Leveraging Fog Computing for Healthcare IoT

Behailu Negash, Tuan Nguyen Gia, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, Tomi Westerlund, Amir M. Rahmani, Pasi Liljeberg, and Hannu Tenhunen

Abstract Developments in technology have shifted the focus of medical practice from treating a disease to prevention. Currently, a significant enhancement in healthcare is expected to be achieved through the Internet of Things (IoT). There are various wearable IoT devices that track physiological signs and signals in the market already. These devices usually connect to the Internet directly or through a local smart phone or a gateway. Home based and in hospital patients can be continuously monitored with wearable and implantable sensors and actuators. In most cases, these sensors and actuators are resource constrained to perform computing and operate for longer periods. The use of traditional gateways to connect to the Internet provides only connectivity and limited network services. With the introduction of the Fog computing layer, closer to the sensor network, data analytics and adaptive services can be realized in remote healthcare monitoring. This chapter focuses on a smart e-health gateway implementation for use in the Fog computing layer, connecting a network of such gateways, both in home and in hospital use. To show the application of the services, simple healthcare scenarios are presented. The features of the gateway in our Fog implementation are discussed and evaluated.

1 Introduction

The Internet of Things (IoT) is already around us. We can see it in many areas of our daily life; from wearable fitness tracking gadgets, smart home appliances, to smart self-driving cars [1, 2]. Healthcare is expected to be among the domains that

Behailu Negash, Tuan Nguyen Gia, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen
University of Turku, Turku - Finland, e-mail: {behneg, tunggi, armanz, imaazi, mizhji, tovewe, pakrli }@utu.fi, hannu@kth.se

Amir M. Rahmani
University of California, Irvine - USA & TU Wien - Austria, e-mail: amirr1@uci.edu

will be remodeled through IoT by enhancing its penetration and lowering the service cost [3, 4]. IoT offers potential to uninterrupted and reliable remote monitoring due to its ubiquitous nature while allowing freedom of movement for individuals. Activity tracking and following up of the heart rate and calorie intake are some of the application areas of commercially available IoT devices. In professional medical environments also, the quality of healthcare services can be enhanced by automating patient monitoring [5, 6, 7]. This enables a coherent healthcare system at home and in the hospital through the cloud [4, 8]. It is predicted that the current hospital-centered practice of medical care will be balanced by its home-based counterpart in 2020. This progress is expected to reach home-centered approach in the next decade[9]. To support such a shift and scaling, new computing approaches need to be developed.

Integration of currently available fragmented solutions towards all inclusive healthcare is a critical requirement, and at the same time potential, in IoT. This integration allows the synchronization of data from wearable and implantable devices with cloud based services [10, 11, 12]. One of the main areas of focus in achieving this integration is the architecture enabling such an interoperability. A common approach is to directly connect the sensor devices to Cloud services. However, due to the resource constraints of the end user devices, monitoring and actuating devices, an architecture with an intermediate computing layer is becoming the most widely used approach. This intermediate layer, known as Fog computing or edge computing [13, 14, 15, 16, 17], provides both generic and domain specific services for devices to enhance usability, reliability, performance and scalability among others.

Several unique characteristics of healthcare demand the use of Fog computing in IoT-based health monitoring systems. First, the nature of the sensor devices (especially many wearable or implantable ones) require resource efficiency more than many other domains. Second, the nature of communication required by these sensors is often streaming type of transmission. For instance, Electrocardiogram (ECG) signal collection requires a continuous communication with 4kbps bandwidth per channel. Third, due to the criticality of the application domain, immediate response to important events gathered by sensor nodes is mandatory. The overall system needs a high level of reliability where patterns of physiological signals need to be recognized in real-time. Moreover, providing the freedom of movement of individual under a medical supervision through IoT devices is also a key requirement. These functions can be mainly supported by services in the Fog computing layer. Some of these services provided by the Fog layer are presented in this chapter focusing on the healthcare application domain. In general, this chapter concentrates on the following main areas:

- Healthcare IoT system requirements, along with services of the Fog computing layer to address them, are described.
- System architecture of a Fog-enabled healthcare IoT system is presented.
- Performance and advantages of the Fog computing layer services via a proof-of-concept full system implementation are demonstrated.

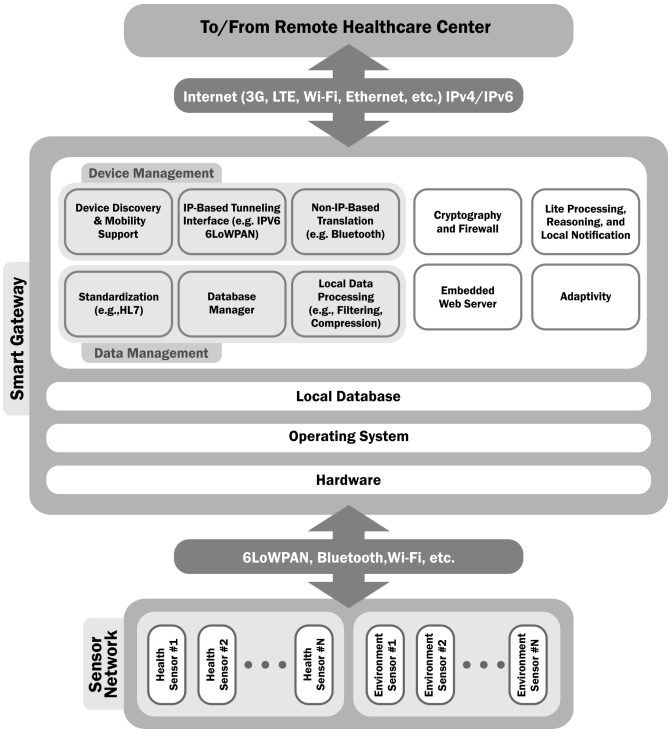

Fig. 1 Services provided by the Fog layer

2 Healthcare Services in the Fog layer

Fog computing layer provides computing, networking, storage and other domain specific services for IoT systems. The healthcare domain has a set of requirements that uniquely identify it from other IoT applications; for instance, one of the use cases of healthcare IoT is remote monitoring, which demands a high degree of reliability. Unlike other domains, the security and privacy aspects of healthcare are also of critically important. This section highlights the services that can be provided by the Fog layer with a focus on healthcare. The physical proximity of Fog layer to Body Area Network (BAN) of sensors and actuators allows us to address the requirements of healthcare IoT. Some of these services are generic and can be used by any application domain. Figure 1 show a generalized view of the services of the Fog layer, which are discussed separately in the following sections.

2.1 Data Management

Data management has an important role in Fog computing by which the sensory data is locally processed to extract meaningful information for user feedback and notifications along with system plan adjustments. According to the system architecture, Fog layer continuously receives a large amount of sensory data in a short period of time from the sensor network, so it should manage the incoming data to provide a fast response regarding various user and system conditions. This task becomes more significant in healthcare scenarios since latency and uncertainty in decision making might cause irreversible damages for the patients. According to the different functionalities of the data management task, we introduce it in five different units, all of which are essential in a smart e-health gateway. These units are local storage, data filtering, data compression, data fusion and data analysis.

