LIGHT WEIGHT HEALTH APPLICATION FOR LOW END CELL PHONES

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Peyman Emamian

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

November 2016

Fargo, North Dakota

NORTH DAKOTA STATE UNIVERSITY

Graduate School

Title

LIGHT WEIGHT HEALTH APPLICATION FOR LOW END CELL PHONES

By

Peyman Emamian

The supervisory committee certifies that this thesis complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Prof. Juan Li
Chair

Prof. Jun Kong

Prof. Sudarshan K. Srinivasan

Approved:

7 December 2016
Date

Prof. Brian Slator
Department Chair

ABSTRACT

Health applications are usually complicated and low end devices do not benefit from them. The focus of this thesis is on expandable health services platform for low end cell phones. Large number of mobile phones in the world are incapable using web or modern operating systems and pre-installed SMS application is the preferred communication medium. Moreover, SMS does not require a stable connection so text-based health information can still be available even during natural disasters. Although our platform is accessible through communication forms other than SMS.

We propose a scalable platform for light weight health applications, providing novel and proactive client communication. Using cloud we assure the scalability, elasticity and reliability of the server side. Our multi-layered architecture provides separation of concerns and decoupling of communication and business logic. Furthermore, plug-ins can expand and customize functionalities.

ACKNOWLEDGEMENTS

I would first like to express my sincere appreciation to my thesis advisor, Professor Juan Li, of the Computer Science Department at North Dakota State University, for her great advice in every step of the way and for her patience and continuous encouragements. I could not have been at this point today without her honest and generous support.

I am grateful to my supervisory committee members, Dr. Jun Kong and Dr. Sudarshan Srinivasan, for their guidance and feedback.

In addition, I thank the faculty, staff and students of the Computer Science Department at North Dakota State University for providing such an excellent and friendly environment to learn and progress.

Finally, I express my gratitude to my parents and to my wife, Aida, for providing me with unfailing support throughout my years of study. This accomplishment would not have been possible without them.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF FIGURES vii

1. INTRODUCTION 1

1.1. Motivation 3

1.2. Thesis Contributions 5

1.3. Thesis Organization 6

2. BACKGROUND AND RELATED WORK 7

2.1. Background 7

2.1.1. Mobile Health 7

2.1.2. Text Messaging for Mobile Health Applications 9

2.1.3. Cloud Computing 11

2.1.4. Infrastructure as a Service (IaaS) 12

2.1.5. Platform as a Service (PaaS) 12

2.1.6. Software as a Service (SaaS) 13

2.1.7. Amazon Web Services 13

2.2. Related Work 14

3. PROPOSED APPROACH 17

3.1. Features of the Proposed Approach 19

3.1.1. Multi-layered Architecture 19

3.1.2. Plug-in-able Feature Development 19

3.1.3. In the Cloud 19

3.1.4. Scalability 19

3.1.5. Expanding to Other Forms of Communication 20

3.2. High Level System Overview 20
3.3. Resource Allocation for the Web Application 21
3.4. Applications 22
3.4.1. Interaction with Softwares 22
3.4.2. Input Data Gathering 23
3.4.3. Proactive Outbound Communication and Outreach 23
3.4.4. Statistical Analysis and Reporting 24
3.4.5. Machine to Machine Communication (Internet of Things) 25
4. SYSTEM DESIGN 26
4.1. System Architecture 26
4.1.1. Communications Layer 27
4.1.2. Dispatch Layer 28
4.1.3. Plug-ins Layer 28
4.2. Flow of Information 29
4.2.1. System Implementation 31
5. EVALUATION 34
5.1. Scalability 34
5.2. Performance and Responsiveness 35
5.3. Load Testing 35
5.4. Elasticity 38
6. CONCLUSIONS AND FUTURE WORK 40
REFERENCES 42

LIST OF FIGURES

Figure Page

2.1. Wireless Body Area Network (BAN). Adapted from [16]. 8

2.2. Cloud Computing Service Levels. Adapted from [9] 12

2.3. Users and Providers of Cloud Computing. Adapted from [5] 13

3.1. High Level System Overview 21

3.2. Drug Information Plug-in 21

4.1. System Architecture 26

4.2. Sequence Diagram 30

4.3. Example Flow of Information 32

5.1. Load Test - Histogram of Response Latency 36

5.2. Load Test - Predefine Number of Requests for Load Testing 37

5.3. Load Test Traffic - Number of Successful and Failed Transactions 38

1. INTRODUCTION

Mobile phones are ubiquitous. In the last few decades, cell phones have changed human's daily life in many ways. Mobile phones have made personal and group communications much easier. They have given the users the freedom to move around while staying connected all the time. This unique capability empowered a lot of important applications. Not only cell phones have kept individuals connected to each other, but also they have played a significant role in conducting and managing business and facing emergency situations such as natural disasters.

Since the emergence of the first commercial cell phone in the market in 1983, mobile phone adoption rate has been growing with an increasingly fast pace. According to The International Telecommunication Union (ITU) estimation in February of 2013, there are 6.8 billion mobile subscriptions worldwide. This means the penetration rate in the global population is about 96 percent [12]. (Note that, these numbers do not show the count of subscribed users because one person may have multiple subscriptions.) These statistics clearly show the huge adoption rate of mobile phones in public and the significant role they can play in today's society.

In the past decades, as the popularity of mobile phones was growing, mobile phones have become more sophisticated. While the very first generation of mobile phones were only capable of telephony, newer generations began to support more services such as text messaging, email, Internet access, short range wireless communications (e.g. Infrared, Bluetooth), photography, and gaming. The newest generation of advanced mobile phones which are called smartphones, run on a mobile operating system (e.g. Android or iOS) and have a lot of computing power to support a variety of complicated applications such as voice recognition, image processing, and navigation among others.

Although smartphones have attracted a lot of interest among public and have a fast growing adoption rate, they are not affordable by a large portion of the society. Moreover, most of the applications on a smartphone require the user to have Internet connection which needs an extra monthly payment for a data plan to the service provider. In contrast, a basic mobile phone which is only capable of voice calling and text messaging is generally very cheap. For example, in the

United States, a basic mobile phone device comes almost free of charge when you subscribe for a basic phone plan.

Since basic cell phones with basic functionalities are the most affordable and the most penetrated communication medium among the public, they have been the most important communication technology for a lot of modern applications. There has been a lot of interest in the research and industry communities to develop light-weight applications for basic mobile phones with limited processing power and with no Internet access. Health care applications enabled by cell phones are one of the most important applications that have been studied and developed during the last decade. The term mobile-health (or m-health) has been introduced to indicate the emergence of a new field in e-health (electronic health) [15].

M-health which is defined as the use of mobile technologies to support public health, has been the topic of a great number of studies in the last few years [15, 15, 23, 16]. Moreover, text messaging (also called short messaging service: SMS) has attracted a lot of attention as the main enabler for a vast majority of mobile health care application [24, 1, 20, 19, 25]. We will describe several examples of existing m-health applications that use SMS as the primary means of communication in chapter 2. For instance, these applications use SMS to remind patients of their appointments or their medicine intake time, to communicate with users to give them useful information about a medicine and its side effects, to inform the individuals about a disease epidemic in the area, to remind them to eat healthy or exercise, and to communicate many other useful health-related messages.

The reason for the importance of text messaging in mobile health care applications is manifold. Text messaging has been shown to be a very popular means of communication among people (especially among teenagers); SMS is very convenient and it fits into the daily routines of the individuals (both patients and clinical practitioners). Short messages are easier to digest and to remember particularly for elderly. Since users receive SMS in their personal devices and can receive and send them privately, they feel comfortable to use SMS-based health applications. Finally, as mentioned earlier, it is cheap to send and receive an SMS on a basic phone and there is no need to have Internet access for text messaging.

1.1. Motivation

Everyone agrees that health is an important issue and taking care of our bodies should be a priority in our lives. In the recent centuries, humans have accomplished to know their bodies better and to take better care of their bodies, cure or prevent diseases, and counter epidemics and eventually humans live longer and healthier lives. In the age of computers, Internet and satellites, it is easier to access the health related data, but sometimes the user is confronted with too much data. You should have enough search skills to find the name of the symptoms you have, or pay a professional to do that for you.

Even with simple data, the user is facing the ultimate challenge of finding meaningful information within the huge bulk of data. For example, what those increasing or decreasing curves mean in real life and should he/she be worried about them. Lastly, the decision making on what would be the next steps is a challenge. Here are a few examples of the health related information that people would like to have: How should we act to prevent a sickness, improve hygiene, stay healthy or treat the disease? How to find the right medicine? Or should we just relax and rest and trust our immune system.

Having too little data is always a problem, but this problem can be solved by paying more attention, recording activities and having more discipline in taking care of ourselves. However, a more important problem to solve is how the data can be converted to information and knowledge and become useful. Everybody can look at a rising curve showing the blood pressure, but it takes a doctor and years of experience to understand it correctly and make the right decision. There is always a gap, between the data and knowledge.

Given this mind-set, this project is trying to provide the first steps to bridging this gap, and to lay the foundation for better software that helps improving health care. It requires gathering, processing, clustering and dicing data and acting upon the results using human knowledge on health care and computer science. The most important factor for software that tries to solve this problem is scalability and expandability. The core processing should happen in the cloud to be as powerful as possible and easily update-able. On the other hand, client should be as simple as it can be, to support the widest range of devices and to preserve power and energy.

The challenge here is that health applications are usually complicated and this fact hinders the users with low end devices from benefiting from these applications. It is complicated to develop a health application that works on all mobile platforms. For instance not all devices have enough amount of memory or processing power to support browsers or user programs. There is a need for a light weight application to track, organize and analyze health related data.

The focus of this work is on low end devices that are incapable of interacting with rich application interfaces for example not all devices can communicate with web applications or applications that need installation on the device. Although speed and processing power of modern mobile devices improve in a fast pace, a large number of people still use low end devices with limited processing power that are incapable of complicated functionalities like web browsing. These devices are especially common among older people who do not keep up with technology updates as fast as younger individuals. In addition, in most of the developing countries still the vast majority of people use basic mobile phones rather than smartphones. Even if users have enough processing power on high end devices, sometimes it is preferred to preserve power to prolong usage cycle of the device without recharging. In addition the owner may prefer not to use expensive Internet connection if he/she can enable a mobile application with a cheaper communication method like SMS.

