This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics

MyHealthAssistant: An Event-driven Middleware for Multiple Medical Applications on a Smartphone-mediated Body Sensor Network

Christian Seeger, Kristof Van Laerhoven, and Alejandro Buchmann

J-BHI Special Issue on "Service Science for e-Health"

***Abstract*—An ever-growing range of wireless sensors for medical monitoring has shown that there is significant interest in monitoring patients in their everyday surroundings. It however remains a challenge to merge information from several wireless sensors and applications are commonly built from scratch. This paper presents a middleware targeted for medical applications on smartphone-like platforms that relies on an event-based design to enable flexible coupling with changing sets of wireless sensor units, while posing only a minor overhead on the resources and battery capacity of the interconnected devices. We illustrate the requirements for such middleware with three different healthcare applications that were deployed with our middleware solution, and characterize the performance with energy consumption, overhead caused for the smartphone, and processing time under real-world circumstances. Results show that with sensing-intensive applications our solution only minimally impacts the phone's resources, with an added CPU utilization of 3% and a memory usage under 7 MB. Furthermore, for a minimum message delivery ratio of 99.9%, up to 12 sensor readings per second are guaranteed to be handled, regardless of the number of applications using our middleware.**

***Index Terms*—Body sensor networks, Cybercare, Medical services, Middleware, Performance analysis, Wireless sensor networks**

I. Introduction

An ageing population and low birth rates are leading to a demographic change in the West that significantly challenges its health care systems [1], [2]. In addition to this, 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 [3]. A paradigm shift towards integrated and 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 these problems. Systems that collect information from a network of on-body and ambient sensors are a promising tool for such solutions: As body sensor network systems are capable of continuously monitoring a person's physiological and physical state [4]–[6], they can provide patients with the required information and motivation. Combined with the additional information of the user's surroundings via ambient sensors, full-fledged Body and Ambient Sensor Network (BASN) health monitoring solutions can be built to face these upcoming challenges in health care systems.

We focus in this paper especially on the need for supporting multiple sensor constellations and the integration in the user's environment as significant features for many medical BASN applications. For a patient with a cardiac disorder for instance, monitoring of blood pressure, ECG, and physical activities would be preferred. As monitoring progresses, data from additional respiration and blood oxygen saturation sensors might become relevant to observe a developing sleep apnea. Additionally, information about the user's environment is important for a correct interpretation of many vital parameters: Coffee consumption before taking a blood pressure reading can for instance influence the results [7]; Readings from ambient sensors help determining such contextual information.

MyHealthAssistant is proposed here as a middleware designed for managing body and ambient sensor networks for user-centric health monitoring. From a systems perspective, it 1) is able to cope with interchanging sets of sensor units, 2) is fast to deploy on a user's personal phone, and 3) supports ambulant, day-long monitoring. The event-based design contains dedicated modules that translate sensor data to events to support adapting the system's functionality, extending the sensor set, and to cope seamlessly with changing sensors. The middleware is furthermore designed to run on a smartphone, making use of its connectedness, processing power, and user acceptance. Additional services for detecting sensing artifacts and a worsening system status aim to support application developers.

This paper is structured as follows: after discussing differences between our work and related work, we describe the design choices for our middleware supporting body and ambient sensor network applications in Section III. Section IV presents three case study applications that were built on top of the middleware, all making use of different sets of sensing units. Section V discusses the middleware's performance in detail with focus on energy consumption and efficiency of information routing through the system.

II. Related Work

Many body sensor network (BSN)-based projects in health care focus on monitoring of a particular disease or set of phys-

Supported by the German Research Foundation (DFG) within research training group 1362: Cooperative, Adaptive, and Responsive Monitoring in Mixed Mode Environment. Manuscript received December 21, 2013; revised May 21, 2014.

C. Seeger, K. Van Laerhoven and A. Buchmann are with the Department of Computer Science, Technische Universität Darmstadt, Germany e-mail: {cseeger,buchmann}@dvs.tu-darmstadt.de, kristof@ess.tu-darmstadt.de.

iological signals [4]–[6]. They benefit from the independence from stationary in-hospital observations, allowing patients to freely move and live their daily life while being monitored over longer times and under more realistic conditions. In the Partnership for the Heart project [8], 710 patients with cardiac disorders were equipped with a stationary scale, ECG, SpO2, blood pressure sensors, a hip-worn activity sensor as well as a PDA for transmitting the daily measurements to a remote health care provider. Less hospital stays, an increased quality of life, and a faster reaction to health changes are promising results of this study. The system contained a fixed set of sensors and a relatively sparse monitoring technique.

In contrast to such a fixed setup, middleware for sensor networks such as MiLAN [9] allow a more flexible combination of sensors and applications. MiLAN allows applications to define their QoS requirements over time and how to meet these requirements using different sensor combinations. Based on different priority levels and information about the available sensors and their status, MiLAN continuously adapts the network configuration to meet the application's needs while maximizing the system's lifetime. In order to provide a proper management of the sensor network, MiLAN requires a tight integration with the sensors and protocols.

The Self-Managed Cell (SMC) [10] is a middleware which consists of a policy-based architecture that supports autonomic management and self-configuration for BSNs. Policies define how the system should adapt in response to specific events and an event bus provides content-based subscriptions.

Waluyo et al. [11] propose a middleware for medical BSNs that supports multiple sensors and applications, plug and play features and resource management. Similar to the use case presented in our introduction, they consider vital parameter monitoring and behavior monitoring as two applications running on the same sensor network. In that project however, parts of the middleware and the applications reside on a PC.

In [12], an Android based body area network for telemedical systems is presented. The authors discuss two sensor network setups that consists of either wired or wireless connections from the sensors to a gateway node, which transmits sensor readings to an Android phone for further processing and data forwarding. Several challenges such as data acquisition, visualization, data storage, and safety are discussed.

