Wireless Sensor Networks in the Wild: Three Practical Issues after a Middleware Deployment

Christian Seeger[*] , Alejandro Buchmann
DVS Group, TU Darmstadt, Germany
{cseeger,buchmann}@dvs.tu-darmstadt.de

Kristof Van Laerhoven
ESS Group, TU Darmstadt, Germany
kristof@ess.tu-darmstadt.de

ABSTRACT

This paper reflects on experiences in deploying middleware for a body sensor network, using commercial biosensors. Three types of issues are highlighted that arose during the deployment, which impact middleware design in particular: 1) How can the architecture cope with different levels of data fidelity and propagate those levels to the applications? 2) What is the optimal way to handle temporary disconnections from sensors? and 3) How should the middleware implement sensor-specific peculiarities? Although these issues are described using a specific and demanding health care scenario, we argue that the underlying causes tend to be archetypal for a generic set of sensor network middleware. Awareness of these problem categories and possible solutions are therefore generally relevant for other researchers working on middleware designs for all kinds of sensor networks.

1. INTRODUCTION

Improvements and new developments of sensor nodes have opened many new opportunities in wireless sensor networks. As new sensor types arise, and as existing sensors become more powerful, more accurate and more energy efficient, those advancements enrich the sensing capabilities and widen the range of possible applications. New and additional sensors empower the network's opportunities but they also increase the overall complexity of the system. A middleware for sensor networks decouples the application from the underlying sensing and communication tasks. This abstraction from the sensor network reduces the complexity for developers which accelerates the application development and, thus, the deployment of the system. Furthermore, a layered middleware architecture increases the system's flexibility and, hence, the adaptability to new sensors and circumstances. Hardware and protocols can be changed without touching the application itself. In addition to this, using a common middleware in a network allows running multiple applications using the same nodes.

An emerging area for sensor networks, especially body sensor networks, is the area of health care [2, 5] and preventive health care applications. The World Health Organization predicts that chronic diseases will become the most expensive problem faced by current health care systems and sees the integration of prevention into health care as the main solution for this problem [9]. A paradigm shift towards integrated, preventive health care as well as equipping patients with information, motivation, and skills in prevention and self-management are described as essential elements for solving this problem. As body sensor network (BSN) systems are capable of continuously monitoring a person's physiological and physical state, they form a promising tool that equips patients with the required information and motivation.

Motivated by the need for BSN-based preventive health care applications, we developed an event-based middleware for such sensor networks. It is straightforward to deploy, capable of serving multiple applications, and is able to cope with interchanging sets of sensors. Hosted by a smartphone, the middleware allows day-long user monitoring with changing sensor configurations, as well as the integration of stationary physiological devices. The applications running on top of the middleware perform activity recognition and monitoring of physiological parameters. The activity recognition is based on up to three customized accelerometers. For the monitoring of physiological parameters, we use off-the-shelf sensors for measuring heart rate, blood pressure, and body weight.

This paper presents the experiences we made by integrating and deploying commercial biosensors to our middleware. Based on those experiences, three problem classes are distinguished, to be handled by the middleware: 1) data fidelity, 2) temporary sensor disconnections, and 3) sensor-specific peculiarities. Since our experiences are certainly applicable for other researchers working on middleware for sensor networks, we present the general problem for each class and the specific issue we observed with our implementation. Some general solutions as well as specific approaches for solving our problems are presented and discussed.

This paper is structured as follows: after presenting related work, an overview of our body sensor network system including middleware, sensors, and a preventive health care application is given. Section 4 presents the experiences we made by deploying the whole system. Based on

[*]supported by the German Research Foundation (DFG) within the research training group 1362 *Cooperative, Adaptive, and Responsive Monitoring in Mixed Mode Environments*

this, three categories of problems are distinguished and their application-specific as well as generic solutions are discussed in Section 5. This paper concludes with a short summary.

2. RELATED WORK

Reporting of practical issues arising during deployments can help other research in design decisions and raise awareness of the emerged issues, in this paper's instance in designing middleware for sensor networks. The authors of [8] reflect on their experiences of deploying ubiquitous computing systems in public areas. They rely on three deployments: a digital signage solution for conference environments, an exhibition system consisting of large public displays and a video diary application for visitors, and a system for providing information and interactive content to people waiting in an underground bus station on campus. As a result, sixteen lessons that help researchers in deploying ubiquitous computing systems were made.