An SMS-based health application makes it possible for a wider range of lower end devices to have access to health related information, gather data and analyze it without the need to install an application or browse web. In order to have a light weight health application, a simple and clear communication and data structure is required. Compared to web application with rich user interface, text-based applications use less network bandwidth. By reducing the complexity of the application, we reduce the cost of the software and ultimately decrease the health care cost.

An SMS-based application can perform as a drug and health-related information repository. Users can send keywords related to medical information to get a concise description about the topic. For example, information about a drug and its side effects or first aid information about a medical situation (e.g. broken leg). In rural areas where it is hard to reach out to health care providers, access to useful health information can be life saving. Internet connection is usually unstable or unavailable in rural areas. Therefore, an SMS-based health application becomes very useful in such locations. In case of unstable signal reception, an SMS can be cached on the device and

sent whenever the cell coverage is available. For example a hiker in a rural area who tries to find information about snake bite treatment can use such an SMS-based application more conveniently.

With an SMS-based health application, we can benefit from proactive communications. Because of the simplicity of the client and the removal of the installation phase, new opportunities show themselves to communicate with client in reverse direction of usual data transfer. While usually there is a request from client and an answer from application, the health application can communicate with potential clients in a proactive way. For example in special situations like a natural disaster or disease epidemic outbreaks, an SMS-based application can send out appropriate health-related messages to the public. For instance, location of shelters and medical stations in case of natural disasters (such as storms, earthquakes) or preventive actions in case of epidemic disease outbreaks. Also, this kind of applications can be used for periodic and automated notices to users. A few examples are medicine intake reminders, healthy diet and lifestyle motivational messages, recurrent health check-up reminders (such as dental and eye care visit reminders).

1.2. Thesis Contributions

In this thesis, we propose a light-weight, scalable and expandable SMS-based health application framework. Below we briefly introduce the main contributions of our work:

- Our proposed multi-layer architecture facilitates the separation of concerns in levels, therefore system is scalable in each layer.
- Our proposed system is plug-in-able and it assures that any additional feature can be added to the system as a plug-in, with minimum impact on the running system and this can happen without the need to stop or update the system.
- We use cloud computing to enhance the availability, scalability and ease of use of the software that serves the clients.
- The combination of the three key benefits of this design, layered architecture, plug-in-able approach for feature development and running the application in the cloud, makes the application easy to scale without any theoretical limit. Moreover, the proposed approach offers high availability and failure-tolerance.

- Our proposed framework is easily expandable. Any other type of communication other than SMS gateway can easily be integrated to communicate with the system (e.g. twitter).

1.3. Thesis Organization

The rest of this thesis is organized as follows: In Chapter 2, we present the background information and review the relate work. In chapter 3, our proposed approach, its benefits and applications are described. Chapter 4 characterizes our proposed multi-layer architecture and the implementation challenges. In chapter 5, we analyze the performance of the proposed system. Chapter 6 concludes the thesis and points out the potential future research directions.

2. BACKGROUND AND RELATED WORK

In this chapter, we present our background study and literature review on the topic of light-weight mobile health softwares and use of text messaging in these applications.

2.1. Background

2.1.1. Mobile Health

The rapid growth of mobile communications on one hand and pervasive and wearable computing on the other hand have greatly influenced medical care systems in the recent years. Mobile health (or m-Health) is defined as the use of mobile computing, wireless communication technologies and medical sensors for health care [15]. The concept of m-Health which was first introduced under a different name, "Unwired e-med", in 2000 [15] is an evolved subset of traditional desktop "telemedicine" which offers the use of wireless and portable technologies.

Since the first introduction of m-Health, wireless technology and pervasive computing have seen significant advances which result in better opportunities in health-care delivery applications. For example data rates of wireless connections is increasing in every new generation of wireless standards and the medical sensors (in both implantable and wearable forms) are smaller, consume lower power, and offer better performance. These advances empower new health related applications that have not been possible in the past and accordingly will have a great impact on reshaping the traditional medical care services in the near future.

For example with today's wireless technology, the medical record of a patient can be accessed by health care professionals from virtually anywhere with a wireless connection to the medical database. Also, handheld wireless devices can be used for home health care purposes for instance through blood sugar monitoring for patients with diabetes. Emergency calls or text messages that warn a community about a disease epidemic or alert thousands or millions of people about an impending storm or tsunami are some other important m-health applications.

Another important m-Health application that has been stimulated by the great advances in wireless and medical sensor technologies is the wireless body area network (BAN). A BAN consists of a network of sensor nodes that measure, process, and communicate different vital signs from the

body (such as heart rate, blood pressure, body temperature, etc.) and ambient parameters (such as location, temperature, humidity, etc.) from the surrounding [23].

When a person uses a BAN, he/she is being continuously monitored by the sensor network on his/her body. The BAN collects and processes physiological data from different parts of the body and transmits this data to the user's mobile phone in his/her pocket. The mobile phone then can periodically communicate this information to a medical center server or a health professional mobile device. Figure 2.1 illustrates a typical body area network system. In this BAN the sensors' data are first aggregated in a wireless personal server which is connected to the user's mobile phone via a wireless link such as Bluetooth.

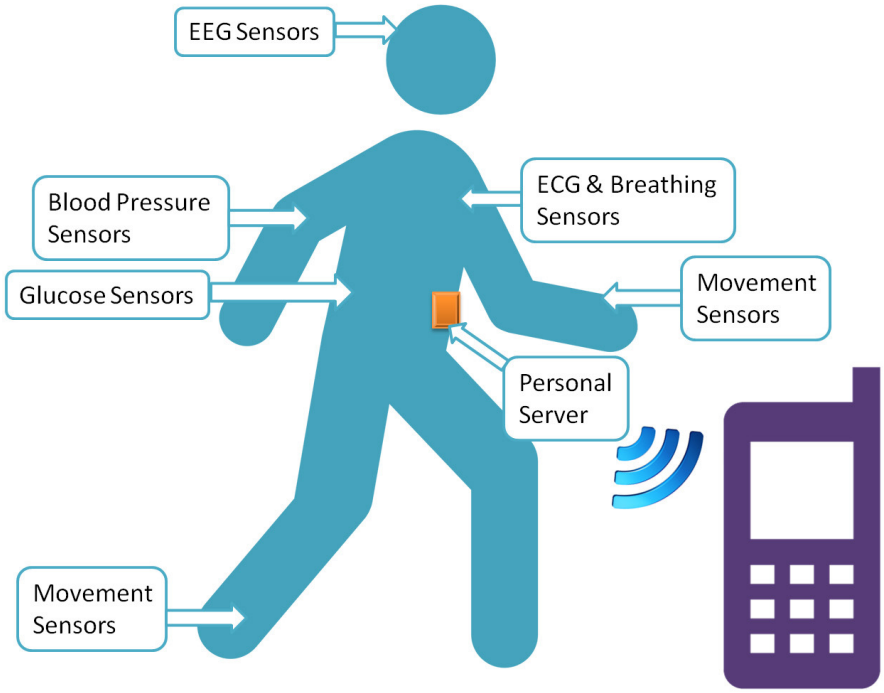

Figure 2.1. Wireless Body Area Network (BAN). Adapted from [16].

In case of a critical condition a BAN can be life-saving because the health practitioner will be alerted of the patient's situation in a timely manner. Such integrated medical sensor networks enable the physicians to diagnose and treat complicated medical conditions remotely and accordingly reduce the cost of health care. In addition, with these technologies the health professionals can treat a larger number of patients in the same amount of time. Therefore, with

emerging technologies like body area networks and wearable computing, m-health reshapes the traditional ways of practicing health care.

M-Health is a promising approach that facilitates the fight against chronic diseases and also communicable diseases [18]. The confrontation with these two categories of health problems is particularly important in developing countries and also rural communities in developed countries. Generally, the communities who are less educated about healthy lifestyle or the communities who do not have an easy access to health professionals face steady growth in chronic and communicable diseases.

Some of the well-known preventable causes of chronic diseases world-wide are poor diet, low physical activity, and smoking. These unhealthy behavior may result in obesity, heart and lung diseases, diabetes, cancer etc. The chronic diseases commonly need early and extensive health interventions and m-Health has proved to be an effective approach regarding interventions. As an example an m-health intervention may include disseminating incentive and informative text messages about health facts to help individuals or communities to have healthier lifestyles and to educate them about self-care [18].

2.1.2. Text Messaging for Mobile Health Applications

The ubiquity of mobile phones empowers m-Health to offer a number of unique opportunities in large geographical areas. Text messaging is one of the most important social networking tools in m-Health [18]. In the previous section, we mentioned a few examples of m-Health applications that are enabled by SMS such as health care intervention with text messaging and dissemination of SMS to the public in the emergency situations like natural disasters.

Cell phones are vastly applied as common clients for many distributed, database-centric health care applications especially in rural areas where there is lack of physicians [19]. Because of the extensive availability of cellular services, mobile phones have become a key element in making health care applications possible in rural areas where the health workers use cell phones to bridge between patients and doctors. Unlike the large cities, in the rural areas the data rates are extremely limited and data connections are expensive, therefore a distributed client/server system is too complicated and heavy to be used in these environments [19]. Accordingly, most of the times only light-weight health applications that use basic communication like SMS are feasible in rural health applications.

Text messaging is a perfect fit for many of m-health applications because of the following reasons: 1) Mobile phones are ubiquitous. They have been adopted by the vast majority of the people. 2) Text messaging is affordable. Every basic cell phone supports SMS and the subscriber does not require connectivity to the Internet in order to send and receive SMS. 3) Text messaging is convenient. It fits into the daily routines of patients and clinical practitioners. 4) Text messaging is very popular, especially among teenagers. 5) Short messages are easier to digest and remember particularly for elderly.

At the community level, people can use social networking via text messaging to exchange information about the local health system and their experiences about how to access their needed health resources. Social networking via SMS enables the members of a community to effectively connect to each other and help them to identify the existing resources and discuss about the quality and costs of different local health providers. These discussions may encourage the local providers to improve their services and to decrease their prices. With social networking, community members can support each other on preventive behavior such as quitting cigarettes or working out. In the same way, patients can support each other in managing chronic diseases such as diabetes or asthma [18].

