Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2016, Article ID 3651591, 13 pages
http://dx.doi.org/10.1155/2016/3651591

Research Article

A Flexible and Scalable Architecture for Real-Time ANT+ Sensor Data Acquisition and NoSQL Storage

Nadeem Qaisar Mehmood, Rosario Culmone, and Leonardo Mostarda

Department of Computer Science, University of Camerino, 62032 Camerino, Italy

Correspondence should be addressed to Nadeem Qaisar Mehmood; nadeemqaisar.mehmood@unicam.it

Received 29 January 2016; Accepted 21 April 2016

Academic Editor: Francisco Falcone

Wireless Personal or Body Area Networks (WPANs or WBANs) are the main mechanisms to develop healthcare systems for an ageing society. Such systems offer monitoring, security, and caring services by measuring physiological body parameters using wearable devices. Wireless sensor networks allow inexpensive, continuous, and real-time updates of the sensor data, to the data repositories via an Internet. A great deal of research is going on with a focus on technical, managerial, economic, and social health issues. The technical obstacles, which we encounter, in general, are better methodologies, architectures, and context data storage. Sensor communication, data processing and interpretation, data interchange format, data transferal, and context data storage are sensitive phases during the whole process of body parameter acquisition until the storage. ANT+ is a proprietary (but open access) low energy protocol, which supports device interoperability by mutually agreeing upon device profile standards. We have implemented a prototype, based upon ANT+ enabled sensors for a real-time scenario. This paper presents a system architecture, with its software organization, for real-time message interpretation, event-driven based real-time bidirectional communication, and schema flexible storage. A computer user uses it to acquire and to transmit the data using a Windows service to the context server.

1. Introduction

The ageing population of a country shows the progress of its healthcare policies. World Health Organization (WHO) gives facts about ageing areas: "between 2000 and 2050, the proportion of the world's population over 60 years will double from about 11% to 22%. The number of people aged 60 years and over is expected to increase from 605 million to 2 billion over the same period." European Commission ageing report states that by 2060 one in three Europeans will be over 65 [1]. The ratio of working people to inactive people is shifting from 4 to 1 today to 2 to 1 by 2060. The challenge is how to keep them active and productive because at an old age people lose self-control and independence due to physical or mental diseases. This affects their mobility and confines them to their homes, especially during the winters in the frigid regions. The research indicates that individual health management, physical exercise sessions, and remote monitoring on the daily basis contribute greatly to prevent diseases.

Another challenge is to minimize the healthcare expense. We can see a complete list of income spendings upon health by each country here [2]. European digital agenda states that the costs of health and social care will rise substantially to about 9% of EU GDP in 2050 [3]. By 2020, Europe will face up to 2 million vacancies in health and social care sector [4]. Under half of all US health spending is by private companies and the situation is almost same with EU countries. This encourages the business stakeholders to invest in the healthcare sector.

Healthcare is more than a medical problem. Information Communication Technology (ICT) is the most powerful tool to provide cost-effective and high-quality remote healthcare services. Healthcare services, such as *Electronic Medicine*, *Electronic Heath*, *Telemedicine*, *Mobile Health*, and *Telehealth*, have taken care of the young and the elder society members, by utilizing telecommunication network, information technology, and data-driven systems [5]. Particularly the Internet

[6], wireless technologies [7], databases [8], and telemetry [9–12] have contributed many fields, such as observation of lives of rare species [13], smart homes [14–16], medical assistance [7, 17, 18], and security surveillance [19, 20]. It is not possible to nurse a society without measuring the body parameters, such as heartbeat, blood pressure, or temperature. The phenomenon is to be "Keep in Touch (KIT)" [6] with the user and to use Smart Objects (SOs), enabled with wireless technologies, to facilitate telemonitoring processes. SOs are in fact wireless devices or sensors having identification, with the capability to detect changings or events in the environment and to transmit the acquired parameters within a certain distance. Therefore to measure physiological body parameters, remote healthcare systems demand a wearable sensor technology, such as described in [7]. Wearable and implementable biosensors are embedded computers used to develop Body Area Networks (BANs), to provide monitoring services, such as personal care, hospital care, clinical care, individual security, sports, and fitness applications [18, 21].

To provide healthcare services to the seniors, there is a list of goals and technical challenges to meet. There is a range of objectives, such as miniaturization of a sensor device to its communication; integration of devices to build the networks; device data interpretation and its delivery; sensor data reliable storage to later predictive or descriptive analysis; and integration of further stakeholders, such as patients, physicians, and hospitals. On the other side, to better meet the goals, we encounter with a list of theoretical and practical challenges, such as less energy consumption, devices integration, data acquisition, general message processing and interpretation, real-time data exchange in the web, and scalable structural independent data storage.

Due to fewer energy resources, we can not replace batteries frequently during long monitoring operations. Different Media Access Control (MAC) protocols have been developed for bandwidth maintenance and low energy consumption [22–25]. A new 2.4 GHz protocol called ANT+ has recently become famous due to its less battery utilization to develop WPANs [26]. Its range is 30 meters (m) and battery prevails up to 3 years comparative to Bluetooth Low Energy (BLE) and ZigBee with 1 year and 6 months, respectively [27]. ANT+ builds its ecosystem through device interoperability by implementing device profiles, which are ANT+ protocol branded standards.

We acquire data using various methods, such as from papers, distributed datastores, and interactive voice response or Electronic Data Capture (EDC) systems. An EDC system collects data in electronic format to increase data accuracy and to minimize the time [28]. A Windows service is a program that operates in the background and is similar in concept to a Unix daemon [29]. System Application Programming Interfaces (APIs) and services [30] are of different types, such as Component Object Model (COM) and Dynamic-link libraries (DLLs). Microsoft Windows 7, 8, and 10 come with many sensor data acquisition services [31]. Windows 7 includes native support for sensors and provides device APIs [32]. Windows 8 comes with preinstalled sensor monitoring service [33]. Windows 10 comes with many preinstalled monitoring services, such as *Sensor Data Service* [34]. We take the specified advantages from the Windows services and wearable computing devices [26, 35] to develop the basic component of our system which is responsible for acquiring, processing, and sending data to the server in real-time in an event-driven manner. Before transferal, it processes the sensor messages and interprets the different devices' data uniformly and retains the data in the main memory objects holding the device profile properties.

The context server accepts the data and stores it in a document-oriented database, using a set of specific storage routines where each one corresponds to a specific device type. Such procedural routines would only execute upon receiving a specific event. Document databases are schema flexible and can store data in different structures [36–38]. Yet to give sanity we also apply a preschema approach which is very effective in context to rationality without losing the schema flexibility advantage.

The paper is organized as follows: Section 2 is about ANT+ protocol, and Section 3 presents a related research overview. We discuss data acquisition aspects in Section 4. It follows with a discussion about the architectural aspects. The proposed methodology is described in Section 6. The next section presents the prototype details. Future work and summary are given in Sections 8 and 9, respectively.

2. ANT+ Protocol

It is a low energy consuming protocol for long time monitoring operations, to send wireless data from one device to another device in a flexible and robust manner. With millions of installed and operating sensor devices, this protocol allows low message rate sensor network topologies—from peer-to-peer or star to mesh—in Personal Area Networks (PANs), such networks are fit for activeness, fitness, wellness, sports, and home or hospital healthcare systems [26].