The work presented so far has a particular focus on optimizing the sensors and the sensor network itself. Further related projects are DexterNet [13], SIXTH [14], Lifeware [15], MobiSense [16], VITRUVIUS [17], and Kamal et al. [18].

In contrast, our middleware approach focuses on providing sensor information to multiple mobile and personal health applications running on a mobile device such as a smart phone. It operates with off-the-shelf sensors from different manufacturers and it does not need an adaptation of the sensors. Therefore, the aim of our middleware is to mediate between sensors from different manufacturers and multiple health-related applications running on a single mobile device.

Jones et al. [19] present the MobiHealth project which consists of a generic BSN for health care as well as a generic mobile health service platform. This system also provides sensor data to multiple applications running on a mobile device with focusing mainly on the network infrastructure among a patient's BSN and health care provider.

Morón et al. propose a smart phone-based telecare system using a body area network [20]. The system consists of commercial off-the-shelf Bluetooth sensors that measure vital parameter and a conventional smart phone which allows using the system without any hardware or software modifications. A target application of that work is the efficient monitoring and management of chronic diseases. The particular focus is on analyzing the impact of different implementations (i.e., Java vs. Python) and different message forwarding techniques between the smart phone and the back-end system.

For health care applications, event-based systems have been presented mainly in areas with high amounts of data. Examples are intensive care solutions, real-time sleep analysis, and solutions for establishing large health care networks. All solutions benefit from the efficient data processing provided by event-based systems. The following describes two examples.

Intensive care units are equipped with numerous devices for monitoring a patient's health parameters. Many of them are stand-alone devices with individual alarming systems triggering their own alarm event. Guerra et al. [21] propose an event-based system that combines the events from individual sensors. It integrates in one place historical data, events, rules, and data mining models and it is highly customizable. In addition, the system performs data mining for identifying possible future risks (e.g. cardiac arrests).

Besides patient monitoring in intensive care units, Singh et al. [22] propose an event-based middleware for patient monitoring in their home environments. The home care system sends monitoring reports and state changes to health care providers and triggers alarms in case of emergencies. By characterizing such a scenario as highly data-driven, the authors chose an event-based system. The particular focus of this work lays on enabling data security by adding dissemination control.

III. Architecture

In [23], the requirements for applications running on a phone-based medical body sensor network were analyzed and an event-driven, layered middleware architecture was proposed. Event-driven systems fit the nature of both body and ambient sensor networks because: a) sensor constellations and running applications change over time, b) most body and ambient sensors send their readings in an event-driven manner (e.g., as alarms when thresholds are exceeded), and c) modular sensor units are agnostic on which applications use their readings. Event hierarchies allow establishing a common, extensible data abstraction on which various applications can be built and by introducing event transformations (e.g., ACTrESS [24]) or ontologies (e.g., CONNECT [25], [26]) to the system, a comprehensive interoperability and integration in other event-based systems would be provided.

A. Event-driven Architecture

Fig. 1 depicts the architecture of MyHealthAssistant, annotated with implementation details as an Android `Remote Service`. On the left, wireless sensor units can be connected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics

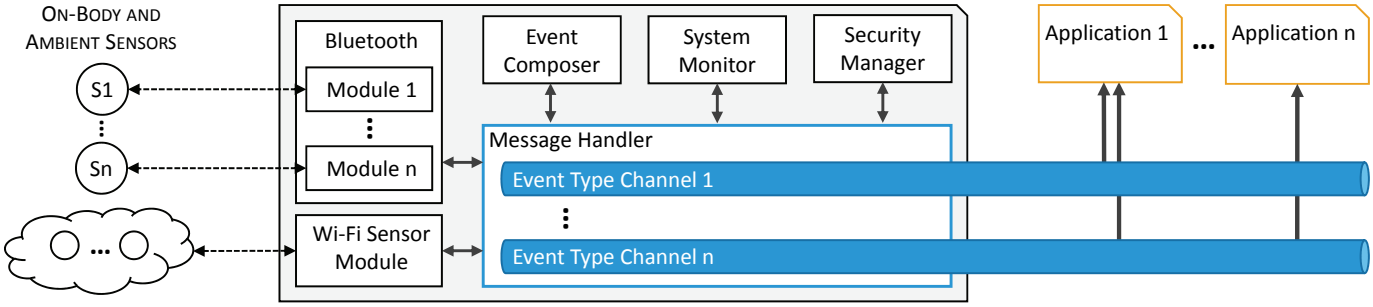


Fig. 1. Event-driven architecture mediating between on-body and ambient sensors and multiple applications. The system's *message handler* utilizes Android `BroadcastReceivers` for inter-application communication. *Sensor modules* translate raw sensor data to sensor events and the *event composer* provides additional sensor fidelity information. The *system monitor* monitors the overall system status and the *security manager* provides information for access control.

to the system: In our implementation, ambient sensors are connected using Wi-Fi, while on-body sensors (*S1*, *Sn*) use the Bluetooth (BT) protocol as it is supported by most Android phones, though also other communication protocols such as ZigBee can be introduced. A *sensor module* handles the sensor communication and creates a corresponding sensor event, which is forwarded to the *message handler*. The latter injects the received event to broadcast channels with respect to the event hierarchy, thus informing subscribed applications about new sensors readings.

The *event composer* in Fig. 1 interprets incoming events, identifies general situations on which the system has to react and creates a corresponding derived event. Events from our case study's heart rate sensor for instance contain also the current battery level: Upon receiving a sensor reading indicating low battery power, an alarm event is created and sent to the *message handler*. The *event composer* also allows to check for inaccurate or invalid sensor data and emits events enriched with fidelity information [27]. This allows applications to only consider sensor readings reaching a certain data fidelity threshold, and throw away low-fidelity data such as heart rate readings with sensing artefacts or blood pressure readings taken after exertion or with a wrong arm posture during the measurement. The *system monitor* measures the overall system status and detects critical situations such as a low battery level. In order to monitor the phone's overall liveliness, heartbeat messages containing status information are periodically sent to a server, allowing a remote detection of a crashed phone or a bad connection to the network carrier. Figure 2 depicts the Android implementation of myHealthAssistant.