Text messaging is also an effective and affordable medium for patient-provider interactions. Good communication between patient and practitioner is a key factor in a successful treatment. Text messaging enables patients to ask questions, get timely responses, send important personal health data, and receive guidance and health facts (e.g. information about drugs and their side effects, information on their next appointment, etc.) without the need to travel to medical clinics and this saves a lot of time and cost for both patient and health practitioner [18].

Text messaging can be used in health application in different ways. The following categories are considered for text messaging techniques for health care [13]:

1) Sending information to the users (e. g. educating people with a health fact, notification of polluted weather or epidemics, and reminding people to work out or encouraging them to have healthy and nutritious food). The techniques in this first class are commonly used in technology-based behavioral intervention projects that have attracted a lot of interest in recent years [24, 7, 21, 17, 11, 26]. We will look at several examples of these projects in Section 2.2.

2) Gathering information from users (e. g. collecting data from people about their flu symptoms in a short survey). This class of techniques can reveal the health patterns for individuals and also for large populations.

3) User questions and expert response (where the response can come from a database or from a real person). Unlike the last two classes, in which the information flows in one direction, this category introduces a two-way interactive SMS technique.

Our proposed platform in this thesis can support all of the above three categories of SMS-based communications.

2.1.3. Cloud Computing

In our computerized world, nowadays, computing has become similar to one of the traditional utilities such as electricity, gas, and water. With this vision, users must be able to access computing services based on their requirements and without knowing or worrying about where the services are hosted or how they are delivered. Cloud computing is a scalable distributed computing paradigm that makes this vision possible [8].

The concept of cloud computing involves a large number of computers that are connected through a network (such as Internet) and run applications that are offloaded to them by external customers. The pool of the computing power is abstracted, virtualized, dynamically-managed and scalable. Cloud computing also delivers on-demand storage, platforms, and services to the users [14].

Since hardware and software maintenance is outsourced to the cloud provider, delivering software as a service in the cloud can reduce IT operational costs. Furthermore, cloud computing allows the software developers/providers to decide on the extent of their resource needs on a short-term basis. A software company can start small and expand its computing resources only when there is a demand to do so. As a result, there will be no waste in investment due to overestimating the required computing infrastructure. On the other hand, using cloud computing, a software provider can decrease the amount of allocated hardware resources if the resources are no longer used and accordingly some of the costs can be cut [5].

Cloud computing refers to both applications that are delivered to users as a service and the hardware and system software in the data center (cloud) where these applications run [5]. Cloud

computing services can be viewed in three different levels. Figure 2.2 illustrates cloud computing service levels. Next we briefly describe each of these three service levels.

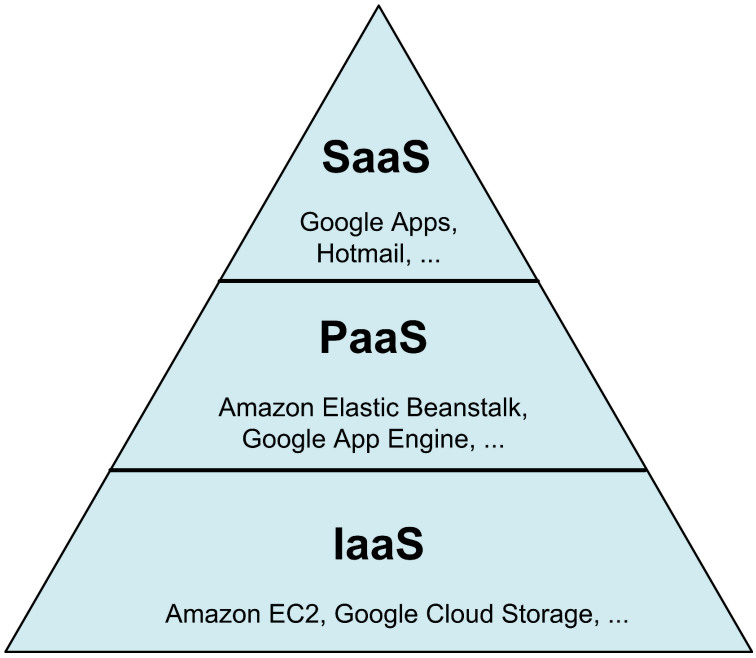

Figure 2.2. Cloud Computing Service Levels. Adapted from [9]

2.1.4. Infrastructure as a Service (IaaS)

IaaS is the most basic (lowest level) cloud service where provider lends physical or virtual computers and other resources such as storage, firewalls, IP addresses and virtual local area networks (VLAN). The on-demand services can scale up and down based on customer's requirements. An IaaS provider delivers the resources from its pool of computers in its data center and bill the services based on the amount of allocated/consumed resources. The cloud users in IaaS service level must install and maintain the operating system and software patches on their own as these are not provided by the IaaS provider. Furthermore, the user is responsible for all upgrades, security fixes, and licensing issues [9].

2.1.5. Platform as a Service (PaaS)

PaaS is the middle level cloud service, in which cloud providers offer a computing platform to users. The platform typically consists of operating system and execution environment. In this level, providers take the responsibility of maintaining the infrastructure and deliver a ready-to-

build/host platform to the application developers. Therefore the developer does not need to worry about the middle-ware, patches, etc. like in IaaS [9]. Moreover, with a pay-as-you-go pricing model, the developers cut the cost of the underlying hardware needed for their applications. Some of the PaaS providers also offer automatic scaling of resources based on the application demand; thus, the developer does not need to adjust the resource allocation manually.

2.1.6. Software as a Service (SaaS)

SaaS is the highest level cloud service. Software applications that run on the cloud platforms and are accessible only through the cloud are referred to as Software as a Service (SaaS). Cloud providers install and maintain the software applications in the cloud and SaaS users access and use the software through a cloud client. Since the users do not need to install and run the software on their own computers, the software maintenance and support is simpler and transparent to the user.

Figure 2.3 illustrates the relations between the providers and users in the cloud. As it is shown in the picture, the cloud user (PaaS user) can be SaaS provider.

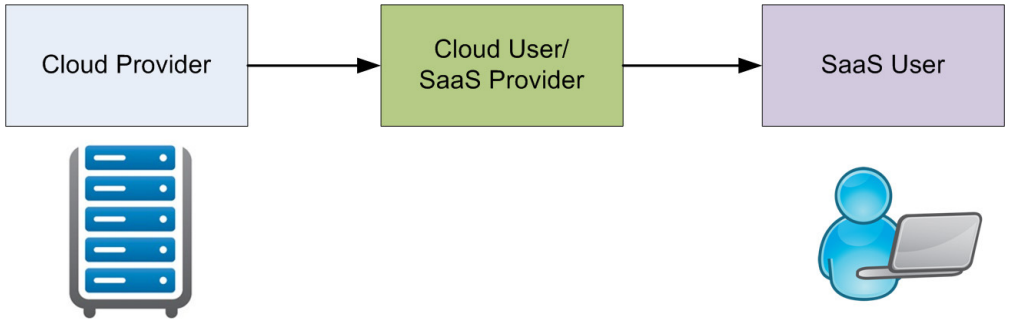

Figure 2.3. Users and Providers of Cloud Computing. Adapted from [5]

2.1.7. Amazon Web Services

Amazon Web Services (AWS) is a collection of tools and services provided online by Amazon.com Inc. to help build scalable online applications. All of the services are accessible through Internet. Amazon provides these services as IaaS and PaaS. Application designers and developers can use building blocks provided to them to make applications based on their own need and requirement.

AWS removes the upfront cost of application development and makes it possible for the companies to pay as they use the service. This eliminates the capital investment for server farming

in medium to large applications and provides the opportunity of scaling up without any hassle or problem for small companies as they grow.

2.2. Related Work

ALIVE (A Lifestyle Intervention via Email) project [6, 22] has shown that simple email reminders can have a significant impact on improving diet and physical activity of individuals. The reported results show a significant increase in physical activity (e.g. walking per week) and consumption of fruit/vegetables.

Delaware Physicians Care, Inc. (DPCI), a Medicaid program, has used text messaging to remind patients with diabetes about their scheduled blood test appointments [1]. After six months the percentage of patients receiving the necessary test was increased by 18 percent. In another project, DPCI uses text messaging to remind pregnant moms of their prenatal and postnatal appointments as well as to provide them with educational information.

Another example of health-related text messaging tool is presented in [3] that has been designed to help educating the youth in the San Francisco area about sexual health. Users can send a simple text message to get information about what to do after unprotected sex or they can get guidelines and information about sexually transmitted infections, including HIV.

UbiFit Garden [10] is another example. It is technology-based health-related behavioral change system which uses on-body sensors and a mobile application user interface to encourage regular physical activity. This system is more sophisticated than a simple text messaging-based system like the above examples and therefore needs more complicated hardware and software components.

An m-health system has been proposed in [20] to help health workers in rural areas of India to do their jobs more efficiently and to have more effective interactions with doctors. The health workers collect the symptoms of the patients; then they use their cell phones to send the symptoms to a remote server where they are stored; The doctors can asynchronously access the server, review each patient's record, ask more questions about the patient, and finally do the diagnosis and initiate the treatment. The server then sends the prescription back to the health worker via SMS.

In the case of this application, the solution consists of tasks that are done by both computer and human. The computer system has a client-server architecture. The XML requests are generated by the Android client and sent to the server via network. On the other side, when the data is received

via HTTP request, the server creates a reply in XML format and sends it back to the client via HTTP response.

As authors stated in their research, their system consist of Mobile client, Server and Database. This separation creates asynchronous layers of access to application that makes it possible for field agent to work with Mobile client at the same time doctors use Server and Database layer to analyze and diagnose, furthermore showing the importance of separation of concerns and layered architecture.

Reference [19] proposes ELMR (Efficient Light-weight Mobile Records) system that offers a light weight database access protocol for accessing and updating records from remote cell phones. The motivation of the design is the implementation of a medical record system that can work on a mobile device where the only data connectivity is via SMS. This is the case in rural areas where the Internet connection on cell phones is scarce and very expensive.

The introduced light weight database access protocol for health care applications is optimized and simplified to be applied under extreme bandwidth constrained SMS service (an SMS packet is confined in 140 bytes only.) The authors have mentioned the application of their proposed system in health care delivery in AIDS care centers in Ghana and South Africa where health workers need to frequently access health databases using low end devices [19]

Reference [24] surveys on several text messaging based intervention systems aimed at HIV prevention, and encouraging healthier sex behavior. In addition, the article introduces several examples of applications for health interventions for diabetes patients, pregnant women, etc.