2.1.1 Local Storage

The gateway needs to store received data from several sources in a local storage to be able to utilize it in the near future analysis. The operating system of the gateway has a file server and a database to store and recover data. The local storage in the gateway can be used to store files in encrypted or compressed format based on the type, size, and importance of data. Using local storage, the gateway is able to export data to medical standard formats such as Health Level Seven (HL7) [18] if required. Other functionalities of the gateway such as data analysis, compression, filtering, and encryption are also dependent on a temporary storage. The speed of data transfer between cloud layer and Fog layer is limited to network speed and most of the local calculations in the gateway are limited by its processing power. So, in case of any imbalance in computation time and transfer time, the local storage acts as a local cache memory to make data flow continuous. Local storage also helps saving data while the Internet connection between gateway and cloud server is not available. The gateways sends saved data to the cloud when it connects to Internet again. This local storage is shown in Figure 1 as database manager.

2.1.2 Data Filtering

Data filtering is the first data processing unit to implement filtering methods at the edge after receiving data from the sensor network. To obtain patient medical condition, various bio-signals such as electrocardiogram (ECG), electromyography (EMG) and photoplethysmogram (PPG) are collected using relevant probes. These signals usually include complex shapes with a small amplitude (i.e., milivolts) and consequently are susceptible to some unavoidable noises and distortions accumulated during the health monitoring. Such noises are thermal noise, electromagnetic interference noise and electrode contact noise. Available light-weight filtering in some of the sensor nodes reduces these accumulated noises although it might be in-

sufficient in practical cases. Therefore, the data filtering unit in the Fog layer enables to remove noise and to increase aspects of the signals (e.g., signal-to-noise ratio) using various filters (e.g., finite impulse response (FIR) filter) before any other local data analysis.

2.1.3 Data Compression

For reducing a large amount of transmitted data over a communication network, data can be compressed by lossless or lossy compression methods. In healthcare IoT applications, lossless compression is more preferable in most of cases because lost data can cause inappropriate disease diagnosis. Although lossless compression algorithms can recover original data accurately, they are often complex. Correspondingly, they are not suitable for sensor nodes which are resource-constrained regarding to battery capacity, computation and memory capacity. For example, lossless ECG compression methods [19, 20, 21] cannot be run in many types of sensors. In some cases, these lossless algorithms can be successfully operated at sensor nodes but they cause a large power consumption and latency. Thanks to Fog computing, all mentioned limitations at sensors can be avoided by switching all burdens of sensor nodes to the gateway which handles these burdens efficiently while respecting the real-time requirement.

2.1.4 Data Fusion

Data fusion is the data processing unit to integrate sensory data from multiple sources to obtain more robust data and meaningful information. This processing unit efficiently decreased the volume of sensory data by removing redundant data and replacing new information. Consequently, this data reduction improves local data analysis and data transmission to the remote servers.

Data fusion can be divided into three classes as complementary, competitive, and cooperative [22]. First, complementary data fusion combines (at least) two different data from different sources to obtain a more comprehensive knowledge in the Fog layer. For instance, combination of patient health parameters with surrounding context data provides more data about patient condition. Second, competitive data fusion improves data quality as well as system decision making by integrating data collected from one source with (at least) two sensors. For example, a more robust heart rate value is extracted using values from a ECG signal and values from a respiration signal. Third, cooperative data fusion provides new information from one source using different sensors. Determination of the patient medical state using vital signs (e.g., heart rate and respiration rate) is an example of an cooperative data fusion.

2.1.5 Data Analysis

Data analysis unit at the edge enables the healthcare system to process the sensory data locally. This unit improves the system performance by decreasing response latency and data transmission to the cloud servers. For example, in case of patient health deterioration, the emergency event is detected and responded early and effectively since the data is processed locally instead of being transmitted to the cloud and awaited for the appropriate response.

In addition, the data analysis unit improves data reliability as well as system consistency. Connectivity losses and limited bandwidth for data transmission are unavoidable events in long-term remote health monitoring because the patient might engage various activities in different environments. Hence, data analysis at the edge could manage the system's functionality locally, store the sensory data as well as the calculations in a local storage and subsequently synchronize the Fog layer with the remote cloud after reconnecting to the network.

2.2 Event Management

Several important events happen during patient monitoring which may be a change in vital signs, activities or environment of the patient. Each event triggers a specific action in gateway or switches to a learned behavior in personalized systems. Fog computing provides low latency communication which helps to notify health experts, caregivers and even patient very fast in case of a serious event. In such case when an immediate response is necessary in form of medical actions or automatic system actuation, the event management service ensures on time and proper signal delivery. The real-time and fast response of actuators are important in some medical events like changing the frequency of nerve stimulation according to heart rate or adjusting automatic insulin pump with blood glucose level. Other emergency events might also happen which required to notify rapid response team, caregivers or family members of the patient.

2.3 Resource Efficiency

In healthcare IoT applications, resource efficiency is one of the most essential requirements because failure in resource management can cause serious consequences from sensor nodes' malfunction to imprecise disease diagnosis. Particularly, energy consumption of sensor nodes and latency of gathered data presented at end-users' terminals must be attentively considered. They are discussed in details as the followings:

2.3.1 Energy efficiency of nodes

Sensor nodes in health monitoring systems are typically small and resource-constrained, such as small battery capacity, but it is required that sensor nodes must be able to operate in an appropriate duration such as a whole day or even a few days. In order to fulfill these requirements, sensor nodes must operate efficiently in terms of energy consumption. Several methods including software and hardware-based techniques can be applied for achieving the task. For instance, sensor node's hardware should be designed for particular purposes instead of general tasks. This method helps to save energy consumption by avoiding unused components or high power consuming components. However, it is more challenging for designing an energy efficient nodes than customizing software running at the nodes. Particularly, the software must be able to perform primary tasks sensor nodes while it must be extremely simple for reducing computation time. For example, sensor nodes in IoT-based fall detection systems only gather digital data and transmit the data to Fog layer which runs complex fall detection algorithms [23]. Accordingly, sensor nodes' power consumption can be reduced dramatically. In another example, several approaches [24, 25] show high levels of energy efficiency at sensors nodes when running ECG feature extractions at Fog layer instead of sensor nodes.

2.3.2 Latency

In health monitoring applications, latency is critical because it can cause inappropriate disease analysis and delay in decision making. Latency in IoT-based application comes from both processing and transmitting of data such as transmission latency from sensor nodes to end-users via gateways, Cloud, and processing at sensor nodes, gateways. In many cases, transmission latency and processing latency are often in a trade-off relationship. However, processing data does not always guarantee to reduce the total latency. In some cases, data processing even increases the total latency. In order to reduce the total latency and fulfil time requirements of real-time health monitoring, a distinct method must be applied for a particular sensor nodes and applications. In other cases, simple filtering methods for eliminating noise and invalid data can help to reduce a large amount of data transmission as well as the total latency. For example in an ECG feature extraction application, the feature extraction algorithm must be run at Fog instead of sensor nodes for reducing the total latency[25].

2.4 Device Management

Device management encompasses many areas of IoT infrastructure. In this section, the focus of the discussion is on device management from the point of view of device discovery and maintaining connectivity during mobility.

2.4.1 Discovery and mobility

The resource constraints of devices in the sensor and actuator network has been mentioned briefly earlier. For battery powered devices, the life time of the battery is precious and needs a proper management. Devices should go to sleep state in a controlled way whenever they become idle. Any communication that occurs when the device is in the sleep state needs to be taken care of by the Fog layer. In healthcare scene, a patient wearing medical sensors and moving from a location to another changes the corresponding gateway that handles communication. This means that a sleeping sensor can wake up at a vicinity of different gateway, i.e. it was connected to another gateway when it entered to the sleep mode. Device discovery service helps another device that requires connection to a sleeping sensor node to discover it and gracefully handle the sleeping cycle. A detailed function of this service can be found at [26].