In [1], experiences in a wireless sensor network for clinical monitoring in a hospital unit are presented. The system collects pulse and blood oxygen saturation readings from up to 41 patients and sends them to a base station which stores the readings in a local database. The authors studied different aspects of the system, analyzed problems that occurred and drew suggestions for improvement based on their experiences. One of the work's results is that the sensing reliability of the biosensors is the bottleneck of the whole system's reliability and not the sensor network itself.

The authors of [3] present the experiences they made while deploying a wireless sensor network of about 100 nodes for precision agriculture. Goal of their project was the protection of a potato crop against a fungal disease. Their work presents a list of hardware and software problems and misunderstandings that occurred during that project. Those problems made them rethink their development process described in their paper and led to a list of lessons learned that can avoid the repetition of mistakes in future projects.

In contrast to the work mentioned above, this paper highlights the experience made in developing and deploying a middleware for (body) sensor networks. A distinction between three problem classes is made and each class is discussed generically as well as specifically for our middleware.

3. SYSTEM SETUP

This section describes the hardware and software setup we used for this paper. It consists of our middleware for body sensor networks, a preventive health care application and the hardware used for it. After the middleware architecture is described, we give a brief overview of the application and biosensors using the middleware.

3.1 Layered, Event-driven Middleware

We developed an event-driven middleware for body sensor networks [7] that is designed to seamlessly handle changing sensor configurations which we argue is a significant feature for many BSN applications. For instance, the preventive health care application we describe in the next section uses a blood pressure device and a scale which are only available when the user is in its proximity. The event-driven architecture inherently supports such ad-hoc connections. Furthermore, having sensor- and application-specific modules as well as a layered structure in our middleware increases extendibility and adaptability. Due to the event-driven design, parts of the middleware and the application are only triggered if data becomes available and does not require those parts to request for data. This saves CPU cycles and, hence, saves energy which is an important resource in BSNs. In order to use our middleware, applications need to subscribe to events they are interested in. In addition to this, applications can forward events to the middleware which allows adjusting sensors, storing data in the database, and information exchange among different applications using our middleware.

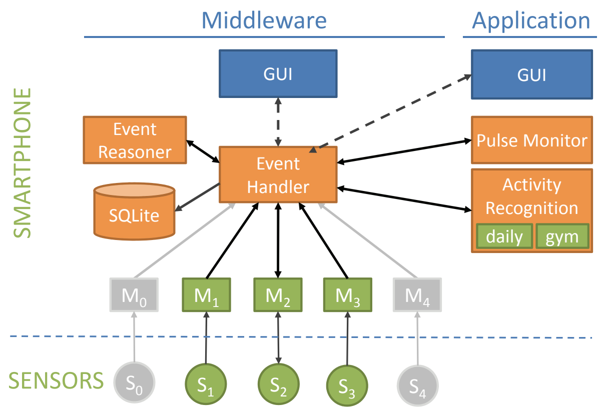

Figure 1: Event-driven middleware for body sensor networks including a preventive health care application with pulse monitoring and activity recognition. Sensor modules simplify adapting to new sensors.

The middleware shown in Figure 1 basically consists of four parts: 1) sensor modules (M_x), 2) an *EventHandler*, 3) middleware services (*EventReasoner*, *SQLite*) and 4) a graphical user interface (*GUI*) for configuration, visualization and user interaction. The sensor modules at the bottom layer are connected to the sensors using the sensor-specific network protocol. They translate raw sensor data to commonly known events which allows abstracting from individual sensors. Sensor events are then forwarded to the *EventHandler* at the service layer. The *EventHandler* acts like a broker that consumes events sent from the sensor modules and other middleware services. Upon receiving an event, it modifies the event, enriches it or simply forwards it to an interested event consumers like the application running on top of the middleware (e.g., *PulseMonitor, ActivityRecognition*). The *EventReasoner* evaluates events and provides alarms, for instance if a sensor's battery level is too low. Those alarms are also events forwarded to the *EventHandler*. The *SQLite* database is used for logging and provides user information to the applications. Middleware services can be used for extending the middleware's functionality.