ANT manages physical, data-link, network, and transport layers of an Open Systems Interconnection (OSI) model. However ANT+, an extension, manages the session, presentation, and application layers to provide data and device interoperability, as shown in Figure 1. This divides its bandwidth into 125 channels of width equal to 1 MHz and operates on 2.4 GHz spectrum with transmission duration less or equal to 150 microseconds per frame (150 μs/frame) for 8 bytes of data. ANT supports the establishment of numerous public or private uniquely identifiable managed networks, where each one is accessible through a special network key. The three ways for the channel usage for any two ANT nodes, being on the same network, are independent, shared, and scan channels [27, 39].

Each node contains an ANT protocol engine and a host Micro Controller Unit (MCU), and it can be both master and slave and can participate in one or more networks. It can communicate differently depending upon factors, such as what channel type to use; how a channel has been configured; what data types to transmit; and in which direction to communicate. There are different channel types, such as bidirectional, unidirectional, and shared. Mostly, nodes use independent, synchronized, and bidirectional channels. ANT sends data packets over Radio Frequency (RF) channel

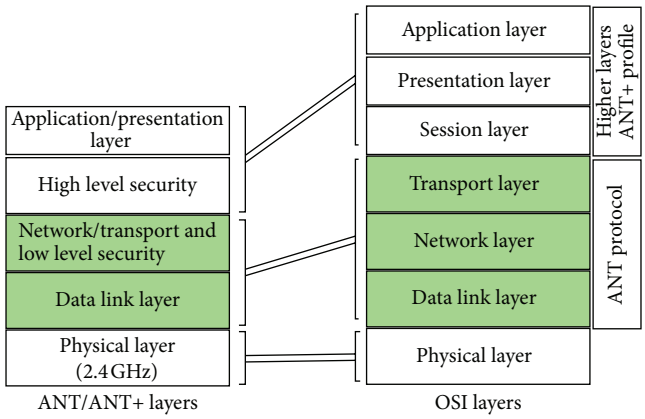

Figure 1: Relationship between OSI layers and ANT/ANT+ profiles [27].

using four different data types exclusively, that is, broadcast, acknowledge, burst, and advance burst [27, 39, 40].

2.1. ANT Data Storage and Sharing. Because ANT devices are embedded devices and possess less energy, less memory, and less other resources, they cannot communicate directly over the Internet. We may lose the advantages of miniaturization of embedded systems if we make them capable of these features. ANT Alliance solves this problem by introducing a technique based upon a compact way of data storage and batch data sharing. It is a format of data representation at the higher layers if agreed upon by vendors and is defined mutually in a set, called as ANT+. Many of the devices are capable of storing the data locally in this compact structure format, called as Flexible and Interoperable Data Transfer (FIT) protocol. This is for data interoperability and compression, whereas to provide this ANT+ uses the notion of device profiles where each profile represents a use case such as heart rate and blood pressure. Therefore, ANT+ profiles allow both sides to understand each other, as both have already agreed upon a specific format [26].

ANT File Share (ANT-FS), an extension, provides a robust framework for transferring FIT files wirelessly between two ANT enabled nodes within a limited distance. It is often used in ultra-low power display devices with limited storage capabilities, such as fitness watches [41]. Therefore, these FIT files are shareable to any other FIT-compliant device. But this technique binds the context data storage servers to be FIT-compliant and to abide by a predefined format and as a result this inflexibility stops them from accepting other formats which are not predefined.

3. Related Work

There are many data acquisition and managing healthcare systems, which have been developed. We already have discussed few Windows services in the introduction section. To the best of our knowledge, a methodology has not been provided by anyone for using Windows service for ANT+ sensors' data acquisition and management. The theoretical work presented in [42] relates to ours in the context of the goals and the application domain; however, it is very high level and hypothetical with untidy lose boundaries and is incomplete, whereas the current work is pragmatic and depicts the results. In [43], we presented a system which acquires and stores ANT+ sensor data locally in a structural compressed format while taking advantage of the ANT+ FIT and FS extension features. It uploads the FIT files on the server at a later stage using batch processing. Although it displays sensor data locally in real-time, it is not real-time in the context of the data delivery. Its distributed context data management server is also inflexible with respect to storage and does not offer a schema independent feature, hence making it unscalable, inefficient, and inflexible.

Kozlovszky et al. [11] acquire cardiac and diabetic patient's data, using many protocols, such as ZigBee/XBee, Bluetooth, and ANT+, with the help of a local data acquisition device called DataHub, based upon a Generic Sensor Network Data Acquisition (DAQ) solution, which sends the data to a distributed context data storage component called Monitoring Datacenter (MDC). They have used both schema approaches, namely, structural or predefined schema and nonstructural or schema flexible, in the form of MySQL [44] and MongoDB [45], respectively. However, they do not use WebSockets' based event-driven approach but use other Internet communication means for the data delivery. Besides this, they do not provide a general methodology to deal with all the sensor device types for any single targeted protocol, such as ZigBee/XBee, Bluetooth, and ANT+, whereas they focus on cardiac and diabetic patients. The provision of a multiprotocol solution with visualization and alert signaling approaches are their additional features.

However Ma and Sun [12] are similar to us technically, but they monitor a building environment in real-time instead of a human body. For the data display and delivery, they use WebSockets in the browser and for the data storage they prefer NoSQL databases like us, but their focus is more in the security domain. The wireless sensor networks are composed of ZigBee and 6LoWPAN [46] instead of ANT+; hence, they miss a general approach, having an explanation about how to deal with all the types of ANT+ devices.

Berndt et al. [20] enable the implementation of a complex system, by conceiving a flexible Telematics Platform, which guarantees secure real-time communication and visualization of the fitness and stress data for both mobile phone and for a website. They use Bluetooth sensors to transmit data to a mobile based gateway, which further transmits the data to a processing database via UMTS/GPRS. Later, the data is stored in a middleware, a Telematic Platform, which allows integration of other applications and services. For predictive analysis, they also propose a new fuzzy logic based stochastic method to model the stress state of an end user. They are similar to us in context to the platform development and its scalable features; however, they do not handle ANT+ sensors. The sensor data storage is not in real-time and the storage server uses a preschema definition approach.

Zhang et al. [47] enables the development of a scalable, real-time, multiparameter, remote healthcare monitoring

system, based upon real-time web communication technology, that is, HTML5-WebSockets. However, the system does not address in general data acquisition and flexible schema storage features. With further advancements, Zhang et al. [16] build a context-aware mHealth System, for the remote monitoring of physiological patient parameters, targeting fitness and stress management application domains. The system provides the technical grounds to perform conditional evaluating the physiological parameters and also provides two main components: (i) an online activity recognition algorithm and (ii) a predictive model, based upon conditional-reasoning knowledge based, to filter potentially dangerous abnormal heart rate instances. Their system supports both Bluetooth and ANT+ sensors and runs on a mobile platform. The system acquires the data in real-time and transmits it in near real-time to the context data server, using HTTP and Sockets. Their server is more mature, in context of the service provision and predictive data mining based decision support; however, they do not provide a general methodology to support all the sensor device profiles.

Gharghan et al. [48] provide a survey about monitoring systems based on PANs, using three common wireless communication protocol standards, that is, Bluetooth [49], ZigBee [50], and ANT. They focus on fitness in general (i.e., by measuring biomechanical and physiological activities of bicycles and cyclists) rather than the core body parameters such as temperature, blood pressure, and heart rate. They review, present, and compare several communication solutions such as low power consumption, long distance communications, small size, and light weight. After observing the examples, of an advanced and adaptive network technology (ANT), the authors conclude that ANT+ protocol is better due to reduced power consumption and prolonged battery life. Furthermore, during their later study in 2015 [51], they repeatedly prove the energy saving advantages of ANT+ protocol over the ZigBee protocol; and for it, they conducted corresponding experiments within the communication range of 65 and 50 meters (m).

