AN ARCHITECTURAL APPROACH FOR THE INTEGRATION OF WIRELESS SENSOR NETWORKS WITH CLOUD COMPUTING FOR A SECURE HEALTHCARE SYSTEM

By

NAVEEN RAJ DHANAPAL

Bachelor of Engineering in Computer Science and

Engineering

Magna College of Engineering, Anna University

Chennai, Tamil Nadu, India

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2012

AN ARCHITECTURAL APPROACH FOR THE

INTEGRATION OF WIRELESS SENSOR NETWORKS

WITH CLOUD COMPUTING FOR A SECURE

HEALTHCARE SYSTEM

Thesis Approved:

Dr. Johnson P Thomas

Thesis Adviser

Dr. Subhask Kak

Dr. Michel Toulouse

Dr. Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Existing processes for gathering patients' health data require a great deal of work that includes collecting the health data inputs, processing and analysis of the information collected [1]. These kinds of processes are usually error prone and considered to be slow. However, today's biomedical sensor solutions are effective for only an individual measurement (for example EEG, ECG, PCG and the like) but are not integrated into a complete body area network, where many sensors work simultaneously collecting information about an individual patient. Patient mobility also brings in the need for sensors in biomedical monitoring to become wireless. This creates the need for the implementation of new biomedical personal wireless networks with a common architecture and the capacity to handle multiple sensors, monitoring different body signals, with different healthcare apps running on smart phones [2]. To overcome this, this work proposes an architecture that integrates wireless sensors that communicate with the patient's smart phone and integrates them with the cloud computing paradigm and the hospital information system. The information is therefore made available in the cloud where it is processed and accessible to doctors and other medical professionals.

Cloud Computing is a commercial extension of computing resources which provides scalable resources and economic benefits to its users over the internet [3]. It acts as software and provides data access and storage services which don’t need the knowledge of the end users physical location and the systems configuration that provides the computing resources. In Cloud Computing, the users use the web browsers as an interface [4], while the software and data are stored on the remote servers and hence it is device independent.

In recent years, many healthcare organizations have started using wireless senor networks to remotely monitor patient health. Many healthcare organizations and insurance companies have also started using the electronic medical record (EMR) system by which the medical records are maintained in a centralized database in the form of an electronic record and the records are stored in the cloud. Applications deployed on the cloud for manipulate electronic medical records [5].

The general scope of our work is to propose an architecture to integrate the healthcare cloud with wireless sensor network (WSN) technology through smart phones. The healthcare apps on smart phones monitor patients’ health wirelessly providing real-time updates of the patients’ health condition to the doctors and other medical professionals via the cloud.

The proposed architecture contains a filter system running on the smart phones, which takes the patient’s health records from the smart phone apps and compares with a lookup table, which contains the normal readings of the different health parameters. If the incoming health readings to the filter are found to be abnormal, then an alert SMS is sent to the doctors with whom the patient is associated and a copy of the record is also sent to the cloud running an EMR system maintained by the hospital. The patient may be in a real emergency situation and he may not be aware that he is in danger. The filter identifies that the patient may be in a serious emergency situation based on the historical and current values of the sensed data. If the filter determines the situation to be critical, a consolidated report is sent to the doctor through SMS, along with the location (address) of the patient, sensed through the GPS sensor running on the smart phone.

Since, the health records have to be confidential and secure, the Health Insurance Portability and Accountability Act HIPAA [6] defines a set of rules on who has access to the patient's health records. In order to comply with the rules and regulations of HIPAA, whenever there is an emergency alert, the alert SMS is sent only to the authorized and appropriate medical professionals who can access that patient's health data. The proposed key search algorithm helps to find the appropriate medical professionals for different health abnormalities and ensures that the patient's abnormal health data is sent only to the appropriate medical professionals. However, this algorithm works only when there is a single abnormal input to the filter system. Our work therefore also proposes a priority ranking algorithm, when multiple historical abnormal data have to be considered. Our system provides a secure framework whilst providing the benefits of the cloud.

1.1 TAXONOMY OF CLOUD COMPUTING

Cloud Computing can be broadly classified into three different layers [5] based on the service models and cloud deployment models. They are

1. Software as a Service

This is the top-most layer in the cloud which allows the users to use the applications running on the cloud using a web browser as an interface rather than installing the software in the local machine or in a local data center. Google Docs is one of the example of the SaaS service model.

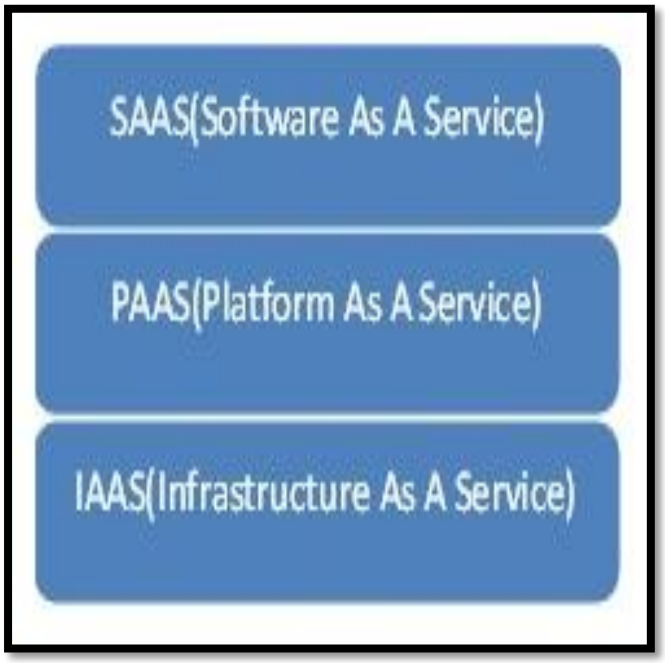

SAAS(Software As A Service)
PAAS(Platform As A Service)
IAAS(Infrastructure As A Service)

Figure 1: Layers in the Cloud

2. Platform as a Service

This middle layer in the cloud offers a development platform for the users to deploy applications in the cloud using different programming languages and the tools supported by the cloud provider. It provides the users with database management, security, workflow management, etc. Microsoft's Azure is an example of PaaS service model.

3. Infrastructure as a Service

This is the bottom-most layer in the cloud and provides components for the basic infrastructure such as CPU, memory, networks and storage. It provides the users to host their own application and store data. Amazon's Elastic Compute Cloud (EC2) is an example for an IaaS service model.

Cloud Computing has four different deployment models [5] as follows.

1. Public Cloud (External Cloud)

The cloud infrastructure is offered over the internet to the general public or an organization and it is owned by the organization providing the cloud services. In this type of deployment model, the application developer and the customers hold the entire responsibility for the security. This type of service is provided by Amazon EC2, Salesforce etc.

2. Private Cloud (Internal Cloud)

The cloud infrastructure is exclusively operated for a single organization which is managed by the organization itself or a third party cloud provider. This is mainly for an organization which has a large number of users and resources. This type of cloud deployment models is more secure.

3. Community Cloud (Semi-private Cloud)

The cloud infrastructure is shared by several organizations that seek to share infrastructure in order to realize some of the benefits of the cloud.

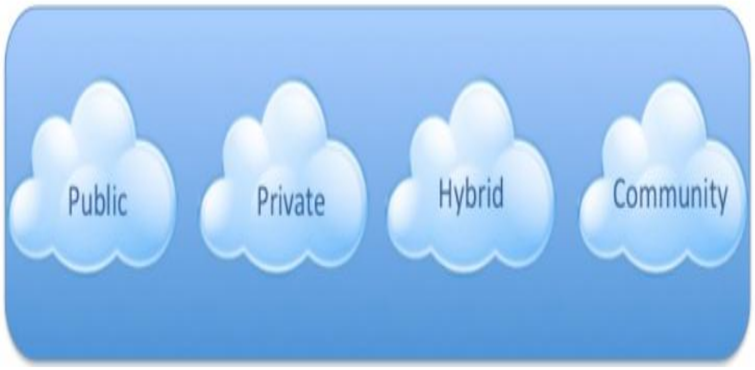


Figure 2: Cloud Deployment Models

This type of cloud is a public cloud which focuses on same domain companies like government or a banking organization.

4. Hybrid Cloud (Integrated Cloud)

The cloud infrastructure is a combination of all or some of the types of cloud. This type of cloud gives the organization more flexibility to share and manage resources between private and public clouds. The organization using this type of cloud keeps all the sensitive data in their private cloud and less sensitive data in their public cloud.

Chapter II overviews the literature. The proposed system architecture is described in Chapter III. Chapter IV contains the simulation and results. The conclusion is in Chapter V.

CHAPTER II

REVIEW OF LITERATURE

2.1 INTEGRATING CLOUD COMPUTING AND WIRELESS SENSOR NETWORKS (WSN)

Cloud computing provides the scalable processing power and variety of connectable services. The cloud computing paradigm has many characteristics that match the typical wireless sensor network, which has a lot of motes responsible for sensing and local preprocessing [9]. Wireless sensor networks are very limited in their processing power, battery life and communication speed whereas cloud computing provides the opposite, and is therefore well suited for long-term observations and analysis. The huge amount of data, which a sensor network delivers, demands a powerful and scalable storage and processing infrastructure.

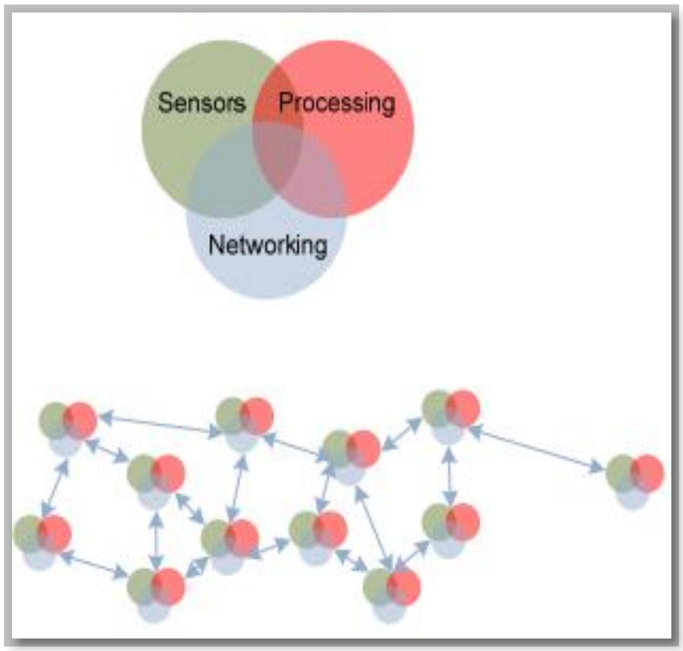


Figure 3: Wireless Sensor Network

The integration of cloud computing with wireless sensor networks [9], provides:

- Integration of sensor network platforms from different vendors.
- Scalability of data storage and processing power for different kinds of analysis.

- Worldwide access to processing and storage infrastructure.
- Optimization of Resources.

One of the services providing data aggregation in low-power and distributed wireless environments is TinyDB [10]. Participating motes within the wireless sensor network run TinyDB on top of TinyOS [11]. TinyDB offers a simple GUI and a Java API for declaring and executing queries in acquisitioned query languages, similar to SQL. The ease of use, efficient use of the available processing and battery power is one of the main advantages of TinyDB.

SPINE [14], a TinyOS-based software framework for body sensor signal processing applications consists of signal processing components for sensor nodes running the TinyOS [11] environment and Java components to manage the sensor nodes from a central coordinator node (star topology). The framework offers a sensor data classification service that is operated in a distributed fashion on individual sensor nodes and a base station computer. The approach doesn't consider aspects like scalability and data security.

2.2 WIRELESS SENSOR NETWORKS IN TELEMEDICINE

The existing system in healthcare involves manual note taking, updating the notes to the computer and maintaining the records under a unique id assigned to every patient. Figure 4, shows the process of data collection in healthcare institutions that use manual note taking [15].

Here,

- The hospital nurse collects the patient’s data at the bedside, writing it on a paper spreadsheet.

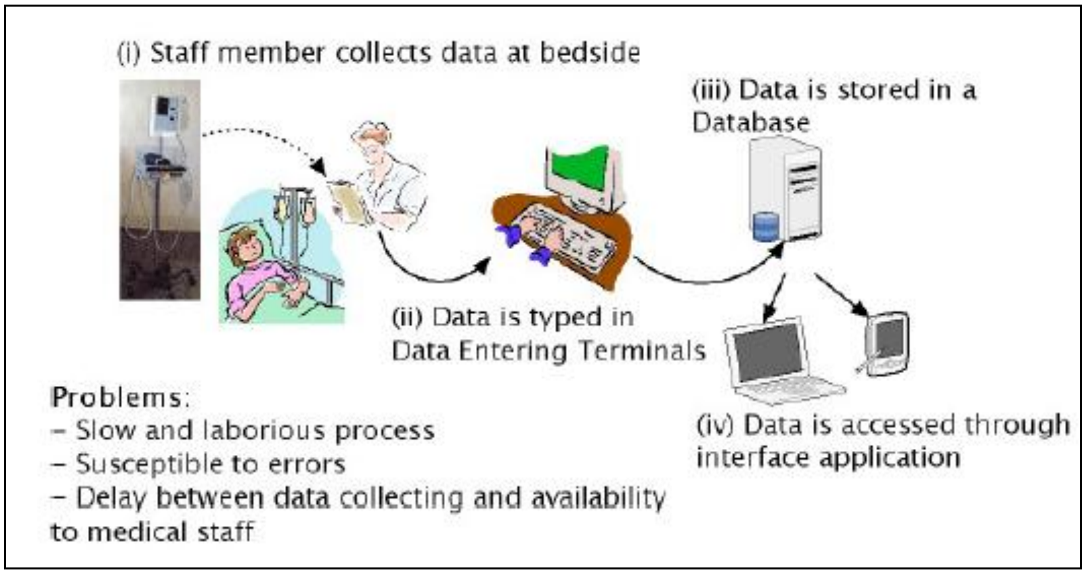

(i) Staff member collects data at bedside
(iii) Data is stored in a Database
(ii) Data is typed in Data Entering Terminals
Problems:
– Slow and laborious process
– Susceptible to errors
– Delay between data collecting and availability to medical staff
(iv) Data is accessed through interface application

Figure 4: Current Scenario for Patient’s Data Collection in Hospitals [15]

- These notes are then typed in data entering terminals.
- The data is transmitted to a database server that organizes indexes and makes it accessible through a database interface.
- Now the medical staff and the doctors can access this information through an interface application.

Usually this process is very slow and error prone and there is latency between data gathering and information accessibility.

Implantable and wear-able monitoring devices such as UbiMon [13] are used to generate vital patients' data required for clinical decision making. With the UbiMon architecture [13], the sensors are placed on the subject and it delivers real-time data to the local processing unit (LPU) via a wireless radio frequency (RF) link. The LPU then processes the incoming data streams and sends it over the wireless networks. The central server (smart phone) receives the real-time sensor data and stores it to the database, with which long-term trend analysis on historical data can be conducted. This allows the prediction and identification of potential life-threatening conditions.

2.3 OVERVIEW OF HEALTHCARE CLOUD

In order to improve the overall productivity and to lower overall IT costs, many healthcare organization are moving into cloud computing. The core idea for the transition to cloud computing is to eradicate the traditional healthcare IT where each application running on different server and management of data were more complex.

We define the concepts [6, 7] of Electronic Health Record (EHR), Electronic Medical Record (EMR) and Personal Health Record (PHR). EHR and EMR are often interchangeably in the health science literature [6]. Strictly speaking, both terms have completely different meanings as given by HIMSS (Health Information and Management System Society) [8].

Electronic Medical Record (EMR):

EMR is the electronic health information of an individual that is created, gathered and managed by an individual healthcare center. It is the legal record of what happened to the patient during the encounter at a healthcare center.

Electronic Health Record (EHR):

EHR is the aggregate of electronic health information of an individual that is created, gathered and managed cumulatively across more than one healthcare organization. This is a composition of some subsets of EMRs which is created and owned by the individual.

Personal Health Record (PHR):

PHR is the cumulative electronic health information of an individual that is initiated and maintained by the individual himself. An effective PHR includes a complete medical history of an individual ranging from the past till the present from different sources including EHRs and EMRs, and control of access to those medical records is the responsibility of the individual. Microsoft's HealthVault and Google Health are two examples of PHRs.

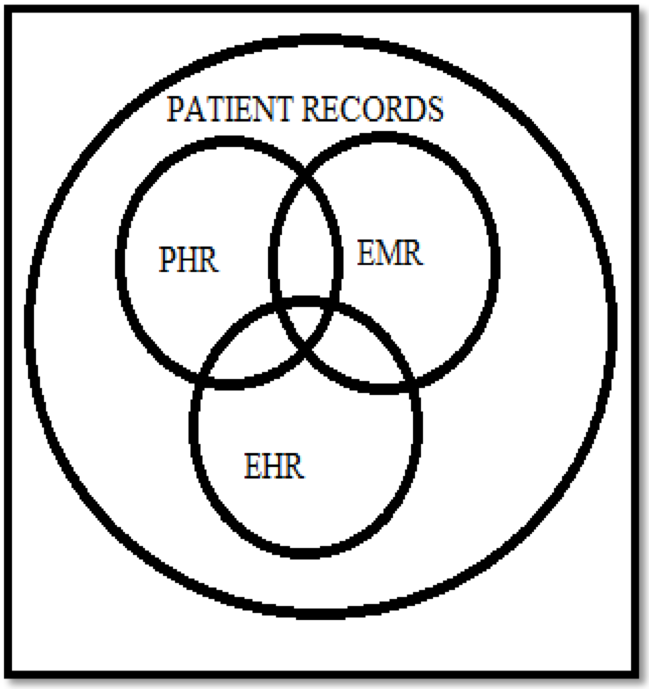


Figure 5: Relationship of EHR, EMR & PHR

The healthcare cloud is a "patient-centric" view, which offers an open platform for the patient to collect, store, use and share health information in a controlled manner. The cloud also offers secure storage and management of the health records for multiple applications such as disease treatment, lab research, insurance etc.

The common security issues shared by the healthcare cloud applications are ownership of information, authenticity, non-repudiation, patient consent and authorization, integrity and confidentiality of data.

2.4 EHR SECURITY REFERENCE MODEL

[5] gives a secure EHR security reference model for both the patient's and healthcare practitioners. This security reference model consists of three core components as follows:

1. EHR secure collection and integration

The first core component of this model is the security reference model which is the collection and integration of EHR data from different sources, which is created and maintained by the hospital. This step is considered to be the first and most important in order to securely share its EHRs with other hospitals. The unique key functional requirement of this component is the EHR integrator. It is responsible for two important tasks:

1. It verifies various EHRs provided by different healthcare organizations in terms of authenticity, confidentiality, integrity, and ensuring non-repudiation as well as HIPAA compliance.
2. It combines and integrates the successfully verified EHR data into a new composite EHR with a security certificate signed by the integrator.

2. EHR secure storage and access management

The EHR storage and management component is composed of two main entities: the secure storage server and the access control engine, where the server stores the encrypted composite EHR data and allows only authorized access and the access control engine manages a collection of role-based or attribute-based access control policies and HIPPA compliance policies, and enforces the access control policies to prevent the data from unauthorized access. Only authorized medical personnel can obtain the access to the authorized portions of the encrypted EHR data through identity and authorization based decryption mechanisms.

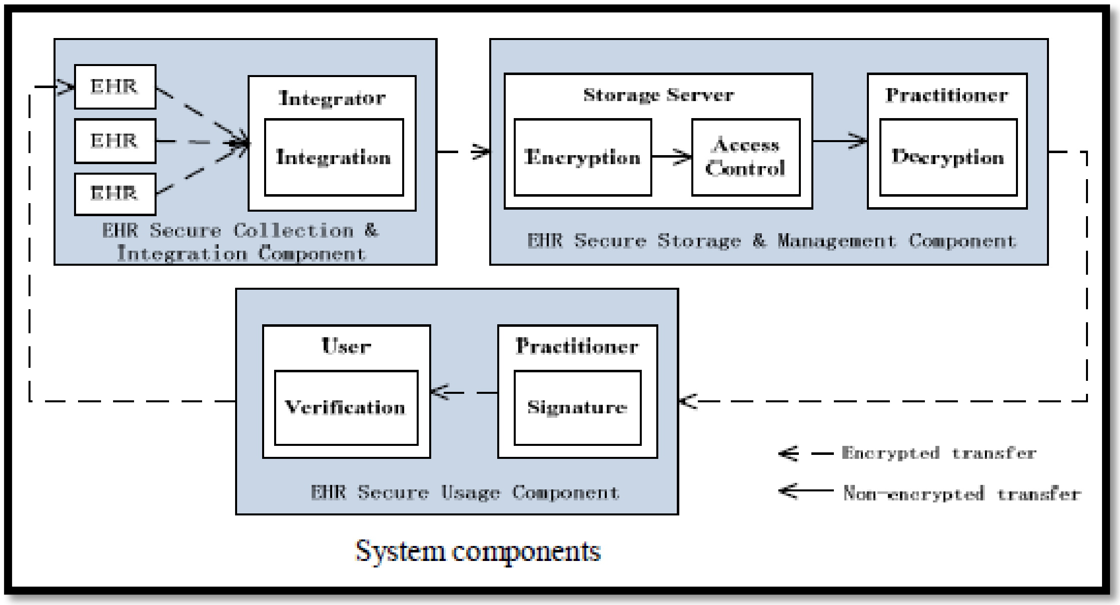


Figure 6: EHR System Integration Model [5]

3. EHR secure usage model

The third component of the EHR security reference model is the secure usage model that provides source verifiable content access for consumers of EHR data, including both patients and healthcare practitioners. Thus the two basic functional building blocks in this component are signature and verification.

Signature:

When the practitioners participating in the consultation of a patient, reach a medical conclusion regarding the next step in treatment, they sign the medical certificate of the corresponding EMR with the appropriate signature algorithm. When the portion of the EMR is used to create an EHR record for that patient regarding this consultation, the certificate is sent to him alone with the corresponding EHR.

Verification:

Consequently, to verify the authenticity of the consultation result made available in the form of an EHR, the patient can use this medical certificate and practitioners' digital signature. Note that the privacy of the practitioners can be respected and, the patient does not need to know the group of practitioners who signed the medical certification of the consultation results.