B. Broadcast Channels

Applications are usually interested in sensor readings of a certain type only. Sensor readings are therefore published to broadcast channels according to their type, allowing applications to simply subscribe to the reading types of their interest. When a hierarchy of reading types is available, applications can also subscribe to a group of sensor types (e.g., cardiovascular readings) to receive all information within this group (e.g., heart rate and blood pressure readings). Management information such as battery alarms or changes in sensor connections are propagated in broadcast channels and structured in an event hierarchy as well. To save redundant

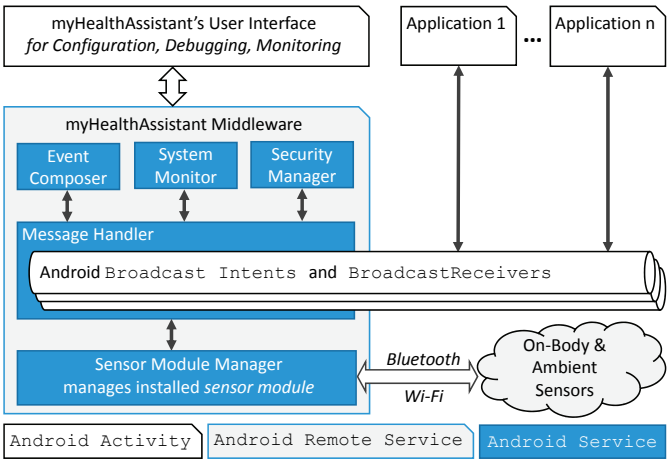


Fig. 2. Android implementation of myHealthAssistant: the middleware is started as an Android `Remote Service` which administers Android `Services` implementing the middleware components. Events sent to applications are encapsulated in `Broadcast Intents` that are received via `BroadcastReceivers`.

and unnecessary calculations, the system also provides a mechanism that allows applications to exchange events.

From an implementation perspective, applications in Android are running in Dalvik process virtual machines that are strictly separated from each other. In order to support inter-process communication, Android provides `BroadcastReceivers` which allow applications to receive information from other applications and the Android system. For receiving information, a BroadcastReceiver needs to be instantiated and registered to the desired broadcast channel (cp. Listing 1). Information is exchanged via Android `Intents` which can contain data of simple data types and `Parcelable` objects. Since events in our system are implemented as Parcelable objects, they are encapsulated in Intents and sent via the Android broadcast service. For an application to inject an event to the system, it thus needs to add this event to an Intent and send it to a receiver using the Android `sendBroadcast()` method (cp. Listing 2). Upon receiving this event, MyHealthAssistant distributes it to the different channels with respect to its event type.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics

Listing 1. Registering an Android `BroadcastReceiver` (ReadingEventReceiver) for receiving events of type *blood pressure*.

```
mReadingEventReceiver = new ReadingEventReceiver();          // create event receiver
registerReceiver(mReadingEventReceiver,                      // register listener for
    new IntentFilter(SensorReadingEvent.BLOOD_PRESSURE));    // blood pressure readings
```

Listing 2. Sending a reading event (myReadinngEvent) to myHealthAssistant using the Android `sendBroadcast()` method.

```
Intent i = new Intent();                                     // create Intent
i.putExtra(Event.PARCELABLE_EXTRA_EVENT, myReadingEvent);    // add event
i.setAction(MyHealthAssistant.RECEIVER_CHANNEL);             // set channel
getApplicationContext.sendBroadcast(i);                      // send Intent
```

C. Application Development

Our middleware solution handles all sensor communication and provides the information in a common data abstraction for the applications to use. For receiving sensor readings, an application needs to subscribe to the broadcast channel of interest. The communication with sensor units, dealing with reading artifacts, and monitoring the liveliness of the system are responsibilities of MyHealthAssistant. Having a middleware as a layer between sensors and applications furthermore allows the multiplexing of sensor information which could otherwise only have been accessed by one application. The event hierarchy of our current implementation is implemented with a tree structure along which the events are distributed in channels.

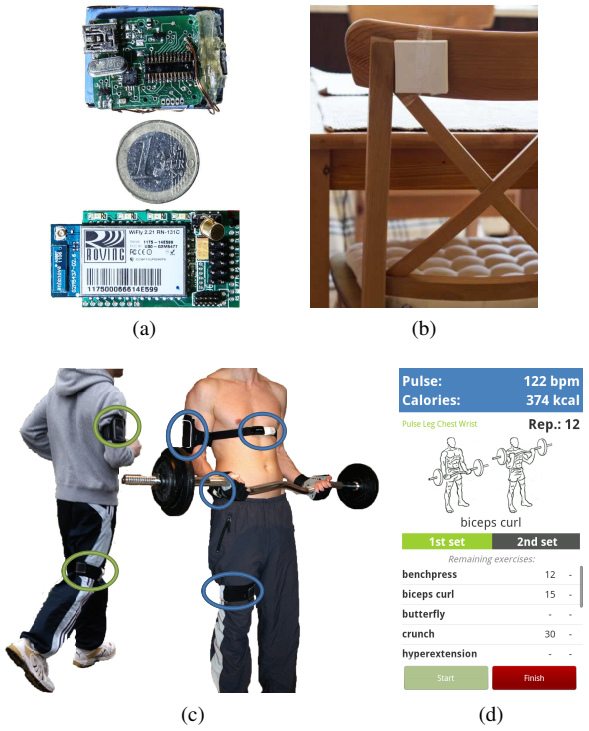
There are two ways for adapting to new requirements following ever-changing network protocols and technologies: The first, middleware-centric method is adding a new or modified *sensor module* to the system. For adaptations to a new protocol based on already implemented technology, a single method that implements the new protocol needs to be written. For adapting the system to a new network technology, connection handling and packet retrieval have to be implemented as well. The second method would involve an additional application which manages the sensor communication and then injects the resulting event to the middleware as described above. Since the latter method would result in a lack of control over the application-based sensor module, the middleware-centric method is identified as the preferred one.