4. Data Acquisition Aspects

As we discussed previously in a distributed environment we lack robust systematic methods for sensor data acquisition and its storage which is due to many factors, such as the diversity of sensor data and its acquisition devices. We are required to develop general solutions to address the common sensor data acquisition concerns, ranging from sensor data reading to the data storage and sharing. In [52], to offer a single-system view in the heterogeneous computer networks, distributed systems are often organized by a layer of software called middleware which is a software glue that is logically placed between a higher level layer, consisting of users and applications, and a layer underneath it, consisting of operating systems and basic communication facilities. Due to the absence of a shared memory, all communication in distributed systems is based on sending and receiving messages. In the communication context, a middleware is an application that logically lives (mostly) in the application layer but contains many general-purpose protocols.

For a distributed system with ANT+ protocol based data acquisition, storage, and sharing functionality, we explore and highlight different problems and requirements, such as sensor communication, real-time data processing and interpretation, data interchange format, real-time data delivery, and scalable schema flexible data storage [9].

4.1. Sensor Communication. It is natural to obtain data from sensor devices in pervasive computing which involves the typical physical linking of carrier's networks, such as computers, printers, and routers, through which a meaningful quantity of data is bidirectionally transmitted. Wireless Sensor Networks (WSNs) now provide many valuable features such as low power consumption, low cost, flexibility, efficiency, and security which are flexible to collect information from the monitoring environments. In short, special importance is given to the high transmission rates for the faster downloads. These objectives must be kept in mind while designing or selecting a protocol from the three Open Systems Interconnection (OSI) layers, that is, physical, data-link, and network [53]. Such aspects are addressed in IEEE 802.11 (WLAN/Wi-Fi) and 802.15.1 (Bluetooth) standards [54]. However, Wi-Fi and Bluetooth consume more bandwidth and power; therefore, such protocols are not suitable for wearable sensor devices. In contrast to this, the different aspects related to the low-cost communication with the nearby devices are addressed in the IEEE 802.15.4 standard (e.g., under ZigBee [55]).

Besides this several MAC protocols have been designed for wearable devices with the objective of bandwidth optimization and low power utilization. Recently a new protocol is proposed, that is, ANT+, which is considered as more robust to bandwidth and optimize energy consumption. It is emerging as a widely used MAC protocol for fitness, wellness, and sports wearable sensor devices. Given the required configuration parameters, its device interoperability capabilities provide many additional features to the ANT+ devices such as for the integration purposes; they get to be configured automatically within a wireless body sensor network to build up an ecosystem. Preinstalled ANT+ enabled devices are available in the market. However, for the other components which do not support ANT+ protocol, we can extend them using ANT/ANT+ adapters so as to make them enable to communicate. Therefore, for the wireless data transmission between the device and the context data server, we have used ANT USB Adapter to enable the PC as a gateway.

We have used ANT+ enabled USB device in two flavors, that is, ANT1 (USB1) and ANT2 (USB2). ANT1 has a limited number of channels (i.e., 4), whereas the newer ANT2 stick has more channels (i.e., 8). Its communication aspects are addressed under IEEE 802.15.4 [27]. Different devices operate upon different channel types, message periods, and radio frequencies (RF) and each of them needs to be configured accordingly [39]. The default RF frequency value is 66 and represents the network operating frequency of 2466 MHz. The channel message rate can range from 0.5 Hz to above 200 Hz. For this, we need to configure channel and USB settings with appropriate parameters to build up a network.

4.2. Data Processing and Interpretation. There are many reasons that a sensor data must be processed in real-time before the final delivery for storage, such as interpretation of data for the enhancement of data semantics; translation of data within a particular situation, to capture a more abstract contextual concept; analysis of data to build relationships with the environment; processing of data to shift some of the load, from the server-side to the data collection or monitoring-side; and defining a data interchange format for more meaningful data transferable objects. Examples of such data processing and transferring features can be seen in [56, 57]. Such type of data processing is always specific to certain factors, such as sensor protocol, a sensor device, and the use case. Therefore, this aspect needs to be discussed with respect to two dimensions: (i) data processing and (ii) the physical location of processing.

In a real-time scenario, the system should distinguish, process, and interpret each message before the storage. ANT+ nodes send a lot of information in the default payload (i.e., 8 bytes) messages, where the first byte (i.e., between 0–255 range) is used for the identification purpose [58]. Different sensors generate a similar or different type of data based on certain factors: same message IDs, different acquired information, same errors, same broadcast messages, or acknowledge messages and so forth. Therefore, the message data processing and interpretation are not simple. This become more complex when communicating with more sensor device types; having many different message pages and categorized data semantics.

The four data types, supported by an ANT protocol, are broadcast, acknowledged, burst, and advanced burst data [39]. Each data type is sent in 8-byte packets over an RF channel. It extends device data type messages and permits an ANT channel to send additional data to the host receiver, besides any other data message payload. ANT+ messages have also a 12-byte extended format with more augmented information, such as device identification, device type, and trans type [39]. Currently, ANT+ protocol supports more than 60 different types and almost same number of generated events. So a separate interpretation and processing module for each device type do not suit and we need a static one for all messages. It can distinguish messages, profile types, acquired data, and pages format.

Secondly, for the processing location, we have two alternatives: (i) to process at the data acquisition end, where the sensors might reside, and (ii) to send unprocessed data to the server, but this will create the load and will slow it down. Adding interpretation logic to the server will make the server inflexible in context to minor updates or any addition, with respect to any sensor type message format, will also cause the server rigid.

4.3. Data Interchange Format. While modeling pervasive real-time environments and thus; there is need for data delivery or data exchange over a communication channel; a complex system has to take account of many nonfunctional requirements, such as (i) heterogeneous systems' interoperability; (ii) flexibility to permit easy modifications; (iii) lightweight data to permit efficient transmissions; (iv) truthfulness to be consistent with the real world domain; and (v) readability to allow data evaluation by its domain experts [56]. Data interchange format is similar to the concept of data wrapping; in the latter case, the data is associated with extra security and management features, without changing the underlying message contents, to encapsulate the context data into an appropriate format, whereas the former is to convert the context data into appropriate consistent and readable format which permits understanding and evaluation to its domain experts. Both these concepts are complementary to each other and are used similarly. Indeed, it involves both data abstraction and a transfer syntax for data exchange.

During the data exchange stages, the source schema based data structures are transformed into the target schema based data structures so that the targeted data may represent an accurate meaning the source data [59]. Data exchange process is very close in concept with the data integration process, where, in the former case, the data is actually restructured (i.e, with the possible loss of some data contents). There are many ways, to transform the data from one structural representation to another one; and dozens of source and target schema representations are possible, even within a single domain.

Therefore, keeping transmission rate and performance as the main goals, we require a data exchange language which defines a set of data encoding rules, for both human-readable and machine-readable formats. In fact, this is in generally suitable with interoperation over the Internet as well as between the heterogeneous platforms. For example, Extensible Markup Language (XML) enables the creation and exchange of domain-specific messages and is very popular over the Internet [60]. However, JavaScript Object Notation (JSON) [61] objects are more lightweight, serializable, and AJAX friendly; hence, they are more suitable for faster transmissions. We should define a JSON object's model holding properties for a device type. Nurseitov et al. [62] present a good comparison between the two technologies.