To provide the freedom of mobility for the patient, and to do so without consuming much of the scarce resources, the Fog layer service is used to handover the locally retained information from one gateway to another. Once the device is discovered in a new location, the handover takes place to seamlessly continue the monitoring process at the new place. A simplified implementation of mobility and device discovery working together is presented in [26]. These services, like some of the other Fog computing services, can also be used for other domains as well.

2.4.2 Interoperability

The Internet of Things is composed of heterogeneous set of communication protocols, platforms and data formats. There are many standardization efforts to establish uniformity among the different components. The current vertical segmentation of applications need to be bridged to ease the creation of holistic healthcare applications. Traditionally interoperability face the challenge the resource constraints of the majority of end devices. The Fog computing layer plays a significant role in providing services that leverage the proximity of this layer to end devices and hence ease interoperability. These services act as an adapter among different communication protocols, data formats and platforms. Preliminary work can also be accomplished at the Fog layer to provide semantic interoperability at the cloud to give meaning to the data collected through the sensors.

2.5 Personalization

System behavior can be configured for different applications of Fog computing in advance or at the run time. This, however, might be insufficient for healthcare scenarios since users might have various medical conditions and engage different activates in different environments. Therefore, a dynamic plan for the system is required

to not only personalize the system behavior according to the user requirements but also adaptively adjust the system over time, especially in emergency situations. In this regard, it improves the health applications (e.g., local decision making) as well as optimizing the system performance (e.g., energy efficiency).

Personalized system behavior can be defined for various health applications using rule-based techniques and machine learning algorithms. To this end, different priorities and modes are defined for the system parameters (e.g., sensor sampling rate and data transmission rate), and according to the patient conditions, the appropriate values are selected. In addition, the priorities become personalized regarding the medical history of the patient. In a simple example, if a heart failure was detected for a patient during the monitoring, the system would learn to increase the priorities to heart related parameters.

2.6 Privacy and Security

In general, security is critical for all applications and it is more essential in cases of healthcare because a single insecure point in a system might cost a human life. For example, it is reported that a insulin pumper in a IoT glucose management system can be hacked within 100 feet [27]. In order to provide a secure IoT healthcare system, the whole system including sensor nodes, gateways, Fog and Cloud must be attentively considered. If one of devices or components is hacked, the entire system can be controlled or manipulated by hackers. For example, several methods such as AES-128 or CMAC-AES-128 can be applied at sensor nodes and gateways for data encryption and decryption, respectively. At gateways, IPtable offered by Linux can be used for configuring IP tables giving grant permissions to particular communication ports [28]. Although these methods can improve some security levels, it cannot be seen as robust methods for protecting the entire system. On the other hand, other approaches in the literature [29, 30, 31] provide a high level of security. However, they cannot be applied in IoT systems because their complex cryptographic algorithms are not suitable to resource-constrained sensor nodes. To address the security issues in IoT healthcare systems, Rahimi *et al.* [32, 33] recently introduce end-to-end secure framework. The framework can provide efficient authentication and authorization for Health-IoT systems while the main parts, consisting of several complex security algorithms, are run at Fog layer.

3 System Architecture of Healthcare IoT

The architecture of a system provides information about the components, interactions and the organization of the parts. It is one of the key elements for achieving graceful scaling and performance. Moreover, it is designed to meet the functional requirements of the application domain. Among the non-functional requirements

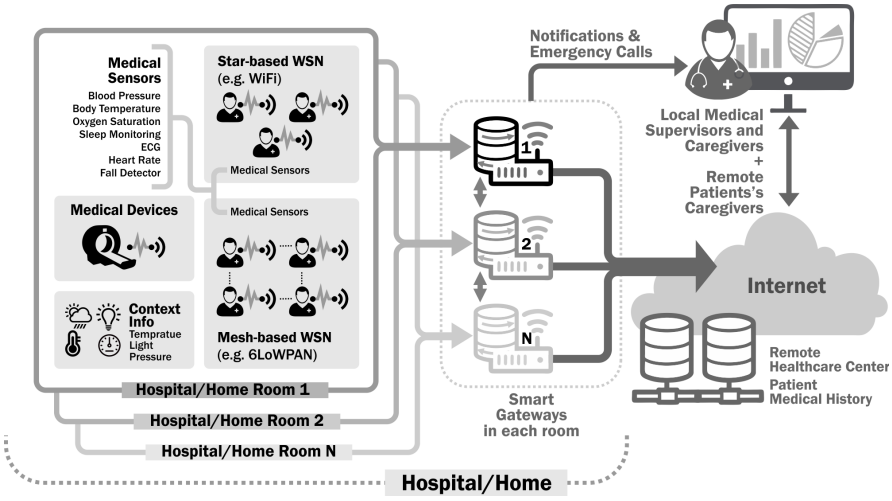

Fig. 2 Healthcare IoT system architecture

that constrain the system architecture design, few of these are scalability, usability and performance. One of the main challenges is the huge number of devices that are getting connected to the Internet. Connecting more devices cause the available resources, such as bandwidth and computing power, to be shared by more nodes leading to quality and performance degradation. However, the criticality of the application domain makes this degraded infrastructure unacceptable. In addition, a large proportion of these devices are resource constrained. This shortage of resources add more design constraints to the architecture design.

The introduction of Fog computing layer between the end devices and the cloud has contributed to a design shift towards IoT based systems thereby alleviating the challenges mentioned above. The Fog layer can be used to provide a wide variety of services to support the resource constrained nodes [13]. In a healthcare IoT application, an overview of the architecture is shown in Figure 2, where resource constrained nodes can be wearable or implantable sensors and actuators. A patient can be at home or in hospital and the sensors send the values of the physiological signals they read to a local gateway in the Fog layer. The Fog layer has local services handling data, events, devices and the network. In healthcare scenario, this architecture is composed of the following three main parts in each layer.

1. **Medical sensors and actuators:** Mainly connected through low power wireless communication protocols, serves in identifying subject, reading physiological signals and act in response to a command from the Fog layer.
2. **Smart e-health gateways:** Distributed network of gateways form the Fog layer and serve the underlying sensor and actuator network. It usually has multiple interfaces that enable it to communicate with different protocols. It hosts a wide

variety of services and act as a bridge to the cloud. This is the main focus of the chapter.
3. **Cloud platform:** Back-end system where the data persists and stakeholders get access to the system, through web or mobile interface. It is the point of interface with other systems, such as hospital information system and patient health records.

4 Case study, Experiments and Evaluation

Previous sections discussed theoretical aspects of Fog computing and highlight its benefits in healthcare IoT systems. In this section, we discuss the detailed implementations and experiments conducted in our work.

The architecture of our system consists of three layers, shown in Figure 4. The first layer is sensor network containing three group of sensors: a set of medical sensors to record vital signs (heart rate, respiration rate, body temperature, blood pressure, blood oxygen level, and ECG), a set of environment sensors to find the situation of the patient (light, temperature, and humidity), and activity sensors to find the body movements, posture, and step count.