In order to connect an application to the middleware, it needs to connect to the *EventHandler* and subscribe to events the application is interested in. The application shown in the figure consumes events from accelerometers for activity recognition and from a heart rate sensor for pulse monitoring. Pulse monitoring does not only rely on the heart rate but also on the current activity. Therefore, the application subscribes to activity events sent from the *ActivityRecognition* service via the middleware's *EventHandler* to the *PulseMonitor*.

3.2 Preventive Health Care Application

Our preventive health care application [6] focuses on monitoring a user throughout a day. It captures the user's activities by using up to three accelerometers, monitors the user's heart rate with respect to the current activity, and it takes blood pressure and weight measurements. In addition to this, the user's calorie expenditure is calculated based on the heart rate and an alarm reminds the user on taking the next measurement. Reminders are important because the scale and the blood pressure sensor are stationary and, thus, measurements cannot be automatically triggered.

3.3 Hardware

Our middleware and the application built upon it are based on Android OS (www.android.com). Android is an open source mobile operating system for smartphones and tablet PCs that uses a modified version of the Linux kernel. Software can be written in Java and executed in a specialized virtual machine. The number and functionality of Android devices grow rapidly and fit very well to the area of BSNs. A smartphone is unobtrusive and, hence, it can be used for daily (patient) monitoring whereas a tablet PC at the doctor's office can be used for better visualization of the patient's health parameters. The Motorola Milestone phone serves as an Android 2.1 device for our application.

Bluetooth is used for the communication among the sensors and the smartphone. It is well integrated in current smartphones and supported by most Android devices. In addition, there are already various Bluetooth-enabled off-the-shelf health care sensors. The runtime of approximately 12 hours for the entire system, consisting of a wireless heart rate sensor, wireless custom-built accelerometers, and the Android smartphone, largely depends on the phone itself.

Besides the accelerometers for detecting the user's activities, three off-the-shelf sensors for monitoring a user's physiological parameters are used. The accelerometers are custom-built for our purposes, and therefore might not reflect general problems with BSN deployments. Therefore, the focus in this paper is put primarily on the three off-the-shelf wireless sensor units.

Heart Rate Sensor. The Zephyr HxM Bluetooth sensor[1] serves as our heart rate sensor. It monitors heart specific parameters including heart rate, calories burned, and R-R intervals as well as the wearer's step counts, speed, and distance. The sensor operates for 24 hours with a full charge.

Blood Pressure Sensor. For blood pressure readings, we use the Corscience Boso-Medicus Prestige + BT device[2] which is an upper arm blood pressure meter equipped with a Bluetooth interface. It automatically measures the blood pressure and afterwards transmits the results to a Bluetooth device.

Scale. The IEM Libro-O-Graph[3] serves as a body weight scale which features four-point weight sensors. It can be used like a usual scale and transmits the current weight to a Bluetooth device after the reading is taken.

[1]http://www.zephyr-technology.com/consumer-hxm
[2]http://www.corscience.de/en/medical-engineering/products/blood-pressure/
[3]http://www.iem.de/libr_o_graph

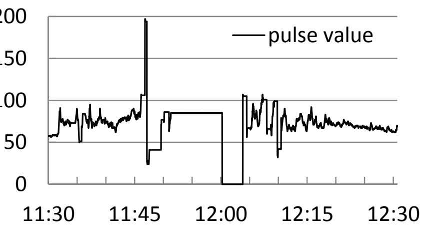

Figure 2: Pulse values from a chest heart rate strap including sensing artifacts.

4. EXPERIENCES FROM THE WILD

In a real-world deployment, a middleware for sensor networks has to cope with different issues such reception of faulty sensor readings, sensors becoming unavailable, and sensor-specific protocol demands. This section presents issues experienced during the deployment of our body sensor network for a preventive health care application. The sensor network consists of a heart rate sensor, a scale, a blood pressure sensor, and up to three accelerometers, all connected to an Android phone hosting our middleware for wireless sensor networks. The following subsections describe three groups of problems that arose: data fidelity, temporary unavailable sensors, and sensor-specific peculiarities.