4.4. Real-Time Data Transfer. One of the basic requirements, for a successful web-based system, is efficient processing and data transferal over a communication channel. Data acquisition systems possess distributed components, where data is delivered to its consumers which can be an application or a context data middleware, as in our case.

A real-time healthcare system is a soft real-time system; therefore, some latency is allowed [63]. Interpretation and detection of critical urgent situations (i.e., a sudden fall, a heart attack, etc.) and taking appropriate automatic steps, such as calling a friend or a hospital emergency service, are always the main goals of a healthcare system. Real-time healthcare systems, based on The web technologies, involve a large amount of sensor data exchange, as well as bidirectional event notifications between the system components. Therefore, real-time data delivery feature is important and always required.

Besides this, as discussed previously, the components' heterogeneity, use case requirements, and limited resources make the data delivery process complicated, especially in the case when many clients try to connect to the same server simultaneously or multiple data sources are accessed over the Internet via different computing nodes.

Middleware communication protocols support high-level communication services, for example, that allow a process to call a procedure or invoke an object on a remote machine in a highly transparent way [52]. Similarly, there are high-level communication services to set and synchronize data streams to transfer real-time data, such as a video stream. The four high-level middleware communication services are remote procedure calls (RPCs), message queuing services (MQS), communication of continuous data streams, and multicasting [52].

The interprocess communication, in the real-time web-based client-server distributed systems, consists of general criteria or agenda which involves a set of phases, such as persistent or transient, synchronous or asynchronous, discrete or streaming communication, interface, connection-orientation, speed, single or bidirection, and persistency. In the following we discuss these communication alternatives with respect to our healthcare scenario.

4.4.1. Persistent or Transient. In persistent communication, the middlewares store the messages in one or several storage facilities until their delivery to the receiver. In contrast to this, we have the transient communication in which both sender and receiver are active and are executing entities. In healthcare, for the real-time scenario, we do not need a middleware to store data temporary until its delivery, so our communication should be *transient*.

4.4.2. Synchronous or Asynchronous. The characteristic feature of asynchronous communication is that a sender continues immediately after it has submitted its message for transmission [52]. As a result, the transmitter is blocked until its request is known to be accepted. Therefore, for the healthcare real-time scenario, we need a transmitter which does not wait for responses and continue its work; hence, in this scenario, the transmission will be *asynchronous*.

4.4.3. Discrete or Streaming Communication. We should also make a distinction between discrete or streaming communication; and in the former case, the distributed components communicate with messages and each message form a complete unit of information. In the latter situation, the streaming communication deals with carrying multiple messages at the same time, one after the other, and the messages refer to each other by the order they are sent [52]. Therefore, in the context of the given problem, we should send the sensor data using a data interchange format, such as XML or JSON formats, and a message will be a complete unit of information. We shall take these options to solve the problem we are dealing with and the body wearable sensors we have planned to use are heart rate, temperature, foot pod sensors, and so forth. In contrast to the above options, we may switch to the other choices and should select a streaming communication while dealing with camera or voice sensors.

4.4.4. Interface. An interface is a collection of procedures, by which a client can call to use its functionality and to pass the value parameters as arguments. A system can provide access to its components, without exposing any further detail, through the procedural/functional interfaces. For the remote access, there are different techniques available to use, to expose a software component's Application Programming Interfaces (API), such as Interface Definition Languages (IDL), as mostly used in RPCs; the Representational State Transfer (RESTful) Service Description Language (RSDL), as mostly used for HTTP-based web applications (i.e., typically REST web services) [64]; and the Web Service Description Language (WSDL), as mostly used in web services [65]. The primary purpose of the REST-compliant web services is to manipulate representations of web resources using a uniform set of stateless operations [66].

Nonetheless, such procedural resources may also be functional and executable upon receiving specific events. A distributed system, having an Event Driven Architecture (EDA), reacts to a set of special events generated by different components. There are different reasons responsible for the generation of such type of events, such as upon detecting strange situations in the environment, a process or a thread which may generate an event, upon receiving a procedure or a function call, and upon receiving a special message with an event from a client.

4.4.5. Speed. Latency is a significant issue in networked systems and even milliseconds (ms) become important while dealing with industrial or healthcare problems. Using the UDP protocol for the transmission purpose is considered very suitable because of its speed, which is due to the lightweight data packets and no data delivery guarantee overhead. For healthcare, when a system has to deliver in real-time over the Internet, in a peer-to-peer manner, latency is very important, as well as the guarantee of the data delivery. But previous research shows that HTTP was not designed for real-time, full-duplex communication due to the complexity of real-time HTTP Web applications [67]. We must seek alternatives for Internet-based real-time ANT+ sensor data delivery with speed and other features (discussed above) related to the distributed systems.

4.4.6. Connection-Orientation or Connectionless. In connection-oriented communication a communication session or a semipermanent connection is established before any useful data transmission; however in contrast to this, in connectionless transmissions the data may be delivered out of the order and independent to each other, over different suitable paths. Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are two divergent and core members of the Internet protocol suite, where the former guarantees the transmission and is connection oriented but slower than the later one, as discussed already [52].

Developing the distributed systems, in the domain of healthcare, we deal with critical and emergency situations. For such kind of real-time scenarios, we do not require a middleware that may lose the data packets or a significant event message during the transmission process, particularly during the transmission of critical event messages regarding life vital sign alerts or fire alarms and so forth. Although there are available other alternate mechanisms, to assure the delivery of the data transmission, a connection-oriented communication

would be preferable in this case, whereas in the case of dealing with the video streaming or voice data we may need more bandwidth as well as efficiency; therefore, a faster communication medium would be preferable, in such a case a connectionless communication will be suitable.

4.4.7. Bidirectional Communication. In simplex communication information travels in one direction at a time, whereas full-duplex communication transmits in both sides and is helpful in situations when either sides want to initiate communication or server also wants to respond quickly [52].

In polling a client continuously checks the other programs or devices to see what state it is in, at the moment; usually, the client wants to see whether the server is still connected and it has some data to share. HTTP polling was considered as a better option for delivering real-time data with both sides communicating [68].

Push technology or server push describes a style of Internet-based communication, in which the request for a given transaction is initiated by the publisher. Push services are often based on information preferences expressed in advance. Instead of asking the client to explicitly request (i.e., pull) the information that they need, the data is to be sent to the client without having them specifically ask for it [69]. The Comet Web application model [70] was invented to push data from a server to a browser without an HTTP request from the browser, but it is generally implemented using long polling to accommodate multiple browsers. Long polling is not believed to provide any substantial improvement over traditional polling [71, 72].

The WebSocket protocol enables full-duplex communication between a client and a remote host over a single TCP socket [73]. The WebSocket API is currently a W3C working draft [74]. WebSocket is more efficient than the HTTP polling with the ratio of three-to-one reduction in latency [72]. The authors in [68, 75, 76] describe that one best way to implement server push for the web browser based real-time application is by using WebSockets.

An EDA is very flexible and scalable to extend a system for the development of publish-subscribe patterned applications because these allow loose coupling between the client-server systems, as depicted in [76]. Such system architectures have lot of advantages in broadcasting and targeting a specific group of people to send the data; for example, one is sending an alert event to a patient's friends as well as to the doctor; one wants to publish the patient's history to more than one group of people including the patient's doctor and his relatives; or one group set of patients subscribes to single source of information or service.