IV. Applications

Three applications were built as case studies on top of MyHealthAssistant, inspired by different application areas: Fitness support, tele-monitoring, and elderly care. They illustrate the type of applications that are targeted by MyHealthAssistant and were used in the evaluation.

A. Fitness Support Application

Studies [28], [29] have shown that an Internet and phone-based user motivation system can significantly increase and maintain the level of physical activity. We developed an application that captures the user's activities and monitors the heart rate throughout a day [30]. It consists of two sensor network setups: one for daily activity detection and another setup for capturing exercises during a gym visit (see Fig. 3c). In the first setup, the user wears a single accelerometer (cp. Fig. 3a upper sensor) and a distinction is made between being idle (i.e. sitting or standing), doing moderate movements (i.e. walking, cycling) and doing sports (i.e. running). In the second setup, two extra sensors embedded in weightlifting gloves and chest strap allow the recognition between 16 gym workouts as well as counting exercise repetitions. Fig. 3d shows a screen-shot during a biceps curl exercise. Detailed evaluation of the system [30] showed that the application's recognition performance matches that of state-of-the-art methods, while being capable of reliable activity and heart rate monitoring with real-time user feedback, for at least 12 hours a day.



(a) (b) (c) (d)

Fig. 3. Hardware modules used in our case studies: (a) Bluetooth accelerometer at the top and a Wi-Fi module with a ball-in-tube sensor attached at the bottom, (b) Wi-Fi module attached to a chair, (c) two sensor configurations of the fitness application, and (d) a screenshot of the fitness application shown during a weight lifting exercise.

B. Telemonitoring Application

Telemedicine is a promising application area for body sensor networks, in which patients' health parameters are collected and transferred to a remote healthcare provider, to observe patients over long periods or remotely detect

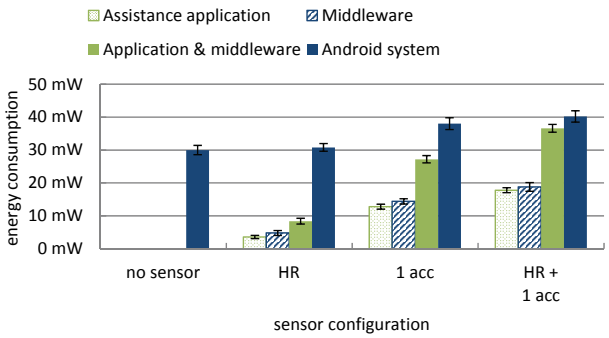
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics

dangerous circumstances. Our second application case study utilizes different sensor units to gather the user's weight, blood pressure, heart rate, ECG, and daily activities. The detected activities are correlated with the user's heart rate, allowing an alarm to be sent whenever the heart rate becomes a-typical for the current activity. Sensor readings are stored in a database and transferred to an existing telemedical platform. Reminders are displayed whenever user-initiated measurements (such as those taken by a scale fitted with bluetooth) are due.

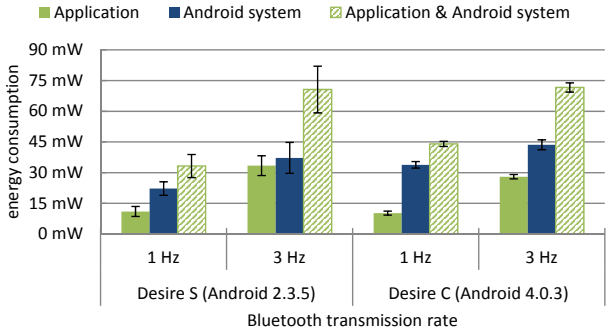
C. Elderly Care Application

The elderly care assistance application monitors both vital parameters and user interactions with the environment, which are interpreted and monitored as activities of daily living. If the user misses to do a specific activity (e.g. tooth brushing) within a given time slot, the system reminds the user. In addition, it creates a list of performed activities at the end of a day which can be used in order to detect changes in daily habits.

In addition to the sensor units mentioned in the previous application, the assistance application makes use of ambient sensors, which are low-power Wi-Fi modules equipped with either a ball-in-tube sensor (cp. Fig. 3a at the bottom), a reed switch, or a passive infrared sensor. Details about the modules are presented in [31]. Upon recognizing a movement, the sensor units send HTTP POST messages to the phone and these events are used together with rules to derive the activities. If for instance the chair in the dining area (cp. Fig. 3b) is used within 45 minutes after the cutlery drawer and the fridge were opened, the application assumes that the person is eating. The system was evaluated in different apartments for activities such as tooth brushing, showering, airing, eating, entering and leaving the apartment, cooking, and desk work, as will be described in Section V. The system makes use of the existing Wi-Fi network infrastructure preserving the phone's Wi-Fi connectivity.



(a) assistance application



(b) impact of different transmission rates

Fig. 4. (a) Shows the energy consumption of the assistance application for different sensor constellations. (b) Depicts the energy consumption for two different Bluetooth transmission rates (1 Hz vs. 3 Hz). Increasing the transmission rate drastically increases the overall energy consumption.

Start	Stop	Duration	Remaining battery power
9:45	22:15	12 hours 30 minutes	30%
7:20	21:20	14 hours 00 minutes	20%
9:45	22:00	12 hours 15 minutes	50%
6:10	22:20	16 hours 10 minutes	15%
6:10	21:20	15 hours 10 minutes	20%

TABLE I
THE REMAINING BATTERY LEVELS AFTER EACH DAY OF USING MYHEALTHASSISTANT WITH 11 SENSOR UNITS FOR PHONE #2 .