Reference [4] explains a sensor-based heart monitoring system in which in the case of irregularity in user's heart rate (for example when the heart rate goes above a certain threshold), an SMS is sent to the user and/or the user's doctor or relatives. Similarly, [25] proposes a method and architecture for a remote health monitoring system that monitors several vital signs of the patient (such as oxygen percentage in blood, heart rate, and temperature) and if any of these parameters are not in the predefined range, an SMS will be sent to the user's doctor/emergency number. The SMS will contain both the values of vital signs and the location of the patient that is extracted from his/her cell phone's GPS (Geographical Positioning System).

Text messaging has shown to be a successful health intervention enabler for smoking cessation programs [7, 21], depression treatment [17], obesity prevention, alcohol recovery [11], and asthma treatment and education [26] among others.

Clearly, it is advantageous to have a multi-layer architecture since there is a need to asynchronously process, store and retrieve data. Looking at the research presented in [20], [4], and the proposed architecture in [25], It is clear that this layered separation benefits the system in the form of increased stability and ease of further development.

3. PROPOSED APPROACH

This section describes the problem at hand and the possible solutions to encounter the problem. We explain what is the purpose of our introduced application and why we have chosen our approach over other possibilities to solve the problem. In addition, we illustrate the features and functionalities of the proposed solution.

This project is trying to lay the foundation for better and easier service development in the area of healthcare. Most important factor that we try to solve is to be scalable and easily expandable. Using cloud computing, the major processing work must be done in the cloud to be as powerful as possible and so that the service can be easily updated and scaled to our needs. On the client side, in order to support the largest range of potential users, it should be able to communicate with a wide range of communication strategies, most importantly the forms of communication that are available to everybody. Moreover, setting up and using the client must be easy and attractive to the customers and they should be able to use it intuitively. Short Message Service (SMS) or texting provides the best solution for a simple and intuitive client (Although the system should be capable of communication in other forms such as web, web service call, etc.)

As one of the functionalities of the desired framework, we need to have a light weight application to track, organize and analyze health related data. The focus of this work is on low end devices that are incapable of interacting with rich application interfaces. For example, not all devices can communicate with web applications or applications that need installation on the device. Also, not all devices have enough amount of memory or processing power to support browsers or user programs. On the other hand, the client that is producing the data could be an entity other than human, for example a sensor or a web service that is inputting data to our system.

Our approach is to benefit from cloud computing to implement complex health applications in the cloud. Thus, the client application becomes simple and light weight and it consumes less power. With this approach, we can benefit from complicated health applications on low end devices. Therefore SMS or any other form of simple communication can be exploited to interact with the system. Note that more complicated devices can use Internet and rich application interface to communicate with the software in the cloud as well.

The benefit of having the software on the cloud is that we can easily update the software without involving the client in the update process. The program is scalable because no matter how many users use it, we can easily increase the throughput by adding instances to respond to the inputs and we can provide a seamless experience for all of the users. The plug-in-able approach in the cloud makes it easy to add new features or functionalities to the program without having to change the core functionalities. This also makes the introduction and propagation of errors harder and less frequent throughout the system and therefore the system core functionalities are not misused. Furthermore, since the data is stored and replicated in the cloud, in case of a failure in the client, all of the user's data remains usable and is not affected by the client failures.

Another benefit of using a thin client and having the software in the cloud is that we remove the installation process. All of the applications are running on the cloud, and the client only needs to use SMS service to make use of the applications. Since all of the cell phone devices support SMS service, users can instantly benefit from applications without installation (e.g. getting medicine information through SMS). Therefore, a large number of applications will be available on the fingertips of the client without any installation hassle or payment for installation. Also, we can benefit from proactive communication with the users. Because of the simplicity of the client and removing the installation phase, new opportunities become available for communicating to the client in the reverse direction of usual data transfer (i.e. request from client and response from application.) Instead, the health applications can proactively communicate to the clients or the potential clients, for example periodic health related updates or emergency notifications of epidemic diseases can be distributed to the users.

Our scalability approach consists of a combination of reactive and proactive response to the increase in the number of requests. Reactive response in the communication layer is to instantiate more instances of responders. The proactive part involves having a few instances warmed up and ready to answer the extra input immediately. Since starting up a new instance (in any layer of the application) is going to cost more time, this approach ensures a consistent response time for excess traffic, while our load balancer instantiates new working machines to respond to future incoming traffic.

3.1. Features of the Proposed Approach

3.1.1. Multi-layered Architecture

Layered architecture that results in separation of concerns and better scalability of the whole system makes it possible to break the system into modules that communicate with each other while having minimum coherence with each other. Communication between components is by sending and receiving messages; therefore, scaling up or down one component is hidden from the other components and does not affect any other module in the system.

3.1.2. Plug-in-able Feature Development

It is obvious that more and more features will be added to the system as it matures and we cannot get the system right from the beginning with all the features. Making the system plug-in-able assures that any additional feature can be added as a plug-in with minimum impact on the running system and this can happen without the need to update the system. It is clear that the core functionalities of the system will most likely remain almost the same during its lifetime (e.g. saving input data to file or communicating with SMS gateways) or at least these functionalities get updated less frequently. On the other hand, the features of the application will change more frequently over time as we, the system designers, figure out new requirements or optimize the current features. For instance, the reporting interface of the application may be updated regularly.

3.1.3. In the Cloud

Although the concepts of client-server application, its usage in business software development, and the server scalability have been around for a while, but with the recent advancements in cloud computing over the last few years, the availability and ease of use have been improved drastically for application development in the cloud. As a result, with a little effort, any computer scientist and software engineer can design, implement and deploy scalable applications without the need for extensive funding or capital investment at the beginning.

3.1.4. Scalability

The combination of three key benefits of this design, layered architecture, plug-in-able approach to feature development, and running the application in the cloud, facilitates efficient scaling of the application with no theoretical bound. The components of the system can be instantiated as much as needed to work in parallel and to handle any increase in input traffic. Load balancing

between layers of the system architecture, can make sure that we are using resources efficiently and we free unused resources and acquire more resources when needed.

3.1.5. Expanding to Other Forms of Communication

The original idea of this thesis is to empower the health-care applications by exploiting SMS communication. But given that we have a communication layer that converts the input messages to a data structure that is understandable by the system, we can essentially extend the input to any means of communication. Therefore, any other type of communication other than SMS gateway can work within the proposed system as the messages are translated to our internal data structure by the communication layer. For example, Twitter has a lot of similarities with SMS and it is accessible through APIs for programmers over the Internet, or for this matter any web-service that can send messages to the system can replace SMS communication. As long as there is an adapter unit inside the communication layer that can convert the input to an understandable form by the system, any device with any communication format can communicate with the system.

3.2. High Level System Overview

In our proposed approach, the application runs in the cloud and the devices communicate by sending messages through SMS gateways. The application running in the cloud processes the input and if necessary will send a response to the device. The application in the cloud consists of core functionalities, communication and plug-ins that communicate with the core. Plug-ins are also in the cloud so they can be easily scaled up/down based on the throughput. Figure 3.1 shows the high level system overview.

We have designed a generic expandable architecture for SMS-based light weight applications. One example application that we have developed based on our generic plug-in-able framework is drug information system. The client sends an SMS containing the name of the medicine in a specific format. The SMS goes through the gateway and reaches the core of the software. Supposing the drug information plug-in is registered to the core of the software, based on the format of the SMS, the core decides to pass the input to the drug information plug-in. The plug-in then searches in its database for the drug name and responds with a brief description of the drug. The core passes the response to the gateway to be sent back to the user. Figure 3.2 shows a snapshot of the functionality of drug information plug-in that we have implemented and deployed.