4.1 Data Fidelity

The fidelity of sensor data is a common problem of sensor networks. Sensor nodes are usually designed for being inexpensive, energy efficient, and compact in size which often contradicts to having accurate sensor data. In addition to this, the sensor's environment and circumstances have strong influence on the sensors and sensor readings. For instance, a temperature sensor which is exposed to direct sunlight might not measure the desired air temperature. The following three paragraphs depict the issues we observed with our off-the-shelf sensors.

Contact-based Fluctuations.

The authors in [1] experienced a high impact of movements on the heart rate sensing from a pulse oximeter attached at the patient's index finger. Movements caused erroneous drops of the measured heart rate. The heart rate sensor we use is chest strap based and more robust against movements, but this one has problems with a too dry contact between the user's skin and the sensor's electrodes. In order to get the sensor started, the user has to moisten the strap before attaching it. This usually works well for a couple of hours and for sport applications when the user is typically sweating. In the case of daily heart rate monitoring, the electrode-skin contact tends to dry out after a while, which results in incorrect readings as shown in Figure 2. We observed that this caused four types of phenomena: 1) extreme outliers, 2) steady heart rate values, 3) a heart rate of zero, and 4) the sensor switching off. Moistening the electrode restores the desired functionality.

Incorrect Sensing Context.

The blood pressure device is a stationary device which measures the systemic arterial pressure at the upper arm. The National Heart Lung and Blood Institute lists several tips for getting precise measurement results [4]. One ad-

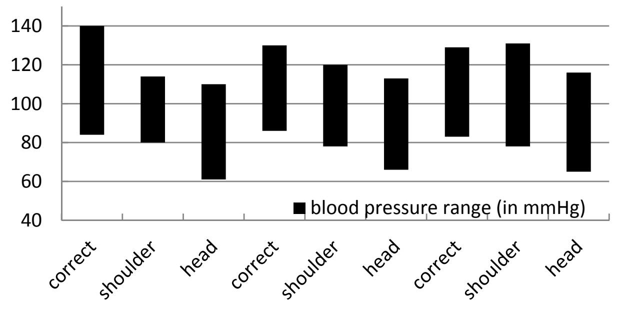

Figure 3: High variations in blood pressure readings taken from the upper arm but with different arm positions: correctly at *heart* level, at *shoulder* level, and at *head* level.

vice is to sit for at least five minutes before doing a test as well as measuring the blood pressure at the level of the heart. Figure 3 shows the ranges between systolic and diastolic blood pressure values taken from a subject within a period of five minutes. The subject had the sensor correctly attached at the upper arm and sat for 10 minutes before the measurements were taken. The first measurement was taken as advised with the sensor in correct position at heart level whereas the second was taken at shoulder level and the third was taken at head level. This procedure was repeated two more times. As shown in Figure 3 the sensor's position has an enormous impact on the measured values and has to be taken into account. Especially when blood pressure sensors become unobtrusive and when they will be worn all the time.

Mapping Environmental Sensors.

Sensors placed in the environment do not necessarily belong to a specific person. They can be shared by many users and their readings can be transmitted to a user's device although the readings do not belong to this user. In our scenario, the scale and the blood pressure sensor could be shared by multiple users. The device itself does not provide user information, which makes it hard to distinguish between valid readings and readings from another person in proximity. This is a general problem in sensor networks that monitor individual subjects/objects. If the sensor readings do not provide information about its subject/object that was sensed, the data fidelity cannot be determined.

4.2 Temporary Unavailable Sensors

Temporary disconnections are a common issue of sensor networks. As a result, sensor readings cannot be transmitted to the sink/aggregator device and, thus, they might be lost for the application. For body sensor networks with environmental sensors, disconnections from stationary sensors are very common and occur every time the user leaves the sensor's proximity.

For our scale for instance, it takes 15 seconds from taking the weight until the value is transmitted to the phone. The transmission time of 22 seconds for blood pressure readings is even worse. Fifteen seconds are enough time to leave the room and be out of transmission range before the transmission is completed.

4.3 Sensor-specific Peculiarities

Off-the-shelf sensors are developed for a specific purpose and they are often not arbitrary customizable. Therefore, the sensor's counterpart which collects their sensor readings has to adapt to sensor-specific peculiarities. The following lists the experiences we made with our sensors.