V. System Evaluation

The main task of our BASN middleware is to collect measurements from the system's sensors and to direct this information to the applications. An important aspect is that the whole system should last for at least a day without re-charging any of the components. We will therefore analyze and discuss in this section the properties of our system with respect to three requirements: energy consumption, performance for different sensor constellations, and execution speed performance for an increasing number of subscribed applications. We chose two representatives of the current smartphone market: **(#1)** HTC Desire S, 1 GHz CPU, 768 MB RAM, 1450 mAh battery, Android version 2.3.5 and **(#2)** HTC Desire C, 600 MHz CPU, 512 MB RAM, 1230 mAh battery, Android version 4.0.3.

A. Energy Consumption

The operating time of a BASN system is critical for its usability and user acceptance. We analyzed the energy consumption of the the third case study application (elderly care), as it is the most power-consuming. Additionally, impact of increased transmission rates are analyzed and energy consumption figures are measured using PowerTutor [32].

Fig. 4(a) depicts the energy consumption of four different sensor configurations and our assistance application running on phone #1. The heart rate sensor sends 59 bytes per second whereas the accelerometer is sending only 6 bytes per second, but triggers the activity recognition which explains the higher energy consumption. It can be observed that with an increasing message workload, the energy consumption of the application, MyHealthAssistant, as well as the Android system increases. The energy overhead for Android is about 30% for the additional communication tasks.

Since the energy consumption of single processes does not indicate how long a device can operate, we measured the remaining battery level at the end of each day. The setup consisted of 6 ambient sensors for detecting interactions with the environment, a heart rate sensor, a scale, a blood pressure sensor, and an accelerometer for continuous activity detection. Weight and blood pressure readings were taken in the morning and in the evening and the accelerometer was worn continusouly, while the heart rate sensor was worn

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics

sporadically. Table I shows the results after one week of monitoring with phone #2: With a significant processing overhead of performing near real-time activity detection, the phone's battery level remained at 15% after 16 hours of operating. On day three, the accelerometer ran out of power, hence the 50% remaining battery level compared to the 30% of day one.

Wireless communication has the biggest impact for the system's energy consumption; MyHealthAssistant largest contribution here comes from avoiding redundant communication between multiple applications with the same sensor unit. Fig. 4 furthermore illustrates the importance of avoiding duplicate or redundant transmission in a comparison of the energy consumption between having a Bluetooth sensor sending measurements at 1 Hz and at 3 Hz for phones #1 and #2.

B. Performance

We will first analyze MyHealthAssistant's performance under an increasing workload: This is done by instantiating a growing amount of event generators, each injecting events with a 1 Hz frequency. The second case study application described in Section IV-B is subscribed to these events and logs the incoming events in order to calculate the delivery ratio. Since our event generators do not require communication with sensors, we also analyze the system's performance in an hybrid approach consisting of real sensors together with event generators. After we discussed the system's behavior on events containing only a single value as payload, we will increase the payload size of the events. In a last test, we will observe the system's behavior on multiple applications being subscribed to it. All results presented in this section are averages over three or more test runs.

1) Number of Events: As a first step, we will observe how the system behaves for an increasing amount of events per second. Since the Bluetooth protocol is limited to seven active connections, event generators were used for this analysis that run in the same process as the middleware and operate in the same way as the *sensor modules* described in Section III. Each event generator injects events once a second, consisting of following fields: `ID`, `type`, `timestamp`, `producer ID`, `sensor type`, `time of measurement`, and an integer value as the `payload`.

Fig. 5 depicts the results for phone #1 and #2, showing that CPU utilization of phone #1 fluctuates around 2% for the whole test: MyHealthAssistant is marginally affected by the increasing workload, as the actual event distribution is done by the Android system while our middleware solution only decides on which channels the event is sent. The throughput is thus limited by the Android system. In our case, the event delivery ratio drops below 99.9% for more than 12 events per second and reaches 98.7% for a workload of 30 events/s. In contrast, the slower phone #2 running Android 4.0.3 shows a different behavior: The CPU utilization increases with an increasing workload, which is likely due to Android 4 performing some tasks within the application process. Most of the processing is done by the operating system, however, and the delivery ratio rapidly drops for more than 20 events/s. The system's memory usage of 5.7 MB/7.4 MB is relatively modest compared to the phone's RAM of 512/768 MB (the slight increase is due to the growing amount of event generators).

In summary, since communication among Android applications goes via the operating system, the maximum number of events per second is limited by the capabilities of Android. With an increasing workload, the delivery ratio starts dropping. For a minimum delivery ratio of 99.9% up to 12 events/s are handled. We believe that this is sufficient for most BASN applications since the energy consumption of wireless communication is the more limiting factor. Furthermore, the phones we used for our analysis are relatively slow compared to current phones with multi-core processors. Faster hardware is expected to speed up this inter-process communication.

2) Hybrid: Bluetooth Sensors and Event Generators: Fig. 6 depicts the test results for a hybrid setup compared to a setup with event generators only. For the hybrid setup, we connected a heart rate sensor and an accelerometer to phone #1. Furthermore, we tested the impact of activity recognition being activated. Having Bluetooth sensors connected to the system increases both the CPU utilization as well as the memory usage due to the additional overhead evoked by the Bluetooth communication, while the impact on the delivery ratio is marginal. An enabled activity recognition leads to a slight impact, mostly because the detected activities are sent to the middleware, thus resulting in an increased workload.