2.5 WSN IN SMARTPHONES MONITORING HUMANS HEALTH

In the last two decades, there has been a steady decline in the number of patients getting admitted and treated in hospitals due to the influence of mobile healthcare. Long-term monitoring of patients' physical, physiological, psychological, cognitive, behavioral process is crucially important for those with chronic diseases [16, 17].

Figure 7, shows an overview of health monitoring architecture with a smart phone [17] which has links to external wireless sensor devices, such as a blood pressure monitor, weight scale, etc. to collect periodically health data. These external devices have sensors, which are Bluetooth enabled. The smart phones running healthcare apps monitor the wellbeing of the patient and transmit the data to the healthcare Data Server maintained by the hospital via the internet. The medical personnel access the Data server via a secure internet connection to monitor the patient's health remotely.

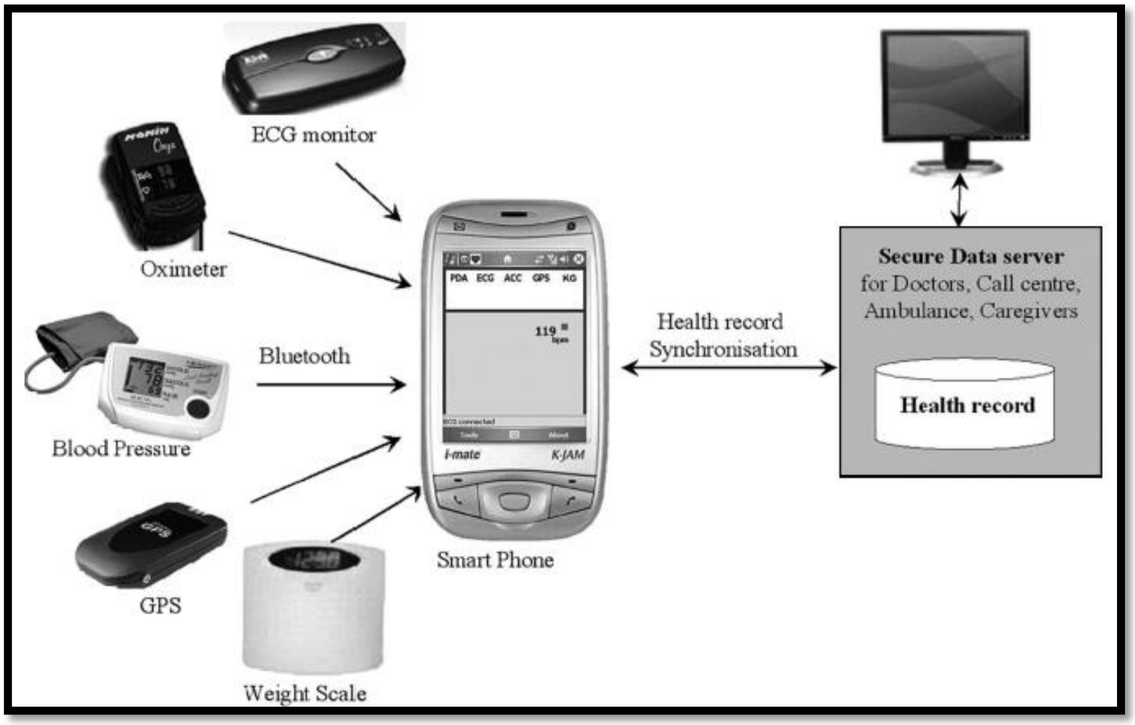

ECG monitor
Oximeter
Bluetooth
Blood Pressure
GPS
Weight Scale
Smart Phone
Health record Synchronisation
Secure Data server for Doctors, Call centre, Ambulance, Caregivers
Health record

Figure 7: WSN in Smart Phones Monitoring Patient's Health [17]

AMON [12] is a wrist-worn WSN medical monitoring system specially designed for patients with cardiac disease and respiratory problems. This system includes continuous collection and evaluation of several vital signs and it is connected to the medical center.

BeTelGeuse ([18], [19]) is a platform for gathering and processing situational data. BeTelGeuse runs on smart phones and gathers health data from a body area network over Bluetooth. It transmits the gathered health data to a server for further processing. The main advantage of BeTelGeuse system is that it can be extended to use new sensors and it runs on several platforms. BeTelGeuse's major disadvantage is that the server does not offer a general architecture for data processing and analysis and also the approach doesn`t consider major aspects like scalability and data security.

2.6 ELLIPTICAL CURVE CRYPTOGRAPHY

In the proposed architecture, we use Elliptical Curve cryptography (ECC), a public key encryption technique based on elliptic curve theory. This is very fast and more efficient [20]. ECC generates the keys by the properties of Elliptical Curve (EC) rather than traditional method of generation as the product of very large prime numbers. Since, ECC establishes more security with less processing power and battery resource usage; it is widely used in smart phones for security.

Neal Koblitz and Victor Miller independently proposed Elliptic curve encryption scheme in 1985. Elliptic curve uses discrete logarithm over finite field [21]. Prime curve over finite field Zp uses a cubic equation of the form of (1), with a, b satisfying (2).

$$y2 \bmod p = (x3 + ax + b) \bmod p \text{………………..…}(1)$$

Where a, b, x and *y* takes integer values between 0 to *p –1*.

$$4a3 + 27b2 \ (\bmod\ p) \ != 0 \text{………………………....}(2)$$

The mode of operation of Prime curve over Zp, can be described as follows. Choose a prime p, curve coefficient a, b, an integer pr and a generator G which is a point on the curve with x and y coordinate. G is multiplied by pr as indicated in (3) to produce *pu*.

$$pu(x1, y1) = pr \times G(x, y) \text{…………………………..}(3)$$

The private key is *pr* and the public key can be denoted as (4).

$$pu(x1, y1), G(x, y), Ep(a,b) \text{…………..……………}(4)$$

In order to encrypt a message *M*, the message is mask as a point with *x* and *y* coordinate, an integer *k* is chosen and the cipher text can be computed using formula in (5).

$$C = k \times G(x, y), M(xm, ym) + k \times pu(x1, y1) \ldots\ldots\ldots (5)$$

To decrypt, *M* is computed from (6) and the mask is reversed to produce the message.

$$\begin{aligned} M &= M(xm, ym) + k \times pu(x1, y1) - pr \times G(x, y) \\ &= M(xm, ym) + k \times pr \times G(x, y) - pr \times k \times G(x, y) \\ &= M(xm, ym) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6) \end{aligned}$$

According to the National Institute of Standards and Technology (NIST) guideline on security strength [22], Elliptic curve offers keys of smaller size and it more secure. ECC is light weight and compatible with lower power systems such as smart phones and PDA’s.

2.7 TESLA PROTOCOL

TESLA (Timed Efficient Stream Loss-tolerant Authentication) is an efficient protocol with low communication and computation overhead for the generation and verification of authentication information, which scales to large numbers of receivers and tolerates packet loss [26]. Tesla provides asymmetric properties in spite of using symmetric cryptographic functions. It uses the MAC (Message Authentication Code) function as a security engine which is very suitable for smart phones that have low-power, processing speed, and storage capabilities.

SENDER:

In this model, the sender composes a stream of data as a message {Mi}. The sender splits the time into equal intervals. The duration of each time interval is denoted with T_{int}, and the initial time for the interval i is T_i. Trivially, we have $T_i=T_0+i*T_{int}$. In each interval, the sender may send zero or multiple message. Before sending the first message, the sender determines the interval duration, and the length N of the key chain, and it picks a random value for K_N. The sender constructs a one-way function F: F(k) = $f_k(0)$ using a pseudo-random function F. The key chain's element is defined as K_i = $F(K_{i+1})$. Now each key can be derived from K_N as $K_i = F^{N-i}(K_N)$, where $F^j(k) = F^{j-1}(F(k))$ and $F^0(k) = k$. Here each key k in the key chain corresponds to one time interval I_j. We use a second pseudo-random function F' to derive the key $K^`_i = F^`(Ki)$ used to compute the MAC, which needs to be appended to every message sent.

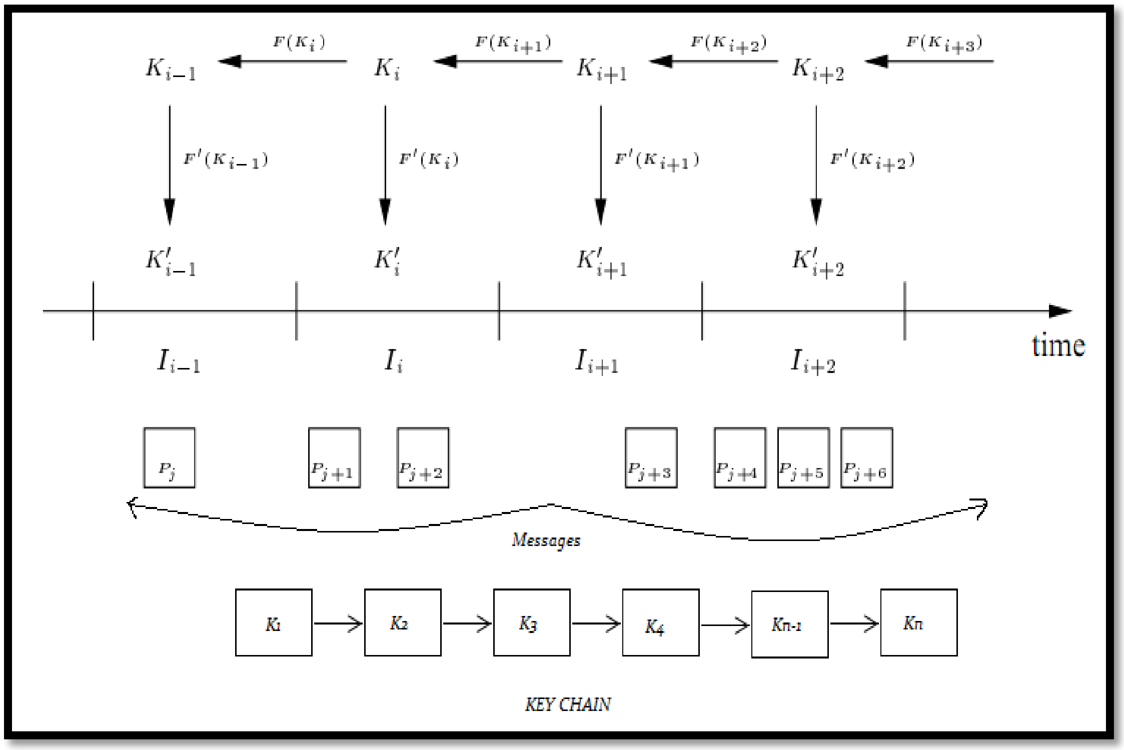


Figure 8: Tesla protocol for securing the alert SMS sent to the hospital.

The initial authentication packet is sent containing time interval duration T_{int}, start time T_i, the length of the key chain N, key disclosure delay d which is the number of intervals, and key commitment to the key chain K_i ($i < j-d$ where j is the current interval index). Each key of the key chain is used in a onetime interval, and the number of messages sent per time interval may vary from 0 to many. Each constructed packet (message) in the time interval I_i will look as follows: $\{M_j \mid MAC(K^{`}_{I}, M_j) \mid K_{i-d}\}$.

RECEIVER:

When the receiver receives the message M_j, it uses the self-authenticating key K_{i-d} disclosed in M_j to determine i. On receiving the disclosed key K_i first, it checks whether it already knows K_i or a later key K_j (j>i). If K_i is the latest key received by the receiver, the receiver checks the legitimacy of K_i by verifying for some earlier key K_v ($v<i$) that $K_v = F^{i-v}(K_i)$. The receiver then computes $K'_i = F'(K_i)$ and verifies the authenticity of packets of interval i and of previous intervals if the receiver did not yet receive the keys for these intervals.

CHAPTER III

PROPOSED APPROACH

3.1 PROBLEM STATEMENT

Existing healthcare systems are paper based which involves manual note taking, updating the notes to the computer and maintaining the records. These processes are error prone and it takes time before the data and notes can be accessed by healthcare professionals. Our goal therefore is to propose a system that will allow a patient to not come to the hospital, provide error free collection of patient data, and enable fast secure access to patient as soon as it becomes available. We use sensor networks and the cloud. The patient therefore does not have to be at the hospital, doctors can read the patient's health data as soon as it is available and the information is subject to be less error prone as the human is not in the chain. Furthermore as the patient is not at the hospital, our goal is to quickly identify emergency situations so that the necessary help can be sent to the patient.

3.2 PROPOSED SOLUTION

We have proposed an architecture which integrates wireless sensor networks running on smart phones, with cloud computing. The health data obtained from the healthcare apps running on the smart phones needs to processed and based on abnormality of the data, necessary actions needs to be taken. A copy of the sensor data is sent to the cloud from where the hospital accesses it. The proposed architecture integrating the smart phones and cloud solves the problem using the proposed filter system. If there is any abnormality in the health parameter, the smart phone sends an alert SMS to the healthcare organization. HIPAA [6] defines a set of rules on who has access to the patient's health records. Therefore, only authorized medical personnel should receive the alert SMS sent by the smart phones. The proposed key search and priority ranking algorithm solves this problem by finding the highest weighted list of medical personnel to whom the alert SMS needs to be send.

3.3 PROPOSED SYSTEM ARCHITECTURE

For the integration of wireless sensor network with the healthcare cloud, we have proposed a three-tiered cloud based system. Each smart phone has a filter app and one or more healthcare apps running on them. All the healthcare apps are tied with the proposed innovative filter app which receives the healthcare data from different healthcare apps running on the smart phone. The filter provides patients and healthcare provider's insight into physiological and physical health states that are critical to the detection, diagnosis, treatment and management of ailments.

In particular the proposed architecture has the following:

- A smartphone layer senses the user's heart rate, blood pressure rate, glucose monitor, body temperature etc. using the healthcare apps. This layer establishes communication with the filter layer by sending the sensor data to the filter.
- The filer layer is designed to meet real-time requirements of the applications running on the smartphones.

    - The filter layer is based on existing smart phone equipment and provides resources for limited data storage, caching, and processing power.
    - The filter running on the smartphone checks the received data from healthcare apps with the thresholds of the respective health record using a lookup table containing different health record and their levels. If there is any abnormality in the health condition, the smart phone sends the health record to hospital. It sends the health record through an emergency alert message (SMS) containing the health parameter and its abnormal value to the hospital.
    - The filter uses elliptical curve, an asymmetric cryptography technique and sends a copy to the EMR System running on the cloud for the hospitals reference. Hence the data is encrypted before sending it to the cloud for security reasons. We used an asymmetric algorithm as they are ideally suited for real-world use, as the secret key does not have to be shared. Every user only needs to keep one secret key in secrecy and a collection of public keys that only need to be protected against being changed. We used Elliptical Curve because it is light-weight and compatible with lower power systems such as smart phones and PDA's.
- The cloud layer can be either a Platform as a Service (PaaS) or an Infrastructure as a Service (IaaS), where the Electronic Medical Record (EMR) System has been deployed. This layer has the feature of large storage, extensive computation, and is used for large-scale computations without real-time constraints.

- The data from the cloud layer is accessed by the insurance companies and hospitals to whom the rights to access the data as per HIPAA [6] compliance policies is granted. Also the hospital keeps monitoring the health condition of the patients.

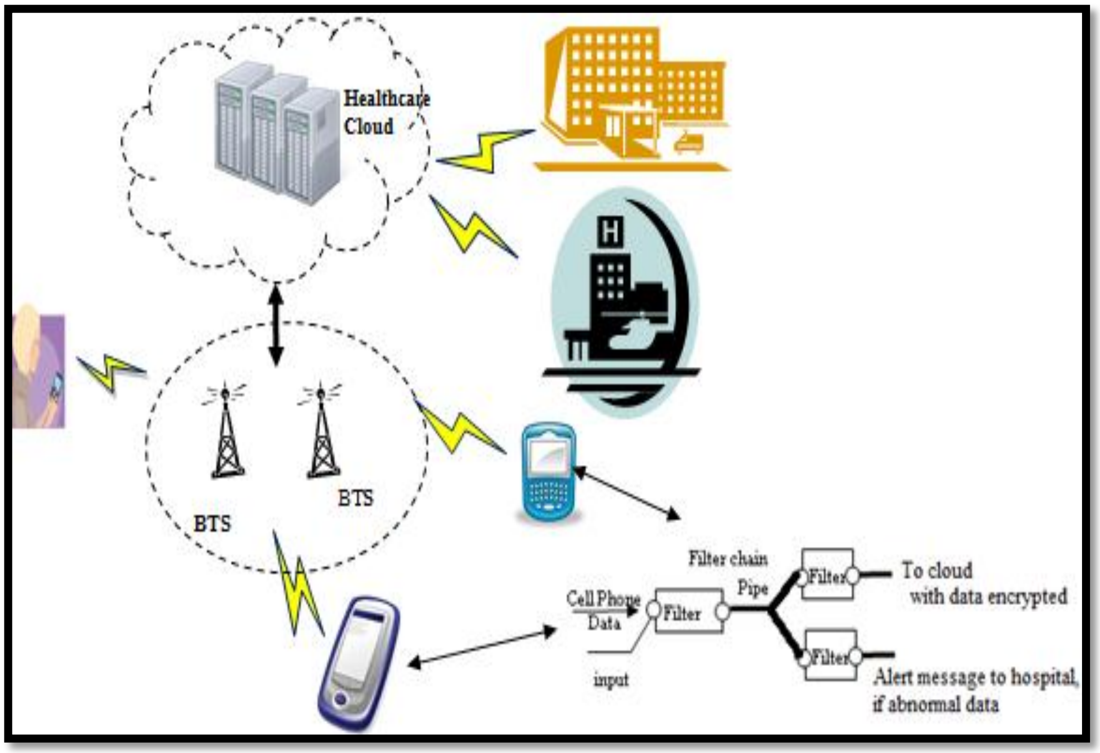


Figure 9: Proposed System Model

3.4 FILTER-BASED HEALTHCARE SYSTEM

- The filter is an application running on the smart phones whose main functionality is to receive the sensed data (usually blood rate, heart rate, glucose etc.) from the healthcare apps and stores them in a temporary buffer.
- The filter analyses the received healthcare data with the help of a look-up table, containing the different health parameter levels and find the severity of the

received data. Table 1, gives the lookup table for Heart rate, Blood pressure and Blood sugar values.

- The smart phone healthcare apps create the xml from the sensor data containing the patient ID, sex, age and health readings. The filter receives this xml data from the healthcare apps.

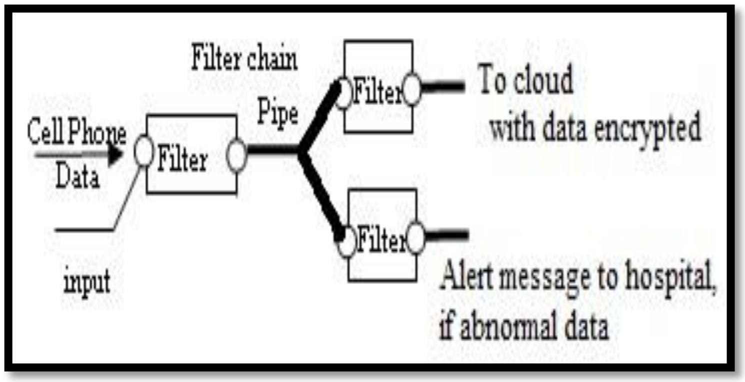

Filter chain
Pipe
Cell Phone Data
input
Filter
Filter
Filter
To cloud with data encrypted
Alert message to hospital, if abnormal data

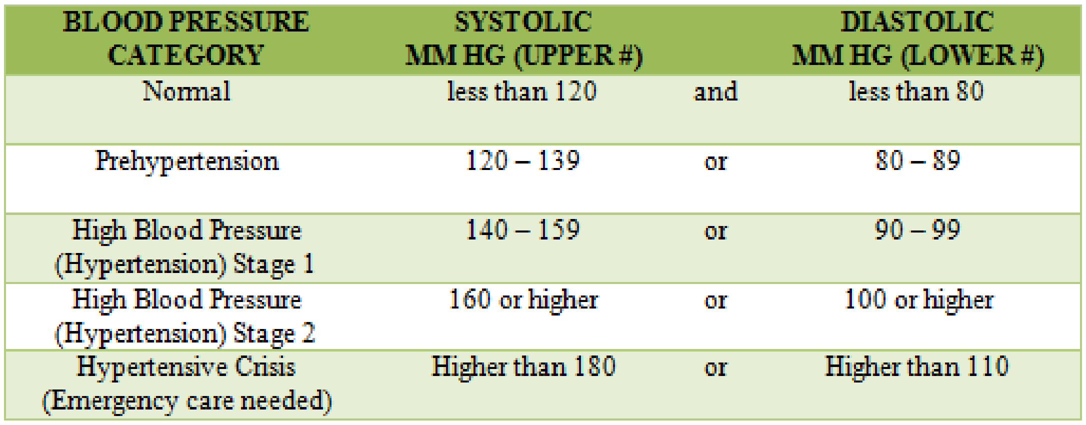
Figure 10: Filter Architecture

XML DATA:

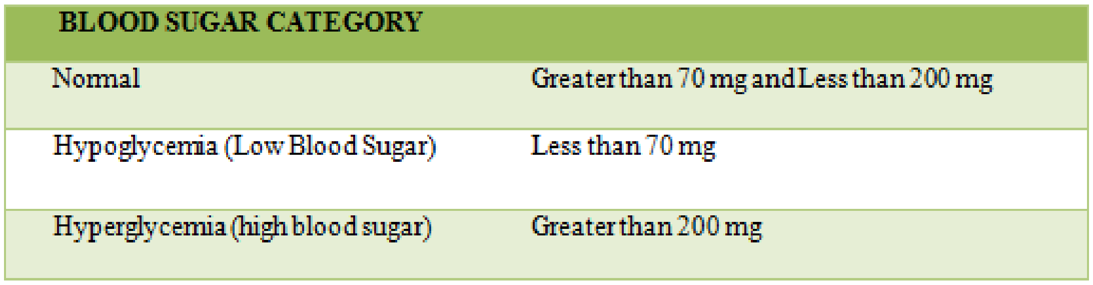
```
<HealthRecord>
        <Pid> P001 </Pid>
        <Age> 45 </Age>
        <Sex> Male </Sex>
        <HealthProblem> HeartRate </HealthProb>
        <Value> 70 bpm </Value>
</HealthRecord>
```

- If the resulting value is found to be severe, an emergency alert SMS is sent to the hospital and a copy of the record is encrypted using ECC and sent to the EMR system running on the cloud. HIPAA [6] defines who should have access to the patients' data. The proposed priority ranking algorithm (section 3.7), decides whom to send the emergency alert message as SMS.
- If the resulting value is found to be normal, encrypt the data using ECC and send it to the EMR system running on the cloud to get stored.