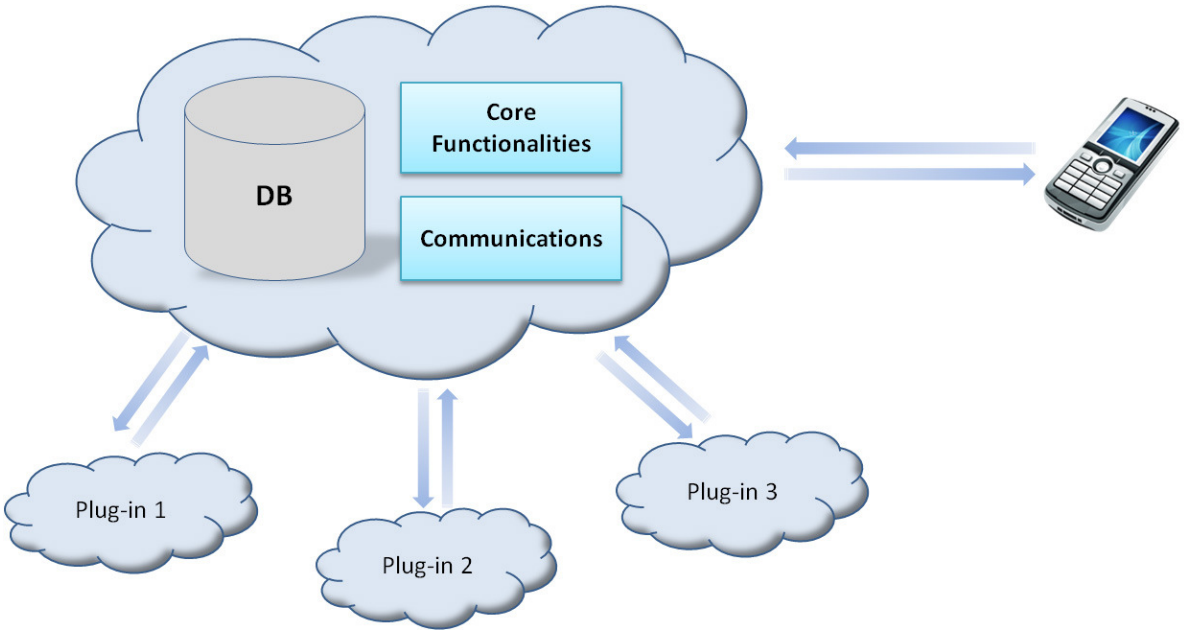

Figure 3.1. High Level System Overview

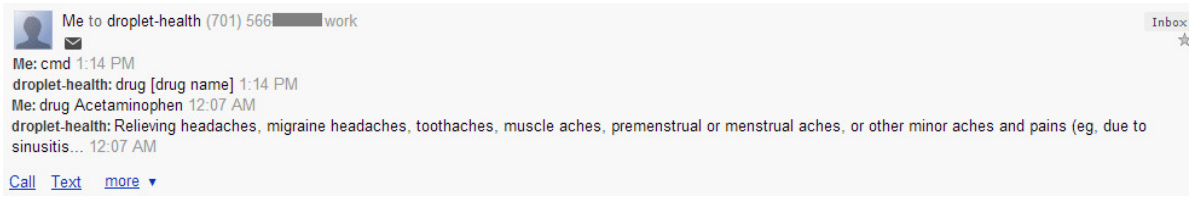

Figure 3.2. Drug Information Plug-in

3.3. Resource Allocation for the Web Application

There are multiple approaches to handle the input messages to the system. One approach is to handle and input message with one process (thread) from the beginning to the end of its lifetime. This is easier to implement and might process input requests faster when the number of request per unit of time is small, but does not guarantee scalability and effective use of memory. The number of processes equals the number of input requests and each additional machine needs to have all of the functionalities to be able to respond to any request. Therefore, using this approach, we have to load all of the features in all of the machines and keep them available in the memory (or virtual memory), so that the system can process any request that might need those features. As we add functionalities to the system, the features of the system need more memory than a normal server can handle. Adding memory and using better machines cannot solve the problem indefinitely as we will reach a point where we cannot load the server application in any machine and we cannot

add any machine. Therefore, this approach is not scalable as each machine should have all of the features of the system, and the machine resources are always limited.

Another approach which is more favorable is to break the process of responding to the requests into multiple layers. Each layer works in connection with the other layers but internal processing is abstract and independent from the other layers. Communication between the layers (and most of the time between the components of a layer) is performed by sending messages. These messages may contain any functionality like triggering a computational action or a request for a database transaction.

One obvious advantage of this approach is that changes can be made in one layer without affecting the other layers. We can replace the whole business logic of a layer, without even slightly changing the other layers. This also limits the area under effect of a bug and helps to find and fix the bugs faster.

Another advantage is that each layer or each component of a layer is specialized in doing a particular task; therefore, it can run separately on a separate machine and the number of machines can be increased as we get more requests for that functionality. As far as the other layers or components are concerned, they ask for the functionality through a message and their request is passed to one of the machines responsible for that task. Moreover, since we assign a machine (or computational unit) to a special work, responses to the frequent computing requests can be cached, so both time and computational power will be saved in responding to repetitive requests.

Each layer is independent, so it can grow or shrink in size as needed to meet the required quality of service criteria such as response time. The other layers always get the same quality of service form this layer regardless of the system load and throughput.

3.4. Applications

3.4.1. Interaction with Softwares

The main functionality of our software is the two-way interaction with the system. The user sends an input to the system, either asking a question or requesting invocation of a functionality, and the system sends the response back to the user. For example, the user can send a text message containing “drug Acetaminophen” and the system replies back with a brief piece of information about the drug and its side effects.

Interaction with the system can involve more sophisticated functionalities than just looking up data. For instance, the response may require geographical information extraction. A user may send "Hospitals 58102" and the system will return with the names and addresses of hospitals in the area of that zipcode.

Sometimes instead of responding to the user with a message, the system triggers an action. For example, in a scenario where the signals received from a user's heart rate monitor indicate a dangerous situation, the application sends a notification to the emergency and medical services along with the user's location (e.g. based on his profile or using his phone GPS). Another example of interacting with the system is extracting useful information about the previously stored data in the system. For instance, the user asks for the amount of calories burnt during the past month.

3.4.2. Input Data Gathering

One of the possible applications of the system is storing data that is extracted from either the users' input or the messages that are sent from a device (e.g. a sensor). For example, a person inputs his/her weight (e.g. weight=200) and the data will be added to the history for that user-parameter combination. In order to use this application, the properties must be defined in the system so that any input can be translated to a property or a special command must be sent from the users to enable the property (e.g. add property weight 200).

Other examples of getting input is user reporting an unhealthy condition (like trash accumulation) in their neighborhood or setting a medication intake reminder. For instance, "reminder acetaminophen 3 times a day, 1 tablet each time" creates the appropriate reminder for the user. In these cases, the system returns an acknowledgment message to the user if the input is correct and accepted.

3.4.3. Proactive Outbound Communication and Outreach

Outgoing messages from the system are not always triggered by an input from the user. Some of the published messages are because of users' subscription to the message lists. For example, topical informational lists (e.g. categories like pediatric health, healthy life style, home environment improvement, elderly care, etc.) or emergency notifications. The subscribed users will receive a message when new information is published in the area of their interest or when there is a medical emergency like a disease outbreak.

An example of proactive outreach is sending motivational messages or healthy tips to the subscribed users (e.g. encouraging them to eat healthy or washing hands more frequently during the flu season). Medical organizations can become sponsors of these proactive health improvement activities. For example after an important surgery or giving birth, it is important that the patient receives follow-up cares such as recurrent visits to the doctor's office or repeat some lab tests. Manually keeping track of these follow-ups and reaching out to the patients is both time consuming and a waist of money and personnel's time. A predefined template (e.g. template for postnatal care) can send appropriate messages to the patients and encourage them to make doctor visit appointments or tests in a timely manner and also to keep track of their compliance.

3.4.4. Statistical Analysis and Reporting

As more users use the system, collection of data stored in the system can provide meaningful information about all health related services that are provided by the plug-ins of the system. Almost all of the information can be summarized or analyzed by their properties. For example, data in a geographical region, can show trends of sickness for the people in the region. Assuming we have a substantial number of users contributing to the system by inputting their daily drug usage, if we see a rise in the number of drugs related to cold and flu, we can determine that there is a flu epidemic happening in the region. If this information could be gathered in time and used properly, can limit the spread of diseases.

3.4.4.1. Demand Prediction

Analyzing data stored in the system could be useful for companies that supply health product and services. Trends of health related data can empower professionals to predict market needs or make guided guesses based on the data. These predictions can be used to forecast the demand for the other products and services as well. Going back to the example mentioned earlier, if our statistical analysis show increasing number of cold and flu patients in an area, this could trigger drugstores in the area to preorder more cold and flu drugs and related products. This will save companies investments by automatically organizing their logistics and being prepared for the rise in demand for their products and also helps customers to get a better service. In long-term this information can help to decrease the cost of medications as well.

3.4.4.2. Targeted Vital Information Propagation

Gathered information and its trend can be considered a service to some companies. For example municipal services can benefit from the reports of trash accumulation reported by the users. On the other hand, companies can send targeted information or messages to the users based on their history in the system (not just advertisements, but also important information). To elaborate this feature, we can consider a case where a drug company has a case of "drug recall" and wants to inform all of its previous customers to recall a specific drug. We can use our users' history to notify the patients who have a history record of using the specified drug. Of course in this manner the users are anonymous to the company, and their privacy is preserved while getting the service.

3.4.5. Machine to Machine Communication (Internet of Things)

Devices that are able to send and receive SMS can communicate with our application and add data to the system. Of course this communication can be in forms other than SMS, but since communication via SMS is simpler than Internet communication, more devices can exploit SMS. For example, a heart rate monitor that publishes data through a GSM modem to the server can only send the raw data instead of making any complicated computation on it. This facilitates lower cost and lower power consumption, thus the manufacturing of the device will have a cheaper and simpler process. In addition, since all of the computations and features are performed in the cloud, the company that uses the devices and the service can operate more easily and more efficiently.

On the other hand, devices capable of receiving and understanding messages, can receive commands or updates through our software. For instance, a heart rate monitor may need to receive updates on the boundary values (maximum and minimum heart rate), so that it can correctly send the out-of-range notifications. This way we can provide an intelligent virtual communication between the devices. The brain of this communication is the software running in the cloud. For example, a heart rate monitor that sends the data to the server, can trigger an alarm for an out-of-range heart rate. This alarm is going to be escalated to the emergency services so that they can help the patient by dispatching the closest ambulance to the patient's location. In addition, the system can send a command to the patient's oxygen mask to increase the flow of oxygen, unlock the doors of his/her house for the emergency first responders' entrance and can also change the room temperature to provide more comfort for the patient.

4. SYSTEM DESIGN

In this chapter, we describe the system architecture and also the flow of information into the system and through all of the layers of the system and output of the application.

4.1. System Architecture

This section provides detailed information about our proposed system architecture, the logical layers of the system and how they communicate with each other. Furthermore, the components and parts of each layer and their role in the system are described. Figure 4.1 illustrates the multi-layer architecture of the system.

