) devices executing commands received from the base station. No additional capabilities are assumed or inferred beyond those described in the model. Any BAN that conforms to this model is guaranteed to be capable of implementing and using our plug and play protocol. The BAN workflow has six main steps:

1. *Base Station Setup:* In this step: (1) the user ensures the base station software (note that this might even be a middleware running on the base station with BAN applications executing with it) is installed on the appropriate device and (2) launches the base station software.

2. *Device Addition:* In this step: (1) the user gathers devices that will be members of the initial BAN configuration, (2) the user introduces the devices into the BAN by switching them on in a discoverable state, (3) the base station scans its environment and discovers the devices in the vicinity and pairs with them at the link-layer[1], and (4) the devices automatically pair with the base station using the underlying link-layer protocol (e.g., Bluetooth, Zigbee).

3. *Activation:* In this step: (1) the base station connects with the devices it has paired with in the previous step thereby being able to send and receive data with them, (2) the base station then automatically recognizes the capabilities of the new device and determines how to deal with the data appropriately, (3) the user can customize the BAN with specific configuration parameters. The BAN is now ready for data streaming. The BAN is called *active* from this point on.

4. *Data Streaming:* In this step the devices in the BAN sample specific stimuli and send data to the base station and beyond.

5. *Device Deletion:* In this step: (1) the user removes one or more existing devices from the BAN at any point in time when the BAN is active (the base station keeps its link-layer pairing with the device for a while, in case the device is reintroduced), and (2) the base station removes all pairing information about the device after a certain duration of time.

6. *Disconnection:* In this step, the base station instructs all the devices to stop sending it data and then destroys all the link-layer pairings between itself and the devices. The intention here is to purge the network completely. The BAN is now said to be *inactive*.

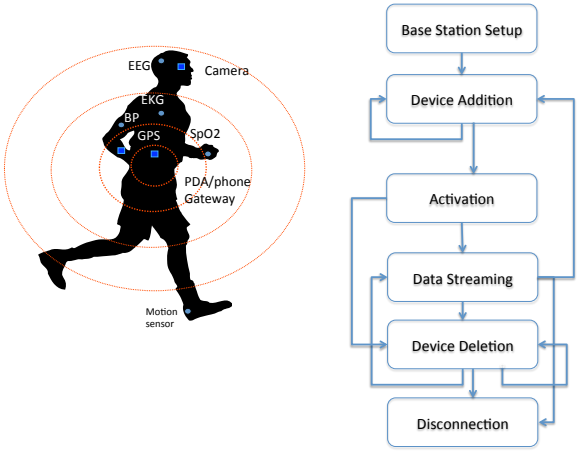

Figure 1: System model and BAN workflow

The user can introduce additional devices at anytime when the BAN is active and the base station will be able to handle it through the steps in the Device Addition and Activation steps. Figure 1 illustrates our system model and the aforementioned workflow.

2.2 Problem Statement

In this paper the goal is to create a protocol that would allow a BAN to be "plug-n-play" where devices can be easily added and

[1]For example, with Bluetooth, the devices will be set to a discoverable mode in which case they announce themselves. The base station will scan for these discoverable devices by scanning for the device announcements.

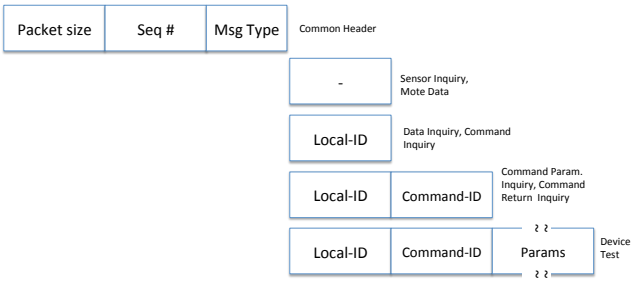

Figure 2: BAN-PnP Protocol Request (base station-node) Messages

removed while minimizing the effort required by the user to reconfigure the BAN. The underlying protocol should be hardware agnostic, requiring only that the device be able to implement this protocol before being added to the BAN. This would minimize the time between the release of a new device and when that device can be integrated in a BAN, while limiting the amount of work required to support the changes. Additionally, the protocol should functional seamlessly in that adding or removing devices from the BAN shouldn't affect existing data streams.