4.4.8. Persistent Connection. Making a new TCP connection every time against each request will add unnecessary latencies and will create an inefficient utilization of the network and host resources. So an option is to have persistent connections; and for this, we have the *HTTP persistent connection or HTTP keep-alive* protocol: it uses a single TCP connection to transmit and receive multiple HTTP request/response messages [77]. There are many advantages of having this technique, such as a decrease in the latency of the messages and fewer opening and closing of the TCP connections; the server may get a notification about when a client leaves; hence this creates a possibility to notice that when either side leaves, the possibility that either side can start a communication and so forth. Beside other many advantages, WebSockets also provide persistent connections and stay at an abstract level to allow traffic for different streams [78].

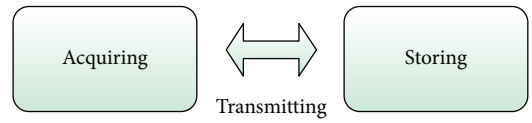

Figure 2: Three main steps of data acquisition.

4.5. Data Storage. It is the physical storage of the data in an organized way. Some of the main factors which have effect upon selecting a good database management system are the data structure or database schema; storage format; scalability; object and instance relationships; and techniques of data gathering.

4.5.1. Storage Schema. An ANT+ protocol deals with many device profile types, to support device interoperability. Thus, we cannot restrict our storage collections to a circumcised circle of specific device formats. We need a flexible database that may allow storage for any schema or format. In other words, the same entity's instance could be storable in more than one format and a single database relation would be able to support this.

4.5.2. Storage Algorithm. Supposing a specific profile type, we require defining a finite set of steps that must be followed to store a message instance related to a profile type. Such steps must perform logically correct updates to a database, leaving it consistent and concurrent after the manipulations. It is significant to mention that the algorithm should be capable enough of holding all the possible schema and such schema oriented algorithm must be valid and abiding to the specifically selected profile type.

5. Architectural Aspects

A software architecture outlines the high-level components, their relationships, and how to create them. This is to understand an environment of a system beside targeting the main goals, such that the system must be scalable and flexible both horizontally and vertically to gather, translate, distribute, and manage sensor data in the context-aware telemonitoring system. Such a context-aware component should allow an integration of services and applications as an addition.

As in Figure 2, the three sensor acquisition abstract level stages are acquiring, transmitting, and storing the sensor data, whereas the acquire and transmit stages can be autonomous components with their own local storage repositories.

The use case scenario unfolds a real-time bidirectional message oriented communication among the two main distributed components, holding the procedures to manage

heterogeneous timestamp data, transmitted by a remote acquisition component. Getting into details, both elements have different goals in different environments.

Although the transient middleware does not store the sensor data temporarily, however, first of all, it separates the sensor data acquisition component from the data storage component clearly, and in this way, this leaves a loosely coupled environment. In healthcare applications, there are many situations when the server has to respond quickly to certain events; hence, an event-driven approach is beneficial. Likewise, an EDA is extremely loosely coupled and highly distributed [79]. To be flexible and to differentiate the control logic from the user interfaces and complex data modeling, a Model-View-Controller (MVC) approach is the most suitable architectural pattern which is very famous for building web applications [80].

As mentioned in Introduction, the data acquisition application will be based upon a Microsoft Windows Service.

The data transfer stage should be capable enough of providing a lightweight and efficient open-source middleware which can handle an event-driven communication in a nonblocking manner. It can accept a WebSocket request and can perform the handshaking to open a persistent full-duplex communication channel. Alongside, it must provide an interface to accept JSON formatted lightweight data messages to store them in a nonrelational schema flexible format.

6. Methodology

Handling different types of diseases and health issues at any of the organizational levels, such as individual, regional, national, or international level, is getting hard and expensive worldwide. Precise research has proved that individual health management, exercise, and physical monitoring help to prevent and cure diseases to improve health conditions, and the key to this is to "Keep in Touch (KIT)," with the patient using the ICT means. Unfortunately, we lack general approaches towards sensor data acquisition, transmission, and storage. We had already talked briefly in the previous sections about the various technical obstacles, that is, sensor communication, information rendering, data interchange format, data transfer, data store, and system environment. To target these technical goals, we have presented and implemented a general methodology which is useful in developing remote healthcare Windows service using Body Area Networks (BAN) for ANT+ enabled sensors. Figure 3 depicts the high-level components and their interactions with respect to the proposed methodology that takes into account the rapid prototyping of ANT+ sensor data provisioning systems.

At an initial phase, the system configures the ANT+ wearable sensor devices to get and parse the data obtained from them. We use an ANT2 stick gateway, to enable the PC to integrate the several sensors with the PC. The manager module opens different communication channels for the sensor devices depending upon their required configuration settings. This uses predefined parameters (i.e., in the default case obtained from an XML file) for the configuration settings of the USB and the communication channels. Due to ANT+ interoperability, any device type from any vendor can be pluggable using the manager provided the configuration settings as parameters. ANT managed library, from Dynastream, helps to communicate to acquire data from wireless sensors through the serial USB stick [26].

The ANTController has a static subcomponent (i.e, Profile Static Interpreter) to process all the event messages of the sensor device or the USB stick. It distinguishes the messages based on their IDs and device types. Since each device type targets a specific use case, such as a heart rate, it will hold different data semantics; therefore, this static module must have a specific message interpretation functionality, as in the current case we require for the heart rate sensor. For each individual device type, the message interpreter is responsible for the extraction of the relevant data semantics and characteristics from the event message data. The interpreter extracts these semantics according to the use-case based device profile specifications, which are defined in the device profile documents provided by the ANT Alliance [58].

After the message data interpretation, the ANTController module extracts and stores each interpreter's specific data in its relevant temporary objects, which hold attributes relevant to a corresponding device profile. Thus, the module's internal organization will deliver a separate temporary object for each device profile.

The ANTPusher component should hold in general JSON modeling system, which additionally carry a separate JSON model with respect to each device profile. Alongside the profile properties, a JSON model may have additional attributes, such as start time, end time, timestamps, or identification. Furthermore, a single model can be subdivided into two models depending upon the requirements, such as that all ANT+ sensors send their device information once after every 65 message events. Therefore, we can differentiate this part of data from the specific JSON model and can send it separately upon another later time interval.

The ANTPusher has a subcomponent called as DataEmitter, which uses a WebSocket protocol client to establish a permanent full-duplex, event-driven based, single communication connection with the remote distributed component after an HTTP-based handshake authentication. DataEmitter transmits the sensor data payload in the form of JSON message, as well as a device-specific *event*. Before the message sending starts, this emitter module fills the lightweight JSON model with the data taken from the corresponding temporary objects. This also contains some specific timers which generate events to trigger and notify the ANTPusher component to transmit the data. Each device profile can have a specific timer depending upon its message rate. A developer can specify the message rates in an XML file and provides those to the component. Therefore, following this method one can develop Windows based sensor data acquisition service for any ANT+ device category.