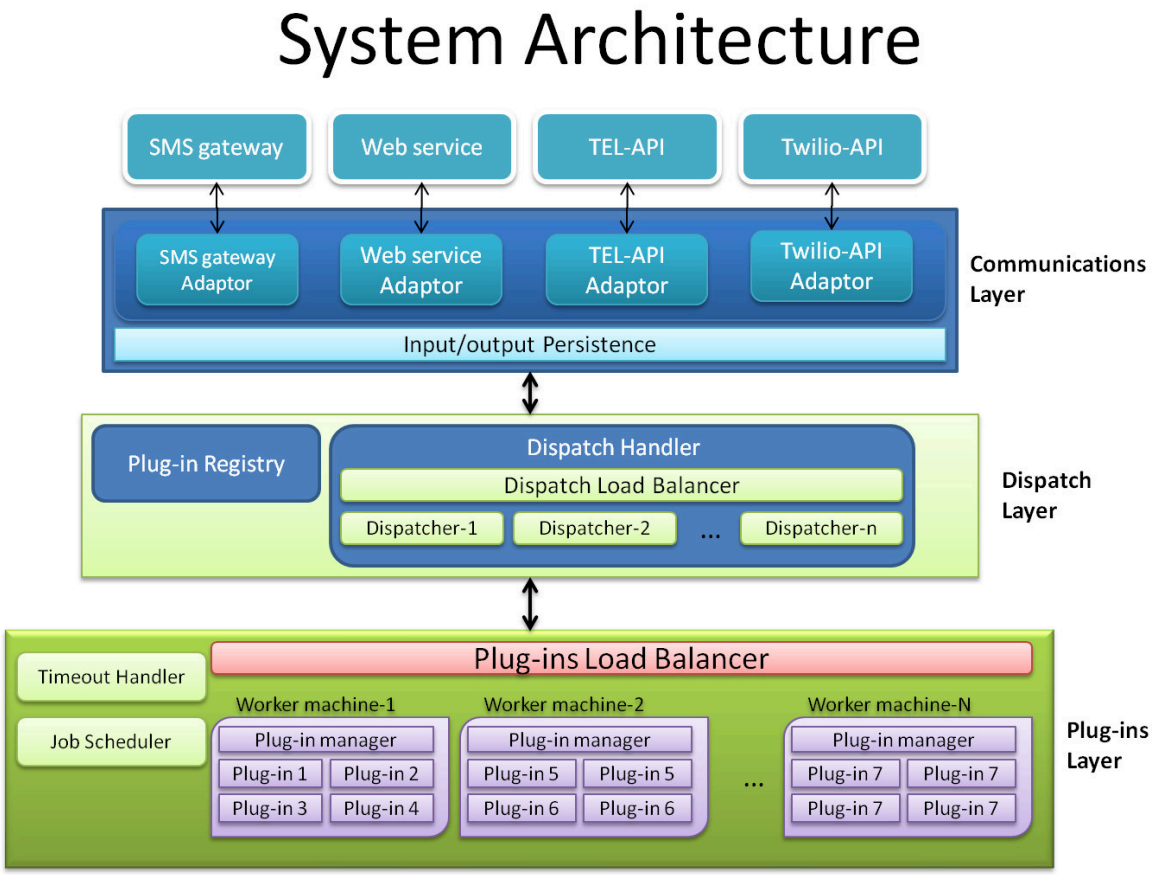

Figure 4.1. System Architecture

4.1.1. Communications Layer

Communications layer is the highest layer in our layered architecture. This layer is responsible for all inbound/outbound external communications. This can include an SMS gateway, web service, or any other kind of external communications. For example Twilio.com provides SMS communication to any cell phone number and is accessible as a web service. Also, Tweeter has programming interface to send and receive tweets that could be used as another form of communication to the system. In the most basic form, an SMS gateway provides SMS communication to the cell phones.

Regardless of the source, any input to the system must be converted to a unified format which is understandable by all of the other components/layers of the system. Any input is transformed to the data structure that is usable by the system and in the context of this thesis we call this data structure the "inputMessage". The inputMessage contains the body of the message and all of the meta information relevant to it. This includes the source information, communication information, and the time stamps. In the same respect, adapters convert responses from the lower level to the format understandable by the external communication device.

Every external communication media has its own adapter unit in the communication layer to do the conversion. This approach unifies the interactions with the lower levels and provides a layer of abstraction. Therefore, the lower layers will not be affected by the changes in the communication layer. Communication layer is also responsible for handling the sessions and cookies if available. Some external communication media have virtual sessions that provide more information about the communication.

One of the important components of the communication layer is persistence unit. It records any communication to/from the system and stores converted inputMessage from the adapter along with the original communication data. The communication history that is recorded by the persistence unit is used for consistency, fault tolerance, accounting, security and statistical analysis. This enable us to recreate any scenario that resulted in an error which is very useful in finding the cause of the errors and how to fix them.

4.1.2. Dispatch Layer

In a nutshell, dispatch layer is responsible for deciding what to do with the inputMessage and it transfers the inputMessage to the appropriate plug-in. Also, it manages how plug-ins respond to the inputMessage (e.g. timeout for generating a response.) The dispatch handler is the main component of the dispatch Layer. It provides the main functionality of the layer which is deciding what to do with the input message and what plug-in(s) should handle it.

The dispatch layer has a registry of plug-ins that contains the matching criteria for each plug-in. The source of the message and its content are the general criteria to decide which plug-in must handle the message.

The dispatch handler consists of multiple dispatcher units. Each unit has a queue of input messages to process and pass to the plug-in layer. "Dispatch load balancer" manages all the dispatchers and distributes the input messages to them and if necessary, it creates new instances of dispatchers to handle more messages or it discards the idle instances.

The dispatch load balancer scales up or down the number of dispatchers in use based on the system load so the dispatch handler can respond to the input messages in the queue as fast as possible and can prevent possible queue overflows or delays in responses. The dispatch Handler can potentially become the bottleneck of the system and needs to be highly scalable and also must be as fast as possible to pass the input messages to the plug-in layer in reasonable time.

4.1.3. Plug-ins Layer

A plug-in represents a specific functionality in the system (for example drug information plug-in that provides information about a drug) and it is encapsulated in the plug-ins layer. The plug-ins layer makes sure that each plug-in responds to the input within the time limit specified for that plug-in. In addition, this layer is responsible for assuring that the plug-ins' errors are handled properly and in case of a physical or logical failure, the plug-in action is passed to another instance of that particular plug-in.

Main processing of plug-ins is done inside a worker machine. A worker machine could be a virtual or physical machine that runs multiple instances of one or more plug-ins. A plug-in manager inside a worker machine manages the number of instances and communicates with the load balancer to instantiate the plug-ins that are needed in the system or too kill the idle plug-in instances.

The plug-ins load balancer makes sure that there is enough instances of each plug-in available to respond to the requests from the other layers. It also routes the plug-in actions to the appropriate worker machine. The plug-ins layer might contain other components to provide additional functionalities.

The job scheduler provides chronological functionalities to the system. Recurrent or scheduled jobs are handled by this unit. For example drug intake reminders are scheduled through this unit by a plug-in request. The timeout handler unit is responsible for handling any timeout event generated by the plug-ins and it triggers the necessary actions when timeouts occur. It works closely with the load balancer such that it helps to identify the plug-ins that are faulty or not responsive. In that case, the load balancer resends the plug-in actions to another plug-in.

If a component in the other layers needs to communicate with the plug-ins layer, it must use the load balancer, so that the consistency of the communication between the layers is maintained and the load balancer can provide the best performance for the system.

4.2. Flow of Information

In this section, we demonstrate how the components of the system interact with each other and we describe the steps of the information flow between the different components. Figure 4.2 illustrates the flow of information in the system.

1. The communication layer receives a message either from a web service or from a device (e.g. SMS gateway) and passes it to its appropriate adapter (e.g. SMS adapter).

2. The a dapter converts the raw input to inputMessage, a unified understandable message by the system.(e.g. it creates SMSInputMessage which extends inputMessage.)

3. The persistence unit stores the inputMessage in the database and passes it to the dispatch layer.

4. The plug-in handler adds the inputMessage to a queue in one of the dispatcher units and based on the plug-in registry criteria, the dispatcher unit decides what plug-in(s) must handle this inputMessage. In most cases, it is decided based on its origin, communication device, input content, and time stamp among other characteristics.

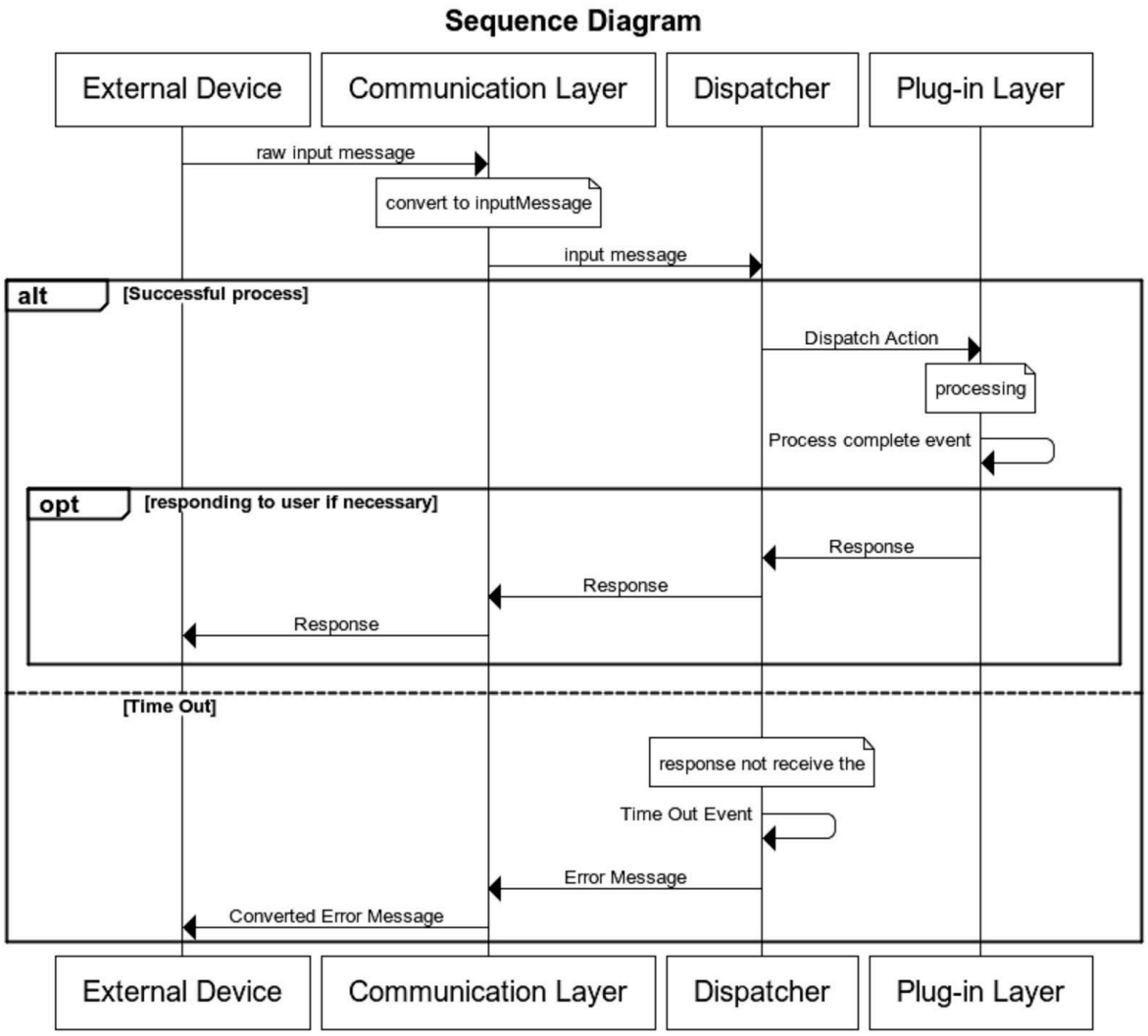

Figure 4.2. Sequence Diagram

5. The dispatcher unit creates a dispatch action and passes it to the plug-in load balancer in the plug-in layer.

6. The plug-in load balancer passes the dispatch action to one of the plug-in instances inside a worker machine (or instantiates one if none exists).