3. PLUG-N-PLAY BODY AREA NETWORKS

The approach we take to enable *plug-n-play (PnP)* in BANs is to make the devices the keeper of functional information within the network. Therefore, information such as what devices are available, what each device is capable of, and what data can be sampled from each device are provided to the base station by the devices. Most current implementations of BANs pre-deploy parts or all this information in the base station, and other parts in the devices [10, 14]. Particular identifiers in messages would specify the functionality to the devices and base station. We have moved all of this functional information to the devices. The device inherently needs to know what sensing elements it has, what they are capable of, and what data it can sample from its sensors. However, now that this information is not assumed by the base station, we have devised a *protocol* for the devices to communicate parts of this information to the base station. In this the rest of the paper we describe our plug-n-play BAN protocol called **BAN-PnP**.

3.1 BAN-PnP Protocol

BAN-PnP can be divided into two categories: (1) learning, and (2) activation. Each of these two categories consist of a range of messages that the base station and the sensing devices can exchange. We describe them next.

Learning: These messages are used by the base-station for learning about the devices being introduced within the system either at the BAN setup time or later when it is active. There are five types of device learning functionalities supported in BAN-PnP. Each of these functionalities has a request and response component to it and a corresponding message in our protocol.

1. *Sensor Inquiry:* Each sensing device may have multiple sensors or sensing elements in it. Therefore it is crucial for the base station to have an understanding of what sensors are available within the BAN being created by the user. The device receiving a sensor inquiry request responds back with: (1) the number of sensors in the device, and (2) the list of sensors in the form of a *local-ID* and a string capturing the *sensor-name*. Note that this *local-ID* is only applicable to this device, and is not global to the BAN. Within the local context of a particular device, *local-ID* 0 is reserved to specify the general device and not a specific sensor on the device. This restriction was made to provide a means with which to handle global actions on a device, for example setting the overall sampling rate of the device.

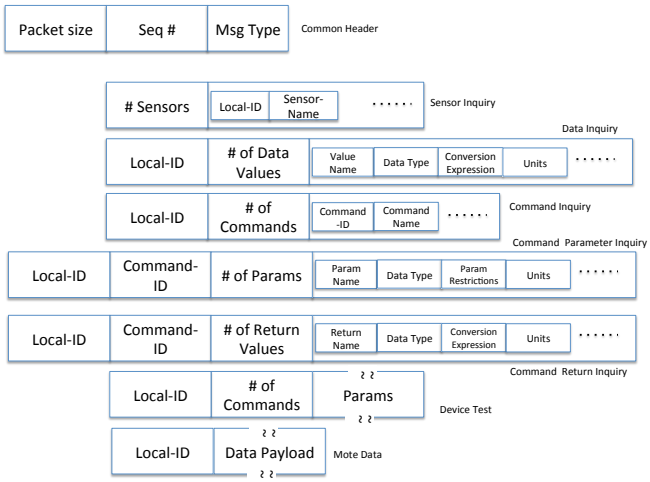

Figure 3: BAN-PnP Protocol Response (node-base station) Messages

2. *Data Inquiry:* The Data Inquiry request message provides the base station with a way to query a device for information pertaining to the data of the device's sensors. The response packet enables the device to specify what type of data a sensor will produce and send to the base station in the form of value mappings. Each value mapping consists of the size, type, name and units of the data sent. Additionally, a raw data conversion expression is also included. This is because most BANs prefer to send raw data to the base station for processing; the base station is usually a more powerful device that can handle complex computations. Conversion equations are specified using ASCII strings and execute and follow the PEMDAS order of operations.

