BSNCloud: Cloud-Centered Wireless Body Sensor Data Collection, Streaming, and Analytics System

Ming Li[1(✉)], Ai Enkoji[1], Matthew Key[1], Aaron Marroquin[1], and B. Prabhakaran[2]

[1] California State University Fresno, Fresno, CA 93740, USA
mingli@mail.fresnostate.edu
[2] The University of Texas at Dallas, Richardson, TX 75080, USA

Abstract. Cloud-assisted body area networks have been the focus of researchers in past years as a response to the development of robust wireless body area networks (WBANs). While software such as Signal Processing in Node Environment (SPINE) provide Application Programming Interfaces (APIs) to manage heterogeneous biomedical sensor networks, others have focused on developing tools that address the issue of sensor connection/control, data receiving, and visualization. However, existing software tools lack sufficient flexibility, scalability, and support for complicated biomedical systems. In this paper, BSNCloud, a cloud-centered heterogeneous and comprehensive wireless body sensor data collection, streaming, and analytics framework is proposed. The system combines the sensor control and data aggregator event detection, real-time data analysis, visualization, and streaming into one Android App and incorporated four key components in the cloud server: data repository, algorithm repository, machine learning engine, and web portal. A prototype has been implemented with preliminary performance evaluation. Results show that the system is promising in its full utilization of the high performance computing power as well as the large volume storage capacity.

Keywords: Body sensor networks · Cloud-assisted · Wireless body area networks

1 Introduction

Wearable computing has gained increasing research and development, especially in biomedical applications. For example, several software tools from Shimmer Sensing [1], Qualcomm [2], Intel [3] and Samsung [4] attempted to address the issue of sensor connection/control, data receiving, and visualization. However, these tools are largely limited in flexibility, scalability, and support for complicated biomedical systems. On the other hand, systems have been recently proposed to enable cloud assisted wireless body area networks. Specifically, BodyCloud [5] provides a general-purpose software that covers a wide range of sensors and provides APIs for the creation of new biomedical applications.

M. M. Alam et al. (Eds.): BODYNETS 2020, LNICST 330, pp. 59–73, 2020.
https://doi.org/10.1007/978-3-030-64991-3_5

Despite of existing research advances, two key issues have not been addressed: (i) medical doctors and practitioners need user-friendly, multiple sensor supported, real-time, and powerful analytics support for high quality patient health monitoring activities. For this purpose, body sensor data should be collected, streamed, processed, analyzed (real-time and on-demand), and archived in cloud server. (ii) biomedical researchers face significant challenges of high cost and overhead performing data analytics. To resolve this issue, both body sensor data repository and data analytics algorithm repository should be created in the cloud server so that researchers can complete the task of large volume data analysis at minimum cost without the overhead of purchasing computing facility, recruiting algorithm developers, as well as obtaining results within a few days instead of months.

We propose BSNCloud, a heterogeneous and comprehensive wireless body sensor data collection, streaming, and analytics framework. BSNCloud is unique in the sense that it provides both real-time Android App based analysis as well as on-demand web portal based data analysis with the potential of supporting a large set of body sensor data and analytics algorithms. The system operates across a set of wireless body sensors, Android phone/tablets, PC desktops/laptops, and a cloud cluster. For implementation, we adopted the widely used Shimmer3 IMU motes that is able to sample multiple signals such as accelerometer. Then, the basic MultiShimmer Template for Android was repackaged to include additional key features such as signal statistics, event detection, real-time data analysis, and multi-signal streaming (using WebSocket [6]). The cloud server is setup with OpenStack operating system, a PHP web server, and MySQL database. Users can perform data analysis using existing data sets and algorithms, upload live data through the Android App, upload their own data on the Web Portal, upload their own algorithms, or modify contributed algorithms for validation and performance evaluation.
Experimental results show that the proposed BSNCloud is very promising. Specifically, it achieves several desirable features:

- *Usability*: given the reasonable cost of Shimmer motes and user friendly web portal, medical researchers and doctors can easily collect data and archive them securely in the cloud server without worrying about storage management.
- *Flexibility*: users have the option of using existing shared data and contributed algorithms as well as experimenting with their own collected data and algorithms, making performance evaluation a fairly easy task. The Android App supports multiple users and multiple signal streaming and therefore fits well for both hospital and home rehabilitation setting.
- *Efficiency*: both real-time and on-demand web portal data analysis are performed in the high performance cluster server, it is possible to run large data volume without significant delay.
- *Scalability*: BSNCloud can be extended to include other Bluetooth enabled sensors. Development of iOS App is also feasible so that data from iPhone sensors can be collected too. The proposed algorithm repository has the potential to grow as more researchers contribute new algorithms.

This paper is organized as follows. Section 2 reviews related works. Section 3 introduces and describes the proposed BSNCloud architecture. Section 4 describes the data

aggregator side design and implementation with an Android App. Section 5 describes the cloud server, which includes a web server, machine learning engine, and MySQL database. Section 6 presents design decisions, implementation details and performance evaluation. Finally, Sect. 7 summarizes the work with future works.