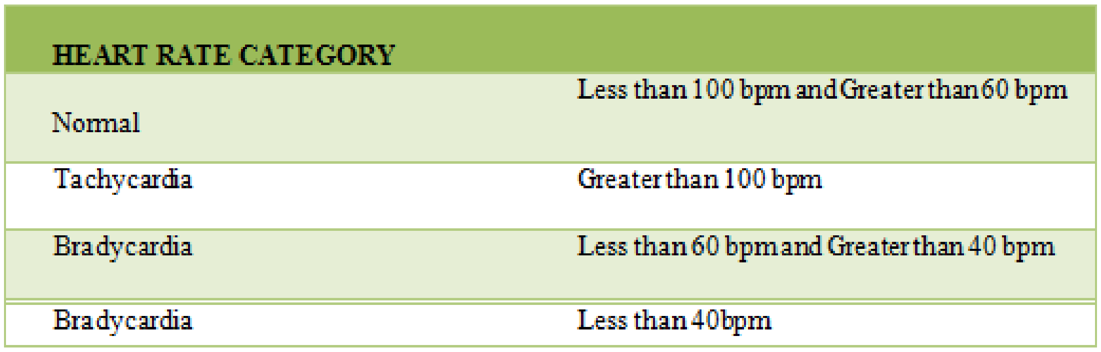
3.5 LOOKUP TABLE FOR FILTER SYSTEM:



BLOOD PRESSURE CATEGORY	SYSTOLIC MM HG (UPPER #)		DIASTOLIC MM HG (LOWER #)
Normal	less than 120	and	less than 80
Prehypertension	120 – 139	or	80 – 89
High Blood Pressure (Hypertension) Stage 1	140 – 159	or	90 – 99
High Blood Pressure (Hypertension) Stage 2	160 or higher	or	100 or higher
Hypertensive Crisis (Emergency care needed)	Higher than 180	or	Higher than 110



BLOOD SUGAR CATEGORY	
Normal	Greater than 70 mg and Less than 200 mg
Hypoglycemia (Low Blood Sugar)	Less than 70 mg
Hyperglycemia (high blood sugar)	Greater than 200 mg



HEART RATE CATEGORY	
Normal	Less than 100 bpm and Greater than 60 bpm
Tachycardia	Greater than 100 bpm
Bradycardia	Less than 60 bpm and Greater than 40 bpm
Bradycardia	Less than 40bpm

Table 1: Lookup Table for Filter System [23, 24]

The lookup table is a part of the filter app running on the smart phones. The table contains the actual values of different levels of blood pressure, blood sugar and heart rate. The main functionality of the lookup table in the filter application is to find whether the incoming data is normal or abnormal.

3.6 KEY SEARCH ALGORITHM:

HIPAA [6] defines that only the healthcare providers, to whom the patient has given permission to access his health data and health plans**,** including health insurance companies, HMOs, company health plans, and certain government programs that pay for health care, such as Medicare and Medicaid, have rights to access healthcare data. This ensures that only the relevant healthcare personnel will receive the patient's health data. For example, if the sensor data indicate that the heart is about to fail, the data should be sent to the primary care physician and the cardiologist.

Our proposed approach maintains privacy and security, in that only the appropriate medical professionals who can help the patient are identified to receive the data. The proposed key search algorithm is implemented as a part of the filter system which on receiving the patient's abnormal data decides to whom to send the data through an alert SMS.

The filter receives the xml data as shown in (section 3.4) from the healthcare app, and it extracts the health data that includes the desired health parameter (Blood Pressure, Blood Sugar, Heart Rate etc.) and its corresponding value from the xml. This health parameter serves as a keyword, and the keyword with its value is sent to the lookup table. The lookup table containing all the health parameters with actual and abnormal values finds the severity of the received value. If the health parameter value is found to be normal, the data is ignored by the algorithm. If the health parameter value is found to be abnormal, its corresponding parameter is used as a keyword for the proposed key search algorithm. Based on the keyword, the algorithm finds the list of doctors to whom the abnormal data is to be sent.

The key search algorithm follows the idea of Google's page ranking algorithm [25], which works on webpages as follows,

- Find all pages matching the keywords of the search.
- Rank accordingly using "on the page factors" as keywords.
- Calculate the inbound anchor text.
- Adjust the results by PageRank scores.

As per Google's page ranking algorithm [25], a graph G = (n, e) is constructed for every search made by the user on the internet, where n is the set of nodes representing webpages and e is the edge between two nodes. Every node (webpage) is given an initial weight. The weights of the nodes change according to the page ranking formula given by,

$PR(A) = (1-d) + d(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$ ……………………….(1) [25]

where,

- PR(A) is the PageRank of A,
- PR(Ti) is the PageRank of pages Ti which link to page A,
- C(Ti) is the number of outbound links on page Ti and
- d is the damping factor that is set between 0 and 1 (normally 0.85).

In our proposed system architecture, every filter system receives healthcare information from the hospital database. Although this information may be sent in XML format, or some other semi-structured format, we represent it as a graph for our purposes. The graph $G_O = (n_O, e_O)$ where n_O is the set of nodes representing doctors and an edge e_O between two nodes represents a relationship between two doctors. Every node n_i is given an initial weight w_{ni} by the hospital's database server. The hospital's database server also provides the availability of doctors or healthcare professionals, and specialization of all the nodes (doctors) in the graph.

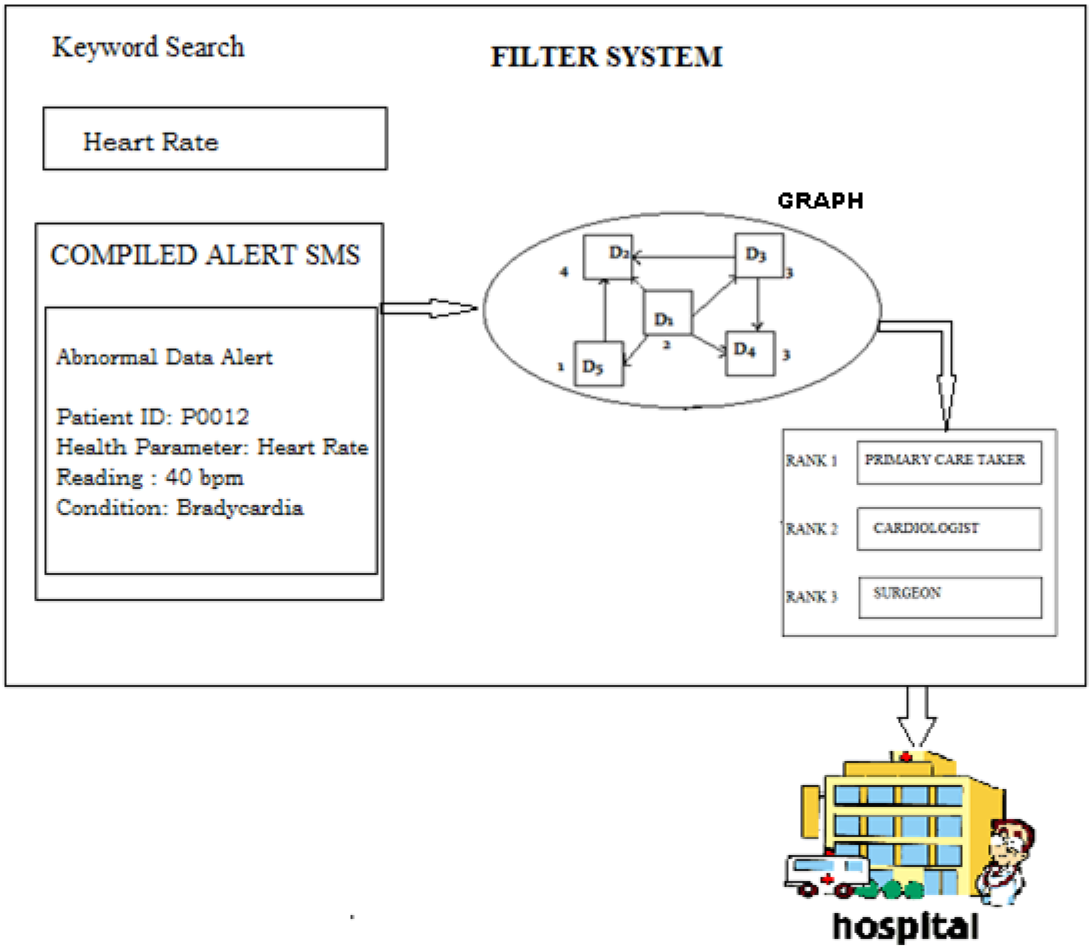


Figure 11: Key Search System Model for Filter System

Our proposed key search algorithm works as follows,

Inputs: Abnormal data keywords A_m and its value x_m. A graph $G_O = (n_O, e_O)$ with n nodes, a start node n_s and other nodes $n_1, \ldots n_m$ and edges $e_1, \ldots e_m$ that connect the nodes. A node contains the following attributes: an unique ID $DocID_i$, the keyword A_i and its value x_i. The start node n_S, could represent the primary caretaker.

Outputs: A ranking of the top three nodes from the generated connected sub-graph.

STEP 1: Find all nodes (doctors) in the graph matching the keyword A_m (represents abnormal data such as Blood Pressure, Heart Rate, etc.).

STEP 2: Construction of the ranking graph,

STEP 2.1: Construct a new graph $G_N = (n_N, e_N)$ where n_N is the set of nodes, and e_N is the set of edges connecting the nodes. Edges are added in step 3. Each node represents a doctor. The new graph G_N must include all the nodes that has the keyword A, in the graph G_O.

STEP 2.2: A weight (w_{nA}) is assigned to the node n_i on the graph G_N based on the doctors availability. A high number means the doctor is very busy. This is assigned to every node.

STEP 2.3: An additional weight (w_{nS}) is added to the node n_i, if the node (doctor) is specialized in the field on which the keyword parameter is. Hence the total weight of a node n_i is $w_{Tni} = w_{ni} + w_{nA} + w_{nS}$ where, w_{ni} is the initial weight of the node n_i.

STEP 3: For all the nodes n_i in $n_1, \ldots n_m$, draw an edge e_{si}, between the start node n_s and node n_i where both n_s and n_i have the same A keyword. At the end of this step, there will be edges between the start node with keyword A and all the nodes that have the same keyword.

STEP 4: In the connected graph, rank the nodes with the highest weight first.

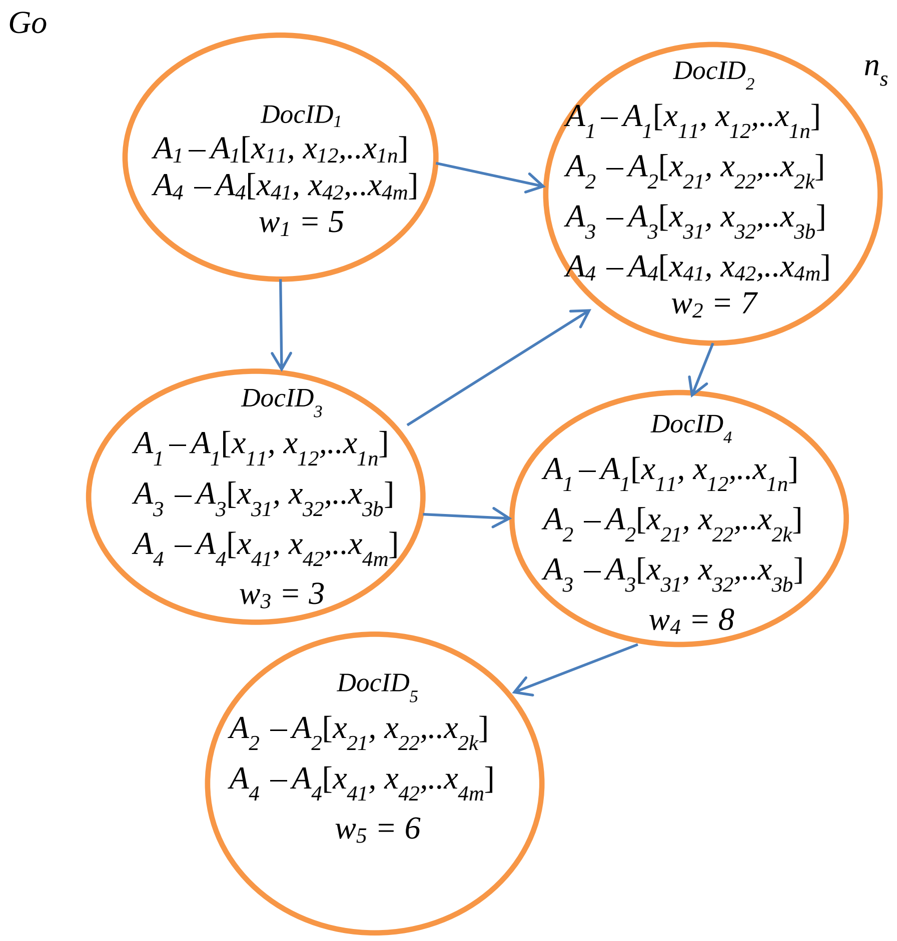
STEP 5: The start node in the graph is given the rank 1, and the next two ranks are given to the next two highest weighted nodes in the connected graph.

The overall complexity can be written as $O(M)$, where M is the total number of nodes.

3.6.1 KEY SEARCH ALGORITHM EXAMPLE:

Inputs

- A directed graph $G_O = (n_o, e_o)$
- A start node n_s
- A node contains the following attributes:
    - an unique ID $DocID_i$
    - Keyword A_i and its value x_i.



Go
n_s
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_1 = 5$
$DocID_2$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_2 = 7$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_3 = 3$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w_4 = 8$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_5 = 6$

Figure 12: Input Graph G_O

- Input keyword A_1:

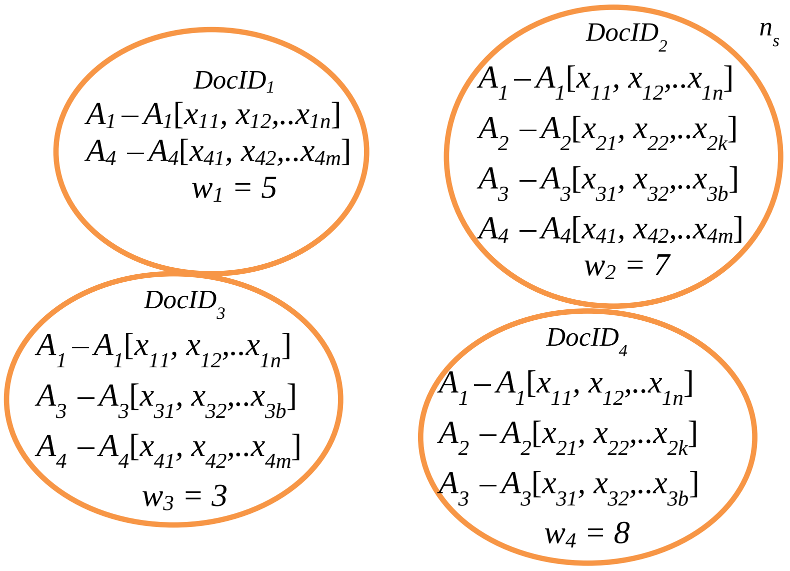

$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_1 = 5$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_2 = 7$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_3 = 3$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w_4 = 8$

Figure 13: Step 1 - Nodes matching Input Keyword A_1

Construction of the new graph G_N ,

- A weight (w_{An}) is assigned to the node n_n on the graph G_N based on doctors availability.

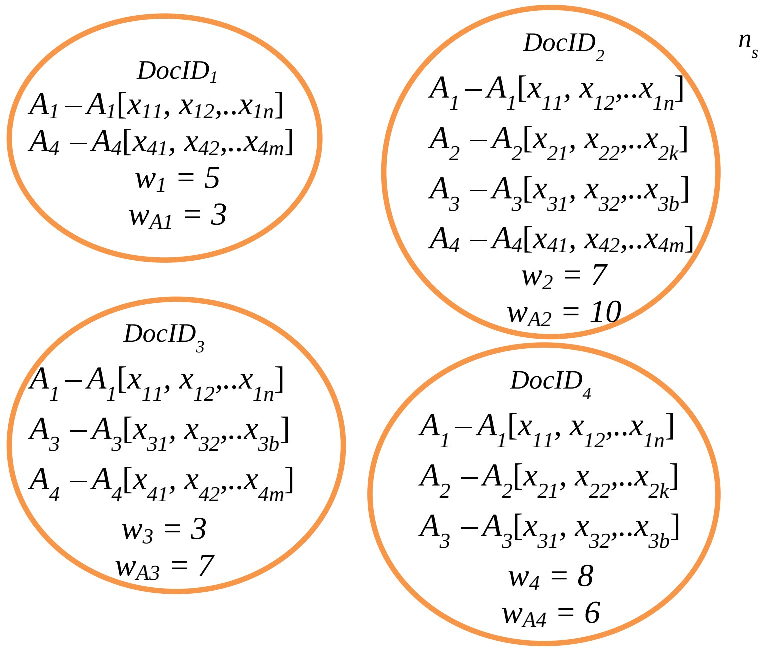
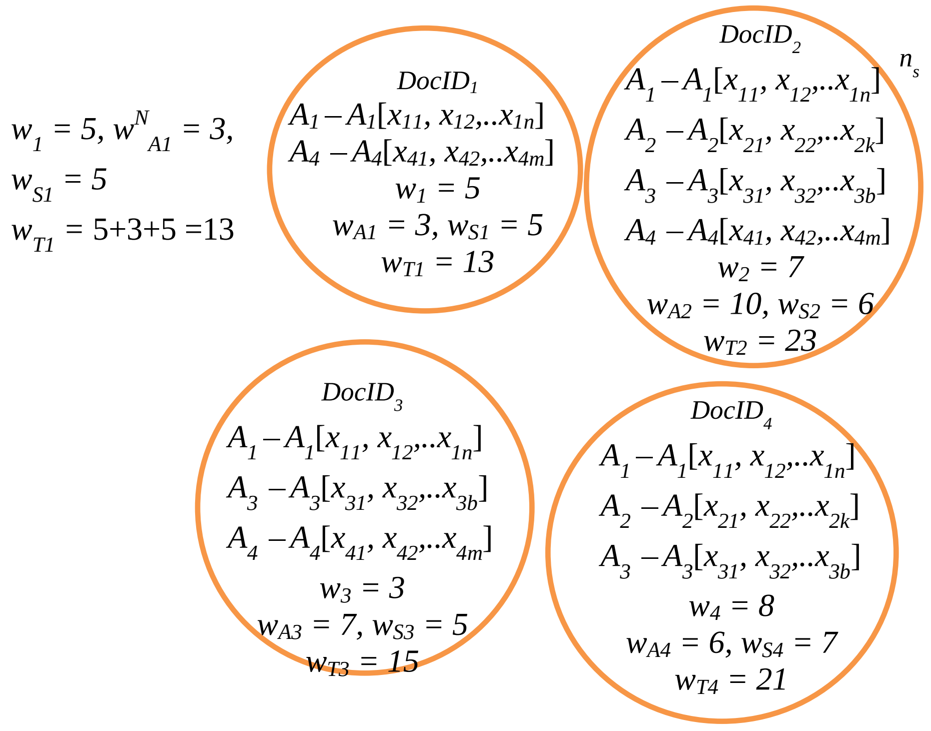

$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_1 = 5$
$w_{A1} = 3$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_2 = 7$
$w_{A2} = 10$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_3 = 3$
$w_{A3} = 7$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w_4 = 8$
$w_{A4} = 6$

Figure 14: Step 2.1 - Construction of New Graph G_N

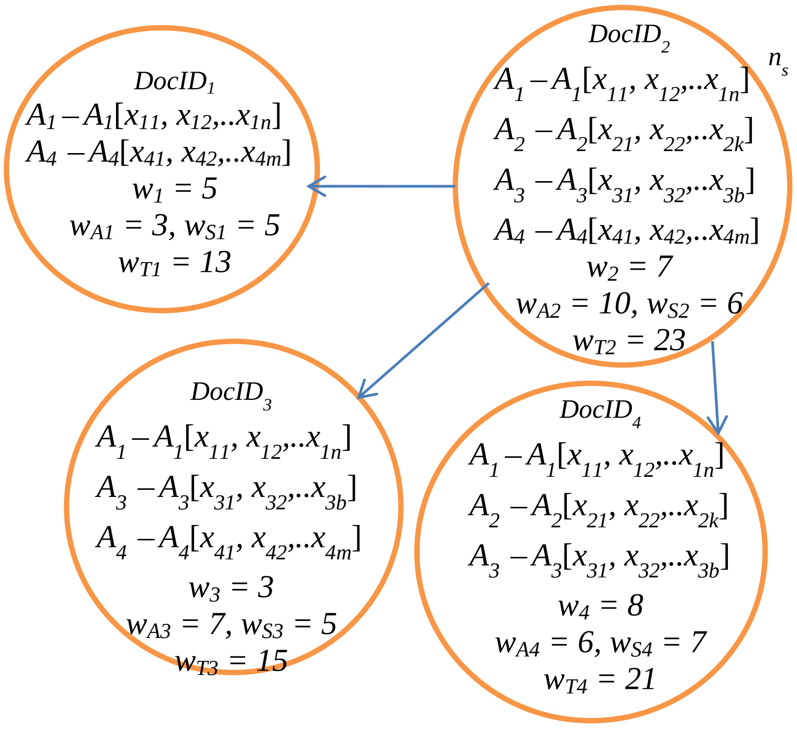
- An additional weight (w_{Sn}) is added to the node n_n, Hence the total weight of a node n_n is $w_{Tn} = w_n + w_{An} + w_{Sn}$



$w_1 = 5$, $w^N_{A1} = 3$,
$w_{S1} = 5$
$w_{T1} = 5+3+5 = 13$
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_1 = 5$
$w_{A1} = 3$, $w_{S1} = 5$
$w_{T1} = 13$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_2 = 7$
$w_{A2} = 10$, $w_{S2} = 6$
$w_{T2} = 23$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_3 = 3$
$w_{A3} = 7$, $w_{S3} = 5$
$w_{T3} = 15$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w_4 = 8$
$w_{A4} = 6$, $w_{S4} = 7$
$w_{T4} = 21$

Figure 15: Step 2.3 - Construction of New Graph G_N with final weights

- Draw an edge e_{si}, between the start node n_s and node n_i where both n_s and n_i have the same input keyword A_1.