3. *Command Inquiry:* The base station learns about the commands available for the devices and its sensors through use of the Command Inquiry. The response by the device provides the base station with the name of each command and an identifier for the command. To enable command inquiry, we associate a variable called *command-ID* with a particular sensor on the device. A *command-ID* is locally significant to a sensor on a device. As such, different sensors may have the same *command-ID*, but this is not a guarantee that those commands are functionally equivalent.

4. *Command Parameter Inquiry:* The Command Parameters Inquiry request message and the associated response message enable the base station to ask a device for information regarding a particular command. The response by the device

specifies what parameters need to be sent with a particular command, in the form of parameter mappings. Included with each mapping is the parameter name, size, type, and units. In addition, a parameter restriction set is specified. Similarly to the raw data conversion, an expression can be specified that restricts the set of valid values a parameter can have. Any values included in the messages from the devices, must be numerical constants matching the parameter type specified. These constants can be defined as a range: $i - j$, which conveys i to j inclusively. They can also be defined as a set: i, j, k, which conveys that exactly i, j, or k are valid. In our implementation of the protocol, when values are defined as a set, these restriction set values are treated as enumerations by both the base station and devices. For example, a sensor sensitivity restriction set of {5, 10, 15, 25} would be treated as an enumeration and mapped accordingly to values {0, 1, 2, 3}, respectively. Therefore, when the base station wishes to change the sensor sensitivity to 10 it would send the value 1 as the command parameter in the command packet.

5. *Command Return Inquiry:* The Command Returns Inquiry packet specifies the return values the device will respond with when a command is sent from the base station. The definition of this message, both request and response, is similar to that of the data inquiry packet, as its mapping is exactly the same as the data inquiry

Activation: These messages are used by the base-station for activating (or deactivating) the devices to sense their environment. These can be executed only when the learning phase has been executed at least once. There are two types of device activation functionalities supported in BAN-PnP. Each of these functionalities has a request and response component to it.

1. *Device Test:* These messages provide the base station with a means to execute a command it has learned from a device. If a command executed has any associated return values, they will be included in the response from the device. Otherwise, the bare response will be returned. This includes the *local-ID* and *command-ID* as an acknowledgement for the request. It provides a means for performing diagnostic tests on the devices with respect to their capabilities in executing specific learning commands.

2. *Data Request:* These messages are utilized to ask a device within the BAN for sensor data. The request message doubles as both a means to ask a device to send and to stop sending sensor data. If the device is not currently sending data, then the request will initiate streaming of data. On the other hand, if the device is currently sending data, the request will halt further sending. The associated response message contains the sensor data for any sensors active and sampling on the device at the time of the request. This data takes the form of the *local-ID* followed by the sample data for that sensor. The *local-ID* provides a means to differentiate between data from different sensors on the same device and is used by the base station to separate and process data according to sensor type.

Figure 2 and Figure 3 illustrate the various request and associated response messages, respectively. Both the requests and response messages have a common header that every request and response

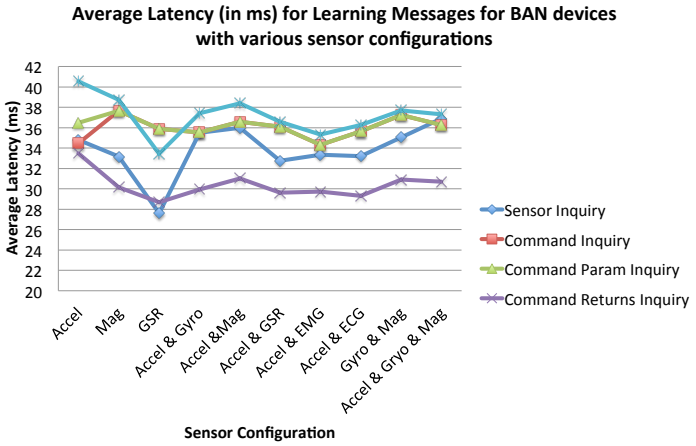

Figure 4: Latency for BAN-PnP Messages and Associated Responses

message possess. This is followed by message-specific request and response elements that pertain to the functionality enabled by each message type.