2 Related Works

Among many potential communication technologies, Zigbee [7] and Bluetooth are most widely deployed. Zigbee is a very low power, collision avoidance protocol optimized for lower power sensors. It has developed a health care specific protocol and is compliant with all IEEE 11073 devices as well as most other IEEE 802.15.4 wireless devices. Bluetooth supports high-bandwidth and many existing devices with a health care compliant version defined. However, it has very high power requirements and uptime for the radios. Bluetooth Low Energy [8] is a new proposed system from Bluetooth for lower energy requirements, while being interoperable with Bluetooth Classic.

SPINE [9], a TinyOS based platform, has enabled the implementation of a heterogeneous body sensor network by abstracting the hardware level of multiple sensors such as TelosB and MicaZ and creating an easy to use software. Furthermore, SPINE contains APIs for general-purpose processing functions such as average, median and RMS. Separately, Shimmer Research [10] delivers two sets of software tools: development drivers for LabView, MATLAB, Android and C#; standalone software such as Shimmer Connect, Shimmer Log and Shimmer Plot for easy connection, storage and visualization of sensor data. While these drivers and software provide a good combination of tools for easy direct use or further API developments, the research community needs a scalable, robust instrument to meet their fast research and development needs.

One WBAN architecture that incorporates SPINE is DexterNet [11]. The system takes a real-time approach and seeks to provide an open-source platform that allows indoor and outdoor persistent human monitoring. The Body Sensor Layer, Personal Network Layer and Global Network Layer provide a three-tier architecture to address the different API needs from the sensors, data aggregator and server. This system is both scalable and reconfigurable in real-time due to the versatility of the layered approach. CodeBlue [12], proposed by Fulford-Jones and Malan, presents an ad hoc infrastructure for emergent medical care. In this project, several types of body sensors (e.g., pulse oximeter, ECG/EKG sensor) are individually connected to Zigbee enabled radio transmitters. Due to the ad hoc architecture and the capability of self-organizing, CodeBlue yields scalability for network expanding and flexibility to connect various wireless devices. Jiang and Cao proposed CareNet [13], an integrated wireless environment used for remote health care systems. CareNet offers features such as high reliability and performance, scalability, security and integration with web based portal systems.

Combining Cloud Computing and Wireless Sensor Networks (CC-WSN) [14] and Open Sensor Web Architecture (OSWA) [15] propose architectures that integrate wireless sensor networks (WSN) with the cloud. CC-WSN has base services including sensor data management run on Google AppEngine and Microsoft Azure, filter chain and filter management and user services including visualization and notification. OSWA provides an architecture for integrating sensor networks with various distributed computing platforms like SOA and Grid-Computing. BodyCloud [5] proposes a SaaS approach that

supports cloud-assisted BSN applications. The purpose of the software is to allow fast prototyping of cloud-assisted BSN applications and flexibility among architectural components utilizing web standards-based procedures and scalability aided by the Google App Engine PaaS infrastructure. Their paper discusses the issues of providing body-side and cloud-side context-aware sensing and adaptation, an issue our proposed architecture will address.

A survey of the architectures presented above reveals that several issues still need to be explored [16]. One such issue is providing an efficient collection of body senor data and implementing an adaptive sensing mechanism. The data transmission throughout the network, including sensor to aggregator and aggregator to cloud, still has potential for further optimization. Contextual adaptation is another area of concern, relating to dynamic adaptation in an array of services depending on the current network and sensing context. Finally, a general software that assists programmers for developing specific healthcare related apps is worth further investigation.

To address data reliability between sensor and data aggregator, an analysis driven adaptive framework [17] was proposed. In this framework, a group data importance based scheme was designed to mitigate the effect of data loss on analysis algorithms. Then, a hybrid transmission protocol was developed to meet analysis needs. Furthermore, a multi-level importance ranking was introduced to allow a fast organization of sensor data into priority and non-priority packets. These packets are transmitted progressively in response to data reliability demands and network constraints. Finally, the concatenation of small data frames from sensors with low sample rates like ECG/EKG, along with an adaptive transmission scheme can provide loss resilience among the sensor network [18].

3 System Architecture

As shown in Fig. 1, our proposed system consists of three basic parts: the sensors, the data aggregator and the cloud server. In a typical scenario, a patient may wear a number of wireless sensors from blood pressure to accelerometer for continuous health monitoring. Depending on the kind of illness and the purpose (rehabilitation vs simple monitoring), some patients may also get scheduled for periodical measurement of specific signals at various frequency in doctor's office. During such scheduled measurement, real-time data analysis is often desired so that more than just visualized signal is displayed. While samples are being collected, a phone/tablet/PC, or the "data aggregator", should be connected with sensors wirelessly (via short range communication such as Zigbee, Bluetooth, or even the newly proposed standard WBAN) so that data can be processed, analyzed, and then streamed to a cloud server for backup. In this case, a software tool (ideally, a tablet App) plays a key role by connecting the sensors and the cloud server. The App should be able to control the data sampling, visualization, analysis, and streaming such that the system can work autonomously and continuously without human intervention.