The second main distributed component (i.e., the context data middleware) is responsible for the storage mechanisms of the JSON sensor data payload, emitted from the ANT-Pusher along a specific *event*. As mentioned previously, we need a database that may be flexible and scalable enough to live with multiple differing structural instances of a same

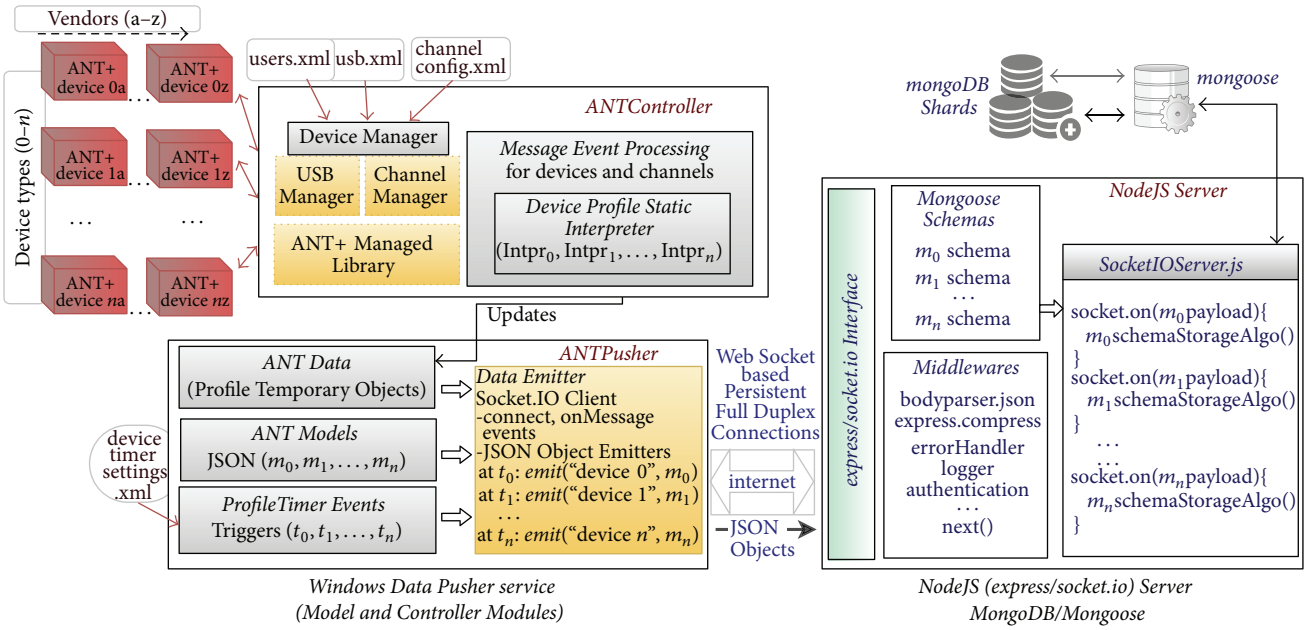

Figure 3: Methodology architecture.

device profile object. For such a complex situation, the best is to use a document-oriented database because of its schema flexible, scalable, and data handling features. There are many such nonrelational databases available; however mongoDB is considered as the best due to its flexible schema support, scalable, replication, and sharding features [45]. For example, MongoDB can store different formats of the heart rate sensor data in a single collection and can provide efficient access to it.

To have a sanity layer before the schemaless data storage, we suggest utilizing a preschema approach which is very effective in context to rationality. For it, we recommend using a middleware, that is, Mongoose, which is a fully developed Object-Relational Mapping (ORM) library for the NodeJS and MongoDB based environments [81]. For a specific device profile, while using Mongoose, we can define as many schemas as we want, according to the requirements. The MongoDB will allow the storage of data according to all defined schemas (i.e., using Mongoose) in any single (or more) collection. In order to store JSON based sensor payloads, according to a Mongoose schema, this component contains a collection of device specific procedures which are responsible for the data manipulation and are also accessible through Socket.IO interface upon receiving an *event* from the Data Emitter. These are set of programming algorithms responsible for a model storage. To allow bidirectional event-driven communication to have access to the data management procedures and to perform the tasks in a nonblocking way, this component must provide such a flexible loosely coupled services. Node.js [82] is a JavaScript-based open-source event driven supported framework which performs operations in an asynchronous fashion. Due to its high-performance network communication and programming environment, we can achieve our required functionality for the complex environment.

ExpressJS is a complete web framework solution which supports the development of applications and services based upon HTTP communication. If one wants to allow access to the NodeJS server through HTTP Web interface, one can use this framework to develop such a component. Middlewares also contain a stack of processors which run on each request made to the server. One can have as many numbers of middlewares that can process each request one by one in a serial manner. Any one of them can change the request, alter data, and then pass it to the *next()* middleware in the chain.

7. Implemented Prototype

A prototype is developed and tested successfully, containing the above-discussed components and the features. We use three types of ANT+ device profile sensors, such as heart rate, temperature, and foot pod [83]. The Device Manager module automatically loads the respective configuration settings from an XML file when the *ANTPusherService* Windows service starts. The system starts accepting data from the sensors simultaneously and parses the messages to populate the specific temporary object. We have developed its Model and Controller components with respect to the MVC architectural pattern. A daemon service mostly does not have the display component because it runs as a background process. However, the prototype is developed with a target to have a flexible and integrated architecture which allow space for the additional features; therefore, there are different techniques available to add views as an additional feature to the system.

In the prototype, we have used about 4 timers to notify the Data Emitter module to emit the JSON data with device specific events. We used 3 timers to send sensor data periodically; however, 1 timer is used to send device-specific information, but once after every 10–15 minutes (min). The

ANTPusher module fills the heart rate, temperature, and foot pod JSON models with the data payload and emits them upon receiving a triggering notification from the respective timers. Roughly speaking, almost each device's data is emitted after every 3-4 seconds (s); but these are not fixed configuration settings since the system is flexible and allows configuration changes to a programmer. For example, when Data Emitter module sends heart rate data, it sends '*heartrateMin*' *event* and for the heart rate sensor's device information it emits '*heartrateHr*' *event*.

We have faced different problem during the development of the prototype, such as how to use WebSockets between a.Net platform and NodeJs based javascript server. To solve this issue, for our windows platform, we use *SocketIO4Net.Client* library which provides a.NET 4.0 C# client for the Socket.IO [84].

To accept WebSocket events, utilizing a full-duplex persistent connection in a nonblocking manner, we have developed the NodeJS based server using Node.js [82] framework. We have defined the listeners for each expected event from the *ANTPusherService*, such as `socket.on ('heartrateMin',function (payload)){}`. According to the Mongoose schema, the server calls the corresponding algorithms to check and store the data in MongoDB collections. We defined three Mongoose schema for the three sensors and have written six different algorithms for their JSON data handling. The three of them relate to storing device's information in the corresponding collection. The communication middleware calls these respective procedures upon receiving the events through the Socket.io interface.

8. Future Work

We can extend our system in different ways such as by targeting other platforms to test our methodology by adding different views at the client side which may allow interactive personalized interfaces to its users; by adding additional in general web interfaces at the data server which may accept data from other sensor devices; and by having more valuable data analysis components. Defining a complete event-response system based upon more specific use case, between the two main components, will make the system more explicit in context to the usage. To perform predictive analysis during data acquisition and to generate the alarming event to trigger healthcare services are the broad focus. There are a lot of opportunities to develop and integrate healthcare services into our very scalable and flexible healthcare system. There are opportunities to develop and integrate more services for different stakeholders, such as patients, doctors, or hospitals.

We want to have different namespaces in the sockets, to support different endpoints or paths to target the stakeholders as different groups, by providing different rooms to the users such as patient's friends' list, physicians list, or service provider group. The different stages within the methodology are flexible to extend with respect to different aspects, for example, in context to the data access point of view. Another facet is to modify and check the system by altering the middleware characteristics as discussed in Section 4.