Many of the protocol packets described include a type identifier. Due to the fact that bytes sent back and forth over the network can be interpreted in several different ways, type IDs are needed to standardize their interpretation per packet. BAN-PnP supports all the basic data types including: signed and unsigned integers, floats, doubles, and strings. More can be defined as needed. By not relying on static message identifiers for the devices, the base station can automatically accomplishes our aforementioned requirements. Since our approach supports virtually every data type, input restriction, and output conversion, almost any device with any type of sensor can be added to the BAN with minimal change to the base station firmware in a plug-n-play manner. As long as a new device can talk to the base station in the predetermined syntax, it can teach the base station everything it needs to know. New sensors and commands need only be appended to the existing inquiry responses. Finally, our approach requires minimal cognitive load from the users in terms of managing the BAN. As the device is expected to teach the base station everything it needs to know, the management task has been moved from the user to the communication protocol, thereby enabling plug-n-play configuration of BANs by the users.

4. PERFORMANCE EVALUATION

In this section, we evaluate the cost of plug-n-play on a BAN, based on the BAN-PnP protocol. Every time a device joins the BAN, it has to teach the base station about its capabilities. This allows us to automate the operation of the BAN. However, this is not for free and the extra control messages exchanged between the device and the base station increase the overhead on the BAN. Our goal in this section is to evaluate how much this overhead is and whether it is "reasonable" in terms of BAN operation.

We implemented the BAN-PnP protocol on a Shimmer platform [1] with an Andoird-based base-station on a Google Nexus smartphone. With respect to the performance analysis of this testbed, we compare its performance with the basic protocol provided by Shim-

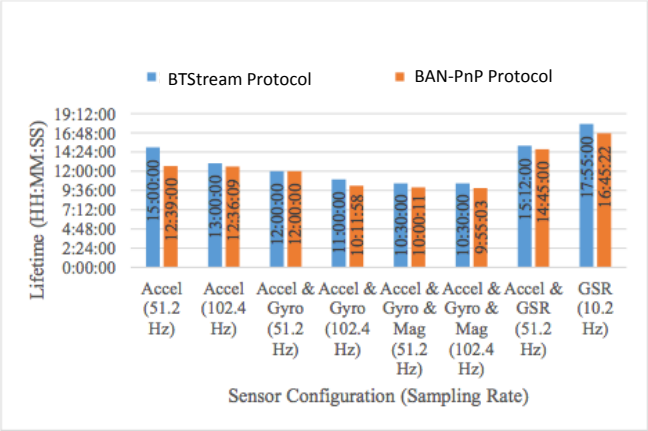

Figure 5: Comparison of BAN device lifetime due to BAN-PnP and BTStream protocols

mer called BtStream [1]. We choose the BtStream protocol because it depicts the traditional non plug-n-play BAN protocol design and it is very simple in its implementation. It therefore provides a good baseline for overhead evaluation compared to some of the other protocols used in more involved BAN applications. Principally, we want to compare the performance of our testbed in terms of three metrics: latency, throughput, and battery life.

4.1 Latency

The control messages designed as part of our protocol provide a way for plug-n-play operation of the BAN. However, the commands used for conveying a device's capability to the base station also increase latency within the network. In this section we evaluate the latency associated with each BAN-PnP command. We define latency as the time-difference between when the command was sent from the base-station to the device and receiving a response back at the base-station. The test was repeated 1000 times for devices with various number of sensors and the average results have been reported. Figure 4 shows the latency associated with the learning messages and their responses. For sensor inquiry messages, these latency varied with different number of sensors per device. In general the larger the number of sensors in the network, the higher the latency in executing individual commands. For command inquiry messages, the time required to process command inquiries was fairly uniform regardless of device configuration. In the case of a command parameter inquiry and a command return inquiry, the latency was consistent across all test scenarios.

Each device has to teach the base-station only once, when it is added to the BAN, and the overall latency for various commands together adds up to up about 200msec. This is quite reasonable price to pay for plug-n-play operation moving forward.