7. The plug-in processes a dispatch action. The processing may include looking up data, storing data, checking for a threshold, deciding on the next action, triggering an event, etc.

8. The plug-in must close the action (before timeout) and return the action to the dispatch layer.

9. The response handler in the dispatch layer receives a closed (or timed out) dispatch action and checks if the actions is correctly completed. If necessary, the response handler passes the appropriate message to the communication layer to be sent out to the device.

10. The adapter creates the appropriate message for the gateway or web service to be sent.

In order to further clarify the flow of information and interactions with users, it would be easier to have and example. Figure 4.3 illustrates an example input to the system. Every element of the figure are inside our application except user and SMS gateway. User initially interacts with the application using a SMS gateway for example Twilio (twilio.com). SMS gateway is a third party application or service and provides services like phone numbers and telecommunication services in order to communicate with users. It is essentially a gateway between user and our application and should provide reliable communication. Internally, we process the incoming SMS and respond to user if necessary. Initially raw SMS is converted to data structure understandable by system and safely stored in Database for future references. Also its ID is added to the appropriate queue for further processing. We may have multiple queues to further categorize processing of plugins. Plugins load balancer works in parallel to the persistance and processing unit and constantly monitors the queue. It fetches an item from the queue and passes it to the appropriate plugin. Every queue item contains sufficient information for plugin to do its job, for example it contains ID of the incoming SMS that is stored in the Database. Plugin load balancer also manages scalability of plugins and initiates or destroys plugin instances if necessary. After processing the data inside plugin, if necessary a response message is generated for the client and passed to SMS gateway adapter. Furthermore, adapter generate necessary communication to SMS gateway in order to communicate with user.

4.2.1. System Implementation

To give the best presentation of the conceptualized architecture, a prototype is implemented as the proof of concept. The implemented prototype contains the base framework for storing messages and interacting with inputs and also includes one module that provides information about the drug names. The features of the system can easily be expanded by adding new modules that interact differently with the user using the same framework.

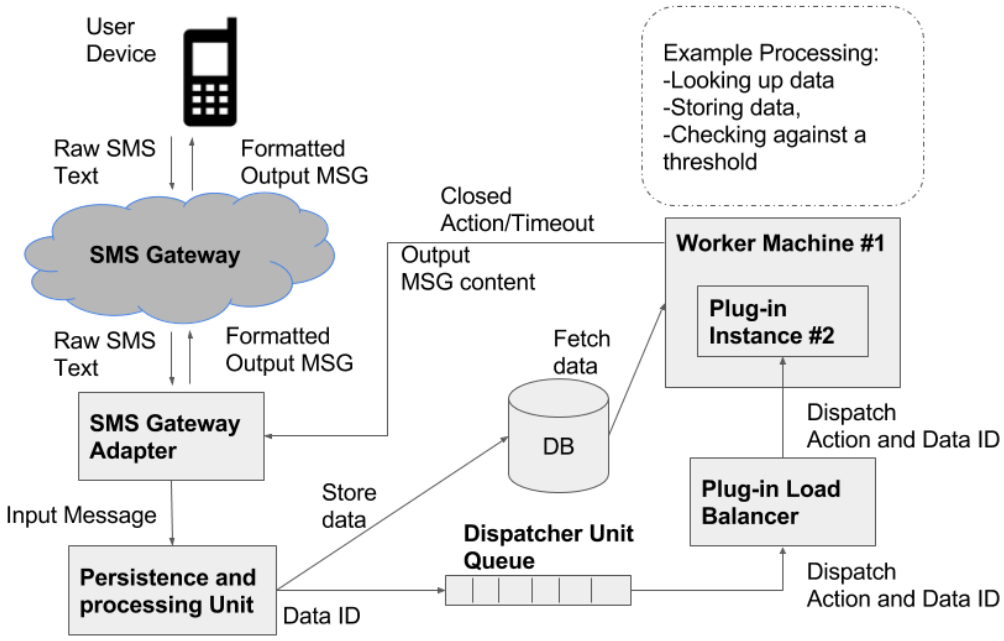

Figure 4.3. Example Flow of Information

In this thesis, the Amazon Web Services platform is chosen as the cloud platform to provide utilities like queuing and scaling options. As for the implementation framework, we used Java Spring to implement the components of the system because of ease of use, clear documentation and rich libraries integrated with the framework. In addition, using Spring makes dependency injection easier by opting in for Inversion of Control (IoC) pattern and auto-wiring reusable sub-components into each of the components.

Each component of the system that interacts with the Internet or the other components of the system through HTTP communication is using the MVC (Model-View-Controller) architectural pattern [2]. The controller accepts input and converts it to commands for model or view. The model manages data related actions like saving or retrieving data and also includes logic of the application. The view is the output of the transaction that in our prototype is mostly simple text communications, but it could contain any page components like tables, charts, etc.

In the context of Amazon Web Services, we are dividing the application into two sub-systems, the first is a web layer that interacts with inputs of the system and receives messages from outside whether it is a service or users interacting with the system. The second sub-system is the worker layer that processes the inputs and takes appropriate measures and actions like sending

a response or updating data in the system. The connection between these two sub-systems is Amazon's Simple Queue Service (SQS) that ensures reliable communication between these two components.

5. EVALUATION

Although most of the processing on the text messages are fairly simple and fast, scalability is the most important factor in the success of an application in this domain. We need to respond to every request within reasonable margin of time. Also we need to preserve processing power while system is used minimally, hence reducing the operation cost.

Text message applications usually have to deal with sudden surges in traffic over a short period of time. Although they might follow a pattern on a daily or weekly basis, but unlike other web applications they have sudden spikes in traffic due to an outside factor. An example would be a voting application for a popular TV show or in case of our application, surges in earth quake related requests a few minutes after an earth quake happens.

In the following sections, we describe each of the factors important to our functioning application. Also, we conducted a load test on the application. For the load test, we emphasize on the criteria described above, such as sudden surges in the input during a relatively short period of time.

5.1. Scalability

To make sure our application is scalable we have to observe its behavior under increasing traffic in single node and clustered environment. The application must be able to handle the growth of traffic by adding more resources and processes to manage and respond to the requests. Given the platform that we are using for handling the incoming traffic, we handle each request in a separate thread, hence increasing traffic triggers more threads in the application to respond to the requests.

We are using multiple layers of components in our architecture. This gives us the flexibility to scale any of those components individually. The gateway layer has its own load balancer and it can scale up/out as much as needed. This accommodates for the incoming SMS traffic without worrying about the processing time of plugins or any other concerns on the other layers. On the other hand, the plugins layer uses a load balancer as well as queues. It can scale based on the number of items that are waiting in the queue to be processed and of course this is independent of the gateway layer. For example, only two instances respond to the gateway layer requests because its process is easy and it is reliant on the network speed. On the other hand, we need more

processing power on the plugins layer. Naturally, we may have ten instances to process the items in the queue and to answer to the users. Using different layers of scalability we prevent the errors from one layer to be extended to another layer. For example, if a plugin uses a third party service to do certain processing, in case of failure or slowness on that service, we only need to add more instances to the plugins layer without affecting the gateway layer. Usually the size of the queue will trigger these actions. As a result, our users will not notice the delay in the response time and we provide a consistent user experience.

5.2. Performance and Responsiveness

It is important that our application responds to the requests under a certain time threshold regardless of the traffic throughput or the number of running instances, thus guaranteeing a uniform and consistent user experience for the customers of the application. For this project, ideally the majority of the response times should be under 200 milliseconds to provide a smooth and fast user experience. Although given the nature of the communication for text messaging, response time up to a few seconds is acceptable.

Figure 5.1 shows the results of our load test in a histogram of latency of the responses. The figure shows the number of requests that have a specific latency (in milliseconds). Ideally we want lower latency for each request and having the majority of the requests (higher number on Y axis) to have a small latency (closer to the left on X axis). Although there are a few requests with higher latency, but the majority of the requests fall under 120 milliseconds. The result of our measured test shows that 98.10% of the tested requests (total of 2372 requests) had a latency less than or equal to 120 milliseconds. Furthermore, 99.74% of the requests were responded under 200 milliseconds.

The data that is used for the histogram shown in Figure 5.1 includes all of the request and response times used for load testing of the application that will be described in the next section. It should be mentioned that the application was under stress of high traffic during a short period of time, yet it was able to maintain the ideal response time.

5.3. Load Testing

For our proof of concept project, we decided to simulate an environment that usually happens with text based messaging systems. An outside factor such as an advertising campaign triggers a sudden growth of inputs in the system. It is usually in a short time period but the throughput

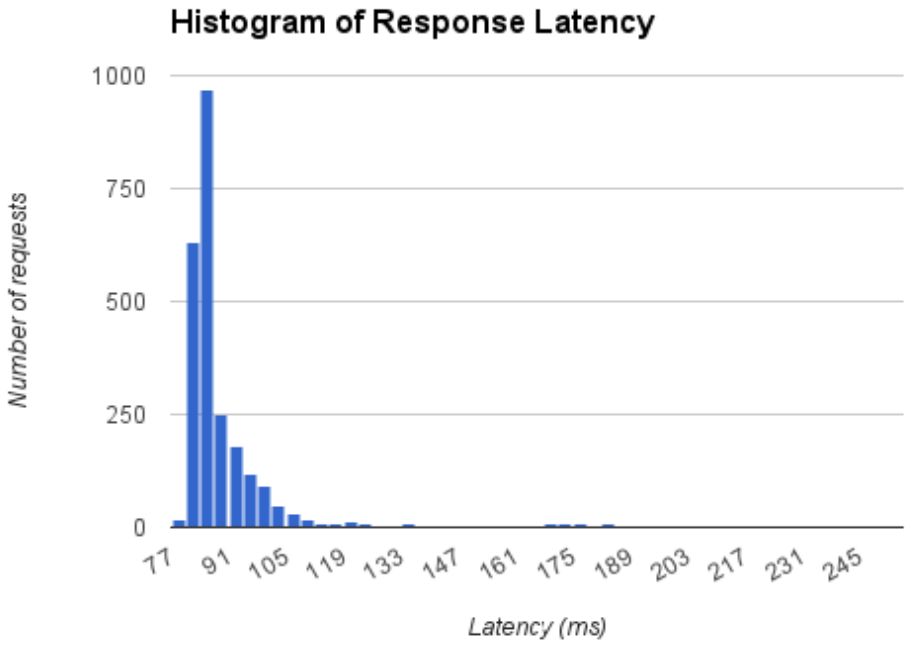

Figure 5.1. Load Test - Histogram of Response Latency

spike is in orders of magnitude and could potentially put the system in the denial of service (DoS) state. It is important that we can scale up/out the application in short period of time without sacrificing user responsiveness, which in this case, is response time to the incoming request.