The Fog computing takes place in the second layer where a network of gateways collect data from sensor nodes via Bluetooth and Wi-Fi wireless communication. A Bluetooth service and Node.js UDP server receive and store the data in separate files for each sensor and each patient. An Apache server also runs a service in the background which calls a Python script to read and process data from local files. Because data is collected from several sources with different properties, the gateway performs an adaptation to the data. The adaptation includes handling different communication protocols (in UDP server script) and unifying the sampling rate which varies from 5 samples per day for step count to 250 samples per second for ECG (via Python script). The python script is also responsible for preliminary data analysis to detect abnormalities before in-depth analysis at the cloud layer. This service first reduces the signal noise from ECG signal using a bandpass filter (0.5 Hz – 100 Hz) with Finite Impulse Response (FIR). Then using RR intervals the heart rate is calculated. The raw and filtered ECG signal are demonstrated in Figure 3.

Data fusion is implemented on the sensory data in the Fog layer. In this case, two heart rate signals are acquired from the user using two different devices. As illustrated in Figure 5 with blue and red dots, the heart rate values from two devices may not be identical due to the noise and measurement inaccuracy. In our approach, first, we remove outliers and out-of-range heart rate values (e.g., zero values) from the data. These outliers are mainly because of loose probe connections over the course of monitoring. Second, we implement a weighted average on the two heart rate values to improve the signal-to-noise ratio. The green line in Figure 5 shows the calculated heart rate. In the calculation, the average weights are defined with respect to the sensor accuracy mentioned in data-sheets.

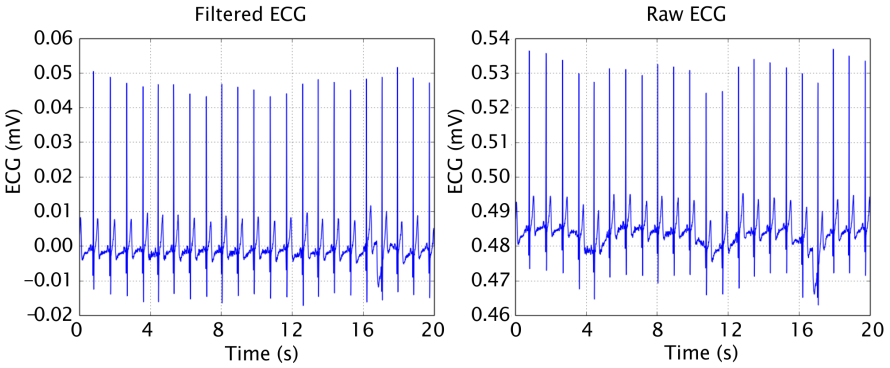

Fig. 3 Raw and filtered ECG data

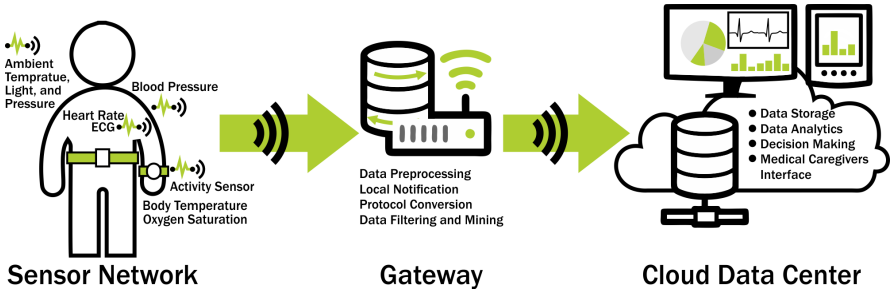

Fig. 4 Healthcare IoT system architecture

In addition to the aforementioned data processing, data compression is implemented in the Fog layer. Before transmitting data to a remote cloud, the data is compressed and stored in the local storage of the Fog layer to provide a backup in case of Internet connectivity is lost. In this case study, *tar* method is utilized to create a temporary file for the sensory data, and then the file is compressed using *tar.gz* method while the size of the file is increased to a certain value. We set this value to 500 KBytes. The relation between the compression ratio and the file size is represented in Table 4. As illustrated, in larger file size, the compression ratio is higher although this improvement is insignificant for the files larger than 500 KByte.

In order to improve the data security, an asymmetric encryption method using *Crypto* library [34] in Python is implemented in the Fog layer. With this method, the compressed data is encrypted with a public key in the Fog layer and is decrypted with a private key by the data collector service in the cloud.

There is also a local storage service in the Fog layer which consists of a file server to store the files and a MySQL database to keep the indexes and attributes of them. One objective of this case study is to store data in the cloud and use the benefit of available processing power of the server.

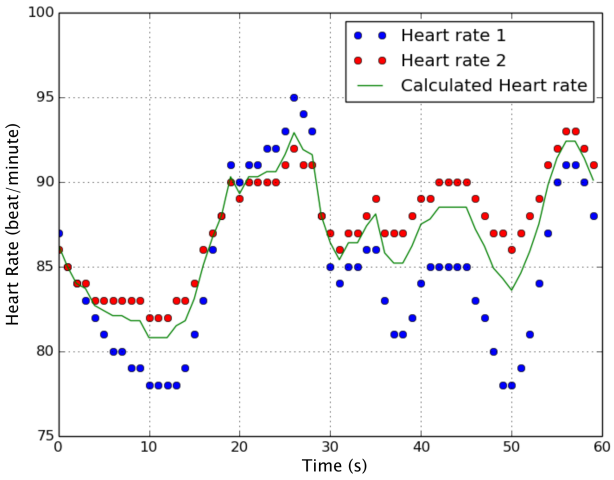

Fig. 5 Heart rate data fusion

The gateway in the Fog layer transmits collected data to the cloud server when the Internet connection is available. When the Internet is unavailable, the gateway marks unsent files in the database and tries to re-transmit once the Internet connection is re-established. The local storage service periodically synchronises the database with the cloud server and removes out-dated and duplicated files from the gateway storage and database.

The third layer in the architecture of our system is the cloud server which is responsible for receiving data from the Fog layer, processing and storing. Cloud server uses stored data together with the history of the patient to analyse the health status of a patient. An interface for caregivers is created to send alerts, reports and plot vital signs and other sensory data in real-time. Figure 6 shows the web-based user interface developed by HTML5 WebSocket. The cloud server provides also information and notifications via a mobile application for patients and caregivers.

4.1 Performance measures

The entire system, including sensor nodes, smart gateways with Fog, Cloud and client back-end parts is implemented. The sensor nodes are able to gather bio-signals i.e. ECG, EMG and contextual data i.e. temperature and humidity. The sensor nodes are created by a combination of medical sensors, micro-controller and wireless communication IC. For evaluating interoperability of our gateways, several types of sensor nodes are used such as Bluetooth nodes, Wi-Fi nodes, and 6LoWPAN nodes. These nodes are formed into a mesh or star network depending on particular wireless communication protocols. In detail, Bluetooth nodes, including