Having better data organization schema, beside robust storage algorithms, can enrich the system performance and usability as a whole. Node.js framework supports a lot of independent middlewares; one can explore such options to enhance the features. Moreover, the platform is highly scalable and flexible; therefore, this can be extendable in many ways. Using the presented distributed middleware architecture, there are many opportunities for the future work in the field of Artificial Intelligence with its applications in the healthcare domain.

9. Summary

An ageing society is lacking with better healthcare means both at homes as well as in the hospitals. There are many responsible factors such as devices and sensor data heterogeneity, robust transmission techniques, lack of available services at the doors or on each computer, and strict inflexible storage mechanisms. In this paper, we have presented a methodological architecture to acquire body parameters using ANT+ sensors without worrying about different vendors. The system interprets all the sensor data simultaneously and communicates it to the context data server in a nonblocking full-duplex event driven manner for a real-time scenario. The storage server accepts the lightweight JSON data and is capable of storing it in any structure in an asynchronous manner. A prototype has been implemented as an evidence to support the methodology. Within a real-time web environment, the system emphasizes the interoperability through many factors, such as device connectivity, device data interpretation, data format, data transmission, and data storage.

Using a schema flexible approach for the data storage is not new, but using it for the vendor-neutral perspectives to handle all ANT+ sensors is significant. On the other hand, we propose a storage context model that organizes the sensor data in the document-oriented schema. Finally, we have made the monitoring and storage experiments to illustrate the superiority of our approach. Such a system can help to decrease the expenditures caused by long-term hospitalization or frequent visits to medical doctors.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is part of the EUREKA (XXVIII Cycle) project which results from the cooperation between Servili Computers s.r.l., Italy; the University of Camerino (Department of Computer Science), Italy; and the state department Regione Marche, Italy. The authors thank the Software Industry; Ministry of Education; University and Research, Italy, for the financial support of this project.

References

[1] "The 2012 ageing report: economic and budgetary projections for the 27 eu member states (2010–2060)," European Commission, Tech. Rep, 2012, https://ec.europa.eu/digital-agenda/en.

[2] S. Rogers, "Health insurance datablog: healthcare spending around the world, country by country," 2012, http://www.theguardian.com/news/datablog/interactive/2012/jun/30/health-spending-map-world.

[3] A. Walker and T. Maltby, "Active ageing: a strategic policy solution to demographic ageing in the european union," *International Journal of Social Welfare*, vol. 21, no. 1, pp. S117–S130, 2012.

[4] E. Commission, "Research and innovation for ageing well with ict," 2015, https://ec.europa.eu/digital-agenda/en/digital-society.

[5] H. Eren and J. G. Webster, *Telemedicine and Electronic Medicine*, CRC Press, 2015.

[6] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier, "The internet of things for ambient assisted living," in *Proceedings of the IEEE 7th International Conference on Information Technology (ITNG '10)*, pp. 804–809, Las Vegas, Nev, USA, April 2010.

[7] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 40, no. 1, pp. 1–12, 2010.

[8] R. Elmasri, *Fundamentals of Database Systems*, Pearson Education India, 2008.

[9] B. Gonc, R. V. Andreao, G. Guizzardi et al., "Ecg data provisioning for telehomecare monitoring," in *Proceedings of the ACM Symposium on Applied Computing (SAC '08)*, pp. 1374–1379, ACM, March 2008.

[10] A. Johansson, W. Shen, and Y. Xu, "An ant based wireless body sensor biofeedback network for medical e-health care," in *Proceedings of the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '11)*, pp. 1–5, Wuhan, China, September 2011.

[11] M. Kozlovszky, L. Kovacs, and K. Karoczkai, "Cardiovascular and diabetes focused remote patient monitoring," in *Proceedings of the 6th Latin American Congress on Biomedical Engineering (CLAIB '14)*, pp. 568–571, Paraná, Argentina, October 2014.

[12] K. Ma and R. Sun, "Introducing websocket-based real-time monitoring system for remote intelligent buildings," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 867693, 10 pages, 2013.

[13] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pp. 88–97, ACM, Atlanta, Ga, USA, September 2002.

[14] F. K. Aldrich, "Smart homes: past, present and future," in *Inside the Smart Home*, pp. 17–39, Springer, Berlin, Germany, 2003.

[15] D. Surie, O. Laguionie, and T. Pederson, "Wireless sensor networking of everyday objects in a smart home environment," in *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '08)*, pp. 189–194, Umeå University, Umeå, Sweden, December 2008.

[16] W. Zhang, K. Thurow, and R. Stoll, "A context-aware mhealth system for online physiological monitoring in remote healthcare," *International Journal of Computers Communications & Control*, vol. 11, no. 1, pp. 142–156, 2015.

[17] Whole System Demonstrator (WSD), "An overview of telecare and telehealth: headline findings," Tech. Rep., UK Department of Health, Whole System Demonstrator (WSD), 2013.

[18] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[19] N. Grang and A. Gupta, "Wireless sensors network: an overview," *International Journal of Modern Computer Science*, vol. 1, no. 2, pp. 20–24, 2013.

[20] R.-D. Berndt, M. Takenga, S. Kuehn et al., "A scalable and secure telematics platform for the hosting of telemedical applications. Case study of a stress and fitness monitoring," in *Proceedings of the 13th IEEE International Conference on e-Health Networking Applications and Services (Healthcom)*, pp. 118–121, Columbia, Miss, USA, June 2011.

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

[22] IEEE, "Strandard on wireless LAN medium access control (MAC) an physical layer (PHY) specifications for low rate wireless personal area networks," IEEE Standard 802.15.4, 2006, http://standards.ieee.org/about/get/802/802.15.html.

[23] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 95–107, ACM, Baltimore, Md, USA, 2004.

[24] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, pp. 1567–1576, New York, NY, USA, June 2002.

[25] W. Ye, J. Heidemann, and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 3, pp. 493–506, 2004.

[26] ThisisANT, "Thisisant: the wireless sensor network solution," Dynastream Innovations Inc, http://www.thisisant.com.

[27] S. Khssibi, H. Idoudi, A. Van Den Bossche, T. Val, and L. A. Saidane, "Presentation and analysis of a new technology for low-power wireless sensor network," *International Journal of Digital Information and Wireless Communications*, vol. 3, no. 1, pp. 75–86, 2013.

[28] T. Bart, "Comparison of electronic data capture with paper data collectionis there really an advantage," 2003, http://www.dream-slab.it/media/docs/eclinica[1].pdf.

[29] Microsoft Corporation, "Services overview," https://technet.microsoft.com/en-us/library/cc783643(v=ws.10).aspx.

[30] Microsoft Corporation, "System apis and services," https://msdn.microsoft.com/enus/library/ee663297(v=vs.85).aspx.

[31] Microsoft default services, "Microsoft Corporation," http://servicedefaults.com.

[32] Microsoft Corporation, "Sensor api," https://msdn.microsoft.com/enus/library/windows/desktop/dd318953(v=vs.85).aspx.

[33] Microsoft default services: Sensor monitoring service (window 8). Microsoft Corporation, http://servicedefaults.com/8/sensrsvc.

[34] Microsoft default services: Sensor services (window 10). Microsoft Corporation, http://servicedefaults.com/10.

[35] Body sensor networks. Imperial College London, http://ubimon.doc.ic.ac.uk/bsn/m621.html.

[36] Document oriented databases, https://en.wikipedia.org/wiki/NoSQL.

[37] C. Strauch, "NoSQL databases," Tech. Rep., Hochschule der Medien, Stuttgart, Germany, 2014, http://www.christof-strauch.de/nosqldbs.pdf.