Different from existing "cloud-assisted" body area network architectures, our proposed BSNCloud system is "cloud-centered", i.e., it fully utilizes the processing power and storage capacity for optimal efficiency and minimum cost. In our design, the cloud server consists of three key components: (i) a web server that keeps receiving data

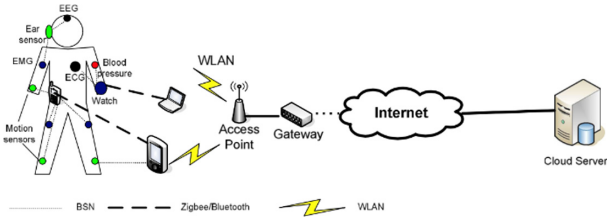

Fig. 1. System architecture of BSNCloud.

streaming and analysis requests from the data aggregator; (ii) a machine learning engine that automatically performs real-time or on-demand analysis as requested and responds with the results; and (iii) a MySQL database that stores various body sensor data sets, specific data analysis algorithms, as well as information about all data analysis sessions.

Essentially, BSNCloud aims at improving sensor-based event monitoring for remote patient care to a new level. Its scalable biomedical body area network computing platform has the potential of eliminating distance barriers and providing health care remotely. Another key aspect of the system is its open source, user friendly, and scalable cloud computing platform for sensor data storage, indexing, and analysis. It also allow researchers to upload, update, and contribute algorithms to the repository and conduct comparative study with the availability of a large number of high volume data sets.

4 Data Aggregator Design

Despite of efforts that enables sensor side programming on various tasks such as data sampling, event detection, data analysis, pre-processing, and data transmission, such approaches are many times limited to specific types of sensor motes. For example, the SPINE open-source software, built on TinyOS [19], provides a set of APIs that allow heterogeneous data collection, multiple forms of data analysis, threshold based event-detection and some forms of data reliability enhancement. However, the widely used TelOsb mote [20] is no longer produced. While Zigbee compatible sensors still exist, they lack enough capability on biomedical sensors. On the other hand, Shimmer Sensing [1] decided not to support TinyOS in their Shimmer 3 IMU motes and instead support Bluetooth only. There are two reasons: (i) there is a significant barrier for mobile devices to communicate with sensor motes in Zigbee while Bluetooth is readily available among all Android and iOS devices; (ii) it is much easier to control sensors at the data aggregator than coding the the sensor itself since vendors may choose different implementation languages such as nesC, C, and Java.

4.1 Data Aggregator Functions

In our system, we adopted Shimmer3 IMU development kit and combined sensor side functions and data aggregator side functions together. Core functions are implemented inside a comprehensive Android App, which is repackaged from the original Multi-Shimmer Template for Android. As shown in Fig. 2, specific data aggregator functions include:

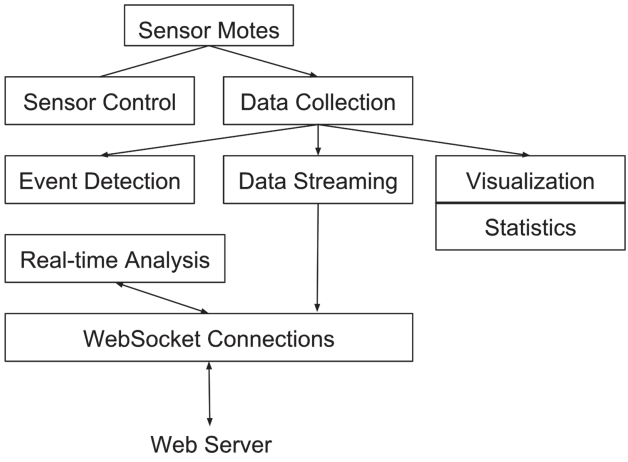

Fig. 2. Data aggregator modules.

- *Sensor Control* that includes turning sensors on/off for sampling as well as manually (through configuration) or automatically (through algorithm) manipulation of the frequency of data sampling. This allows the system to find the appropriate sampling rate for a specific application dynamically and sensors to be emulated for experimental purposes.
- *Data Presentation* that includes data visualization and signal statistics (e.g. min, max, average, standard deviation, time between peaks, binned distribution, etc.), which allows users to directly view the time series sensor data as well as key measures.
- *Event Detection* that handles lightweight analysis of data samples (e.g. threshold-based detection algorithms) to detect emergent events. It can be for both general and application specific functions. General functions will detect abnormal sample values based on configuration of sensors. Application specific functions will focus on identifying certain simple patterns indicating the occurrence of specific events. In the case when a critical event occurs, an alarm message is triggered for emergency and further real-time data analysis may be automatically initiated for diagnosis.
- *Data Analysis* that perform real-time on-demand analysis from users. It can also be triggered automatically when certain abnormal events are detected. In any case, a request is sent to the cloud server so that reasonably sophisticated, capable, and trained algorithms can be executed to analyze the N number of most recent or upcoming samples. This allows the data aggregator to utilize the processing power of the cloud server to run some complex machine learning algorithms for activity classification such as Fall Detection. A list of algorithms specific to each body sensor data can be available to the App user for selection. This makes the App scalable and eliminates the overhead of implementing a large set of analytics algorithms locally on the App.
- *Data Communication* that includes both sample streaming and real-time data analysis requests. For sensors with multiple signals (e.g., accelerometer), multiple socket threads should be used to transmit different signals of the same samples (e.g., X, Y, Z) simultaneously. Streaming can also be adapted based on battery power and network connectivity (i.e. offline or online with WiFi/3G/4G).