4.2 Lifetime

The purpose of the device lifetime test was to measure the battery life of the devices; specifically when a device is active, sampling, and streaming data to the base station. We tested how different sensor configurations affect the battery life of devices within the BAN. We calculate the battery life of a device by comparing the initial timestamp, and the timestamp of the last received data packet. The device stopped sending data when its battery life was expended. The elapsed time since the base station requested data to the last data received is synonymous with the device lifetime. Figure 5 shows the comparison of device lifetime for different sensor combinations between BAN-PnP and BtStream. As expected our protocol has a negative impact on battery life (up to 18% less than BtStream in the worst case). This is due to the additional overhead required when implementing our protocol.

4.3 Throughput

Table 1 shows the throughput results of our protocol and Shimmer's BtStream for a selected set of sensors groupings. As expected, our protocol introduced overhead, which lowers the effective maximum sampling rate. As a direct result, the net throughput and goodput are lower than other protocols, such as BtStream. Our throughput was approximately one half of that achieved by Shimmer with BtStream [1] in all testing scenarios. The goodput we achieved was a little worse than half of the goodput in BtStream due to the additional overhead. Our investigation revealed that buffering samples on the devices before transmission significantly increased overall throughput and goodput.

5. RELATED WORK

Little work has been done in making user configuration of BANs plug-n-play. The focus thus far has been largely in implementing BANs and using them for specific monitoring applications from home care [4, 19] to ambulatory monitoring [5, 18, 20] to rehabilitation [15] . None of these works make any assumptions about how the BAN is put-together by the user and how easy it is to add or remove devices from them.

The closest work to ours has been in the middleware domain. The idea is to develop a platform that simplifies application development by providing abstractions of low-level operations of the network [7, 11]. The middleware usually resides on a mobile device (base station) and interacts with the sensing devices. The goal of middleware typically is to execute BAN applications and shield off the device hardware or OS/protocol stack from the applications. The middleware also provides resource control and management functions. The middleware can be adaptive in nature and change behavior based on changes to the environment [3, 9, 12] and application requirements [7, 17]. Most of these works have been done in the context of wireless sensor networks. However, given the typical scale of wireless sensor networks, which can be many orders of magnitude larger than BANs, these middleware solutions are quite complex. Further, they have not been designed with lay end-users in mind and require considerable configuration to implement.

In [17] we do find the authors implementing a middleware solution for BANs which makes application development easy for smartphone based base stations. This enable plug-n-play of devices from an application development stand-point. However, the middleware still has to be configured for each type of device that is part of the network. This makes our work highly complementary to BAN middleware designs.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented **BAN-PnP**, a communication protocol for enabling devices and the base station (or middleware) to communicate effectively with minimal user involvement. The protocol allowed the devices being added to the BAN to "teach" the base station about its capabilities. By adding a few extra control messages, we are able to transform a traditional BAN into a plug-n-play BAN that is easy for the usually non-tech-savvy users of such systems to

Table 1: Throughput Comparison

	Config	Accel	Accel & EMG	Accel & ECG	Accel & Gyro	Accel & Gyro & Mag
BAN-PnP	Packet Size (bytes)	124	125	125	117	110
	Throughput (KB/s)	3.735	3.333	3.2	3.744	2.816
	Goodput (KB/s)	3.072	2.560	2.560	3.072	2.304
BtStream	Packet Size (bytes)	9	11	13	15	21
	Throughput (KB/s)	9.216	5.632	6.656	7.680	5.376
	Goodput (KB/s)	6.144	4.096	5.12	6.144	4.608

use. The performance analysis of the BAN-PnP protocol demonstrates that the protocol enable plug-n-play operation of BANs with less less than 200ms one-time latency. When compared to one of the simplest non plug-n-play BAN data gathering protocols, we observed a 18% drop in node battery-life and 50% overall throughput and good put drop, in the worst-case. However, with the improving battery life and improving data rates on newer BAN devices, the gain of minimal configuration requirements in comparison to the overhead is affordable.