Heart Rate Sensor. The heart rate sensor is easy to use. As soon as it measures a heart rate, it switches on the Bluetooth module and provides a Bluetooth Serial Port Profile (SPP). Once the middleware discovers the sensor, it connects to it and collects heart rate readings sent every second.

As mentioned previously, the skin-electrode contact is important for a proper functionality. Figure 4 shows the heart rate readings of two days from the same subject. The upper graph shows a day with the strap moistened with water and the lower one shows the results of using a contact gel. The moistening with water was first done with a spray at 7am, but then re-moistened with water from a tap at 8am since the sensor stopped working very soon. Afterwards it wasn't moistened anymore. The gel was used only in the morning. Both graphs show different sensor behaviors; both instances need to be handled by the middleware. Gel performed very well during the day, but at the end of the day the sensor stopped working. User interaction could have solved the problem, but it needs to be decided when user interaction is necessary in order to disturb the user as less as possible.

Scale. The scale provides an active and passive transmission mode. In active mode, the scale itself initiates the Bluetooth connection and transmits the data upon a measurement was taken. In passive mode, the requesting device has to connect to the scale as long as it is switched on. We decided for the active mode since the middleware does not need to search for the scale all the time nor does the user have to tell the system that a measurement was taken. For the active mode it is required that the paired counter-device provides a special Bluetooth profile with ID 1234. SPP is mentioned as a valid profile but it does not work. Therefore, our middleware needs to provide this additional profile. The information sent by the scale is encoded as an SMS message which needs to be parsed in order to convert ASCI characters to hexadecimal numbers.

Blood Pressure Sensor. The blood pressure device we use also provides an active and a passive communication mode. Again we decided for the active mode that listens for SPP and opens a connection upon new sensor readings becoming available. Unfortunately, the device only connects to the phone if Bluetooth visibility is switched on, even though the devices are already paired.

5. SOLUTIONS IN MIDDLEWARE

This section discusses potential solutions for the problems we discovered and presented in the previous section. First the general problem and a general approach for resolving them are given, and then we describe the way it was handled in our event-based middleware and further ideas. This section reflects the structure of the previous one.

5.1 Data Fidelity Levels

Having the application to decide on the data fidelity is one way of handling data of low quality and for some applications it is even necessary to work on the raw sensor data. In those applications the middleware should directly deliver

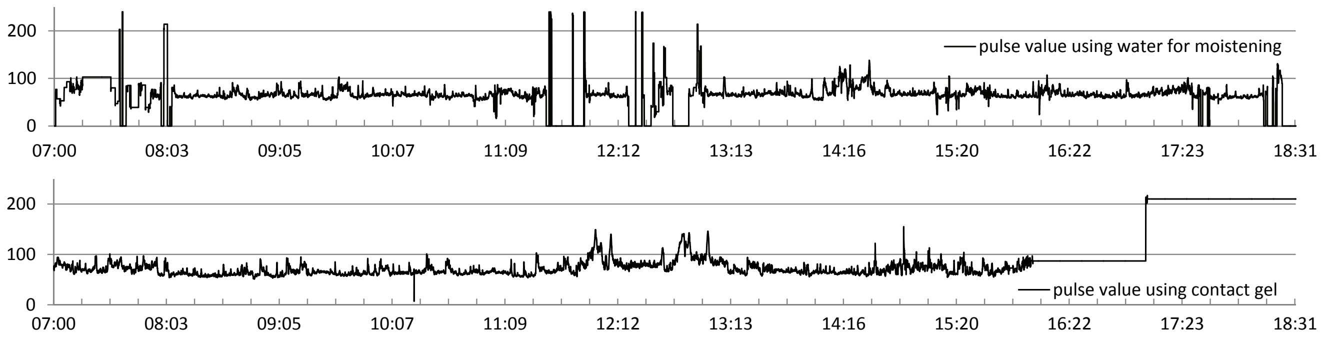

Figure 4: Heart rate readings from a chest strapped unit using (top plot) water for moistening and (bottom plot) contact gel. Water moistening shows more outliers and steady values whereas gel performed well during the day but stopped working at the end. Both reading behaviors needs to be handled by the middleware.

unfiltered sensor data. In many other applications, however, inaccurate or invalid sensor data are not valuable for the application and can be discarded. Therefore, the middleware should detect inaccurate sensor data and mark it as such. The application can decide on how accurate the information has to be for its purposes and discard sensor readings of a lower fidelity. Nevertheless, detecting the level of data fidelity usually requires some processing which causes a time delay. For some applications this delay might be too high which points to the need of having both, the fast raw sensor data as well as sensor data with additional fidelity information.