4.2 Data Streaming

Compared with traditional socket implementation that requires a server program stays running and waits for requests, WebSocket has several desirable features: (i) it's full-duplexed and enables streaming on top of TCP, which ensures the reliability of data streaming; (ii) it is supported by all browsers as well as programming languages such as Java, Javascript, and Python; (iii) its secure version is supported by Firefox 6, Safari 6, Google Chrome 14, Opera 12.10, and Internet Explorer 10. Essentially, WebSocket protocol offers a standard and universal implementation for interaction between a web browser (or client applications such as Android Apps) and a web server with lower overheads, facilitating real-time data transfer from and to the server.

In BSNCloud, a XAMPP Apache based PHP web server is setup and configured to receive requests from both Android App side real-time analysis and also Web Portal side on-demand analysis request. Whenever a user decides to stream and select the specific sensor data, a Java WebSocket connection is created for each signal (3 connections for accelerometers). Collected samples are temporarily stored locally in the data aggregator before delivered to the server. If the data aggregator goes offline, samples that have not been streamed yet will stay in local storage and will be streamed once Internet connection is restored. Each data streaming is uniquely indexed by, a stream_id, and then a sensor_id for each sensor, as well as the name of the specific signals (e.g., X/Y/Z for accelerometers).

4.3 Real-Time Data Analysis

Real-time analysis is a key feature of data aggregator. A user may follow a predefined schedule or decide based on certain observation. It is also possible that abnormality is identified by the event detector and therefore an analysis request is automatically initiated. In either case, a message is sent to the web server and then wait for the response. Frequent requests may make a separate WebSocket connection a necessity. Otherwise, existing WebSocket for data streaming can be used. Upon the receipt of the result, a message box is displayed in the App showing the result.

Due to its time sensitivity, real-time data analysis does not use a large data set and instead focus on either some most recent samples or upcoming samples. In addition, the size of such samples may affect the response time. While most recent samples can be analyzed immediately at the server side, upcoming samples will need time to accumulate and then get started.

5 Cloud Server Design

As the most important component of the proposed BSNCloud system, the cloud server aims at achieving large scale, distributed, and even parallel biomedical data management and analytics. In addition to the hardware resources that offer high performance computing power and large volume storage capacity, we designed four key modules for web server:

- *Data Repository* that stores, indexes, and manages body sensor data storage. MySQL is used to manage all data, which consists of both data sets and data analysis sessions. All raw data are stored as files locally in the hard drive with file location stored in the database tables. Data are classified as "Healthy", "Patient", and "Synthesis" to indicate their source. Data sets can be streamed from the Android App, uploaded data from lab/hospital or online repositories [21].
- *Algorithm Repository* that includes both contributed and user uploaded algorithms. These algorithms can be public and therefore open to all users or private and only for the owner to modify and update. Private algorithms can be contributed and become public once tested and approved. Sample algorithms include distributed clustering algorithms (e.g. K-means algorithm), parallel EM algorithms, parallel SVM algorithms, distributed random forest algorithms and distributed deep learning algorithms (e.g. Deep Convolutional Network, Auto Encoders, Restricted Boltzmann Machines). Based on the specific application, there may be multiple algorithms corresponding to the same body sensor data. To facilitate the contribution of algorithms, an online domain specific language (DSL) based template can be designed for uploading algorithm modules and test of algorithms. A biomedical DSL with uniform language interface [22] will support multiple data sources such as time series data from accelerometers, image data from vision sensor networks, as well as different types of biomedical body sensors.
- Machine Learning Engine that is able to execute various algorithms over selected data in the repository. The machine learning engine works closely with data repository and algorithm repository. It pre-processes raw data so that it fits specific input requirements by data analysis algorithm. For example, when a request is sent to analyze accelerometer data, the machine learning engine combines the raw data of the three signals X, Y, and Z and combine it to a CSV file, before feeding it to a specific algorithm. It should also incorporate major implementation environments such as Python, Java, C ++, and Matlab. Integration with existing frameworks such as TensorFlow is another possibility. The engine may also draw from MPI [23] for computational-intensive algorithms and the MapReduce model [24] for data-intensive algorithms.
- Web Portal that offers account management, data management, data analysis, and algorithm development. The web portal provides a software as a service (SAAS) level of user experience so that minimum overhead is achieved. Through the web portal, a user can view data sets by sensor types or algorithms, manage data analysis sessions, upload data sets, upload/contribute analysis algorithms, and perform new analysis.

The modules presented in Fig. 3 provide the basis for the cloud server's flexible and scalable design. Data received from the data aggregator will be formatted and stored on the server. Through the web portal, a user can request real-time analysis as data is being received, triggering result messages as desired. Furthermore, users will have the opportunity to upload algorithms or select algorithms from the expansive existing libraries and make analysis requests. The results will then be displayed in the web portal and the user can retrieve the data in a standard format. In addition, a user can request raw data from the cloud server. This provides an opportunity not only for researchers to analyze multiple data sets at once, but also for professionals to make real-time judgements based on patient data.

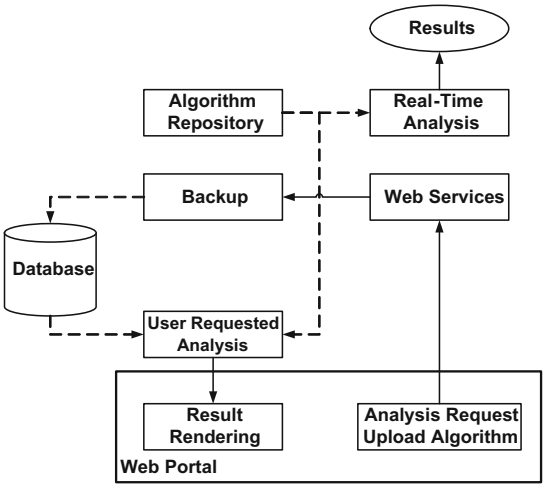

Fig. 3. Cloud server modules.