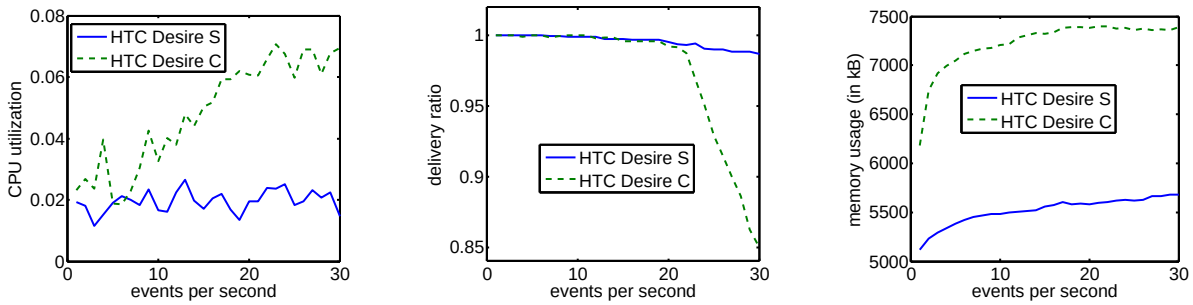
3) Event Size: The events injected so far were consisted of either one or six integer values (the accelerometer sends 6 values/s). Fig. 7 depicts the system's behavior for bigger event sizes. We increased the payload size of the injected events from 1 to 200 integer values. The CPU utilization is not affected by the event size since MyHealthAssistant decides to which channel an event has to be sent only based on the event type and does not inspect the payload. The delivery ratio decreases slightly, which can be explained by the higher amount of transferred data which leads to a slight increase in memory usage: Increasing event sizes have only a marginal impact on the system.

4) Applications: The support of multiple applications is a big advantage of having a middleware solution like MyHealtAssistant as a layer between applications and sensors. We therefore tested our system for four different workload setups and up to eleven Android `BroadcastReceivers` subscribed to the events and running in different processes. As Fig. 8 shows, the amount of subscribed applications has little impact on the system: CPU utilization and message delivery ratio remain while the memory usage increases slightly.

VI. Conclusions and Future Work

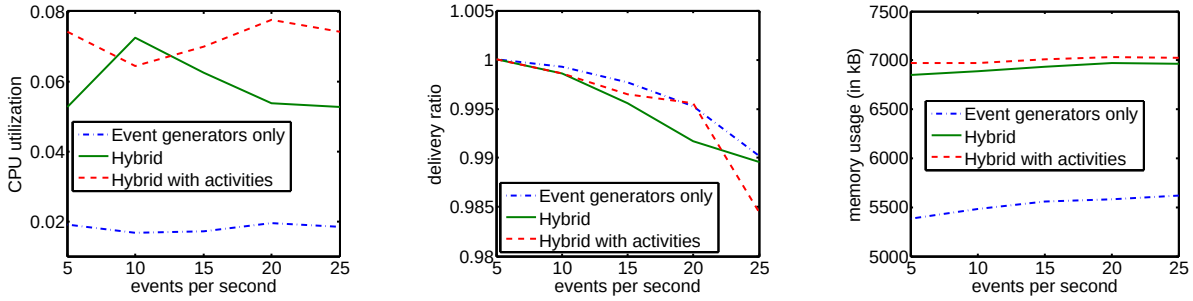
A growing variety of on-body and ambient sensor units is leading to new and promising ways to monitor patients in their natural surroundings. These sensors also result in several challenges to application developers, however, as the increasing heterogeneity of data formats, protocols, and communication channels hinders them in a swift application development. As the currently-available platforms are embedded systems that operate on batteries, an additional challenge is the limitation of available resources. We propose a middleware solution,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics



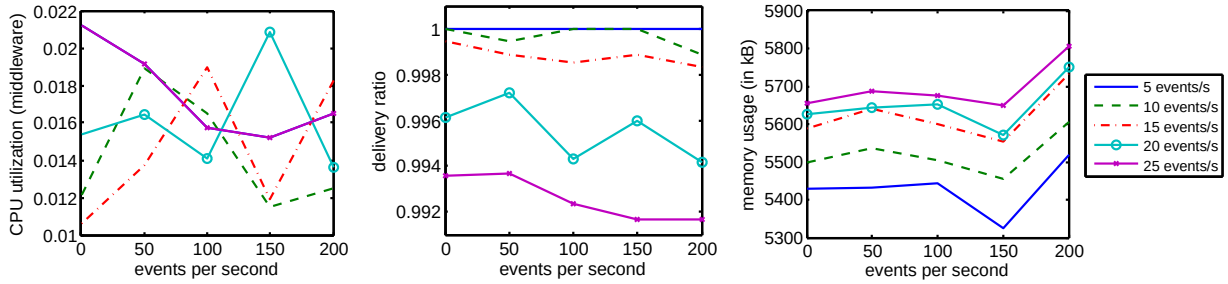
(a) CPU utilization (b) event delivery ratio (c) memory usage

Fig. 5. Performance analysis for an increasing amount of event generators. For a low-end phone (HTC Desire C, 600 MHz CPU, 515 MB RAM) and more than 20 events per second, the delivery ratio starts dropping rapidly.



(a) CPU utilization (b) event delivery ratio (c) memory usage

Fig. 6. Comparison of the system's performance for: I) event generators, II) a hybrid setup with an increasing amount of event generators from a Bluetooth heart rate sensor and a Bluetooth accelerometer, and III) the hybrid setup plus activity recognition as described in Section IV-C.



(a) CPU utilization (b) event delivery ratio (c) memory usage

Fig. 7. Increasing event size from 1 to 200 integer values per event and different throughput configurations.

MyHealthAssistant, which is designed for phone-based deployment and focuses on the efficient management of wireless sensor data for multiple healthcare applications.

The event-driven middleware architecture is designed to aggregate and provides information from both body sensor networks and ambient sensor networks to subscribed applications via broadcast channels. The liveliness of the phone as well as individual sensors is monitored and an event composer module provides the calculation of fidelity levels for sensor readings. A back-channel allows applications to share events to avoid redundant processing.