$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_1 = 5$
$w_{A1} = 3, w_{S1} = 5$
$w_{T1} = 13$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_2 = 7$
$w_{A2} = 10, w_{S2} = 6$
$w_{T2} = 23$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_3 = 3$
$w_{A3} = 7, w_{S3} = 5$
$w_{T3} = 15$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w_4 = 8$
$w_{A4} = 6, w_{S4} = 7$
$w_{T4} = 21$

Figure 16: Step 3 - Final constructed Graph G_N with final weights and edges

Ranking: Rank1: Node2 (*DocID*$_2$), Rank 2: Node4 (*DocID*$_4$), and Rank3: Node3(*DocID*$_3$)

This approach works only for a single health parameter with a single value, and when there are multiple health parameters with multiple input values, the proposed priority algorithm serves as a solution.

3.7 PRIORITY RANKING ALGORITHM

The patient may be in an emergency situation, and he may not be aware that he is in danger. The filter identifies that the patient is in a serious emergency situation based on the historical and current values of the sensed data. If the filter determines the situation as critical, a consolidated report is sent to the hospital, along with the location (address) of the patient, sensed through the GPS sensor running on the smart phone.

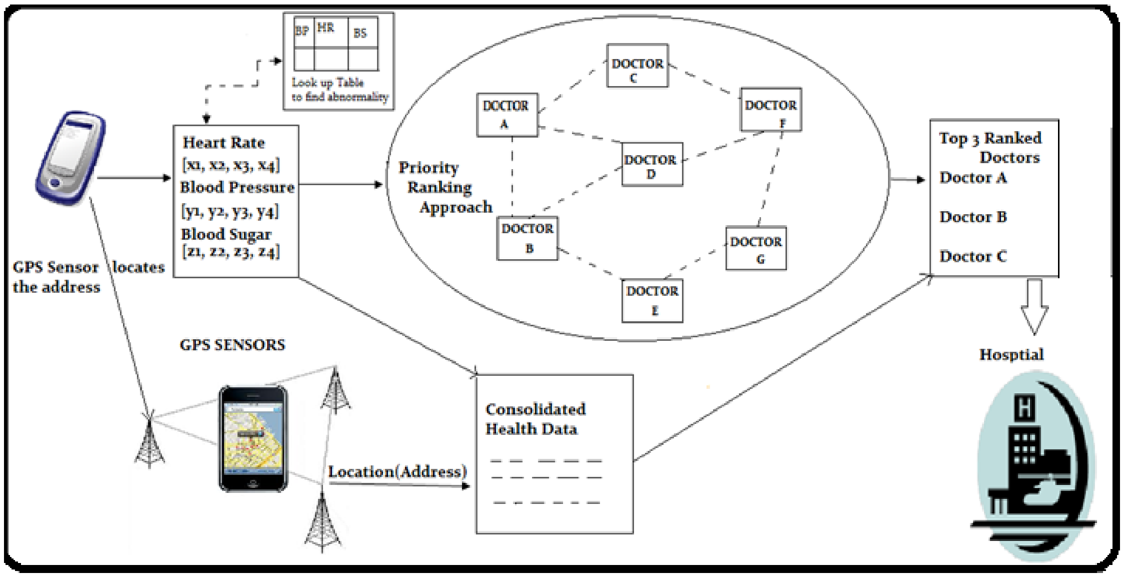

BP
HR
BS
Look up Table to find abnormality
Heart Rate [x1, x2, x3, x4]
Blood Pressure [y1, y2, y3, y4]
Blood Sugar [z1, z2, z3, z4]
GPS Sensor locates the address
Priority Ranking Approach
DOCTOR A
DOCTOR B
DOCTOR C
DOCTOR D
DOCTOR E
DOCTOR F
DOCTOR G
Top 3 Ranked Doctors
Doctor A
Doctor B
Doctor C
Hosptial
GPS SENSORS
Location(Address)
Consolidated Health Data

Figure 17: Priority Ranking System Model for Filter System

The main idea of this proposed algorithm is that the healthcare data needs to be confidential, and only the authorized and appropriate healthcare professionals, who are connected to the patient, can have access to the patient's health data.

Figure 17, explains how the priority ranking algorithm works in an emergency situation. The filter stores the health values of the patient for the last *n* hours (n is determined by the medical profession), in vector form ([x_1, x_2,..x_n]) where x_i is values of different health parameters. The vector is sent to the lookup table, which eliminates the normal values and retains the abnormal values in the vector. The GPS sensor gets activated, and it finds the patient's location and generates the address. The system then produces the report containing the consolidated data which is simply a collection of the data sent by the sensor over the last n hours and the address.

In the Priority Ranking Algorithm, the algorithm constructs a new graph with the input graph and input keywords with its vectors. A new graph is constructed by taking all the nodes matching the input keyword and weights on each node are calculated. This process is repeated for all input keywords. If there are n input keywords, then n number of graphs are constructed and all n graphs are superimposed to get a final connected graph. Then weights on each node of the final superimposed graph are calculated. The nodes are then ranked based on the weights calculated on each node with the highest weight first.

The proposed priority ranking algorithm works as follows,

Inputs: A directed graph $G_O = (n^o, e^o)$. Each node has an initial weight $w^o i$ where i is the node which has a id $DocID_i$. Each edge has an initial weight $w^o_{k,m}$ where the edge is a directed edge between nodes k and m. A node contains the following attributes: an unique ID $DocID_i$, multiple keywords $A_i \ldots A_m$ and its associated vectors $A_i[x_1, x_2, .. x_n] \ldots A_p[y_1, y_2, \ldots y_n]$ respectively.

Outputs: The top 3 nodes are output.

STEPP 1: Find all nodes (doctors) in the graph matching the keyword A_i (represents abnormal data such as Blood Pressure, Heart Rate, etc.).

STEP 2: Construction of the graph,

STEP 2.1: Construct a new graph $G_i = (n^i, e^i)$.

The graph G_i is constructed as follows:

STEP 2.2: A weight (w^i_{An}) is assigned to the node n_n on the graph G_i based on doctors availability. A high number means the doctor is very busy. This is assigned to every node.

STEP 2.3: An additional weight (w^i_{Sn}) is added to the node n_n, if the node (doctor) is specialized in the field on which the keyword parameter is. Hence the total weight of a node n_n is $w^i_{Tn} = w^i_n + w^i_{An} + w^i_{Sn}$

STEP 2.4: For all the nodes n_n in $n_i, \ldots n_m$, draw a directed edge e^i_{sk}, between the start node n_s and node n_m where both n_s and n_m have the same A_i keyword. At the end of this step, there will be edges between the start node with keyword A_i and all the nodes that have the same keyword.

STEP 2.5: An initial edge weight $w^i_{k.m}$ is associated with each edge (k,m). This weight is a function of the input vector $A_i[x_1, x_2,.. x_n]$ associated with the keyword A_i.

STEP 2.6: Add all nodes and edges in G_O to G_i such that if there is an edge e_{on} in G_O that links a node n_o in G_O that does not exists in G_i, to a node n_n in G_O that also exists in G_i, then the directed edge e_{on} and node n_o are added to G_i such that $e^i_{on} = (n_o,n_n)$.

Similarly, if there is an edge e_{no} in G_O that links a node n_n in G_O that exists in G_i also, to a node n_o in G_O that does not exist in G_i, then the directed edge e^i_{no} and node n_o are added to G_i such that $e^i_{no} = (n_n,n_o)$.

If there is an edge e_{pq} in G_O that links a node n_p in G_O that exists in G_i, to a node n_q in G_O that also exists in G_i, then the directed edge e_{pq} is added to G_i such that $e^i_{pq} = (n_p,n_q)$, only if there is no edge connecting n_p and n_q already in G_i.

STEP 2.7: For all initial incoming edges e^i_{kj} to a node n^i_j, the final weight of the node is,

$$w^i_{TF_j} = w^i_{T_j} + \sum_{e^i_{1j}}^{e^i_{kj}} w^i_{kj}$$

that is, the weight of the node added to the weight of all the initial incoming edges.

STEP 2.8 : Assign to each outgoing edge e^i_{jk} of a node n^i_j a new final weight $w^i_{j,k} = (w^i_{TFj} * m) / k$, where there are k outgoing edges from the node n_j, w_{TFj} is the final weight of node n_j that we calculated in the above step 2.7 and m is the multiplication factor which can be between 0 and 1. It is a probability that a doctor is selected by the patient.

STEP 3: Calculate the total weights of each node n_i in the graph G_i :,

$$w^{i}_{Totalj} = m * w^{i}_{TFj} + \sum_{e^{i}_{1j}}^{e^{i}_{kj}} w^{i}_{kj}$$

where,

- w^{i}_{Totalj} is the total weight of the node n_j.
- w^{i}_{TFj} is the final weight of the node n_j in the constructed graph G_i.
- w^{i}_{1j}, w^{i}_{2j}, …, w^{i}_{kj} are the weights of the incoming edges to a node n_i in the constructed graph G_i.
- m is the multiplication factor which can be between 0 and 1. It is a probability that a doctor is selected by the patient.

STEP 4: In the connected graph G_i, rank the nodes with the highest total weight w^{i}_{Totaln} first.

STEP 5: The start node in the graph is given the rank 1, and the next two ranks are given to the next two highest weighted nodes in the connected graph G_i.

STEP 5.1: If there is only one keyword given with its input vector, stop at step 5. However, if multiple keywords are given as inputs, proceed to step 6.

STEP 6: Repeat steps 1 to 5 for all keywords $A_i \ldots A_m$ with input vectors $A_i[x_1, x_2, \ldots x_n] \ldots A_p[y_1, y_2, \ldots y_n]$ respectively. The constructed connected graphs for these keywords will be G_i, G_j..., G_p taking the same graph G_O as input.

STEP 7: Now we have multiple graphs {G_i, G_j..., G_p} constructed using different health parameters of the same patient. Our goal is to find the final list of doctors to whom the abnormal data needs to be sent.

STEP 8: Construct the final graph $G_F = (n, e)$ with the final weights by superimposing the constructed sub-graphs.

STEP 9: Superimpose the graphs {G_i, G_j,...,G_p}, and construct the final graph G_F. The graphs are superimposed as follows:

STEP 9.1: For all nodes n^i, n^j,...,n^p of graphs G_i, G_j, ..., G_p where $DocID_i$ of node n^i = $DocID_i$ of node n^j, = $DocID_i$ of node n^k …,$n^p{}_i$, create a single node $n^f{}_n$ with id $DocID_i$. In other words, all the nodes with the same id from graphs G_i, G_j, ..., G_p, are merged into a node in graph G_f. The merge includes any nodes which have not been merged with any other nodes in the final graph G_f.

STEP 9.2: After superimposing the nodes $n^i{}_i$, $n^j{}_i$,...,$n^p{}_i$ to $n^f{}_n$ which has the identical IDs, calculate the weight of $n^f{}_n$. $w^f{}_{Tn} = w^i{}_i + w^j{}_i \ldots w^p{}_i$, where $w^i{}_i$, $w^j{}_i$ …$w^p{}_i$ are the total weights $w^i{}_{Totali}$, $w^j{}_{Totali}$,…, $w^p{}_{Totali}$, of nodes $n^i{}_i$, $n^j{}_i$,...,$n^p{}_i$ calculated from their respective graphs.

STEP 9.3: Merge the edges $e^i{}_{pq}$, $e^j{}_{pq}$ …. $e^p{}_{pq}$ to a single edge $e^f{}_{pq}$, if nodes p and q exists in all or any of the graphs G_i, G_j, ..., G_p. The weight of edge $e^f{}_{pq}$, $w^f{}_{pq} = w^i{}_{pq} + w^j{}_{pq} + \ldots + w^p{}_{pq}$, where $w^i{}_{pq}$, $w^j{}_{pq}$, …, $w^p{}_{pq}$ are the weights of edges $e^i{}_{pq}$, $e^j{}_{pq}$ …. $e^p{}_{pq}$ calculated from their respective graphs.

STEP 9.4: In the fully constructed final superimposed graph G_f, for all initial incoming edges $e^f{}_{ij}$ to a node $n^f{}_j$, the final total weight of the node $n^f{}_j$ is,

$$w^f_{Total_j} = w^f_{T_j} + \sum_{e^f_{1j}}^{e^f_{kj}} w^f_{kj}$$

that is, the weight of the node added to the weight of all incoming edges. $w^{f}_{T_j}$ is obtained from step 9.2

STEP 9.5: G_f is the final graph as a result of superimposing all the graphs $\{G_i, G_j,...,G_p\}$ as outlined above.

STEP 10: In the connected graph G_f, rank the nodes with the highest total weight w^{f}_{Totaln} first.

- The start node in the graph is given the rank 1, and the next two ranks are given to the next two highest weighted nodes in the connected final graph G_f.

The overall complexity can be written as $O(Ml)$, where M is the total number of nodes, and l is the total number of values in the input vector.

3.7.1 PRIORITY RANKING ALGORITHM EXAMPLE

Inputs:

- A directed graph $G_O = (n^o,e^o)$
- A start node n_s
- A node contains the following attributes:
    - an unique ID $DocID_i$
    - multiple keywords $A_i \ldots A_m$ and its associated vectors $A_i[x_1, x_2,..x_n] \ldots A_p[y_1, y_2,...y_n]$ respectively.
    - Each node i has an initial weight w^o_i
    - Each edge $e_{k,m}$ has an initial weight $w^o_{k,m}$

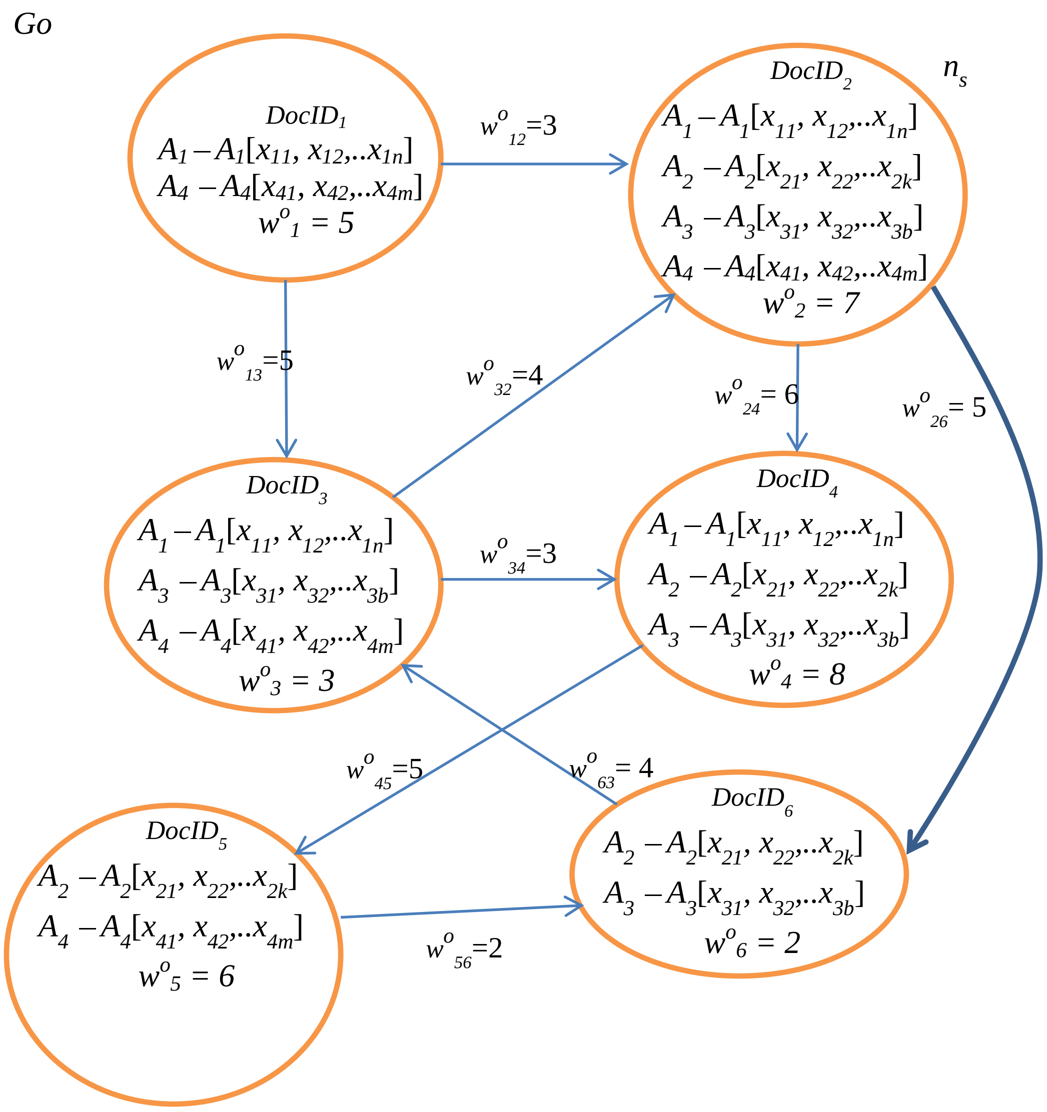
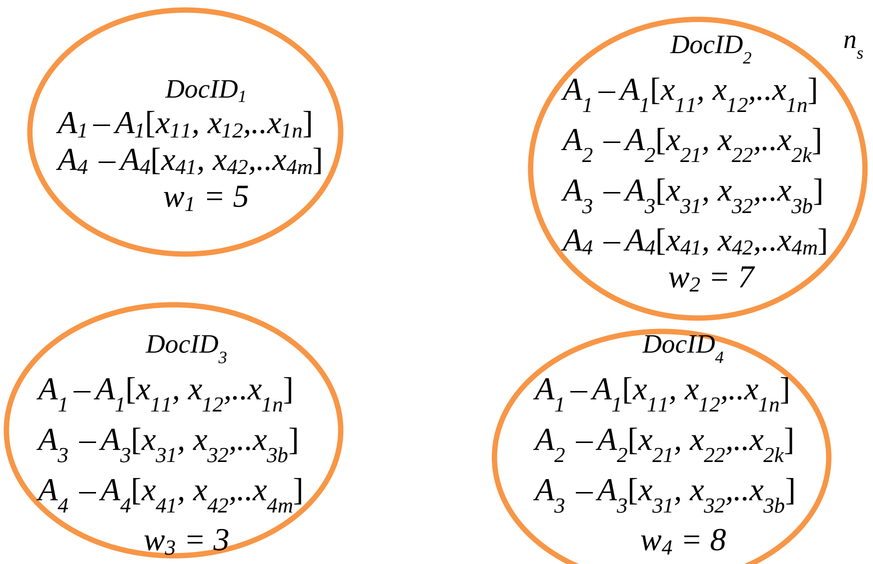

Go
n_s
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^o{}_1 = 5$
$DocID_2$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^o{}_2 = 7$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^o{}_3 = 3$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^o{}_4 = 8$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^o{}_5 = 6$
$DocID_6$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^o{}_6 = 2$
$w^o{}_{12}=3$
$w^o{}_{13}=5$
$w^o{}_{32}=4$
$w^o{}_{24}= 6$
$w^o{}_{26}= 5$
$w^o{}_{34}=3$
$w^o{}_{45}=5$
$w^o{}_{63}= 4$
$w^o{}_{56}=2$

Figure 18: Input Graph G_O

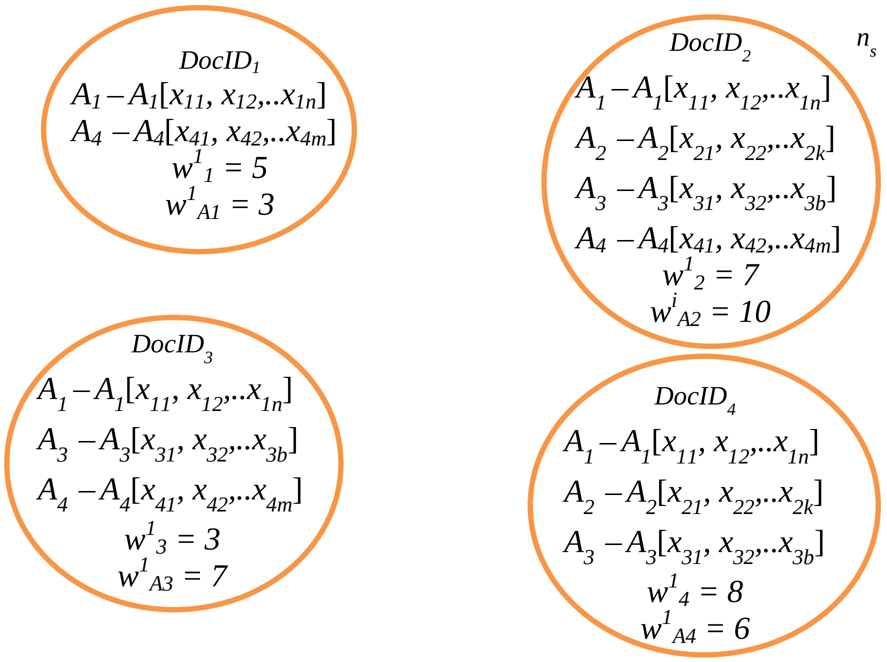
- Given input Keyword is A_1



$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_1 = 5$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_2 = 7$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w_3 = 3$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w_4 = 8$

Figure 19: Step 1 - Nodes matching the input keyword A_1

- Construction of the new graph $G_1 = (n^1, e^1)$ for keyword A_1



$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1{}_1 = 5$
$w^1{}_{A1} = 3$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1{}_2 = 7$
$w^i{}_{A2} = 10$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1{}_3 = 3$
$w^1{}_{A3} = 7$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1{}_4 = 8$
$w^1{}_{A4} = 6$