6 Implementation and Performance Evaluation

We have completed the prototype implementation of the proposed BSNCloud system and performed preliminary study on its performance. has acquired a 5-server cloud cluster with one server being the master and four others as slaves. Each server has 2x EightCore Intel® Xeon® Processor E52640 v2 2.00 GHz 20 MB Cache (95 W) and 8 × 4 GB PC314900 1866 MHz DDR3 ECC Registered DIMM. The total storage space is over 40 TB. Servers run Ubuntu and OpenStack systems. XAMPP-Apache PHP web server is setup for receiving both Android App and Web Portal.

Shimmer3 IMU mote is used for streaming. The mote has the dimension of 51 mm × 34 mm × 14 mm, runs 24 MHz MSP430 CPU with very low power consumption, light weight, and is wearable at different part of human body with a short/long strap. 5 colored LEDs are used to indicate the device status and operating mode. The data is highly accurate and scientifically reliable with a SD card of 7 GB capacity. It offers integrated 9DoF inertial sensing via accelerometer, gyroscope, magnetic and pressure sensors. The supported communication standard is Bluetooth.

A fall detection algorithm is implemented in Python and Java for experimenting the machine learning engine, real-time data analysis, and on-demand web portal data analysis. While multiple sensor data can be used, accelerometer data is the primary focus in this study. The algorithm is firstly trained with labeled data and then read csv files combined from the X, Y, Z raw data stored in the cloud server.

6.1 Android App

The developed Android App supports synchronized login with web portal. Figure 4 illustrates the sensor configuration as well as the main page with multiple functionalities such as event detection, set streaming on/off, enable statistics, perform real-time analysis, etc.

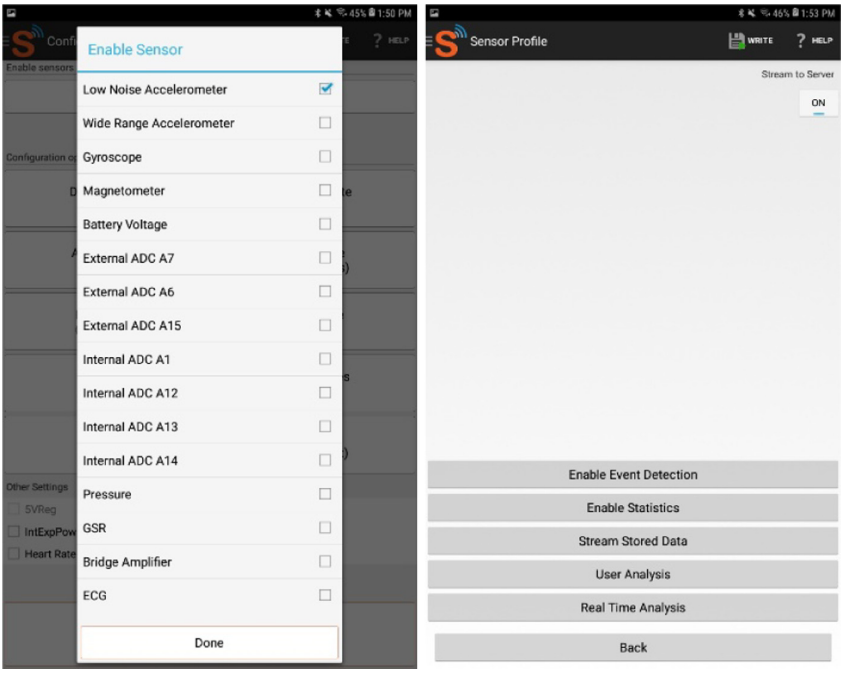

Fig. 4. Android App main page and sensor configuration.