We evaluated MyHealthAssistant with respect to energy consumption, message throughput and served applications. It was shown that the burden of hosting the middleware solution as well as a monitoring application including activity recognition on a usual Android phone is low enough for at least 16 hours of monitoring. Additionally, using a case study analysis of three target applications implemented on our system, we have shown that performance is sufficient to enable many applications on current phone models, assuming that the system is charged overnight. For high requirements on the event delivery ratio (at least 99.9%) on a single-core phone, the maximum message throughput is limited to 12 events per second whereas the event size does not impact the delivery ratio. We believe that this is sufficient for many current applications since most sensors aggregate their data before transmission. Our ECG sensor, for instance, sends readings twice a second including 107 values per message. We had up to 11 applications subscribed to our middleware with no impact on its performance.

The accelerometer modules using in the case studies are open-source (both hardware and firmware) and publicly available for download[1]. The proposed middleware architecture's

[1]HedgeHog Project: http://www.ess.tu-darmstadt.de/hedgehog

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JBHI.2014.2326604, IEEE Journal of Biomedical and Health Informatics

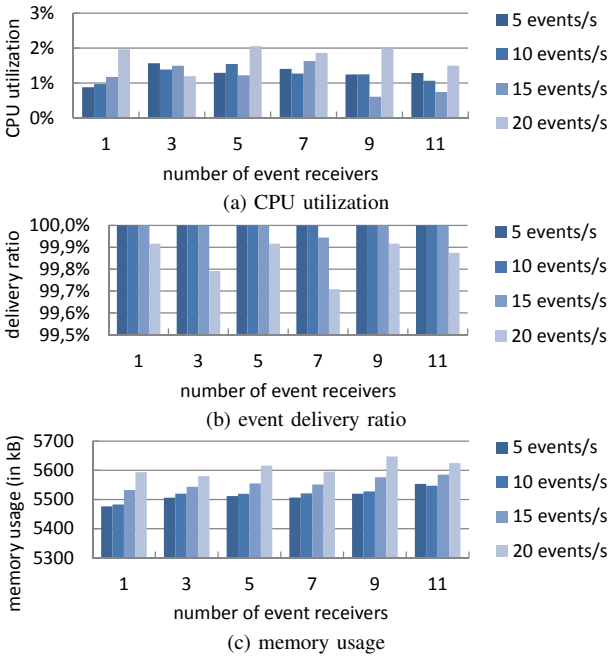


Fig. 8. System behavior on an increasing amount of subscribed applications: no impact for (a) and (b); a marginal impact for (c).

Android implementation will be available as open-source software[2]. The telemedicine platform is a prototype kindly provided to us for this project by Robert Bosch GmbH.

REFERENCES

[1] European Commision Staff, *Demography report 2010*. European Commission, 2011.

[2] L. B. Shrestha and E. J. Heisler, *Changing Demographic Profile of the United States*. Congressional Research Service, 2011.

[3] World Health Organization, "Integrating prevention into health care," http://www.who.int/chp/about/integrated_cd/en/, 2012, [03-23-2014].

[4] M. Chen, S. Gonzalez, A. Vasilakos, H. Cao, and V. C. Leung, "Body area networks: A survey," *Mobile Network Applications*, vol. 16, no. 2, pp. 171–193, Apr. 2011.

[5] P. Khan, A. Hussain, and K. S. Kwak, "Medical Applications of Wireless Body Area Networks," *International Journal of Digital Content Technology and its Applications*, vol. 3, no. 3, pp. 185–193, 2009.

[6] P. Neves, M. Stachyra, and J. Rodrigues, "Application of wireless sensor networks to healthcare promotion," *Journal of Communications Software and Systems (JCOMSS)*, vol. 4, no. 3, pp. 181–190, 2006.

[7] National Heart Lung and Blood Institue, "Tips for Having Your Blood Pressure Taken," http://www.nhlbi.nih.gov/hbp/detect/tips.htm, 2013.

[8] Partnership of the Heart, "Telemedical interventional monitoring," http://www.partnership-for-the-heart.de/en/, 2013.

[9] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo, "Middleware to support sensor network applications," *Network, IEEE*, vol. 18, no. 1, pp. 6 – 14, jan/feb 2004.

[10] S. L. Keoh, N. Dulay, E. Lupu, K. Twidle, A. Schaeffer-Filho, M. Sloman, S. Heeps, S. Strowes, and J. Sventek, "Self-managed cell: A middleware for managing body-sensor networks," in *Intl. Conf. on Mobile and Ubiq. Systems: Networking Services*, 2007, pp. 1–5.

[11] A. Waluyo, S. Ying, I. Pek, and J. K. Wu, "Middleware for wireless medical body area network," in *Biomedical Circuits and Systems Conference, 2007. BIOCAS 2007. IEEE*, nov. 2007, pp. 183 –186.

[12] M. Wagner, B. Kuch, C. Cabrera, P. Enoksson, and A. Sieber, "Android based Body Area Network for the evaluation of medical parameters," in *Proceedings of the Tenth Workshop on Intelligent Solutions in Embedded Systems (WISES)*, 2012, pp. 33–38.

[2]The MyHealthAssistant Website: http://www.dvs.tu-darmstadt.de/research/myhealthassistant/

[13] P. Kuryloski, A. Giani, R. Giannantonio, K. Gilani, R. Gravina, V.-P. Seppa, E. Seto, V. Shia, C. Wang, P. Yan, A. Yang, J. Hyttinen, S. Sastry, S. Wicker, and R. Bajcsy, "Dexternet: An open platform for heterogeneous body sensor networks and its applications," in *Sixth International Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2009, pp. 92–97.

[14] D. Carr, M. O'Grady, G. O'Hare, and R. Collier, "SIXTH: A Middleware for Supporting Ubiquitous Sensing in Personal Health Monitoring," in *International Workshop on Advances in Personalized Healthcare Services*. Springer, 2012.

[15] J. Rodríguez-Molina, J.-F. Martínez, P. Castillejo, and L. López, "Combining wireless sensor networks and semantic middleware for an Internet of Things-based sportsman/woman monitoring application." *Sensors (Basel, Switzerland)*, vol. 13, no. 2, pp. 1787–835, Jan. 2013.