Figure 20: Step 2.2 - Nodes with initial weight and availability weight

- Total weight of a node n_n is $w^1{}_{Tn} = w^1{}_n + w^1{}_{An} + w^1{}_{Sn}$

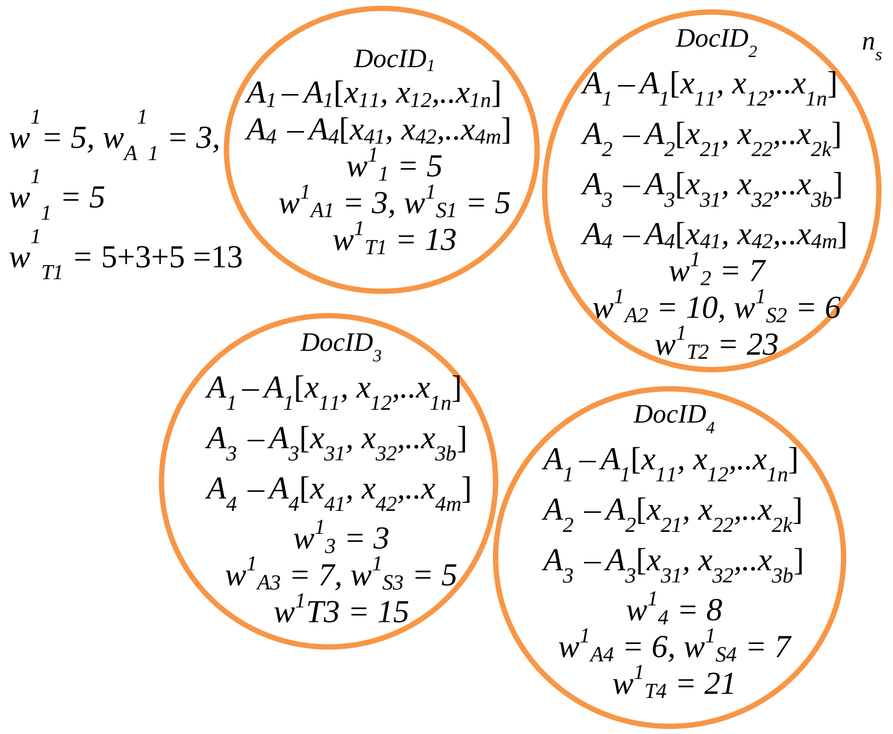


Figure 21: Step 2.3- Nodes with the total weights of availability and specialization

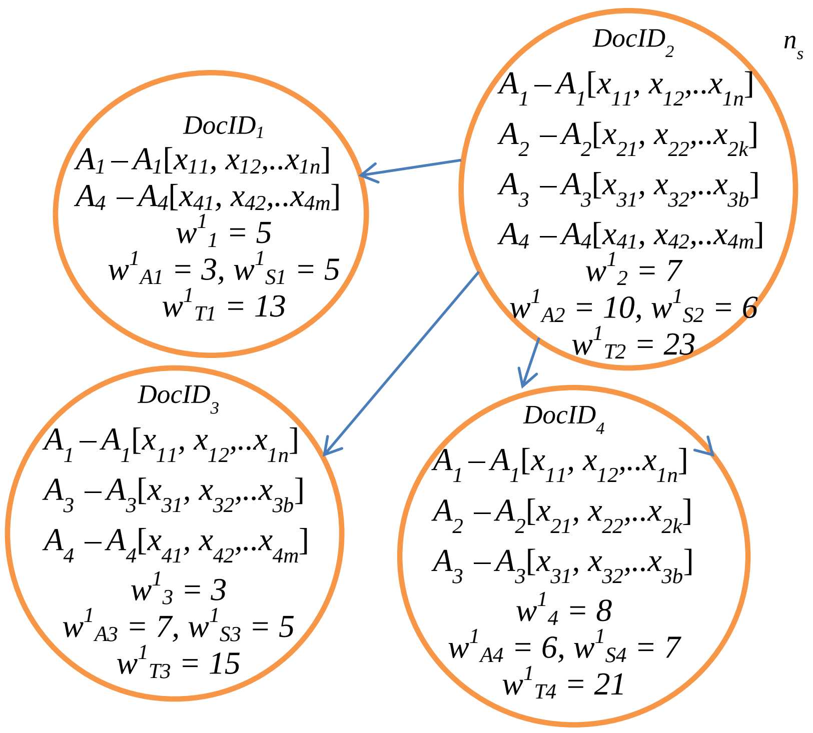
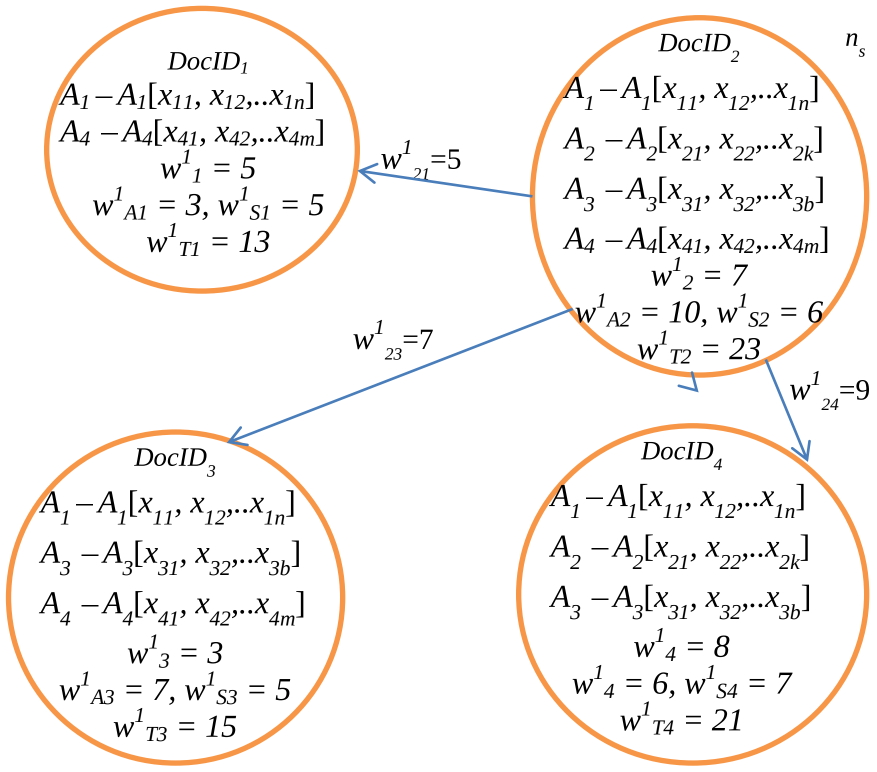


Figure 22: Step 2.4 - Graph G_1 with edges

- An initial edge weight $w^1_{k.m}$ is associated with each edge (k,m).



$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_1 = 5$
$w^1_{A1} = 3, w^1_{S1} = 5$
$w^1_{T1} = 13$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_2 = 7$
$w^1_{A2} = 10, w^1_{S2} = 6$
$w^1_{T2} = 23$
$w^1_{21}=5$
$w^1_{23}=7$
$w^1_{24}=9$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_3 = 3$
$w^1_{A3} = 7, w^1_{S3} = 5$
$w^1_{T3} = 15$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_4 = 8$
$w^1_4 = 6, w^1_{S4} = 7$
$w^1_{T4} = 21$

Figure 23: Step 2.5 - Graph G_1 with edges and its initial weights

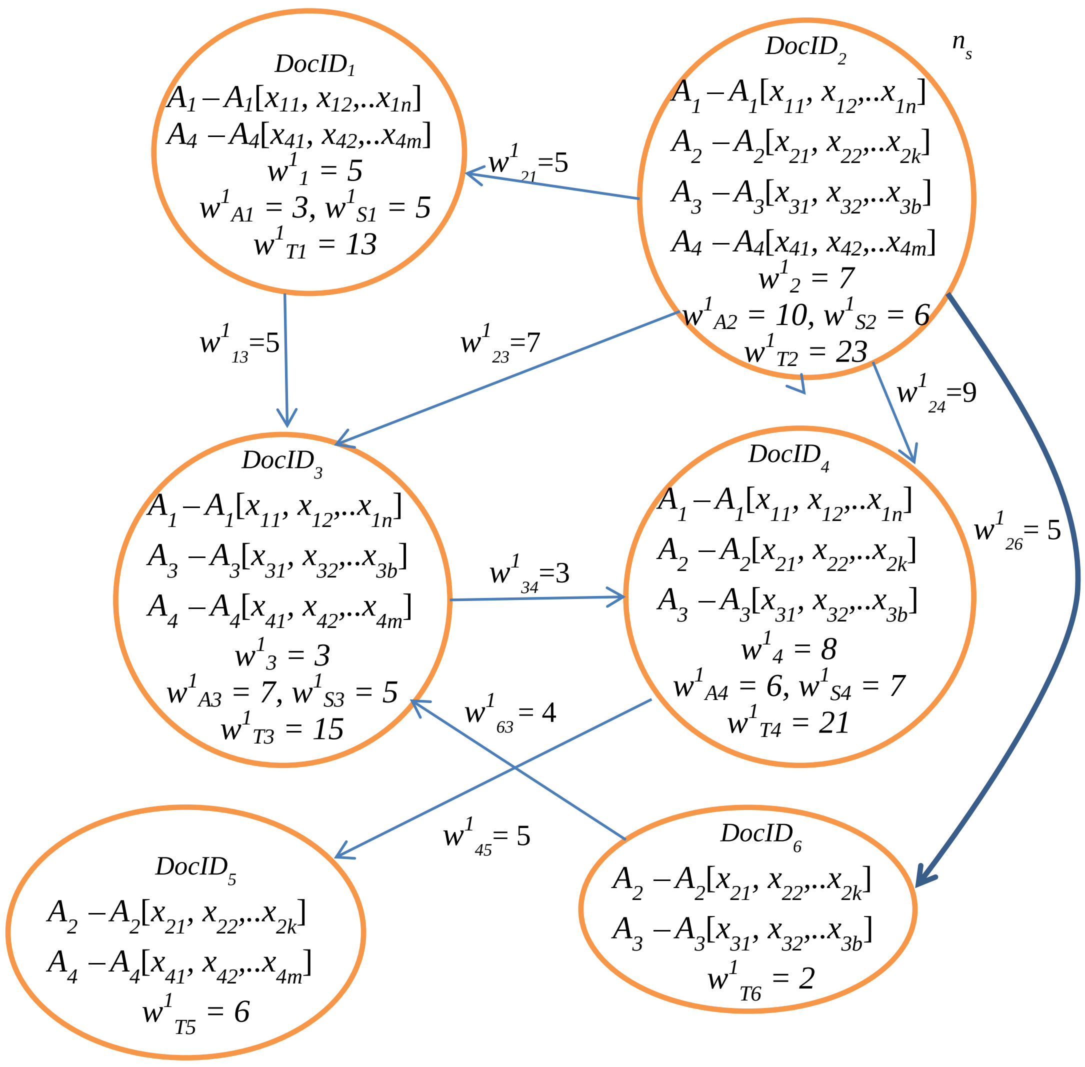
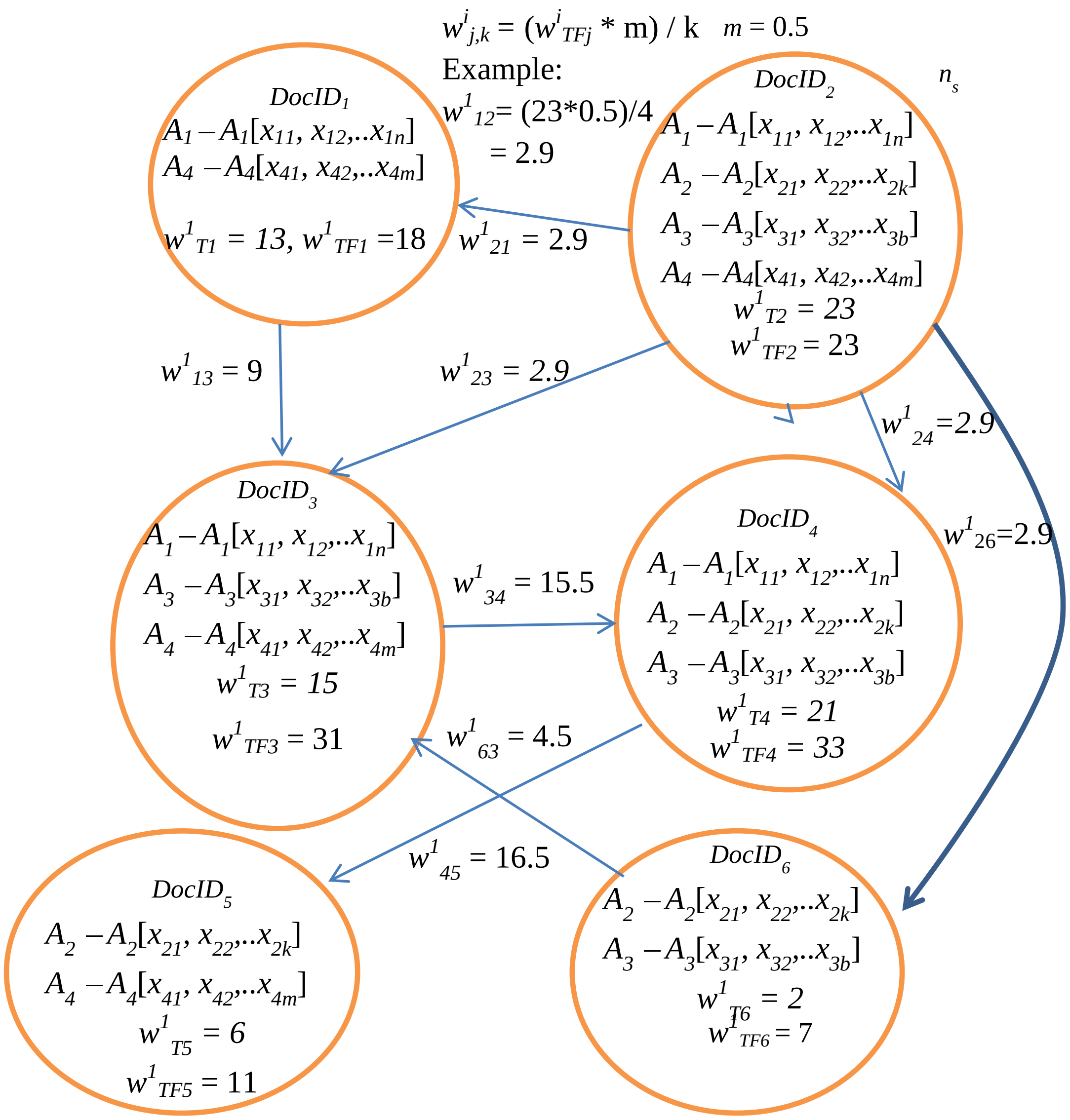


Figure 24: Step 2.6 - Constructed Graph G_1 with initial weights on edges and total weights on nodes.

- Final weight of the node is, $w^i_{TF_j} = w^i_{T_j} + \sum_{e^i_{1j}}^{e^i_{kj}} w^i_{kj}$

  $w^i_{TF_j}$ - The weight of the node added to the weight of all the initial incoming edges. $w^1_{TF_1} = w^1_{T_1} + w^1_{21} = 13+5 = 18$

- New final weight on edges, $w^i_{j,k} = (w^i_{TFj} * m) / k$



$w^i_{j,k} = (w^i_{TFj} * m) / k$ $m = 0.5$
Example:
$w^1_{12} = (23*0.5)/4 = 2.9$
n_s
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T1} = 13, w^1_{TF1} = 18$
$DocID_2$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T2} = 23$
$w^1_{TF2} = 23$
$w^1_{21} = 2.9$
$w^1_{13} = 9$
$w^1_{23} = 2.9$
$w^1_{24} = 2.9$
$w^1_{26} = 2.9$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T3} = 15$
$w^1_{TF3} = 31$
$w^1_{34} = 15.5$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_{T4} = 21$
$w^1_{TF4} = 33$
$w^1_{63} = 4.5$
$w^1_{45} = 16.5$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T5} = 6$
$w^1_{TF5} = 11$
$DocID_6$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_{T6} = 2$
$w^1_{TF6} = 7$

Figure 25: Step 2.7 - Graph G_1 with new weights on edges

- Total weights of each node n_i in the graph G_N using the formula,

$$w^i_{Totalj} = m * w^i_{TFj} + \sum_{e^i_{1j}}^{e^i_{kj}} w^i_{kj}$$

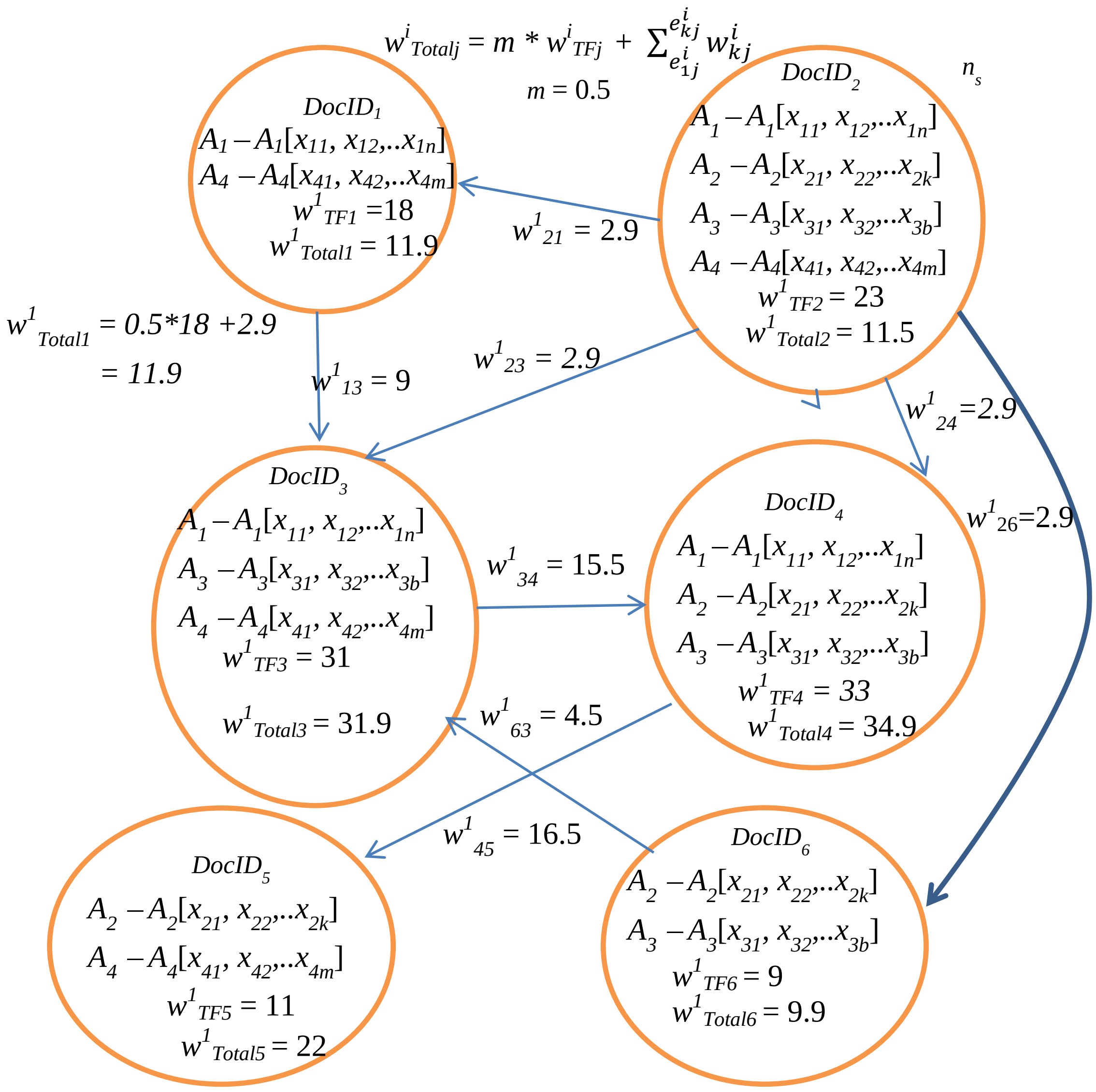
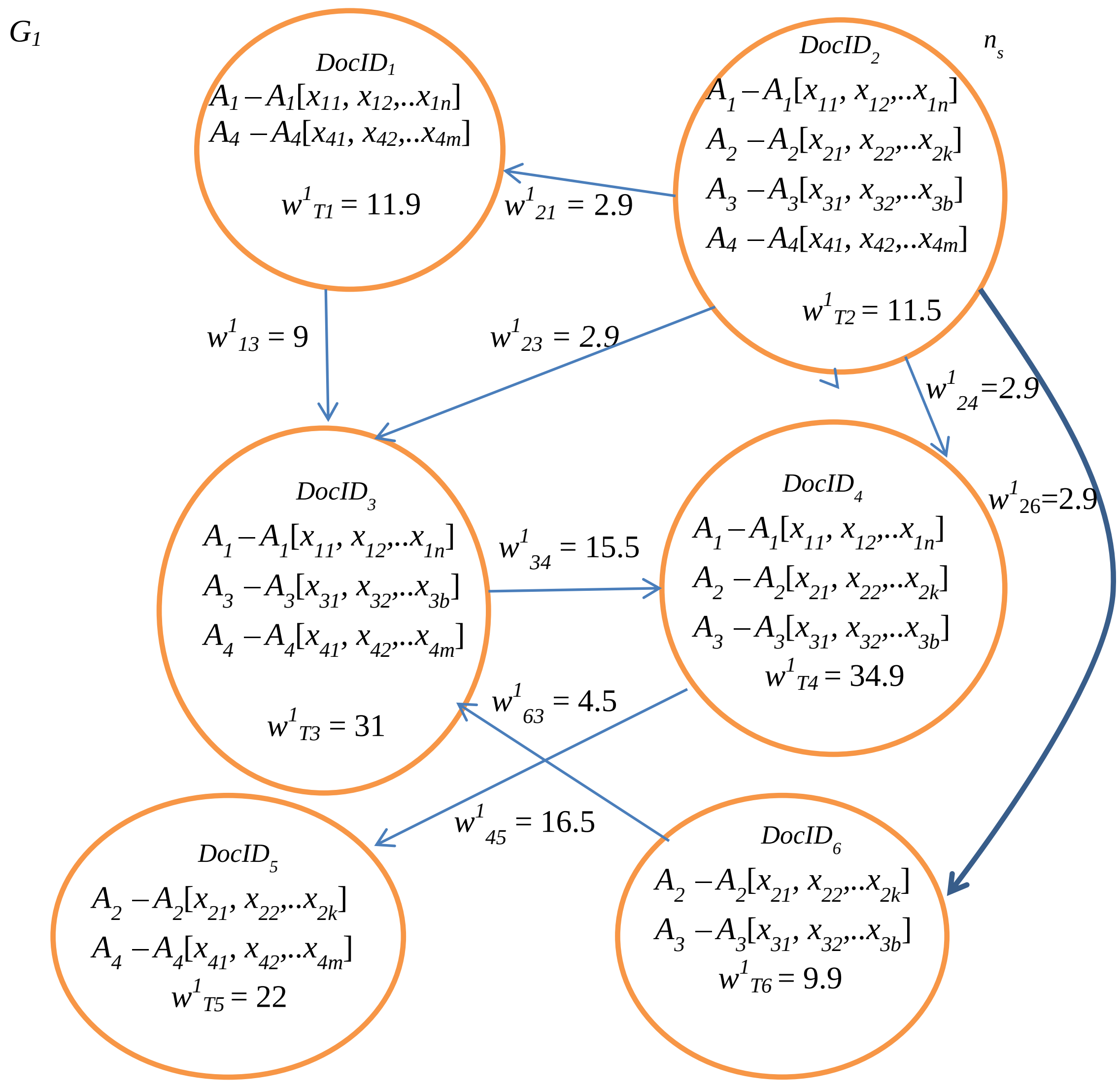

$w^{i}_{Totalj} = m * w^{i}_{TFj} + \sum_{e^{i}_{1j}}^{e^{i}_{kj}} w^{i}_{kj}$
$m = 0.5$
n_s
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{TF1} = 18$
$w^1_{Total1} = 11.9$
$DocID_2$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{TF2} = 23$
$w^1_{Total2} = 11.5$
$w^1_{21} = 2.9$
$w^1_{Total1} = 0.5*18 + 2.9 = 11.9$
$w^1_{13} = 9$
$w^1_{23} = 2.9$
$w^1_{24} = 2.9$
$w^1_{26} = 2.9$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{TF3} = 31$
$w^1_{Total3} = 31.9$
$w^1_{34} = 15.5$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_{TF4} = 33$
$w^1_{Total4} = 34.9$
$w^1_{63} = 4.5$
$w^1_{45} = 16.5$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{TF5} = 11$
$w^1_{Total5} = 22$
$DocID_6$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_{TF6} = 9$
$w^1_{Total6} = 9.9$

Figure 26: Step 3 - Final constructed graph G_1 with final total weights

- Repeat the same for all keywords $A_i \ldots A_m$ with input vectors $A_i[x_1, x_2, \ldots x_n] \ldots A_p[y_1, y_2, \ldots y_n]$ respectively.
- Superimpose the graphs G_1, G_2, G_3, and construct the final graph G_f.