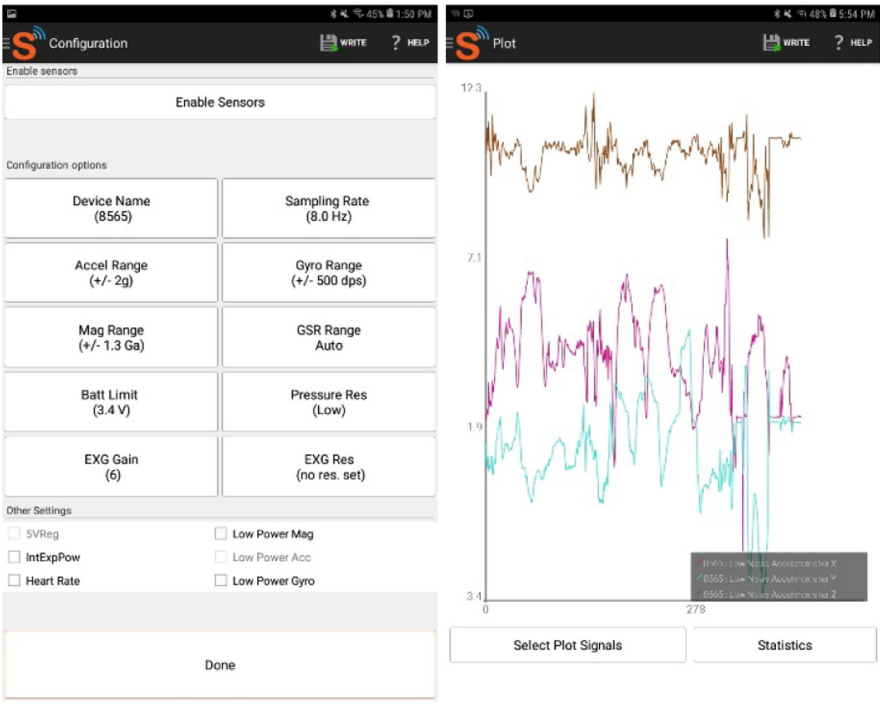

Fig. 5. Android App sensor configuration and data visualization.

Figure 5 illustrates the sensor configuration and data visualization that is provided by the MultiShimmer Template. However, MultiShimmer does not provide functions other than sensor control, data collection, and visualization. We have repackaged the app to include many additional functions, especially on streaming multiple body sensor signals to the cloud server side.

Figure 6 illustrates the real-time data analysis. A user may selects one sensor signal, sends a request to the server, and then wait for the response. Please be noted that for accelerometers, choose one signal X also requests Y, Z signals for analysis together.

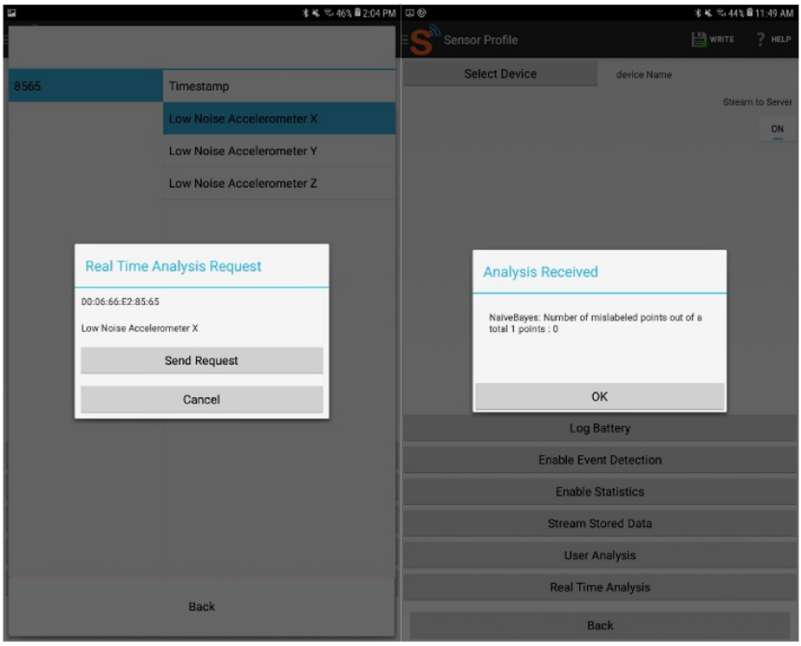

Fig. 6. Android App real-time analysis.

6.2 Web Portal

Figure 7 shows the dashboard page of the web portal. A user enters the system and select data and algorithms for data analysis. First, a specific sensor is chosen, then followed by a specific algorithm, and finally one data set. Each step of selection helps filter non-related algorithms or data sets. Algorithms are labeled by both method and application (e.g., fall detection). Before an analysis is initiated, a summary is presented for verification. The analysis results are displayed in the textbox at the bottom.