We believe that there is significant potential for future extensions to BAN-PnP preliminary protocol and implementation. In particular, incorporating security on top of the protocol would increase the appeal for usage in a commercial body area network. Further, the protocol can be made more efficient by making the base station keep profiles of the device capabilities. Now, if a device is added which has very similar capabilities as an existing profile, it can even skip some of the commands in the learning stage thereby increases the speed as well as decreasing the overhead of the protocol. Finally, the ability to update the plug-n-play protocol on-the-fly is a big area of potential expansion and experimentation. In particular, the ability to modify the learning commands during BAN operation.

7. REFERENCES

[1] Shimmer Research. http://www.shimmer-research.com/.

[2] N. Betzler, S. Kratzenstein, F. Schweizer, K. Witte, and G. Shan. 3D Motion Analysis of Golf Swings. Poster at the 9 thInternational Symposium on the 3D Analysis of Human Movement. Valenciennes, France, 2006.

[3] F. C. Delicato, P. F. Pires, L. Rust, L. Pirmez, and J. F. de Rezende. Reflective middleware for wireless sensor networks. In *Proceedings of the 2005 ACM Symposium on Applied Computing*, SAC '05, pages 1155–1159, 2005.

[4] R. DeVaul, M. Sung, J. Gips, and A. Pentland. Mithril 2003: Applications and architecture. In *IN WEARABLE COMPUTERS, SEVENTH IEEE INTERNATIONAL SYMPOSIUM ON,. IEEE, 2003*, pages 4–11. IEEE, 2003.

[5] H. Ghasemzadeh and R. Jafari. Physical movement monitoring using body sensor networks: A phonological approach to construct spatial decision trees. *IEEE Transactions on Industrial Informatics*, 7(1):66–77, 2011.

[6] H. Ghasemzadeh, V. Loseu, and R. Jafari. Structural action recognition in body sensor networks: Distributed classification based on string matching. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):425–435, 2010.

[7] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *Network, IEEE*, 18(1):6–14, Jan 2004.

[8] R. Hillestad, J. Bigelow, and et al. Can electronic medical record systems transform health care? potential health benefits, savings, and costs. *Health Affairs*, 24(5):1103–1117, 2005.

[9] C. Jaikaeo, C. Srisathapornphat, and C.-C. Shen. Querying and tasking in sensor networks, 2000.

[10] E. Jovanov, A. Milenkovic, C. Otto, and P. C. De Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 2(1):6, 2005.

[11] T. Liu and M. Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. *SIGPLAN Not.*, 38(10):107–118, June 2003.

[12] D. Malan, T. Fulford-jones, M. Welsh, and S. Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[13] S. Nabar, A. Banerjee, S. K. S. Gupta, and R. Poovendran. GeM-REM: Generative model-driven resource efficient ECG monitoring in body sensor networks. In *International Conference on Body Sensor Networks*, pages 1–6, 2011.

[14] S. Nabar, A. Banerjee, S. K. S. Gupta, and R. Poovendran. Resource-efficient and reliable long term wireless monitoring of the photoplethysmographic signal. In *2nd Conference on Wireless Health*, pages 9:1–9:10, 2011.

[15] A. Parnandi, E. Wade, and M. Mataric. Motor function assessment using wearable inertial sensors. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 86–89, Aug 2010.

[16] H. Sagha, J. del R Millan, and R. Chavarriaga. Detecting and rectifying anomalies in body sensor networks. In *2011 International Conference on Body Sensor Networks*, pages 162–167, 2011.

[17] A. B. Waluyo, I. Pek, X. Chen, and W.-S. Yeoh. Design and evaluation of lightweight middleware for personal wireless body area network. *Personal and Ubiquitous Computing*, 13(7):509–525, 2009.

[18] A. B. Waluyo, W.-S. Yeoh, I. Pek, Y. Yong, and X. Chen. Mobisense: Mobile body sensor network for ambulatory monitoring. *ACM Trans. Embed. Comput. Syst.*, 10(1):13:1–13:30, Aug. 2010.

[19] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *Network, IEEE*, 22(4):26–33, 2008.

[20] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 281–286. IEEE, 2007.