G_1
n_s
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T1} = 11.9$
$DocID_2$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T2} = 11.5$
$w^1_{21} = 2.9$
$w^1_{13} = 9$
$w^1_{23} = 2.9$
$w^1_{24} = 2.9$
$w^1_{26} = 2.9$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T3} = 31$
$w^1_{34} = 15.5$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_{T4} = 34.9$
$w^1_{63} = 4.5$
$w^1_{45} = 16.5$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^1_{T5} = 22$
$DocID_6$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^1_{T6} = 9.9$

Figure 27: Step 7 - Constructed Graph G_1 for keyword A_1

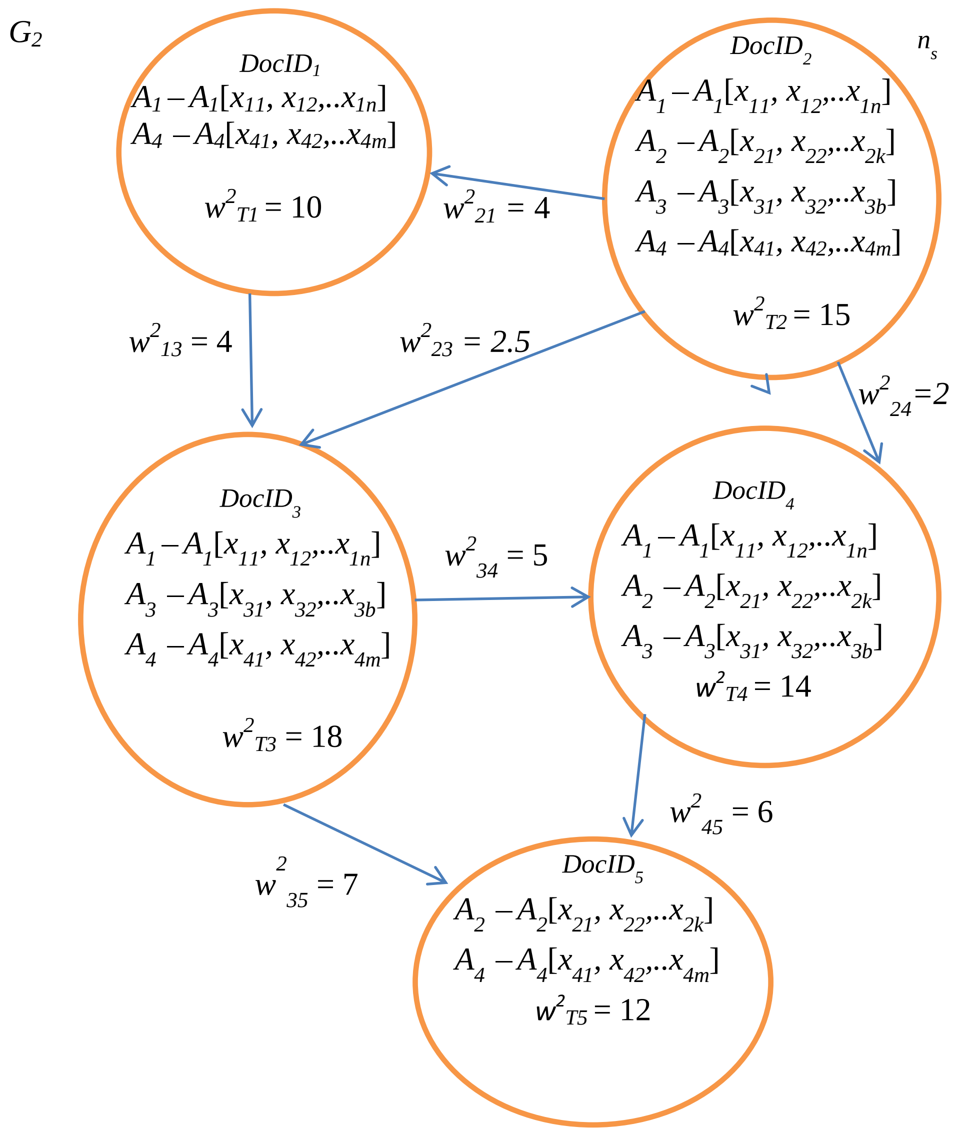
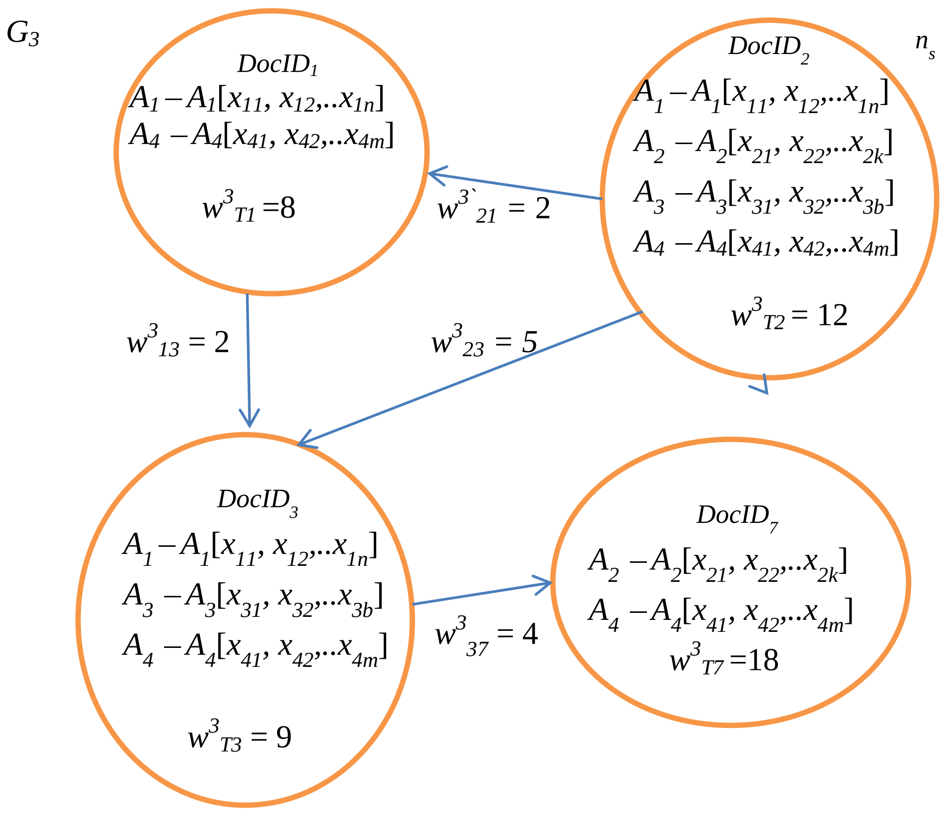

G_2
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^2_{T1} = 10$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^2_{T2} = 15$
$w^2_{21} = 4$
$w^2_{13} = 4$
$w^2_{23} = 2.5$
$w^2_{24} = 2$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^2_{T3} = 18$
$w^2_{34} = 5$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^2_{T4} = 14$
$w^2_{45} = 6$
$w^2_{35} = 7$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^2_{T5} = 12$

Figure 28: Step 7 - Constructed Graph G_2 for keyword A_2



G_3
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^3_{T1} = 8$
$DocID_2$
n_s
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^3_{T2} = 12$
$w^3_{21} = 2$
$w^3_{13} = 2$
$w^3_{23} = 5$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^3_{T3} = 9$
$w^3_{37} = 4$
$DocID_7$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^3_{T7} = 18$

Figure 29: Step 7 - Constructed Graph G_3 for keyword A_3

The Final graph G_f:

Superimposing identical nodes of G_1,G_2,G_3 And Merging edges Of identical nodes

$w^f_{T1}=w^i_{T1}+w^j_{T1}+w^p_{T1}$

$w^f_{T1}=w^1_{T1}+w^2_{T1}+w^3_{T1}$

$11.9+10+8=29.9$

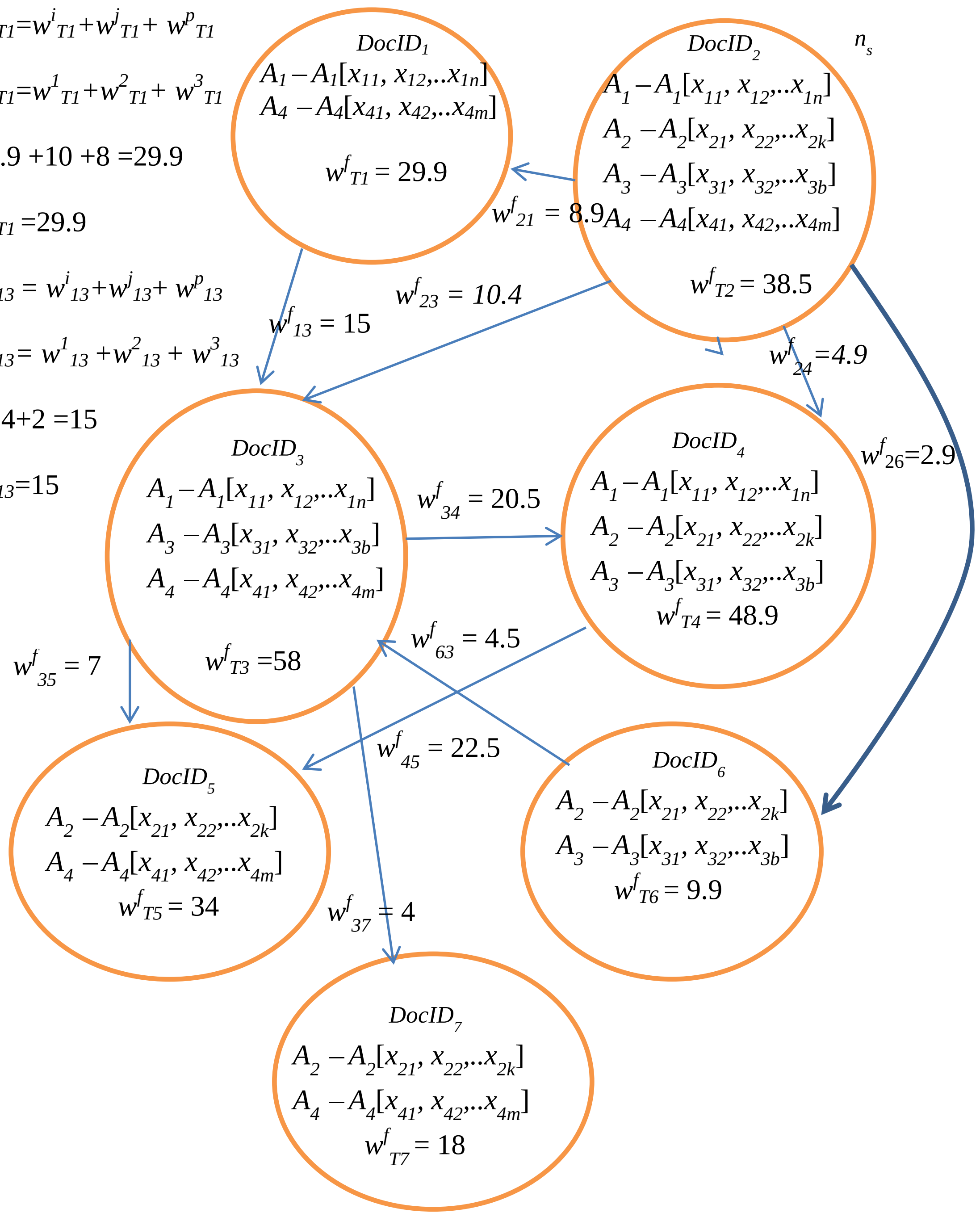
$w^f_{T1}=29.9$

$w^f_{13}=w^i_{13}+w^j_{13}+w^p_{13}$

$w^f_{13}=w^1_{13}+w^2_{13}+w^3_{13}$

$9+4+2=15$

$w^f_{13}=15$



$DocID_1$
$A_1-A_1[x_{11}, x_{12},..x_{1n}]$
$A_4-A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{T1} = 29.9$
$DocID_2$
n_s
$A_1-A_1[x_{11}, x_{12},..x_{1n}]$
$A_2-A_2[x_{21}, x_{22},..x_{2k}]$
$A_3-A_3[x_{31}, x_{32},..x_{3b}]$
$A_4-A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{T2} = 38.5$
$w^f_{21} = 8.9$
$w^f_{23} = 10.4$
$w^f_{13} = 15$
$w^f_{24} = 4.9$
$w^f_{26} = 2.9$
$DocID_3$
$A_1-A_1[x_{11}, x_{12},..x_{1n}]$
$A_3-A_3[x_{31}, x_{32},..x_{3b}]$
$A_4-A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{T3} = 58$
$w^f_{34} = 20.5$
$DocID_4$
$A_1-A_1[x_{11}, x_{12},..x_{1n}]$
$A_2-A_2[x_{21}, x_{22},..x_{2k}]$
$A_3-A_3[x_{31}, x_{32},..x_{3b}]$
$w^f_{T4} = 48.9$
$w^f_{63} = 4.5$
$w^f_{35} = 7$
$w^f_{45} = 22.5$
$DocID_5$
$A_2-A_2[x_{21}, x_{22},..x_{2k}]$
$A_4-A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{T5} = 34$
$DocID_6$
$A_2-A_2[x_{21}, x_{22},..x_{2k}]$
$A_3-A_3[x_{31}, x_{32},..x_{3b}]$
$w^f_{T6} = 9.9$
$w^f_{37} = 4$
$DocID_7$
$A_2-A_2[x_{21}, x_{22},..x_{2k}]$
$A_4-A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{T7} = 18$

Figure 30: Step 8 - Construction of Superimposed Final Graph G_f

- In the fully constructed final superimposed graph G_F, the final total weight of the node n_{fi} using the formula, $w^f_{Total_j} = w^f_{T_j} + \sum_{e^f_{1j}}^{e^f_{kj}} w^f_{kj}$

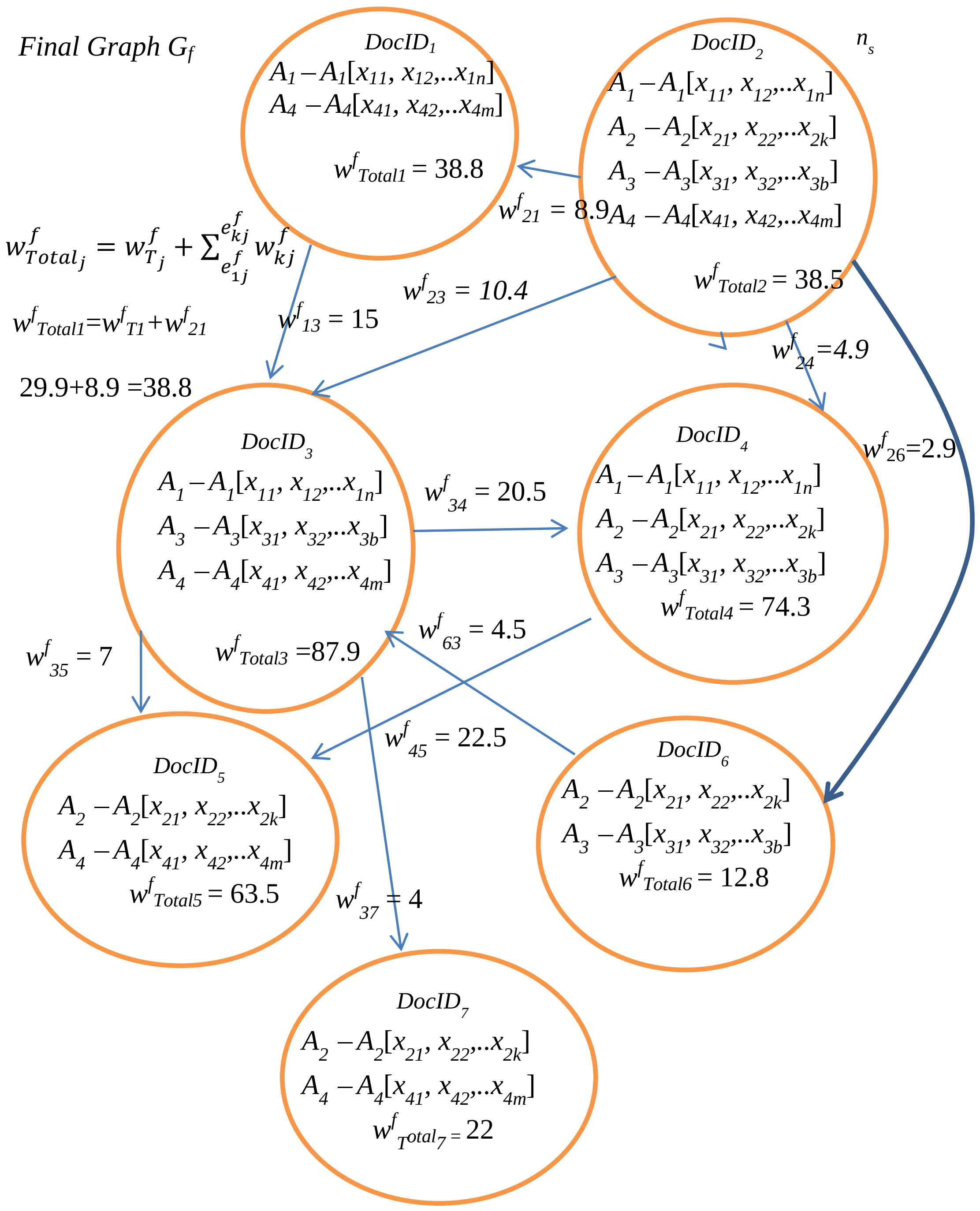

Final Graph G_f
n_s
$DocID_1$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{Total1} = 38.8$
$DocID_2$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{Total2} = 38.5$
$w^f_{21} = 8.9$
$w^f_{Total_j} = w^f_{T_j} + \sum_{e^f_{1j}}^{e^f_{kj}} w^f_{kj}$
$w^f_{Total1}=w^f_{T1}+w^f_{21}$
29.9+8.9 =38.8
$w^f_{13} = 15$
$w^f_{23} = 10.4$
$w^f_{24} = 4.9$
$w^f_{26} = 2.9$
$DocID_3$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{Total3} = 87.9$
$w^f_{34} = 20.5$
$DocID_4$
$A_1 - A_1[x_{11}, x_{12},..x_{1n}]$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^f_{Total4} = 74.3$
$w^f_{63} = 4.5$
$w^f_{35} = 7$
$w^f_{45} = 22.5$
$DocID_5$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{Total5} = 63.5$
$DocID_6$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_3 - A_3[x_{31}, x_{32},..x_{3b}]$
$w^f_{Total6} = 12.8$
$w^f_{37} = 4$
$DocID_7$
$A_2 - A_2[x_{21}, x_{22},..x_{2k}]$
$A_4 - A_4[x_{41}, x_{42},..x_{4m}]$
$w^f_{Total_7} = 22$

Figure 31: Step 9.3 - Constructed Graph G_f with final total weights

- In the connected graph G_f, rank the nodes with the highest total weight w^f_{Totaln} first.

Ranking: Rank1: Node2 ($DocID_2$), Rank 2: Node3 ($DocID_3$), and Rank3: Node4($DocID_4$) Since Node2 ($DocID_2$) is the start node n_s, it is given the rank 1, as per the algorithm.