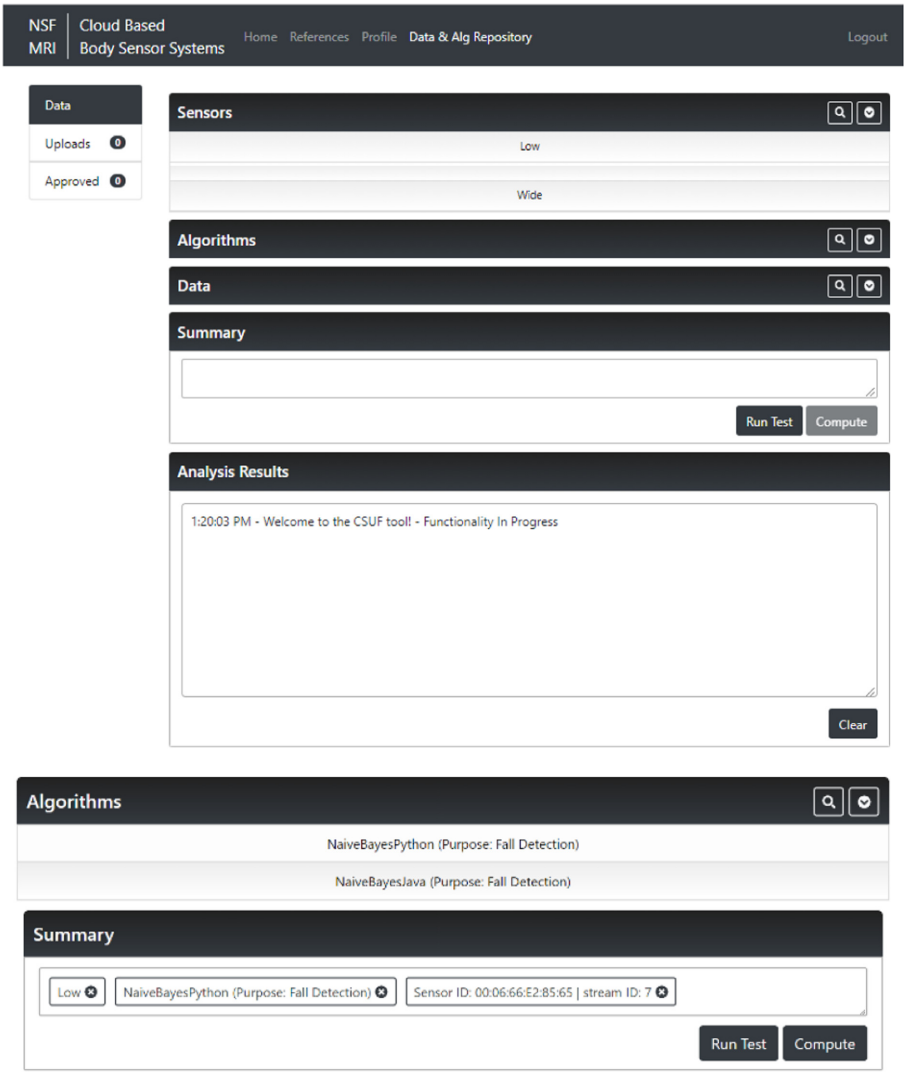

Fig. 7. Web Portal dashboard and data analysis pages.

6.3 Experimental Results

We have also conducted experiments to evaluate some key performance measure for the system. Figure 8 shows the measured throughput of a 8 Hz sampling rate accelerometer data streaming with 3 parallel WebSocket connections by varying the number of samples. It can be seen that the throughput is quite consistent over time for different sample sizes. It should be note that this throughput is not equivalent to the network capacity since it takes time to collect samples.

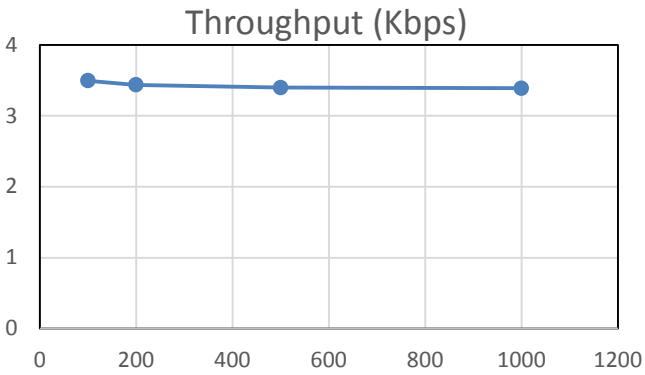

Fig. 8. Data streaming throughput vs sample sizes

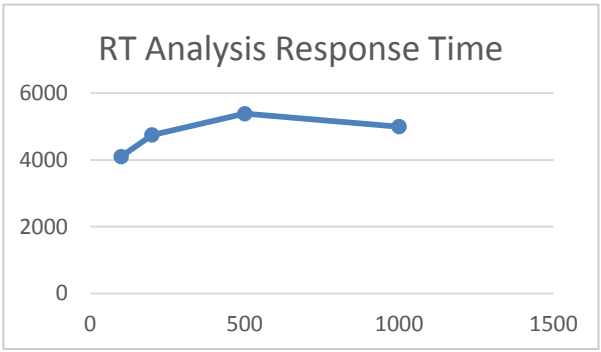

Fig. 9. Real-time data analysis response time

Figure 9 shows the real-time analysis response time in ms. It can be seen the results are sent back from the cloud server within a very short period of time. We then increases the sampling rate from 8 Hz to 16 Hz and 51.2 Hz, results show that the response time increases to over one min. Further analysis reveals that since the real-time analysis request is actually sent over an existing streaming WebSocket, which is already in near saturation status under high sampling rates. We will experiment with a separate WebSocket connection for real-time analysis and/or prioritize the message so that it is sent before queued sample packets.

Figure 10 shows the web portal data analysis response time for accelerometer samples collected for 1, 5, 10, 20, 30, and 60 min, respectively. It can be seen that results come back within 1–2 min. For reference, the data size of 60 min correspond to 2.6 Mbytes. Therefore, the high performance of the cloud server is clearly validated.