[16] A. B. Waluyo, W.-S. Yeoh, I. Pek, Y. Yong, and X. Chen, "MobiSense: Mobile Body Sensor Network for Ambulatory Monitoring," *ACM Transactions on Embedded Computing Systems*, vol. 10, no. 1, pp. 1–30, Aug. 2010.

[17] V. T. Bui, R. Verhoeven, and J. J. Lukkien, "A Body Sensor Platform for concurrent applications," *IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pp. 38–42, Sep. 2012.

[18] R. Kamal, N. Tran, and C. Hong, "Event-based middleware for healthcare applications," *IEEE Journal of Communications and Networks*, vol. 14, no. 3, pp. 296–309, 2012.

[19] V. Jones, A. Van Halteren, N. Dokovsky, G. Koprinkov, J. Peuscher, R. Bults, D. Konstantas, W. Ing, and R. Herzog, *MobiHealth: Mobile Services for Health Professionals, in M-Health: Emerging Mobile Health Systems*. Springer, 2006.

[20] M. J. Morón, A. Gómez-Jaime, J. R. Luque, and E. Casilari, "Development and Evaluation of a Python Telecare System Based on a Bluetooth Body Area Network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, pp. 1–10, 2011.

[21] D. Guerra, U. Gawlick, and P. Bizarro, "An integrated data management approach to manage health care data," *DEBS*, pp. 40:1–40:2, 2009.

[22] J. Singh and J. Bacon, "Event-based data dissemination control in healthcare," *Electronic Healthcare*, pp. 167–174, 2009.

[23] C. Seeger, A. Buchmann, and K. Van Laerhoven, "An event-based bsn middleware that supports seamless switching between sensor configurations," in *2nd ACM SIGHIT Intl. Health Informatics Symposium*, 2012.

[24] T. Freudenreich, S. Appel, S. Frischbier, and A. Buchmann, "Actress - automatic context transformation in event-based software systems," in *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*. ACM, July 2012, pp. 179–190.

[25] G. S. Blair, A. Bennaceur, N. Georgantas, P. Grace, V. Issarny, V. Nundloll, and M. Paolucci, "The role of ontologies in emergent middleware: supporting interoperability in complex distributed systems," in *ACM/IFIP/USENIX Intl. Middleware Conf.* Springer, 2011, pp. 410–430.

[26] A. Bennaceur, G. Blair, F. Chauvel, H. Gang, N. Georgantas, P. Grace, F. Howar, P. Inverardi, V. Issarny, M. Paolucci, A. Pathak, R. Spalazzese, B. Steffen, and B. Souville, "Towards an architecture for runtime interoperability," in *4th intl. conf. on leveraging applications of formal methods, verification, and validation*. Berlin, Heidelberg: Springer, 2010, pp. 206–220.

[27] C. Seeger, A. Buchmann, and K. Van Laerhoven, "Wireless sensor networks in the wild: Three practical issues after a middleware deployment," in *the Sixth International Workshop on Middleware Tools, Services and Run-time Support for Networked Embedded Systems (MidSens 2011)*, ACM Press. Lisbon, Portugal: ACM Press, 12/2011 2011.

[28] R. Hurling, M. Catt, M. De Boni, W. B. Fairley, T. Hurst, P. Murray, A. Richardson, and S. J. Sodhi, "Using Internet and Mobile Phone Technology to Deliver an Automated Physical Activity Program: Randomized Controlled Trial," *J. of Med. Internet Research*, vol. 9, no. 2, Apr. 2007.

[29] D. F. Tate, R. R. Wing, and R. A. Winett, "Using Internet Technology to Deliver a Behavioral Weight Loss Program," *Journal of the American Medical Association*, vol. 285, no. 9, pp. 1172–1177, Mar. 2001.

[30] 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 *The 6th International Conference on Body Area Networks*, ACM Press. Beijing, China: ACM Press, 11/2011 2011.

[31] B. Ostermaier, M. Kovatsch, and S. Santini, "Connecting things to the web using programmable low-power wifi modules," in *2nd Intl. Workshop on the Web of Things*. ACM, Jun. 2011.

[32] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *8th IEEE/ACM/IFIP Intl. Conf. on Hardware/software Codesign and System Synthesis*. ACM, 2010, pp. 105–114.

Christian Seeger received the B.Sc. and M.Sc. degrees in computer science from Technische Universität Darmstadt, Germany. He is currently a Ph.D. student in the Databases and Distributed Systems group led by Prof. Alejandro Buchmann at TU Darmstadt. His research interests are middleware approaches and applications for on-body and ambient sensor networks. His research project, called myHealthAssistant (http://myhealthassistant.net/), received Best Paper Awards at ACM BodyNets in 2011 and at IEEE ICHI in 2013.

Kristof Van Laerhoven obtained his Ph.D. at Lancaster University (UK) and his M.Sc. degree at the University of Brussels (Belgium). He heads the Embedded Sensing Systems lab at the TU Darmstadt (Germany), funded by the Emmy Noether Programme of the German research foundation DFG. His research combines sensing systems with pattern recognition and machine learning, to obtain adaptive and power-efficient systems. These are especially applied in the challenging scenarios of wearable systems and wirelessly connected networks. More information on this can be found on http://www.ess.tu-darmstadt.de

Alejandro Buchmann is Professor in the Department of Computer Science of Technische Universitt Darmstadt since 1991 and is responsible for the area of Databases and Distributed Systems. Alejandro studied chemical engineering at the Universidad Nacional Autnoma de Mexico and received his PhD from the University of Texas, Austin, in 1980. He was an Assistant/Associate Professor from 1980 to 1986 at IIMAS/UNAM and held positions as a senior researcher at Computer Corporation of America, Cambridge Mass. (86-89) and GTE Laboratories, Waltham Mass. (89-91) before joining TU Darmstadt.