3.8 TESLA APPROACH FOR SENDING SECURE SMS:

SMS transmission in plain text through GSM is not considered to be secure. In order to maintain confidentiality and data security, we use TESLA (Timed Efficient Stream Loss-tolerant Authentication) broadcast authentication protocol for the proposed filter architecture to send secure SMS. Since the key generation for TESLA approach is dynamic using the pseudo-random function, we don't need to store the keys. For this reason, we used the TESLA approach to send secure SMS across the GSM network rather than using Elliptical Curve Cryptography (ECC), where the public and private key need to be stored.

SENDER (PATIENT'S SMART PHONE):

In this model, a sender's (patient) smart phone composes the alert SMS {Mi}. The sender splits the time into equal intervals. In each interval, the patient's smart phone may send zero or multiple SMS. The smart phone sends the initial authentication SMS packet to the hospital containing, time interval duration, start time, the length of the key chain N, key disclosure delay and key commitment to the key chain Each key of the key chain is used in a onetime interval, and the number of SMS sent per time interval may vary from 0 to many.

RECEIVER (HOSPITAL):

When the receiver (hospital) receives the alert message M_i, it uses the self-authenticating key disclosed in M_j to determine i. When the hospital receives the disclosed key first, it checks whether it already knows the key K_i or a later key K_j (j>i). If K_i is the latest key received by the hospital, the hospital checks the legitimacy of K_i. The hospital then computes $K'_i = F'(K_i)$ and verifies the authenticity of packets of interval i and of the previous intervals if the receiver did not yet receive the keys for these intervals.

By using the TESLA approach, the alert SMS sent to the hospital is authenticated using an asymmetric cryptography scheme and the transmission of SMS over the GSM network is achieved in a secure manner.

CHAPTER IV

SIMULATION

4.1 INTRODUCTION

The objective of the simulation is to implement parts of the proposed EMR (Electronic Medical Record) system on the cloud, implement part of the proposed filter system on android and perform a simple case study on the proposed priority ranking algorithm. For building the EMR system on the cloud, we used cloud foundry [27], an open source PaaS (Platform as a Service) which gives an environment to deploy java-spring based web-applications. We built this EMR system for the hospital to manage their patients' health information.

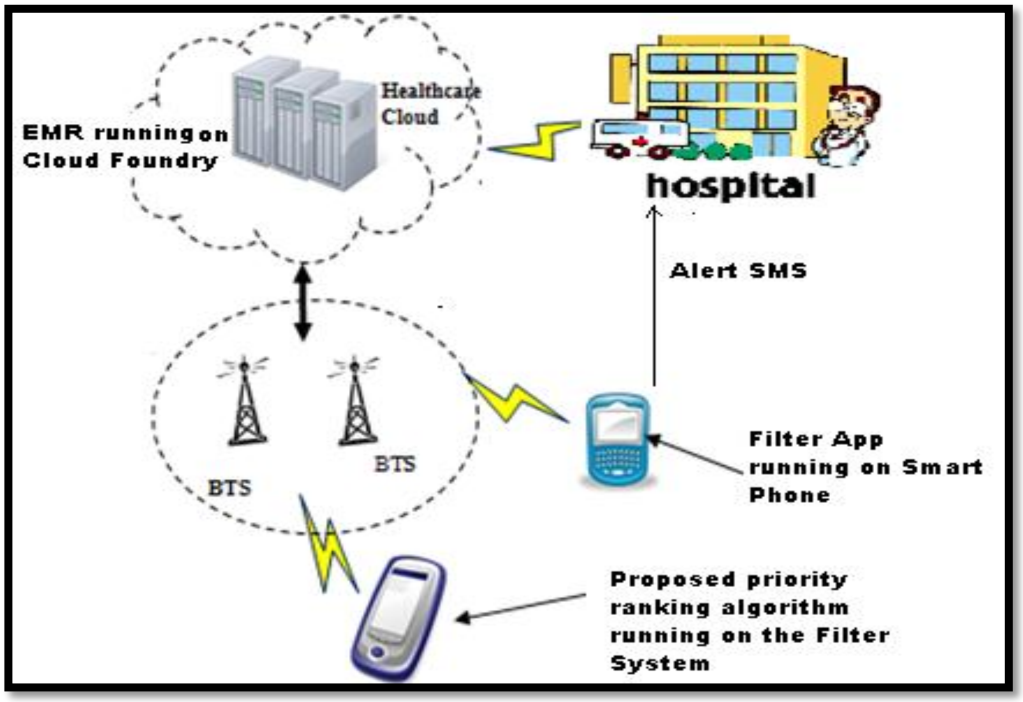

Healthcare Cloud
EMR running on Cloud Foundry
hospital
Alert SMS
BTS
BTS
Filter App running on Smart Phone
Proposed priority ranking algorithm running on the Filter System

Figure 32: Simulated Architecture

Simulation for the filter system was performed on the android emulator on Eclipse IDE, by developing an android filter app which sends an alert SMS on seeing abnormal health data. We tested the app by running two instances (one serves as a patient and the other serves as the hospital) of the emulator and sending alert SMS from one instance to another. We also simulated the proposed priority ranking algorithm using Java 1.6 on Eclipse IDE, using some simulated abnormal health data.

4.2 CLOUD FOUNDRY, AN OPEN SOURCE PaaS DEPLOYMENT MODEL

Cloud Foundry is an open-source platform-as-a-service environment offered by VMware that provides the environment to host multiple languages and frameworks in an open stack of application software that can run on both outside and inside the firewall [27, 28]. The main features of Cloud Foundry are: Choice of developer frameworks, Choice of application infrastructure services, and Choice of clouds

By offering an open architecture in all three dimensions, Cloud Foundry overcomes major limitations found in today's PaaS solutions. The distinct property about Cloud Foundry is the support for the Spring Framework and MySQL, which helps to deploy Java applications based on the Spring Framework on Cloud Foundry.

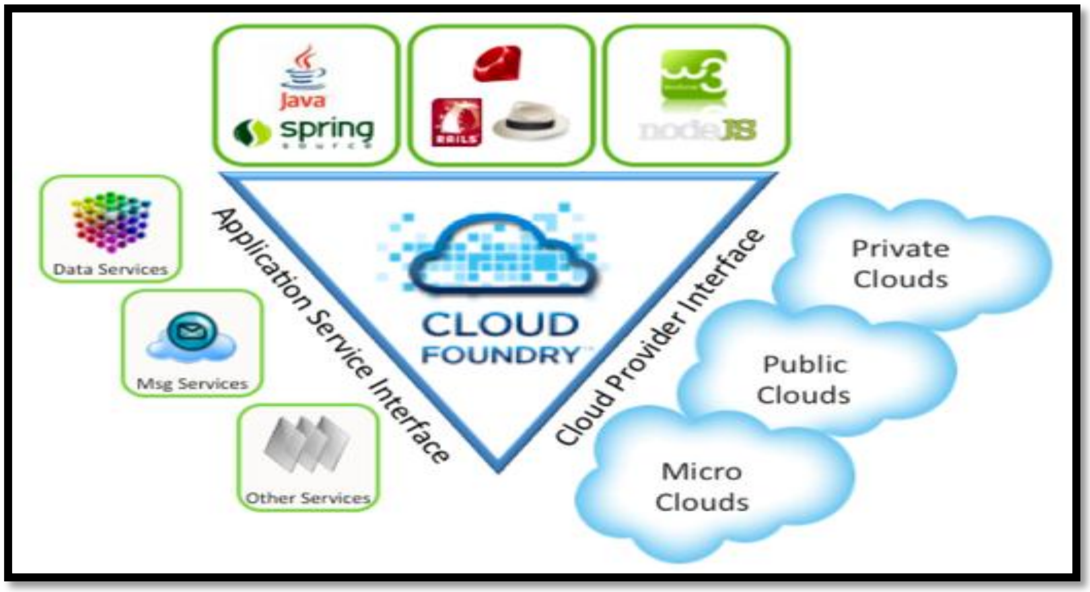


Figure 33: Cloud Foundry, Open Source Deployment Model [27]

4.3 CLOUD FOUNDRY ARCHITECTURE:

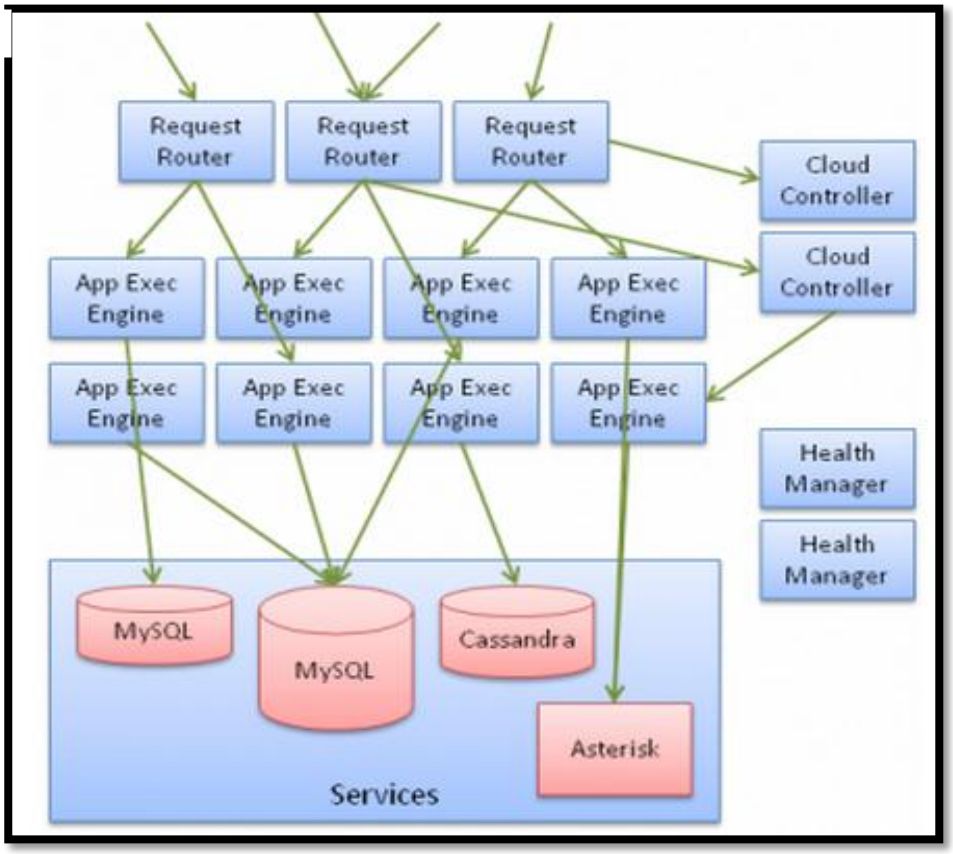


Figure 34: Cloud Foundry Architecture [28]

The pieces that make up Cloud Foundry [28] are:

App execution engine: This is the core that runs the application and launches and manages the Rails, Java, and other language app servers. Scaling up the app, results in more app execution engines launching an app server with the individual's code.

Request router: Being the front door to the PaaS, it accepts all the HTTP requests for all the applications running in the PaaS and routes them to the best app execution engine that runs the appropriate application code. The hostname used by each application is told to the request router, and it keeps track of the available app execution engines for each app.

Cloud controller: The external API used by tools to load/unload apps and control their environment, including the number of app execution engines that should run each application is implemented by the cloud controller. It creates the bundles that app execution engines load to run an application as part of taking in new applications.

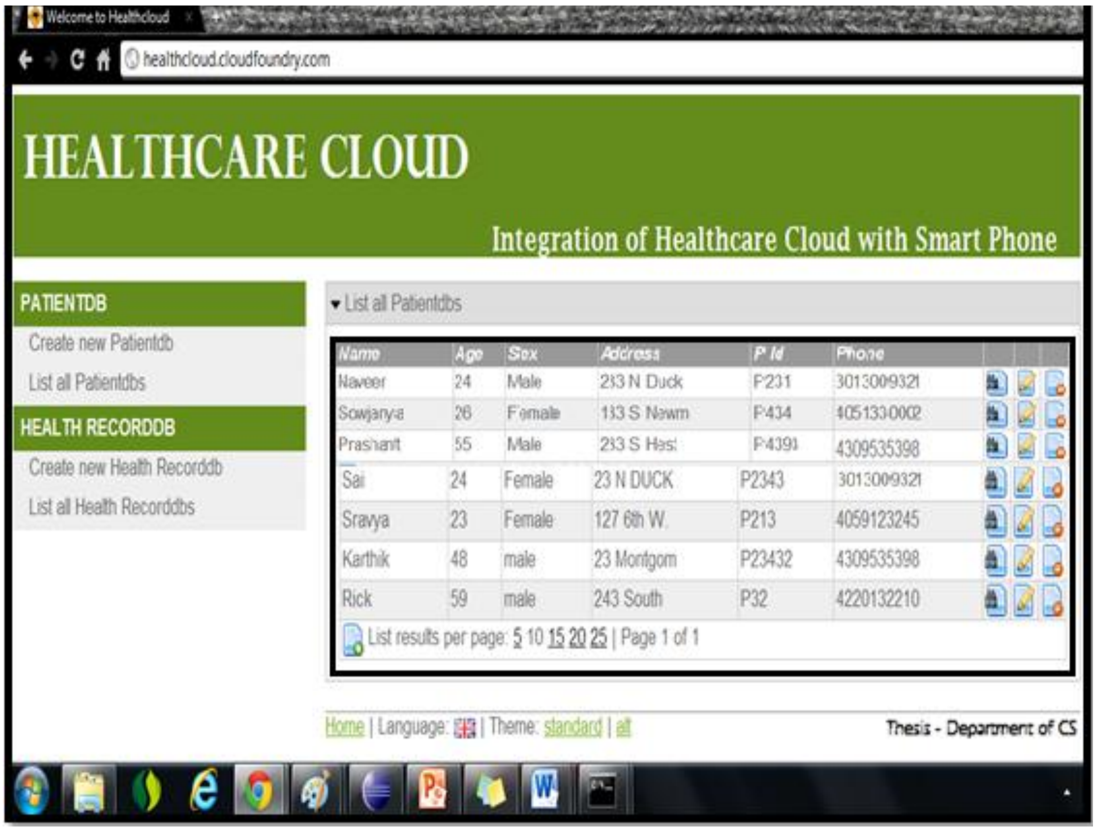
Services: Data storage and other functions that can be leveraged by applications are provided by a set of services. In an analogy with operating systems, these are the device drivers. Each service consists of: the application implementing the service itself (like MySQL, MongoDB, redis, etc.) and a Cloud Foundry management layer that establishes the connections between applications and the service itself.

Health manager: Health manager is responsible for keeping applications alive. It also ensures that if an app execution engine crashes, then the applications it ran are to be restarted elsewhere.

All these parts are tied together using a simple message bus, which, among other things, allows all the servers to find each other.

4.4 EMR HEALTHCARE SYSTEM RUNNING ON CLOUD FOUNDRY

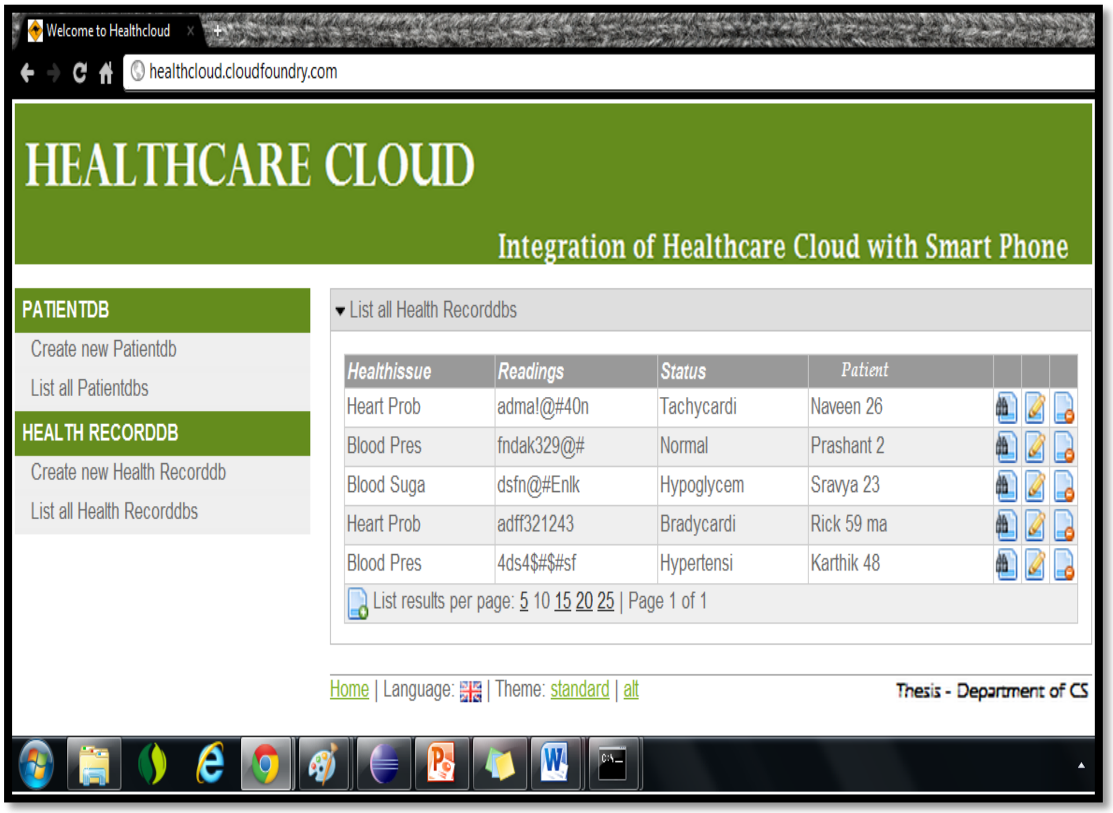
The developed EMR Healthcare system for the proposed architecture is a spring based java web application running on cloud foundry. This EMR system was developed for the hospital to manage each patient's profile and electronic medical records (EMR). This receives the encrypted health data from the patient's smart phone and the data gets stored in the cloud. Here, cloud foundry acts as a centralized database to store and manage a patient's personal and health information. The following two screenshots shows the EMR running on cloud foundry.


Welcome to Healthcloud
healthcloud.cloudfoundry.com
HEALTHCARE CLOUD
Integration of Healthcare Cloud with Smart Phone
PATIENTDB
Create new Patientdb
List all Patientdbs
HEALTH RECORDDB
Create new Health Recorddb
List all Health Recorddbs
▾ List all Patientdbs
Home | Language: | Theme: standard | alt
Thesis - Department of CS

Name	Age	Sex	Address	P Id	Phone
Naveer	24	Male	233 N Duck	P231	3013009321
Sowjanya	26	Female	133 S Newm	P434	4051330002
Prashant	55	Male	233 S Hast	P4391	4309535398
Sai	24	Female	23 N DUCK	P2343	3013009321
Sravya	23	Female	127 6th W	P213	4059123245
Karthik	48	male	23 Montgom	P23432	4309535398
Rick	59	male	243 South	P32	4220132210

List results per page: 5 10 15 20 25 | Page 1 of 1

Figure 35(a). EMR Healthcare Cloud for Hospital (Patient Data)


Welcome to Healthcloud

healthcloud.cloudfoundry.com

HEALTHCARE CLOUD

Integration of Healthcare Cloud with Smart Phone

PATIENTDB
- Create new Patientdb
- List all Patientdbs

HEALTH RECORDDB
- Create new Health Recorddb
- List all Health Recorddbs

▾ List all Health Recorddbs

Healthissue	Readings	Status	Patient
Heart Prob	adma!@#40n	Tachycardi	Naveen 26
Blood Pres	fndak329@#	Normal	Prashant 2
Blood Suga	dsfn@#Enlk	Hypoglycem	Sravya 23
Heart Prob	adff321243	Bradycardi	Rick 59 ma
Blood Pres	4ds4$#$#sf	Hypertensi	Karthik 48

List results per page: 5 10 15 20 25 | Page 1 of 1

Home | Language: | Theme: standard | alt

Thesis - Department of CS

Figure 35(b): EMR Healthcare Cloud for a Hospital (Health Record)

4.5 ANDROID EMULATOR

Android is a software stack for mobile devices that includes an operating system, middleware and key applications [29].

The main components of the Android architecture are:

1. Applications: Android applications are written in Java. Calendar, email client, SMS program, graphs, making phone calls, accessing the Web browser, accessing your contacts list and others are some of the basic applications.

2. Application Framework: Android developers follow a skeleton or framework and can access all framework APIs and manage phones' basic functions like resource allocation, switching between processes or programs, telephone applications, and keeping track of the phone's physical location.

3. Libraries: This layer consists of Android libraries written in C and C++, which are used by various systems and are exposed to Android developers via Android Application framework. Handling different kinds of data are specified by these libraries. For example, media, graphics, 3d, SQLite, web browser library etc. This layer also includes the Android runtime layer with a set of core java libraries and DVM (Dalvik Virtual Machine).