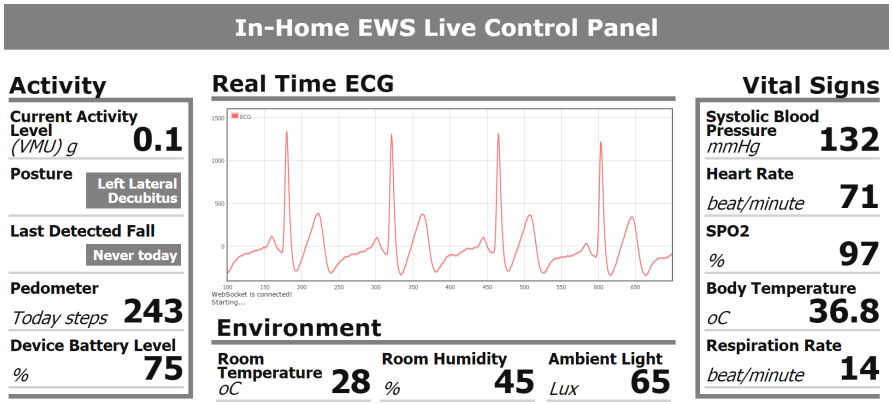

Fig. 6 Live control panel web interface

Bluetooth Classic and Bluetooth low energy (BLE), are created by integrating low-cost HC-05 and BLE ICs into micro-controllers (i.e. ATMEGA128, ATMEGA328P, ARM Cortex M3), respectively. Similarly, Wi-Fi nodes are constructed by using the same micro-controllers and low cost ESP8266 boards supporting a high speed Wi-Fi communication while the 6LoWPAN node is built from a CC2538 module. For gathering data, these nodes are connected with several sensors via SPI, I2C or UART. For example, in order to collect EEG and EMG signals, these devices are connected with analog front end devices (i.e., TI ADS1292 [35] and TI ADS1298 [36]). With the purpose of efficient managing resources of sensor nodes i.e. power consumption and hardware distribution, embedded operating systems are installed in sensor nodes. For instance, RTX, FreeRTOS and Contiki are used in Wi-Fi, Bluetooth and 6LoWPAN nodes, respectively [37, 38, 39]. Specification and power consumption of these sensor nodes when transmitting data at 8.7kbps are presented in Table 1 and Table 2.

Gateway of the system is built from several devices such as Pandaboard [40] and Texas Instruments (TI) SmartRF06 board integrated with CC2538 module [41] and MOD-ENC28J60 Ethernet Module [42]. Due to the Pandaboard's advantages of high performance 1.2Ghz dual ARM Cortex-A9 core processors, large amount of memory and hard disk capability, the Pandaboard is used as a core component of the gateway for performing algorithms and services [40]. In addition, Pandaboard can support several operating systems comprising of Windows CE, WinMobile, Symbian, Linux, and Palm. By applying one of the operating systems, resources can be managed efficiently. For example, collision caused by simultaneous access to the same hardware can be avoided by hardware abstraction in Windows and Linux. Although the Pandaboard cannot support all wireless communication protocols, it support popular wireless and wire communication protocols such 802.11 b/g/n , Bluetooth and Ethernet. For dealing with other wireless protocols, the Pandaboard is equipped with Texas Instruments (TI) SmartRF06 board which is integrated with CC2538 module for supporting Zigbee and 6LoWPAN [41].

Table 1 Hardware specification of sensor nodes and UT-GATE

Device	Micro-controller	Flash (KB)	RAM (KB)	EEPROM (KB)	Clock (MHz)	Voltage (V)
Zigduino R2	ATMega 128	128	16	4	16	3.3
Arduino Uno R3	ATMega 328P	32	2	1	16	5
Arduino Mega	ATMega 1280	128	8	4	16	5
Arduino Due	ARM CortexM3	512	86	-	84	3.3
Zolertia Z1	MSP430	92	8	-	16	3.3
TI-CC2538	ARM Cortex M3	Up to 512	32	-	32	3.3
Pandaboard	Dual-core ARM Cortex-A9	Up to 32000	1000	-	1200	5

Table 2 Power consumption of sensor nodes when transmitting at 8.7kbps

Communication type	Current (mA)	Voltage (V)	Power consumption (mW)
6LoWPAN node	24.6	3.3	81.2
Wi-Fi node	114	3.3	376.2
Bluetooth 2.0 node	56.9	3.3	187.7
BLE node	31.6	3.3	104.4

Table 3 Sensing to actuation latency comparison for local Fog computing based vs. remote cloud based scenarios

Latency of the sensing-to-actuation loop	using Wi-Fi (ms)	using BLE (ms)
Fog-based (locally via UT-GATE)	21	33
Cloud-based (remotely via the Cloud)	161	176

Although Fog computing is capable of providing a large number of advanced services such as local storage and push notification, roles of Cloud can not be lessened. For example, limitations of the local storage of Fog computing units, such as a limited storage capacity and limited accessibility, can be solved by Cloud connection. In our implementation, the remote server is built in Cloud for handling client requests, running complex algorithms and enhancing Fog layer services. and responding with data by providing the requested data along with graphical user interface. The free service provided by ”heliohost.org” including MySQL server with remote access facility is used.

Depending on particular services, Fog or Cloud or a combination of Fog and Cloud computing should be used to implement the services. For example, table 3 shows that decision making at Fog layer is more efficient in terms of latency than at Cloud. Two protocols including Wi-Fi and BLE are used during latency measurements. The currently implemented functionalities of the gateway, called UT-GATE ,such as data fusion, data compression, local storage are discussed in detail in the following subsections.

Table 4 Compression results at UT-GATE and latency reduction

Number of senors nodes connected to UT-GATE	**1**	**2**	**5**	**10**	**50**
Number of analog channels	8	8	8	8	8
Data size (120 samples) (B)	8400	16800	42000	84000	420000
Compressed data size (B)	808	1597	3893	7696	38333
Compression time (ms)	3.1	4.4	9.2	16.6	73.0
Decompression time (ms)	3.3	4.6	11.3	23.0	83.9
Total time of comp., tran., and decomp. (ms)	12.86	21.77	51.64	101.16	463.5
Transmission time of non-processed data (ms)	67.2	134.4	336	672	3360
Total latency reduction (%)	80.8	83.8	84.6	84.8	86.1

4.2 Data Compression at Fog layer

To reduce the amount of transmitted data, and hence to make the system more energy efficient, data compression is applied at the Fog layer. The compression rate depends on a particular data compression method. For example, lossy compression methods have a high compression ratio such as 50:1 while lossless compression method can achieve compression ratio about 8:1 to 9:1. As mentioned, loss of data can cause serious consequences in a healthcare domain. Therefore, a lossless compression is applied in our application for compressing bio-signals such as ECG, and EMG. However, not all lossless compression methods can fulfill the latency requirements of real-time applications defined by IEEE 1073 [43]. In our implementation, an LZW [44] algorithm is used at Fog layer because it provides rapid compression and decompression without losing any data. Computation efficiency of the LZW compression algorithm increases when the number of inputs increases. For example, computation time increases 8 times when the input size increases 10 times. However, in this case, latency raises dramatically. Hence, it is recommended that the data size of inputs used for data compression must be carefully chosen to achieve the computational efficiency and fulfil the real-time requirements of health monitoring.

Table 4 shows time results of LZW compression and decompression at the gateway. In our application, EMG data is acquired from 8 channels with a data rate of 1000 samples/second per channel in which each sample size is 24bit. The LZW algorithm uses 120 samples as its inputs for achieving a high computational efficiency. The number of sensor nodes connected to the gateway varies from 1 to 50 nodes. The results show that when the gateway receives more data, it operates more efficiently.