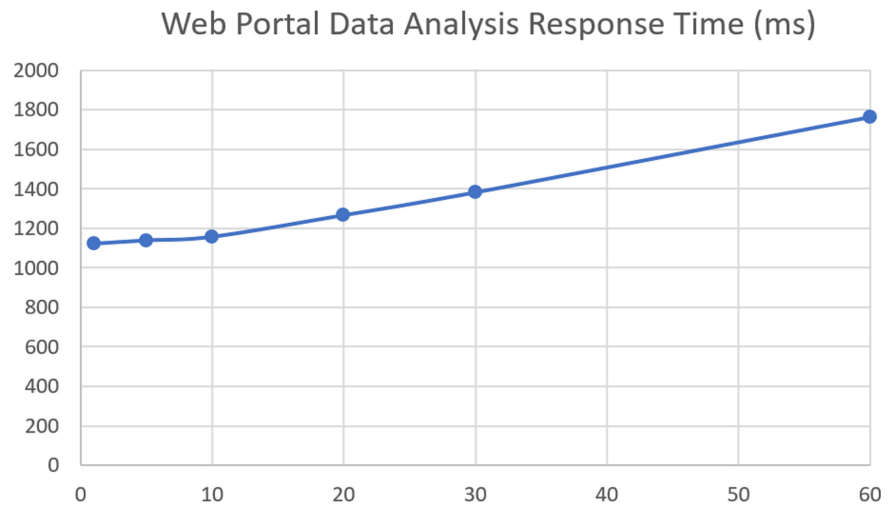

Fig. 10. Web portal data analysis response time for accelerometer samples.

7 Summary

In this paper, we have presented the design, implementation, and performance evaluation of BSNCloud, a cloud centered system for biomedical body sensor data collection, analysis, streaming, and storage. Key design decisions were made to maximize usability, efficiency, and scalability. The system helps facilitate the data analysis research and significantly reduces the cost and overhead by researchers. It is expected to grow with active use and contribution from the research community. In the future, we will address the issue of reducing the real-time data analysis response time under high sapling rate. We will also include more analysis algorithms with the proposed DSL based algorithm contribution template.

Acknowledgments. This work was supported by the National Science Foundation, CNS division (Award No. 1626586). We also would like to thank Rittika Shamsuddin and Barbara Mukami Maweu at University of Texas at Dallas for providing the Naïve Bayes fall detection algorithm.

References

1. Shimmer Enabling Softwares, Available http://www.shimmersensing.com/research-and-education/applications/data-aquisition-software-sensor-systems/
2. Qualcomm Wearables, Available https://www.qualcomm.com/products/wearables/
3. Intel IoT Developer Kit, Available https://software.intel.com/en-us/iot/devkit
4. Samsung Architecture for Multmodal Interactons: An open data platorm for innovaton, Available http://www.samsung.com/us/globalinnovation/innovation_areas/#digital-health
5. Fortino, G., et al.: BodyCloud: a SaaS approach for community body sensor networks. Future Gener. Comput. Syst. **35**, 62–79 (2014)
6. WebSocket. Available http://www.websocket.org/

7. Baronti, P., et al.: Wireless sensor networks: a survey on the state of the art and the 802.15. 4 and ZigBee standards. Comput. Commun. **30**(7), 1655–1695 (2007)
8. Bluetooth Low Energy Core Specification Version 4.0. Available http://www.bluetooth.com/English/Technology/Works/Pages/Bluetooth_low_energy_technology.aspx
9. SPINE Project. Available http://spine.deis.unical.it/
10. Shimmer Sensing. Available http://www.shimmersensing.com/ (2013)
11. Kuryloski, P., et al.: DexterNet: an open platform for heterogeneous body sensor networks and its applications. In: 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks (2009)
12. Fulford-Jones, T., et al.: CodeBlue: an ad hoc sensor network infrastructure for emergency medical care. In: Proceedings of International Workshop on Body Sensor Networks (2004)
13. Jiang, S., et al.: CareNet: an integrated wireless sensor networking environment for remote healthcare. In: Proceedings of the ICST 3rd international conference on Body area networks, ICST, Tempe, Arizona (2008)
14. Kurschl, W., Beer, W.: Combining cloud computing and wireless sensor networks. In: Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services, p. 512–518. ACM, Kuala Lumpur, Malaysia (2009)
15. Chu, X., Buyya, R.: Service Oriented Sensor Web, in Sensor Networks and Configuration: Fundamentals, Standards, Platforms, and Applications, N.P. Mahalik, Editor. 2007, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 51–74
16. Fortino, G., Di Fatta, G., Pathan, M., Vasilakos, A.V.: Cloud-assisted body area networks: state-of-the-art and future challenges. Wirel. Networks **20**(7), 1925–1938 (2014). https://doi.org/10.1007/s11276-014-0714-1
17. Li, M., Cao, Y., Prabhakaran, B.: Multi-level sample importance ranking based progressive transmission strategy for time series body sensor data. In: 2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM) (2015)
18. Read, N., et al.: Loss resilient strategy in body sensor networks. In: Proceedigs of 2011 ACM/IEEE International Conference on Body Area Networks (BodyNets 2011, accepted), Beijing, China (2011)
19. Levis, P., et al.: TinyOS: an operating system for sensor networks. In: Weber, W., Rabaey, J.M., Aarts, E. (eds.) Ambient Intelligence, pp. 115–148. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/3-540-27139-2_7
20. TelosB sensor data sheet. DOI=http://www.willow.co.uk/TelosB_Datasheet.pdf
21. Repository, UC Irvine Machine Learning Laboratory. Available https://archive.ics.uci.edu/ml
22. Huang, X., Zhao, T., Cao, Y.: PIR: a domain specific language for multimedia retrieval. In: Proceedings of IEEE International Symposium on Multimedia (ISM 2013), Anaheim, California, USA (2013)
23. Forum, M.P.I.: MPI: A Message-Passing Interface Standard - Version 2.2 - Sep.4. 2009, Message Passing Interface Forum (2009)
24. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation, San Francisco, California, USA (2004)