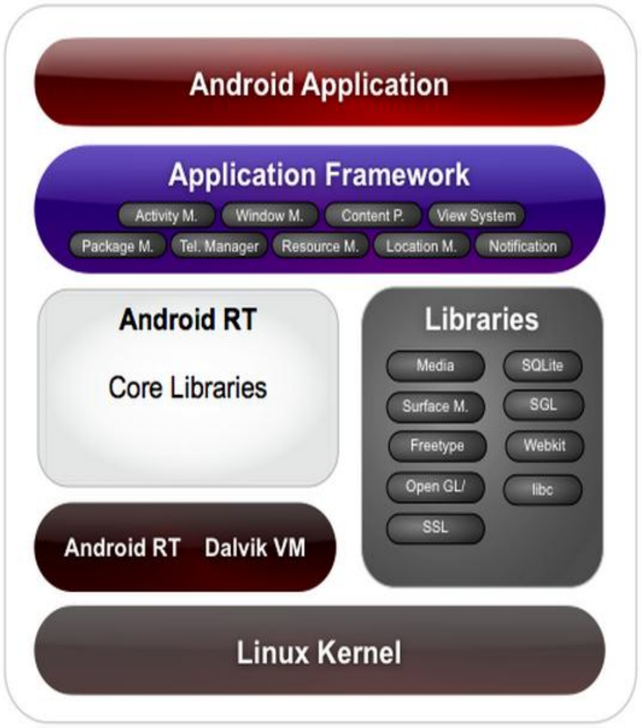


Figure 36: Android Architecture [29]

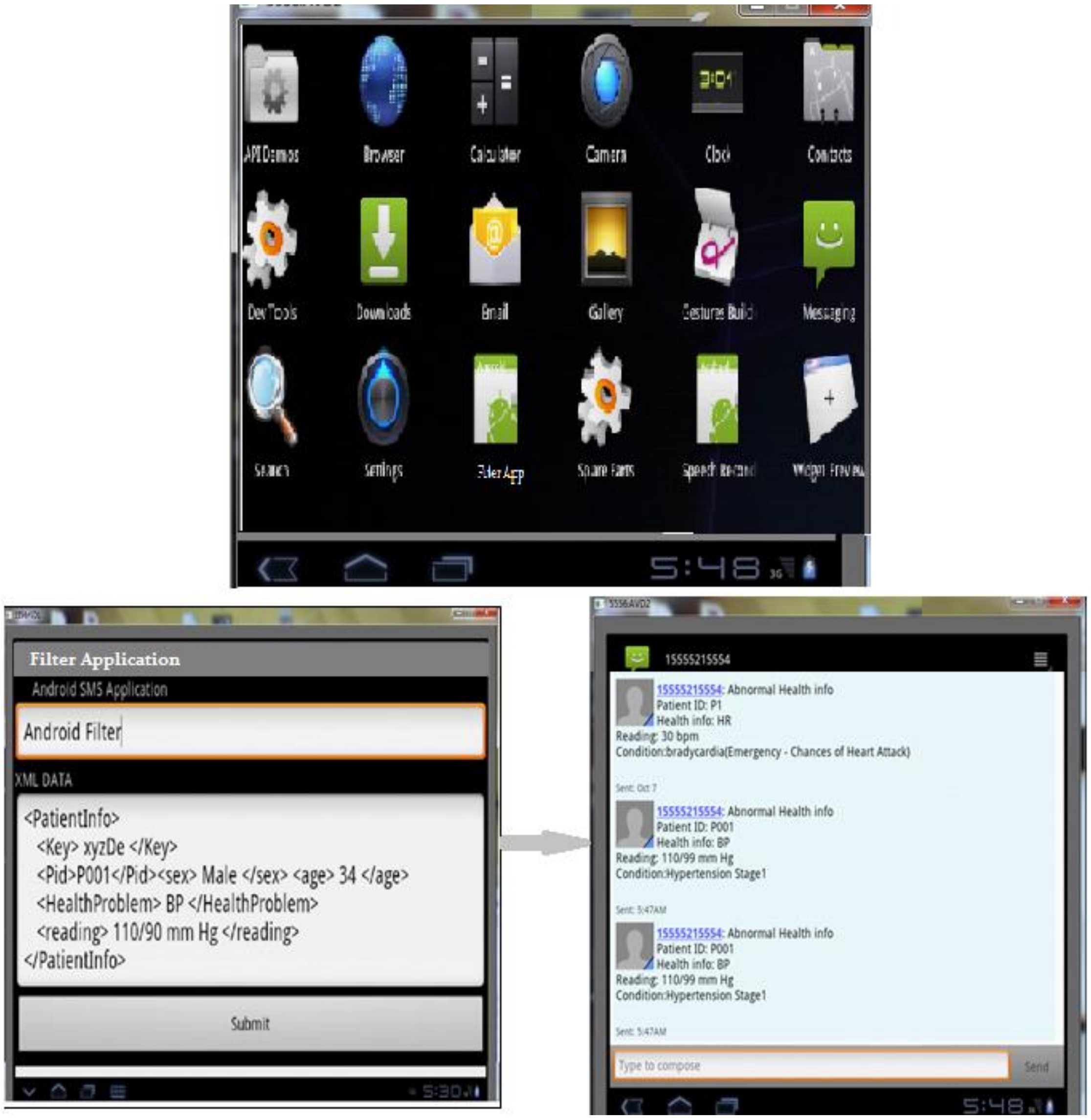
4. Runtime Android: The java libraries require a set of base libraries that are located in this layer. Every Android application gets its own instance of DVM, which has been written so that a device can run multiple VMs efficiently.

5. Kernel – Linux: Android's Memory Management Programs, security settings, power management software, and several drivers for hardware, file system access; networking and inter-process-communication are included in this layer. The kernel also acts as an abstraction layer between hardware and the rest of the software stack.

In order to simulate the proposed filter system, we used Android Development Tools (ADT), a plugin for the Eclipse IDE that is designed to give one a powerful, integrated environment in which to build Android applications.

4.6 FILTER APPLICATION ON ANDROID:

The developed filter application running on the android emulator is for the proposed filter system. It takes the data from the healthcare app running on the emulator instance and sends it to the lookup table to find the severity level. If the data is abnormal, an alert SMS is sent to the hospital. Figure 37 shows the developed filter running on android.



```
<PatientInfo>
  <Key> xyzDe </Key>
  <Pid>P001</Pid><sex> Male </sex> <age> 34 </age>
  <HealthProblem> BP </HealthProblem>
  <reading> 110/90 mm Hg </reading>
</PatientInfo>
```

15555215554: Abnormal Health info
Patient ID: P1
Health info: HR
Reading: 30 bpm
Condition:bradycardia(Emergency - Chances of Heart Attack)
Sent: Oct 7

15555215554: Abnormal Health info
Patient ID: P001
Health info: BP
Reading: 110/99 mm Hg
Condition:Hypertension Stage1
Sent: 5:47AM

15555215554: Abnormal Health info
Patient ID: P001
Health info: BP
Reading: 110/99 mm Hg
Condition:Hypertension Stage1
Sent: 5:47AM

Figure 37: Android App Simulating Filter App

The XML data coming into the filter system would be,

```
<HealthRecord>
        <Pid> P001 </Pid>
        <Age> 45 </Age>
        <Sex> Male </Sex>
        <HealthProblem> HeartRate </HealthProb>
        <Value> 70 bpm </Value>
</HealthRecord>
```

The filter has a XML parser which gets all the patient information, and sends it to EMR system running on cloud, and the filter checks the health parameter. If any abnormality occurs, it sends an emergency alert to the hospital. Figure 37 shows how the filter app sends an alert SMS in android emulator.

4.7 PRIORITY RANKING ALGORITHM SIMULATION

The priority ranking algorithm was implemented and a simple case study is presented below.

EXPERIMENTATION UNITS:

Heart Rate Vector: [40 bpm, 65 bpm, 48 bpm, 70 bpm, 39 bpm]

Blood Sugar Vector: [170 mg, 190 mg, 240 mg, 260 mg, 300 mg]

The simulation was performed with Java 1.6 on Eclipse IDE. First, the input vector is given to the lookup table running on filter java class, which eliminates the normal data and takes only the abnormal value and sends it to graph. In Figure 38, graph G_A shows the constructed sub-graph for Heart Rate, and the weights are found by the simulator after iterating all the nodes in the graph using the priority ranking approach. Graph G_B shows the constructed sub-graph for Blood Sugar, and the weights are found by the simulator after iterating all the nodes in the graph.

In Figure 38, graph G_F shows the final graph which is constructed as a result of super-imposing the first two sub-graphs to find out the weights of each node after combining the graphs. Based on the resultant graph's weights, the nodes are ranked for the filter to find whom to send the alert SMS.

The input vector of abnormal data for Heart Rate is [40 bpm, 48 bpm, 39 bpm]and the initial weights for the graph having eight doctors are: w_{i1}=3, w_{i2}=2, w_{i3}=1, w_{i4}=4, w_{i5}=2, w_{i6}=1, w_{i7}=3, and w_{i8}=2.

The Simulation result for the above input is,

Final Weights: w_{f3}=1.20, w_{f2}=1.94, w_{f5}=2.0, w_{f8}=2.40, w_{f1}=2.4, w_{f6}=2.4, w_{f7}=2.80, w_{f4}=6.4, and w_{f8}=2.

Ranking: RANK 1: Default D1; RANK 2: Doctor D4; and RANK 3: Doctor D7.

The input vector of abnormal data for Blood Sugar is [240 mg, 260 mg, 300 mg] and the initial weights for the graph having six doctors are, w_{i1}=4, w_{i2}=1, w_{i3}=2, w_{i4}=3, w_{i5}=4 and w_{i6}=2

The Simulation result for the above input is,

Final Weights: w_{f6}=1.0, w_{f2}=2.0, w_{f3}=2.24, w_{f4}=3.20, w_{f1}=3.20 and w_{f5}=4.0.

Ranking: RANK 1: Default D1; RANK 2: Doctor D5; and RANK 3: Doctor D4

Now, we need to find the final list of doctors, to whom the data needs to be sent. This is achieved by superimposing the above two graphs, which gives

Final Weights: w_{f8}=2.80, w_{f7}=2.88, w_{f3}=2.88, w_{f2}=3.80, w_{f6}=4.0, w_{f1}=5.6, w_{f5}=5.6 and w_{f4}=11.20

Final Ranking: RANK 1: Default D1; RANK 2: Doctor D4; and RANK 3: Doctor D5

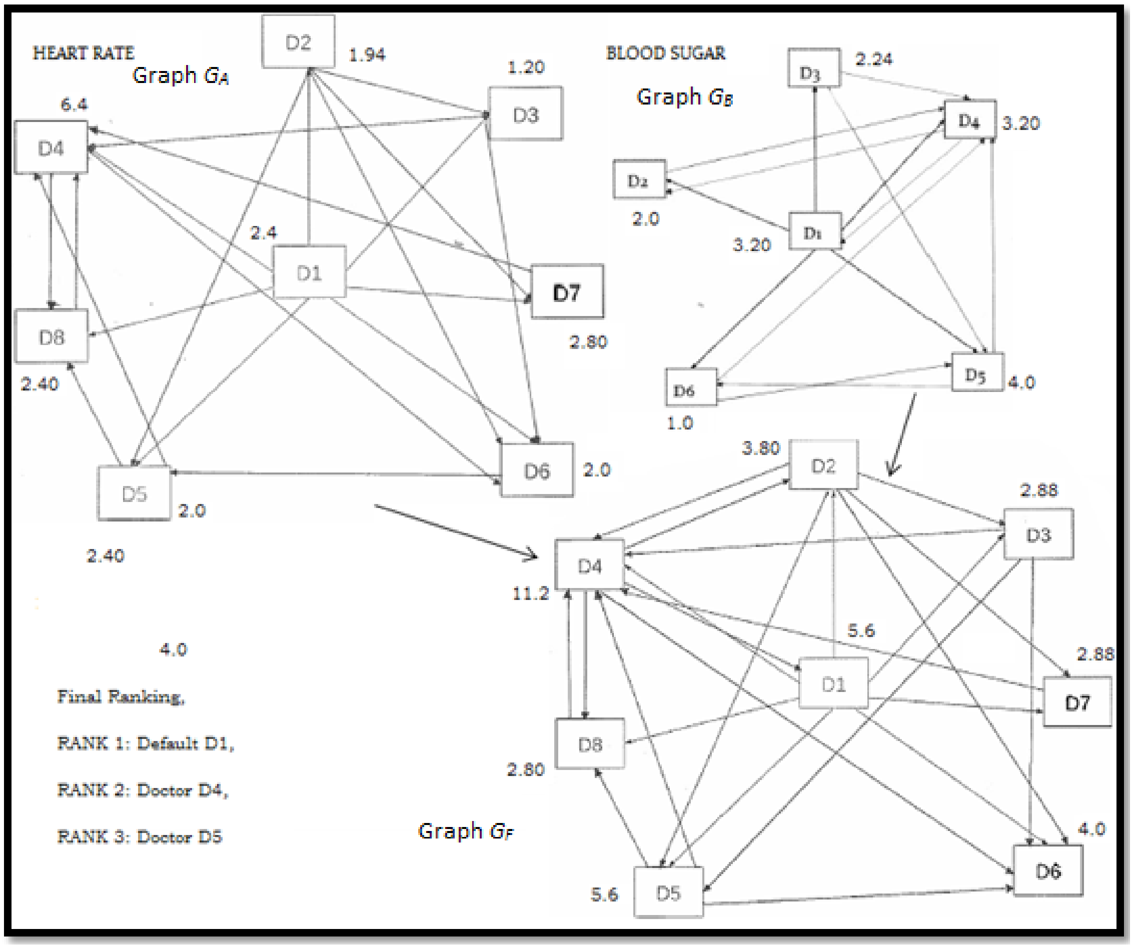


Figure 38: Simulated Graph

The proposed priority ranking algorithm is simulated with sample Heart Rate and Blood Sugar values and final ranks determined.

CHAPTER V

CONCLUSION

In this thesis, we have proposed a filter-based cloud system architecture for integrating wireless sensor networks with cloud computing for a secure healthcare system. Smart phones were considered as the sensor device monitoring the patient's health signs. The sensed health data are sent to the cloud, where the hospitals EMR system running on the cloud manages the patient's medical records. Whenever there is an emergency detected in the sensor data, the smart phone, using the filter app, sends the abnormal data to the hospital through an alert SMS. HIPAA [6] compliance policies enforces the control to access the patient health data. In order to complement this, the filter system uses the proposed key search and priority ranking algorithm. These two algorithms helped to rank the doctors based on the weights and find the list of doctors to whom the alert SMS needs to be sent. Additionally, we used TESLA [28] approach to encrypt the alert SMS being sent to the hospital.

The proposed system model is simulated by creating an EMR system for a hospital using Java Spring web services and deploying it on Cloud foundry, PaaS environment. The proposed filter system is simulated using the filter app running on the android emulator sending alert SMS to another emulator instance on detection of abnormal patient health data. The proposed priority ranking algorithm for finding the list of doctors to whom the abnormal data is to be sent is implemented using Java 1.6.

Our work has left several directions for future work. The smart phone integration with healthcare cloud for monitoring patient's vital health information is expected to have significant impact in the healthcare field. Future work may include real-time implementation of the proposed approach, and integrating Electronic Health Record (EHR) and Patient Health Record (PHR) running on healthcare cloud, with the patient's smart phone. Extensive research needs to be carried out on integrating healthcare monitoring sensors within the smart phones itself. Future research needs to be done on improving and optimizing the proposed ranking algorithms for the filter system. Security model for the Electronic Medical Record (EMR) for the healthcare organizations and insurance companies needs to be developed.

REFERENCES

[1] Carlos Oberdan Rolim, Fernando Luiz Koch, Carlos Becker Westphall, et al. “A Cloud Computing Solution for Patient’s Data Collection in Health Care Institutions”. *Second international conference on ETELEMED*, pp. 95-99, 2010.

[2] Biomedical wireless sensor network, http://ibmmsrit2010.wordpress.com/2010/08/26/biomedical-wireless-sensor-network-in-home-and-health-care-industry-j-v-alamelu-lecturer-s-elavaar-kuzhali-senior-lecturer-department-of-instrumentation-technology-msrit/. (Date last accessed August 26th 2010)

[3] Jensen. M, Schwenk. J, Gruschka. N, et al. “On Technical Security Issues in Cloud Computing”. *IEEE International Conference on Cloud Computing*, pp. 109-116, 2009.

[4] AjayKumar. S, Nachiappan. C, Periyakaruppan. K, et al. “Enhancing Portable Environment Using Cloud and Grid”. *International Conference Signal Processing Systems*, pp. 728-732, 2009.

[5] Zhang. R and Ling. L. “Security Models and Requirements for Healthcare Application Clouds”. *IEEE 3rd International Conference on Cloud Computing*, pp. 268-275, 2010.

[6] Summary of the HIPAA Privacy Rules, http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/privacysummary.pdf (Date last accessed May 2003)

[7] ANSI, ISO/TS 18308 Health Informatics-Requirements for an Electronic Health Record Architecture, ISO 2003.

[8] D. Garets and M. Davis, A HIMSS Analytics White Paper. Electronic Medical Records vs. Electronic Health Records: Yes, There Is a Difference. January 26, 2006. http:// www.himssanalytics.org/docs/wp_emr_ehr.pdf

[9] Werner. K and Wolfgang. B. “Combining Cloud Computing and Wireless Sensor Networks,” *proceeding of iiWAS*, 2009.

[10] Samuel Madden, J. Franklin, Joseph M Hellerstein, et al."TinyDB: An Acqusitional Query Processing System for Sensor Networks". *ACM Transactions on Database Systems*, Vol 30, Issue 1, 2005.

[11] Philip Levis, et al. "TinyOS: An Operating System for Wireless Sensor Networks" *in Ambient Intelligence*, 2005.

[12] U. Anliker et Al., “AMON : a wearable multiparameter medical monitoring and alert system", *IEEE Transactions on Information Technology in Biomedicine*, Vol. 8, Issue 4, pp. 415-427, 2004.

[13] L. Kristof, P. L. Lo Benny, W. P. Ng Jason, T. Surapa, K. Rachel, K. Simon, G. Hans-werner, S. Morris, W. Oliver, N. Phil, P. Nick, D. Ara, T. Chris,Y. Guang-zhong, “Medical Healthcare Monitoring with Wearable and Implantable Sensors,” *Proceedings International Conference on Ubiquitous Computing Ubicomp*, 2004.

[14] R. Giannantonio, F. Bellifemine, and M. Sgroi, "SPINE - (signal processing in node environment): A framework for healthcare monitoring applications based on body sensor networks", http://spine.tilab.com/papers-2008.htm (Date last accessed December17 2011).

[15] Rolim. C, Koch. F. L, Sekkaki. A, et al. “Telemedicine with Grids and Wireless Sensors Networks”. *International Conference on e-Medical Systems, IEEE Tunisia Section*, 2008.

[16] Medvedev .O, Kobelev. A, Schookin. S, Jatskovsky. M, Markarian. G, Sergeev. I, “Smart Phone-based Approach for Monitoring Vital Physiological Parameters in Humans,” *IFMBE Proceedings,* Volume 14, Part 28, 2007

[17] GAY. V and LEIJDEKKERS. P, “A Health Monitoring System Using Smart Phones and Wearable Sensors,” *International Journal of ARM,* VOL. 8, NO. 2, June 2007.

[18] Petteri Nurmi, Joonas Kukkonen, and Eemil Lagerspetz, "BeTelGeuse – A Tool for Bluetooth Data Gathering," *Proceedings of the ICST 2nd international conference on Body area networks*, 2007.

[19] Joonas Kukkonen, Eemil Lagerspetz, Petteri Nurmi, et al. "BeTelGeuse: A Platform for Gathering and Processing Situational Data" *IEEE Pervasive Computing*, vol. 8, pp. 49-56, 2009.

[20] Agoyi. M and Seral. D, “SMS SECURITY: AN ASYMMETRIC ENCRYPTION APPROACH,” *Sixth International Conference on Wireless and Mobile Communications*, 2010.

[21] W. Chou, Elliptic curve cryptography and its applications to Mobile Devices, University of Maryland, 2003,Technical report

[22] E. Barker and A. Roginsky, Recommendation for the Transitioning of Cryptographic Algorithms and Key Sizes, NIST SP 800-131, 2010, Technical Report.

[23] Understanding blood pressure readings, http://www.heart.org/HEARTORG/Conditions/HighBloodPressure/AboutHighBloodPressure/Understanding-Blood-Pressure-readings_UCM_301764_Article.jsp#.Tsr2BGPNltM (Date last accessed August 26th 2010)

[24] Hypoglycemia and hyperglycemia, http://www.friedmandiabetesinstitute.com/learn_about_diabetes/hypoglycemia_hyperglycemia/index.html (Date last accessed December17 2011).

[25] Page Rank Explained, http://netsavy.net/Graphics/PageRank.pdf (Date last accessed November 9th 2001)

[26] Perrig. A, Canetti. R, Tygar. J. D., Song. Dawn, "The Tesla Broadcast Authentication Protocol," *Cryptobytes,* Vol. 5, No. 2, 2002.

[27] Open PaaS, http://blogs.vmware.com/console/2011/04/cloud-foundry-delivering-on-vmwares-open-paas-strategy.html (Date last accessed April 12 2011).

[28] Cloud Foundry Architecture and Auto-scaling, http://blog.rightscale.com/2011/04/14/cloud-foundry-architecture-and-auto-scaling/ (Date last accessed April 14th 2011).

[29] Android Architecture, http://blog.zeustek.com/2010/11/11/android-architecture/ (Date last accessed November 11th 2010).

VITA

NAVEEN RAJ DHANAPAL

Candidate for the Degree of

Master of Science

Thesis: AN ARCHITECTURAL APPROACH FOR THE INTEGRATION OF WIRELESS SENSOR NETWORKS WITH CLOUD COMPUTING FOR A SECURE HEALTHCARE SYSTEM

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December 2011.

Completed the requirements for the Bachelor of Engineering in Computer Science and Engineering at Magna College of Engineering - Anna University, Chennai, Tamil Nadu, India in 2009.

Experience:

Graduate Teaching Assistant for CS 1113-Java Programming at the Department of Computer Science in fall'11, Oklahoma State University.
Stillwater, Oklahoma. August 2011 – December 2011

Graduate Assistant at Institute for Teaching and Learning Excellence (ITLE), Oklahoma State University.
Stillwater, Oklahoma. September 2010 – December 2011

Application Developer - Intern at Rackspace Hosting Inc.
San Antonio, Texas. May 2011 – August 2011

Web Developer for the Department of Zoology, Oklahoma State University.
Stillwater, Oklahoma. September 2010 – January 2011

Name: NAVEEN RAJ DHANAPAL Date of Degree: May 2012

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: AN ARCHITECTURAL APPROACH FOR THE INTEGRATION OF WIRELESS SENSOR NETWORKS WITH CLOUD COMPUTING FOR A SECURE HEALTHCARE SYSTEM

Pages in Study: 56 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

The general scope of our work is to propose an architecture to integrate the healthcare cloud with wireless sensor network (WSN) technology through smart phones. This will enable monitoring patient health wirelessly with the smart phone devices providing real-time updates of patient's health via the cloud to doctors and other medical professionals. To achieve this, we propose a filter system on a smart phone running different healthcare apps which monitor and record the patient's vital signs.

Findings and Conclusions:

The proposed system model is simulated by creating an EMR system for a hospital using Java Spring web services and deploying it on Cloud foundry, a Platform as a Service (PaaS) environment. The proposed filter system is simulated using the filter app running on the android emulator sending alert SMS to another emulator instance on detection of abnormal patient health data. A priority ranking algorithm for finding the list of doctors to whom the abnormal data is to be sent, is proposed and implemented using Java 1.6 on Eclipse IDE.

ADVISER'S APPROVAL: Dr. JOHNSON P THOMAS