In order to cope with both classes of application, we decided for providing two different types of sensor readings for the same sensor: raw sensor events and weighted sensor events. Raw sensor events contain the sensor data as it is sent directly from the sensor itself whereas weighted sensor events consist of the original sensor data plus an additional fidelity level. This allows an application to discard sensor readings which might be inaccurate. As already mentioned, calculating a reading's fidelity level requires time and might depend on other readings. This makes weighed sensor events not suitable for time critical applications which justifies keeping raw sensor events in the middleware. The following three examples illustrate how the confidence measures can be implemented.

Contact-based Fluctuations.

For heart rate events we defined three fidelity levels: high, moderate, and low. High fidelity levels are indicated when no artifacts in the heart rate curve was detected. As soon as a peak is detected or if the heart rate drops to zero within a short period of time (cp. Figure 2), the corresponding heart rate event is classified as low since it is usually caused by a too dry skin-electrode contact. If no peaks are detected but the heart rate stays at a constant value for several seconds, the fidelity level decreases from high to moderate. Having a steady heart rate values does not necessarily mean that the heart rate is not detected correctly, but it might be an indicator since the heart rate usually fluctuates within limited range. If it still remains at a steady value, the fidelity level is finally decreased to low until the heart rate value changes again. In our middleware implementation, an additional middleware service consumes the raw sensor events produced by sensor modules. For heart rate events, it computes the fidelity level based on the last 20 heart rate readings and produces weighted heart rate events for every incoming raw event.

Incorrect Sensing Context.

There are several factors that influence blood pressure readings [4]. Two of them are physical exertion just before a reading is taken and a wrong sensor position with respect to the heart. Since our preventive health care application requires heart rate and activity monitoring, both a heart rate sensor and an accelerometer are connected to the middleware. Both sensors deliver indicators for physical activity which can be used for calculating the fidelity level of a blood pressure reading with respect to the physical exertion before the reading. For detecting a wrong sensor position, an additional gyroscope or accelerometer attached at a fixed position of the blood pressure cuff would give information about the cuff's orientation. Having the arm in another position than the required one changes the orientation and, therefore, decreases the reading's fidelity level.

Mapping Environmental Sensors.

Sensor readings of environmental sensors that monitor more than one subject/object need to be assigned to the specific subject/object the readings belong to. If the sensor itself does not provide this information, an additional information source is needed. There are several approaches for identifying subjects/objects in the proximity of a sensor (e.g., radio signal strength, video information, tagging, and context extraction). For our purposes, RFID tags attached at the person and the phone might be a good solution. RFID tag readers that are able to read a user's tag while the user is in the proximity of the scale or blood pressure sensor would send the tag information to our middleware which than checks whether this tag belongs to the person that is monitored.

5.2 Buffering Sensor Data

Many sensors provide enough storage capabilities for buffering readings if they cannot be transmitted due to a leaking network connection. As soon as the network connection is re-established, the missing readings are transmitted. In order to ease the development of sensor network applications, the underlying middleware can handle the transmission of buffered sensor readings and deliver them to the application if they are still of interest for the application. Some applications might not rely on old sensor readings.

In many health care applications, the history of biosignals is important and makes buffered sensor data still valuable. Therefore, our middleware collects old sensor readings, delivers them as events to the application and stores them in the database. Since the time of measurement of an old reading does not correspond to the timestamp of its event, we introduced a *time_of_measurement* field in addition to the usual timestamp which allows identifying old sensor readings. In our scenario both devices, the scale and the blood pressure sensor provide buffering of old readings and add time information to the old readings. The scale transmits the time difference to the current reading, the blood pressure sensor transmits a timestamp which requires synchronized clocks or at least an offset calculation.