4.3 Benefits of Data Processing at the Fog layer

In real-time health monitoring applications, real-time streaming data is captured and processed for health indicators. In an IoT-based monitoring system shown in

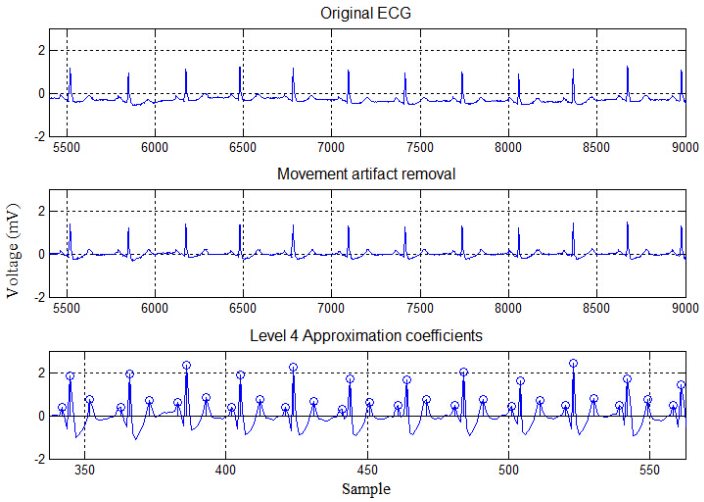

Fig. 7 ECG processing implementation

Figure 2, computation resources for signal processing are distributed in each layer but with different capacities. As illustrated in Section 2.3, the benefits of processing streaming data in Fog layer on the system is twofold:

- Bring energy efficiency to sensor nodes by shifting streaming data processing from sensor node to Fog layer;
- Shorten data transmission latency from sensor nodes to cloud by reducing data size.

To demonstrate these two benefits, tests in three scenarios are conducted in the implemented system. They are, applying signal processing on i) a sensor node, where gateway receives and passes processed data to cloud, ii) gateway, where raw data is received and processed data is transmitted to cloud, which can be also considered as Fog-assisted cloud computing, and iii) the cloud, where raw data is directly passed to it through sensor node and Fog layers. The algorithm of ECG signal processing for Q, R and T wave extraction is applied on MIT-BIT Arrhythmia database [45]. The signal processing on ECG data includes noise reduction, wavelet decomposition and peak detection for extracting waves, as shown in Figure 7. In addition to peak detection, heart rate is calculated from R to R interval. In these three scenarios, energy consumption of sensor node, sample size delivered from gateway to cloud and its data transmitting time are measured and calculated for comparison.

Regarding energy efficiency test on sensor nodes, 1000 samples of ECG data (360 samples per second) are saved at Aruidno Due, which together with ESP8266 Wi-Fi module acting as a sensor node. To simulate real-time ECG data processing on sensor node for every 1000 samples, the sensor node first waits for data gathering and then proceed with processing and transmission. The overheads of signal processing on sensor node regarding energy consumption is accessed by monitor-

Table 5 Providing energy efficiency for sensor nodes

	Time	**Current**	**Energy**
Processing at sensor node (collection+execution)	2.78s+101ms	10.1mA+106.8mA	127.34mJ
Process at gateway/cloud (transmitting raw data)	95ms	180mA	56.34mJ

Table 6 Number of transmitted samples from Fog to cloud

	Processing at sensor node or UT-GATE	Processing at cloud	Improvement (%)
No. of Samples	259	1000	74.1

Table 7 Latency reduction between gateways and cloud server for 240KB of raw samples

Network condition	Data rate (Mbits/s)	Raw samples trans. time (ms)	Raw samples proc. time + trans. time of processed samples (ms)	Latency reduction (%)
Light load	18	106.6	96.3+6.6	3.5
Medium load	12	152.2	96.3+9.5	30.5
Heavy load	9	213.3	96.3+13.5	48.5

ing execution time and current consumption. The results from three scenarios can be seen in Table 5, where about 55.7% can be saved for sensor node when the outsourcing signal processing task to posterior layers.

To look into the benefits of Fog computing on data transmission latency between gateway and cloud server, sample size is first calculated in three scenarios, presented in Table 6. The saved data transmission time from Fog to cloud in practice under three different network conditions is presented in Table 7. It can be seen that, for every 1000 ECG data samples, sample size is reduced by 74.1% at cloud due to signal processing at two prior layers. Both signal processing time in Fog layer and data transmission time are considered as transmission latency. As shown in Table 7, transmission latency is evidently reduced especially in Wi-Fi network with heavy load.

4.4 Local Storage, Notification, and Security at the Fog layer

The majority of the data management Fog services explained in Section 2.1 make use of the local storage function of the gateway in the Fog layer. Incoming data from 6LoWPAN modules through a UDP server in the gateway, on port 5700, and from RTX WiFi modules is aggregated in the storage. The Bluetooth module on the gateway also receives data from Bluetooth sensor modules. The local storage is implemented using MySQL database, to leverage the federation feature provided by the database engine for data synchronization with the cloud storage. The same table schema is used both in the cloud and in the gateways allowing easy migration of the data without mapping to other format.

The data stored locally is stored for about 30 minutes while the synchronization takes place. In the condition that there is a failure in connectivity with the cloud, the

gateway stores the data for as long as there is connectivity or it runs low on memory. In the case that the connectivity is cannot be maintained, the gateway removes the old data and also provides additional user services, such as notification and access to the local data. The notification service can be used as mentioned above or can also be configured to run along with the cloud based notification service.

For protecting the system, an end-to-end secure schema is implemented. The schema provide a high level of authentication and authorization for end-users. In addition, the schema guarantee that the system is secure whilst sensor nodes does not need to be reconfigured in cases of sensor node mobility. The detailed information and an implementation of the schema are presented in [33].

Table 8 XML Status code and description

Code	Description
0	Invalid request or Error
1	Notification - {total in number}
2	No new nofication

Notification service in the gateway can be immediately triggered when receiving an alert signal from other services such ECG feature extraction or an early warning service. In order to reduce the burden of Fog layer and provide a global notification, the notification service is primary built at Cloud. The push notification at Fog layer is merely responsible for notifying the push notification in Cloud by sending signals in XML formats, shown in Table 8. In addition to the push notification service at Fog and Cloud, a mobile application is created for supporting the the push notification service. When it receives the incoming signal from the notification service at Cloud, it immediately pops up a text message on an end-user's phone to inform about an alert case.

4.5 WebSocket Server for interoperability

To enhance the interoperability of the healthcare IoT implementation, we implemented an embedded websocket server at the Fog layer. It is written using a Python framework, known as Tornado, which is an asynchronous web server framework. The server listens to UDP connections and a full duplex communication can be established with a client node. This setup is used to collect ECG signals coming from sensor nodes. Each message contains 800 bytes of data from 400 samples of ECG, at an average speed of 1.1KB/s. Client nodes with an interface can also access an HTML page hosted on the gateway to see the received information rendered into a graph. This can be extended to enhance the service to listen to various transport layer protocol sockets to enable interoperability.

5 Related applications of Fog computing

Three layers of physical separation in IoT architectures is a common approach in recent years. However, prior to that, a client server approach between sensor nodes and the cloud was a usual implementation. In both cases, amalgamation of the cloud computing and IoT technologies provides significant benefits in the healthcare application domain [46]. The IoT allows integrating objects into an information network, so in health scenarios, different medical sensors could collect health data from the patient continuously. The data is transmitted to a remote cloud server for analysis and decision making concerning requirements of the use cases. Using data analytics and machine learning approaches, patient medical condition is estimated, decision making is fulfilled and notifications are provided for the patient and health professionals [47, 48]. In another work, Bimschas *et al.* [49] implement basic learning and intelligence in the gateways of the Fog layer through application code execution to allow them convert between protocols, smart caching, and discovery.