[38] P. Neubauer, "Graph databases, nosql and neo4j," Tech. Rep., infoQ, 2010, http://www.infoq.com/articles/graph-nosql-neo4j.

[39] ThisisANT, *Ant Message Protocol and Usage: Application Notes*, Dynastream Innovations Inc, 2007, http://www.thisisant.com.

[40] ThisisANT, "Ant device pairing: application notes," Tech. Rep., Dynastream Innovations Inc, 2013, http://www.thisisant.com.

[41] ThisisANT, "Ant file share (ant-fs) technical specification," Dynastream Innovations Inc., online doc, January 2012, http://www.thisisant.com/resources/ant-fs-technicalspecification.

[42] N. Q. Mehmood, R. Culmone, and L. Mostarda, "An ontology driven software framework for the healthcare applications based on ANT+ protocol," in *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops (AINA '14)*, pp. 245–250, Victoria, Canada, May 2014.

[43] N. Q. Mehmood and R. Culmone, "An ANT+ protocol based health care system," in *Proceedings of the IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA '15)*, pp. 193–198, Gwangju, Republic of Korea, March 2015.

[44] S. Suehring, *MySQL Bible*, John Wiley & Sons, New York, NY, USA, 2002.

[45] K. Chodorow and D. B. Mongo, *The Definitive Guide*, O'Reilly Media, 2013.

[46] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*, vol. 43, John Wiley & Sons, New York, NY, USA, 2011.

[47] W. Zhang, R. Stoll, N. Stoll, and K. Thurow, "An mhealth monitoring system for telemedicine based on WebSocket wireless communication," *Journal of Networks*, vol. 8, no. 4, pp. 955–962, 2013.

[48] S. K. Gharghan, R. Nordin, and M. Ismail, "A survey on energy efficient wireless sensor networks for bicycle performance monitoring application," *Journal of Sensors*, vol. 2014, Article ID 153604, 16 pages, 2014.

[49] Y. Zhang and H. Xiao, "Bluetooth-based sensor networks for remotely monitoring the physiological signals of a patient," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 1040–1048, 2009.

[50] K. Malhi, S. C. Mukhopadhyay, J. Schnepper, M. Haefke, and H. Ewald, "A zigbee-based wearable physiological parameters monitoring system," *IEEE Sensors Journal*, vol. 12, no. 3, pp. 423–430, 2012.

[51] S. K. Gharghan, R. Nordin, and M. Ismail, "An ultra-low power wireless sensor network for bicycle torque performance measurements," *Sensors*, vol. 15, no. 5, pp. 11741–11768, 2015.

[52] M. van Steen and A. Tanenbaum, *Distributed Systems, Principles and Paradigms*, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands, 2001.

[53] H. Zimmermann, "OSI reference model—the ISO model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, 1980.

[54] J. M. Tjensvold, "Comparison of the IEEE 802.11, 802.15.1,802.15.4 and 802.15.6 wireless standards," September 2007, https://janmagnet.files.wordpress.com/2008/07/comparison-ieee-802-standards.pdf.

[55] ZigBee Alliance, *Zigbee Specification*, vol. 558, 2006, http://www.zigbee.org.

[56] B. Gonçalves, J. G. Pereira Filho, and G. Guizzardi, "A service architecture for sensor data provisioning for context-aware mobile applications," in *Proceedings of the ACM Symposium on Applied Computing (SAC '08)*, pp. 1946–1952, ACM, March 2008.

[57] A. Schmidt and K. Van Laerhoven, "How to build smart appliances?" *IEEE Personal Communications*, vol. 8, no. 4, pp. 66–71, 2001.

[58] ThisisANT, *Ant+ Common Pages*, Dynastream Innovations, 2007, http://www.thisisant.com.

[59] A. Doan, A. Halevy, and Z. Ives, *Principles of Data Integration*, Elsevier, 2012.

[60] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml)," World Wide Web Consortium Recommendation REC-xml-19980210, vol. 16, 1998, https://www.w3.org/TR/1998/REC-xml-19980210.

[61] T. Bray, "The javascript object notation (JSON) data interchange format," Tech. Rep., IETF, 2014, https://tools.ietf.org/html/rfc7159.

[62] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML data interchange formats: a case study," *Caine*, vol. 9, pp. 157–162, 2009.

[63] K. G. Shin and P. Ramanathan, "Real-time computing: a new discipline of computer science and engineering," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 6–23, 1994.

[64] M. Pasternak, "Restful service description language," US Patent App. 13/656,032, October 2012.

[65] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana et al., "Web services description language (WSDL) 1.1 W3C," Note 15, 2001, https://www.w3.org/TR/wsdl.

[66] W. W. W. Consortium, "Web services glossary," 2004, http://www.w3.org/TR/ws-gloss.

[67] P. Lubbers, B. Albers, F. Salim, and T. Pye, *Pro HTML5 Programming*, Springer, 2011.

[68] V. Pimentel and B. G. Nickerson, "Communicating and displaying real-time data with websocket," *IEEE Internet Computing*, vol. 16, no. 4, pp. 45–53, 2012.

[69] M. Franklin and S. Zdonik, "'Data in your face': push technology in perspective," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '98)*, pp. 516–519, Seattle, Wash, USA, June 1998.

[70] R. Gravelle, *Comet Programming: Using Ajax to Simulate Server Push*, Webreference Tutorial, 2009.

[71] S. Loreto, P. Saint-Andre, S. Salsano, and G. Wilkins, "Known issues and best practices for the use of long polling and streaming in bidirectional HTTP," Tech. Rep., 2011.

[72] P. Lubbers and F. Greco, *Html5 Web Sockets: A Quantum leap in Scalability for the Web*, SOA World Magazine, 2010.

[73] I. Fette and A. Melnikov, "Protocol overview: Rfc 6455 the websocket protocol," Tech. Rep., IETF, 2011, https://tools.ietf.org/html/rfc6455.

[74] I. Hickson, "The websocket api," W3C Working Draft WD Websockets 20110929, 2011.

[75] K. Shuang and F. Kai, "Research on server push methods in web browser based instant messaging applications," *Journal of Software*, vol. 8, no. 10, pp. 2644–2651, 2013.

[76] R. M. Lerner, "At the forge: real-time messaging," *Linux Journal*, vol. 2013, no. 225, article 5, 2013.

[77] J. C. Mogul, "The case for persistent-connection HTTP," *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 4, pp. 299–313, 1995.

[78] I. Castillo and V. Pascual, "The websocket protocol as a transport for the session initiation protocol (sip)," 2014, https://tools.ietf.org/html/rfc7118.

[79] B. M. Michelson, *Event-Driven Architecture Overview*, vol. 2, Patricia Seybold Group, 2006.

[80] Model-view-controller. Microsoft Corporation, https://msdn.microsoft.com/en-us/library/ff649643.aspx.

[81] A. Mardan, "Boosting your node.js data with the mongoose ORM library," in *Practical Node.js*, chapter 7, pp. 149–172, Springer, Berlin, Germany, 2014.

[82] S. Tilkov and S. Vinoski, "Node.js: using JavaScript to build high-performance network programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, 2010.

[83] ThisisANT, Thisisant: The facts. Dynastream Innovations Inc, http://www.thisisant.com/business/whyant/ facts.

[84] G. C. Mayol Ramis, *Design and Implementation of a Bidirectional, Secure and Real Time Communication between Windows Phone 8 App and Windows Store App*, Polytechnic University of Catalonia, Barcelona, Spain, 2014.