5.3 Adapting to Sensor-specific Peculiarities

A middleware for sensor networks often has to cope with a variety of different sensors. Although there are standards for identifying and communicating with sensors, usually it takes some effort to achieve a smooth sensor discovery and communication. Since sensors can change over time but applications might remain, it is the middleware's task to handle the sensor communication and to deliver sensor data to the application despite changes in the sensor-aggregator communication. Those changes could be a new communication technology/protocol or a new type of information sent by the sensor.

Sensor Modules. The sensor modules described in Section 3.1 simplify adapting to new communication protocols. They act like drivers for each individual sensor and translate the raw sensor readings to sensor events. The other parts of the middleware as well as the application are working on sensor events. This abstracts from the individual sensor and allows changing sensors without the need of changing the application. Only the corresponding sensor module needs to be changed. Therefore, changing the Bluetooth profile from SPP to the scale-specific profile with ID 1234 was easy to do and had not influenced the other devices or other parts of the middleware.

System's State. Having a modular architecture for easily adapting to a sensor's characteristics does not solve every sensor communication problem. Our blood pressure device for instance requires a visible Bluetooth counter-part for establishing a connection. One solution would be to have the phone's Bluetooth visibility always enabled but this simplifies tracking the user and should be avoided. Another solution is to switch on visibility before the connection will be established, but for this solution the middleware needs to know when the user is taking a blood pressure measurement. In other words, for blood pressure readings the system's state needs to be known and has to be changed (visibility switched on). Since our application reminds the user to take a measurement, it knows the state and automatically asks for switching on the visibility when the user is reminded.

User Interaction. Another problem is the heart rate strap that loses skin-electrode contact (cp. Figure 4). Depending on the application, re-moistening the electrode in order to get the contact re-established might be desired. One solution would be to implement a reminder on application level that fires when re-moistening is needed. A drawback of this solution is that the application is then limited to a specific sensor or sensor type. If a sensor with different peculiarities is used, this reminder might not work anymore. Therefore, we propose a middleware-oriented solution that allows the application to request instructions for user interaction when it is desired (for instance, if the fidelity level is too low). Then, changing a sensor means just changing the middleware's sensor module including user interaction information. This solution also applies for the blood pressure cuff.

6. CONCLUSIONS

In this paper we presented our experiences in developing a middleware for body area networks using off-the-shelf sensors. We defined three classes of problems that a middleware has to cope with in order to simplify and support the development of applications built on top of it. A distinction was made between the problems of handling data fidelity, temporary unavailable sensors, and sensor-specific peculiarities. Each class of problems was discussed and possible solutions presented. We believe that the awareness of those problems helps researchers in designing middleware not only for body sensor networks, but for all kinds of sensor networks.

7. REFERENCES

[1] O. Chipara, C. Lu, T. Bailey, and G. Roman. Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2010.

[2] P. Khan, A. Hussain, and K. S. Kwak. Medical Applications of Wireless Body Area Networks. *International Journal of Digital Content Technology and its Applications*, 3(3):185–193, 2009.

[3] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture. In *IPDPS*. IEEE, 2006.

[4] National Heart Lung and Blood Institue. Tips for Having Your Blood Pressure Taken. `http://www.nhlbi.nih.gov/hbp/detect/tips.htm`, 2011. [Online; accessed 08-August-2011].

[5] P. Neves, M. Stachyra, and J. Rodrigues. Application of wireless sensor networks to healthcare promotion. *Journal of Communications Software and Systems (JCOMSS)*, 4(3):181–190, 2008.

[6] C. Seeger, A. Buchmann, and K. Van Laerhoven. myHealthAssistant: A Phone-based Body Sensor Network that Captures the Wearer's Exercises throughout the Day. In *Bodynets*, 2011.

[7] C. Seeger, A. Buchmann, and K. Van Laerhoven. An Event-based BSN Middleware that supports Seamless Switching between Sensor Configurations. In *ACM International Health Informatics Symposium*, 2012.

[8] O. Storz, A. Friday, N. Davies, J. Finney, C. Sas, and J. Sheridan. Public Ubiquitous Computing Systems: Lessons from the e-Campus Display Deployments. *IEEE Pervasive Computing*, 5(3):40–47, July 2006.

[9] World Health Organization. Integrating prevention into health care. `http://www.who.int/mediacentre/factsheets/fs172/en/index.html`, 2011. [Online; accessed 01-April-2011].