The application of Fog computing to different domains is also an active research area. Related to this chapter of the application of Fog computing to healthcare, Shi YingJuan et al. [50] have used it in a similar context. Their work motivates the need for Fog computing in general and discuss the latency sensitivity of healthcare applications. In addition, it also highlight the resource constraints of sensor and actuators at the end of the IoT network. To meet the latency requirement and support the end devices, their implementation of Fog computing provides services, such as online data analytics and interoperability of various communication protocols. In another work [51] that applies Fog computing for healthcare, they have developed a gateway to be used in the Fog layer to provide services for healthcare domain. Similar to the previous work, the gateway provides interoperability to various network protocols of sensor nodes and communicate with the cloud through WiFi or Ethernet. The gateway is used for a medical application, emergency attendance in a patient care facility. It is also used in monitoring patients to follow up on their medication. Moreover, their work considers semantic interoperability of the data for better quality of the medical information collected. In contrast, this chapter begins with more comprehensive general Fog computing services that enable healthcare applications and show specific use cases that consume these services. Moreover, Stantchev *et al.* in their work [52] present, the advantages of applying the three tiered approach for healthcare scenario. It focuses on servitization and business aspect of healthcare IoT. In contrast to the work presented in this chapter, their work focuses on high level architectural aspects while this chapter presents a real world case study and experimental evaluation of the services.

Leveraging Fog computing in a healthcare applications can enhance latency of the system and energy efficiency to sensor node devices, as discussed in this chapter. These benefits of Fog computing are particularly important to healthcare applications which are sensitive to response time and need distributed analytics. In addition to the case studies of early warning score and ECG feature extraction, some other remote healthcare monitoring applications have been proposed to employ Fog-assisted architecture. Yu C. *et al.* [53] implemented a fall monitor application for

stroke mitigation in an architecture with Fog computing with minimal response time and energy consumption. Moneiro A. *et al.* [54] showed the efficacy of Fog computing in their speech tele-treatment application, where speech data collected from smartwatch is stored and processed in Fog layer and only speech features are sent to secure cloud. Similarly, applications including pervasive brain monitoring with EEG, emergency response system integrating heterogeneous data and ambient assisted living for seniors have considered to utilize system structure with Fog [55, 56, 57]. Moreover, as mentioned above, security plays an important role in all health monitoring IoT-based systems. In [33], authors provides an end-to-end Fog-based schema for enhancing security levels of health monitoring IoT-based system. Their results show that by applying the secure schema in the Fog layer, the entire IoT system can be protected even in case of sensor nodes' mobility. In [25], the authors show that latency and a large amount of transmitted data can be decreased dramatically by applying the services of the Fog layer.

6 Conclusions

This chapter focused on applying Fog computing to healthcare Internet of Things domain. As a case study, to highlight the benefit of Fog computing, a set of services was presented that enable healthcare IoT and utilized in an implementation of a smart gateway for Fog computing. These services are designed to address the key challenges in IoT, with a focus on addressing healthcare functional requirements. A geographically distributed network of these smart gateways, each handling a group of sensor nodes or patients, form the Fog layer in this architecture. This cluster of gateways provide a continuous patient monitoring means without limiting the movement of the patient in the coverage area.

Fog computing provides services for remote patient monitoring by reducing communication latency and improving system consistency. To this end, the patient vital signs were processed locally, and a local notification was provided for the user. Moreover, for further analysis, the sensory data along with the obtained results were transmitted to the cloud server. The details of the Fog services and the obtained benefits were analyzed and performance measures presented. Some services can be duplicated or partitioned to multiple, such as sensor layer with the Fog layer or the Fog layer with the Cloud. This implementation specifics depend on the particular services and the functionality. Incorrect decisions of implementation location can cause inefficiency in terms of energy consumption, latency, and performance. For example, before sending data from sensor nodes to Fog layer, noise should be initially removed at nodes. Fog can be used to implement advanced and complex noise elimination and signal processing methods for enhancing the quality of collected data. More advanced services, such as complex machine learning algorithms, the cloud layer should be used for the implementation. In order to achieve a high level of energy efficiency at sensor nodes, processing and communication must be taken

into consideration. Finally, specific medical use cases can have additional design constraints that demand fine tuning of behaviors of the Fog layer services.

References

1. European Commission Information Society. Internet of Things Strategic Research Roadmap, 2009. *http://www.internet-of-things-research.eu/ [accessed 2015-07-14]*.
2. European Commission Information Society. Internet of Things in 2020: a Roadmap for the Future, 2008. *http://www.iot-visitthefuture.eu [accessed 2015-07-14]*.
3. A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier. The internet of things for ambient assisted living. In *Proceedings of the International Conference on Information Technology: New Generations*, pages 804–809, 2010.
4. D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497 – 1516, 2012.
5. M. Carmen Domingo. An overview of the internet of things for people with disabilities. *Journal of Network and Computer Applications*, 35(2):584 – 596, 2012.
6. Hairong Yan, Li Da Xu, Zhuming Bi, Zhibo Pang, Jie Zhang, and Yong Chen. An emerging technology â“ wearable wireless sensor networks with applications in human health condition monitoring. *Journal of Management Analytics*, 2(2):121–137, 2015.
7. Y. J. Fan, Y. H. Yin, L. D. Xu, Y. Zeng, and F. Wu. Iot-based smart rehabilitation system. *IEEE Transactions on Industrial Informatics*, 10(2):1568–1577, 2014.
8. B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, and K. Mankodiya. Towards Fog-Driven IoT eHealth: Promises and Challenges of IoT in Medicine and Healthcare. *Elsevier Future Generation Computer Systems*, 2017.
9. C.E. Koop, R. Mosher, L. Kun, J. Geiling, E. Grigg, S. Long, C. Macedonia, R. Merrell, R. Satava, and J. Rosen. Future delivery of health care: Cybercare. *IEEE Engineering in Medicine and Biology Magazine*, 27(6):29 – 38, 2008.
10. European Research Cluster on the Internet of Things. IoT Semantic Interoperability: Research Challenges, Best Practices, Solutions and Next Steps. *IERC AC4 Manifesto - "Present and Future"*, 2014.
11. B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu. Ubiquitous data accessing method in iot-based information system for emergency medical services. *IEEE Transactions on Industrial Informatics*, 10(2):1578–1586, 2014.
12. L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu. An iot-oriented data storage framework in cloud computing platform. *IEEE Transactions on Industrial Informatics*, 10(2):1443–1451, 2014.
13. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, pages 13–16, 2012.
14. M. Aazam and E. N. Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 687–694, 2015.
15. M. Aazam and E. N. Huh. Fog computing and smart gateway based communication for cloud of things. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 464–470, 2014.
16. A.-M. Rahmani, N.K. Thanigaivelan, Tuan Nguyen Gia, J. Granados, B. Negash, P. Liljeberg, and H. Tenhunen. Smart e-Health Gateway: Bringing Intelligence to IoT-Based Ubiquitous Healthcare Systems. In *Proceeding of 12th Annual IEEE Consumer Communications and Networking Conference*, pages 826–834, 2015.
17. A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 2017.

18. Health Level Seven Int'l. Introduction to HL7 Standards, 2012. *www.hl7.org/implement/standards [accessed 2015-07-30]*.
19. M.L. Hilton. Wavelet and wavelet packet compression of electrocardiograms. *IEEE Trans. Biomed.*, 44(5):394 – 402, 1997.
20. Z. Lu, D. Youn Kim, and W.A. Pearlman. Wavelet compression of ECG signals by the set partitioning in hierarchical trees algorithm. *IEEE Trans. Biomed.*, 47(7):849 – 856, 2000.
21. R. Benzid, A. Messaoudi, and A. Boussaad. Constrained ECG compression algorithm using the block-based discrete cosine transform. *Digital Signal Processing*, 18(1):56 – 64, 2008.
22. H. F. Durrant-Whyte. Sensor models and multisensor integration. *Int. J. Rob. Res.*, 7(6):97–113, 1988.
23. Tuan Nguyen Gia, Tcarenko Igor, Victor K. Sarker, Amir-Mohammad Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Iot-based fall detection system with energy efficient sensor nodes. In *IEEE Nordic Circuits and Systems Conference (NORCAS'16)*, 2016.
24. Tuan Nguyen Gia, Mingzhe Jiang, Amir-Mohammad Rahmani, Tomi Westerlund, Kunal Mankodiya, Pasi Liljeberg, and Hannu Tenhunen. Fog computing in body sensor networks: An energy efficient approach. In *IEEE International Body Sensor Networks Conference (BSN'15)*, 2015.
25. T. Nguyen Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *Proceeding of International Conference on Computer and Information Technology*, pages 356–363, 2015.
26. B. Negash, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. Lisa: Lightweight internet of things service bus architecture. *Procedia Computer Science*, 52:436 – 443, 2015. The 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015).
27. N. Paul, T. Kohno, and D. C. Klonoff. A review of the security of insulin pump infusion systems. *Journal of Diabetes Science and Technology*, 5(6):1557–1562, 2011.
28. netfilter/iptables - nftables project. *http://netfilter.org/projects/nftables/ [accessed 2015-07-24]*.
29. G. Kambourakiset, E. Klaoudatou, and S. Gritzalis. Securing Medical Sensor Environments: The CodeBlue Framework Case. In *Proceeding of The Second International Conference on Availability, Reliability and Security*, pages 637–643, 2007.
30. R. Chakravorty. A programmable Service Architecture for Mobile Medical Care. In *Proceeding of Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 5 pp. – 536, 2006.
31. J. Ko, J. H. Lim, Y. Chen, R. Musvaloiu-E, A. Terzis, G. M. Masson, T. Gao, W. Destler, L. Selavo, and R. P. Dutton. Medisn: Medical emergency detection in sensor networks. *ACM Trans. Embed. Comput. Syst.*, 10(1):11:1–11:29, 2010.
32. S.R. Moosavi, T.N. Gia, A. Rahmani, E. Nigussie, S. Virtanen, H. Tenhunen, and J. Isoaho. SEA: A Secure and Efficient Authentication and Authorization Architecture for IoT-Based Healthcare Using Smart Gateways. In *Proceeding of 6th International Conference on Ambient Systems, Networks and Technologies*, page 452–459, 2015.
33. S.R. Moosavi, T.N. Gia, E. Nigussie, A. Rahmani, S. Virtanen, H. Tenhunen, , and J. Isoaho. Session Resumption-Based End-to-End Security for Healthcare Internet-of-Things. In *Proceeding of IEEE International Conference on Computer and Information Technology*, pages 581–588, 2015.
34. PyCrypto API Documentation. https://pythonhosted.org/pycrypto/, accessed: 26-05-2016.
35. Texas Instruments. Low-Power, 2-Channel, 24-Bit Analog Front-End for Biopotential Measurements, 2012.
36. Texas Instruments. ECG Implementation on the TMS320C5515 DSP Medical Development Kit (MDK) with the ADS1298 ECG-FE, 2011.
37. RTX Real-Time Operating System. *http://www.keil.com/rl-arm/kernel.asp, 2015-08-04*.
38. R. Barry. *Using The FreeRTOS Real Time Kernel, Microchip PIC32 Edition*. FreeRTOS Tutorial Books, 2010.

39. A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceeding of International Conference on Local Computer Networks*, pages 455–462, 2004.
40. OMAP®4 PandaBoard System Reference Manual, 2010. *http://pandaboard.org [accessed 2015-08-04]*.
41. SmartRF06 Evaluation Board User's Guide, 2013. *http://www.ti.com/lit/ug/swru321a/swru321a.pdf [accessed 2015-08-04]*.
42. Olimex. MOD-ENC28J60 development board, Users Manual, 2008. *https://www.olimex.com/Products/Modules/Ethernet/MOD-ENC28J60/resources/MOD-ENC28J60.pdf [accessed 2015-08-04]*.
43. IEEE standard for medical device communication, overview and framework. In *ISO/IEEE 11073 Committee*, 1996.
44. V. S. Miller et al. Data compression method. *US4814746 A, Filing date Aug 11, 1986, Publication date Mar 21, 1989*.
45. G. B. Moody and R. G. Mark. The impact of the MIT-BIH arrhythmia database. *Engineering in Medicine and Biology Magazine, IEEE*, 20(3):45–50, 2001.
46. S. Luo and B. Ren. The monitoring and managing application of cloud computing based on internet of things. *Computer Methods and Programs in Biomedicine*, 130:154–161, 2016.
47. j. Gomez, B. Oviedo, and E.Zhuma. Patient monitoring system based on internet of things. In *Procedia Computer Science*, volume 83, pages 90–97, 2016.
48. G. Ji, W. Ouyang, K. Yang, and G. Yang. Skin-attached sensor and artifact removal using cloud computing. In *7th International Conference on e-Health (eHealth2015)*, 2015.
49. D. Bimschas, H. Hellbrück, R. Mietz, D. Pfisterer, K. Römer, and T. Teubler. Middleware for Smart Gateways Connecting Sensornets to the Internet. In *Proceedings of the International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks*, pages 8–14, 2010.
50. Y. Shi, G. Ding, H. Wang, H. E. Roman, and S. Lu. The fog computing service for healthcare. In *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech)*, pages 1–5, May 2015.
51. Lisardo Prieto González, Corvin Jaedicke, Johannes Schubert, and Vladimir Stantchev. Fog computing architectures for healthcare: Wireless performance and semantic opportunities. *Journal of Information, Communication and Ethics in Society*, 14(4):334–349, 2016.
52. Vladimir Stantchev, Ahmed Barnawi, Sarfaraz Ghulam, Johannes Schubert, and Gerrit Tamm. Smart items, fog and cloud computing as enablers of servitization in healthcare. *Sensors & Transducers*, 185(2):121, 2015.
53. Yu Cao, Songqing Chen, Peng Hou, and Donald Brown. Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on*, pages 2–11. IEEE, 2015.
54. Admir Monteiro, Harishchandra Dubey, Leslie Mahler, Qing Yang, and Kunal Mankodiya. Fit a fog computing device for speech teletreatments. *arXiv preprint arXiv:1605.06236*, 2016.
55. Octavian Fratu, Catalina Pena, Razvan Craciunescu, and Simona Halunga. Fog computing system for monitoring mild dementia and copd patients-romanian case study. In *Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2015 12th International Conference on*, pages 123–128. IEEE, 2015.
56. John K Zao, Tchin-Tze Gan, Chun-Kai You, Cheng-En Chung, Yu-Te Wang, Sergio José Rodríguez Méndez, Tim Mullen, Chieh Yu, Christian Kothe, Ching-Teng Hsiao, et al. Pervasive brain monitoring and data sharing based on multi-tier distributed computing and linked data technology. *Frontiers in human neuroscience*, 8, 2014.
57. Mervat Abu-Elkheir, Hossam S Hassanein, and Sharief MA Oteafy. Enhancing emergency response systems through leveraging crowdsensing and heterogeneous data. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International*, pages 188–193. IEEE, 2016.