In order to evaluate the scalability related performance of the implemented prototype we are using load testing techniques to generate loads similar to the real environment to observe the behavior of the system under different situations. We are using JMeter to generate random targeted traffic comparable to actual traffic for similar applications. The input data is generated randomly from a set of actual requests that are answerable by the application. We used our drug lookup plugin to answer questions about drug names. Although requests were sent to the system randomly, but all of the questions are valid and have a valid answer in our drug information database.

We need to be able to intentionally create request spikes for the application at certain periods of time. We also should be able to repeat the exact test based on the number of requests at each point of time in test. JMeter and related plugins provide the environment to create an exact number of Requests Per Second (RPS) for our tests. We can create a model of desired load at each time period of the test. As a result, we are able to ensure the consistency and repeatability in our

test, and exactly measure the behavior of the system at an exact load. Moreover, we can repeat the test in the same exact condition after each improvement. An improvement could be to the application and related systems like networking and routing, or can be a change in scaling policy or an update to the plugins.

Figure 5.2 shows the desired number of requests at each point of time (Requests Per Second or RPS) used for load testing. This model is generated using the JMeter plugin to shape and generate random requests. It uses a data set of actual drug names to test the system in random order.

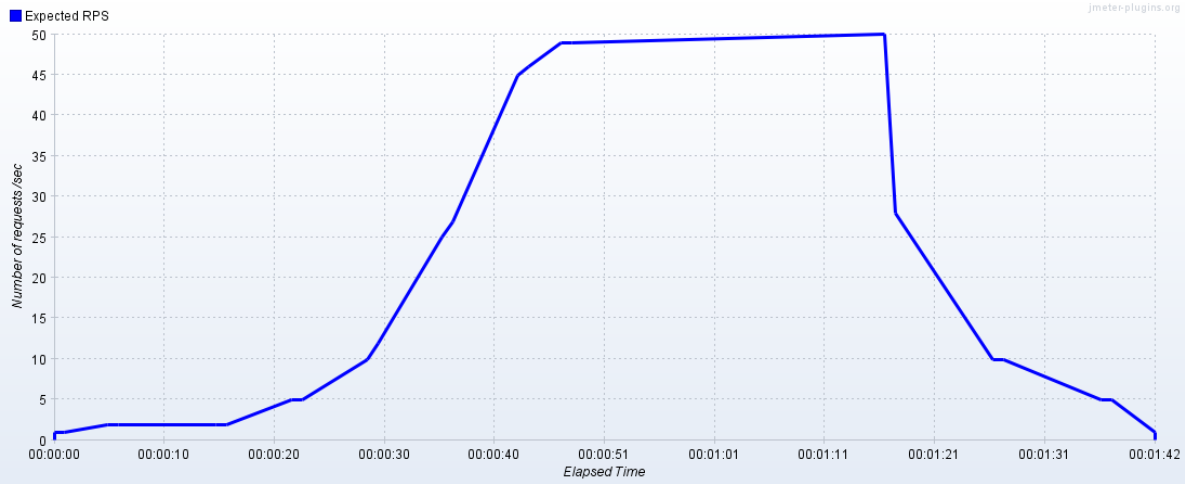

Figure 5.2. Load Test - Predefine Number of Requests for Load Testing

As shown in the Figure 5.2 we increase the input rapidly and then keep the input load on the stress level for some time. Both of these phenomena have an impact on how the application handles the load. First, the application should be able to cope with the increasing traffic and the increasing rate of the traffic increase. Second, the application should also be able to maintain the stress level for longer period of time as a sign of handling and reusing resources used in the previous requests/responses. Finally, the system should recover when the load is decreased.

As mentioned above, Figure 5.2 shows the intended input traffic; in contrast, in Figure 5.3 we show the input traffic that was actually used in the testing based on the real-time measurements. The figure shows both successful and failed transactions per second. As shown in Figure 5.3 none

of the requests have failed and the application was able to successfully handle all of the requests. As the results the corresponding line to the failures is constantly zero.

Figure 5.2 and 5.3 are direct outputs of the JMeter application used for load testing and measuring the latency of the responses. The raw data results of those tests are presented in the form of a histogram in Figure 5.1 to evaluate the responsiveness of the application during the test.

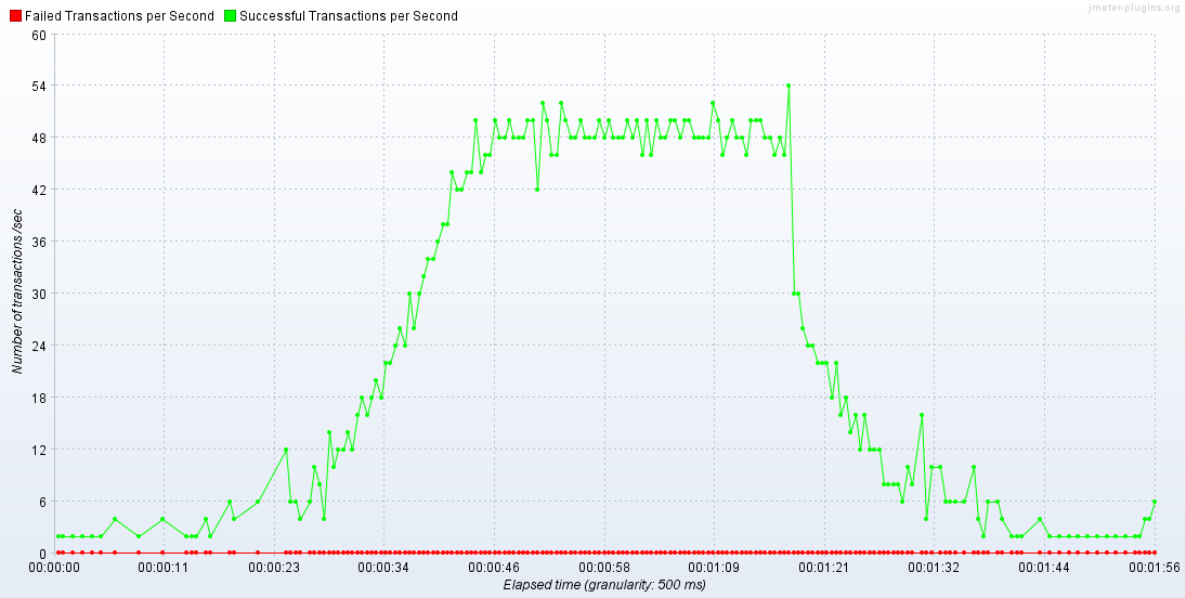

Figure 5.3. Load Test Traffic - Number of Successful and Failed Transactions

5.4. Elasticity

Elasticity is the behavior of the system when the traffic is more than the amount that could be handled by one instance. In this case, the application should scale out and initiate new instances to properly handle the traffic. On the other hand, when the traffic is less than the processing power of the system, it should scale in to preserve resources and cut the cost. Therefore, only sufficient processing power is used at any input level while the efficiency and responsiveness of the application is guaranteed to be at the desired threshold as shown in Figure 5.1.

We are using Amazon's internal signals such as CPU usage and response delays to decide if we must scale out and create more instances or we should scale in and shutdown the unused instances. It is important that during low traffic, specifically after a peak traffic, we shutdown the unused or lightly-used instances to save resources, otherwise our elasticity strategy is not practical

for real environments. The scaling policy in this situation is based on the amount of time an instance is idle, or alternatively the amount of traffic (or lack there of) that the instance receives in a certain period of time. For example, if an instance has received less than 10 requests per minute in the last 5 minutes, it is an ideal candidate for shutdown. After an instance-shutdown there is a grace period that scale in policy is put to hold (e.g. 5 minutes.) The grace period is necessary because this way we prevent shutting down too many instances all together. This would be a regular problem for equally load balanced instances, since all of the instances get roughly the same amount of requests at any time.

During our test shown in Figure 5.3, the scaling policy that we put in place, increased the number of instances and added two more servers to handle the peak load of the inputs. It also terminated the instances and reduced to only one server when the load returned to minimum. Our scaling policy is based on parameters of the system running the instance. We used CPU utilization and response delay of instances provided by AWS instances to actively determine whether new instances are needed or we should terminate idle instances. An alternative approach would be to measure the input traffic of the instances. Since SMS messages are relatively the same size, an increase in the input traffic would correlate to the number of inputs, hence the requirement for instances.

Although the number of servers used in our proof of concept application is relatively low, but it is comparable to real world SMS applications. Moreover, the combination of the scaling policy and the load balancers in each layer of the application, theoretically would have the same consistent behavior if the number of the nodes were higher. The reasoning behind the anticipated consistency is that we use Amazon Web Services components such as load balancers and SQS (Simple Queue Service) for our platform which guarantee the same behavior.

6. CONCLUSIONS AND FUTURE WORK

This thesis presents a proof of concept for capabilities of a lightweight mobile health application with the focus on light communication and scalable server side processing. The proposed approach is in particular advantageous for low-end devices with basic capabilities such as text messaging. However, the proposed framework is generic and expandable to support other technologies. Exploiting the communication layer in our proposed multi-layered architecture, which converts input messages to a data structure understandable by the system, we can extend the input to any means of communication that can be translated to our internal data structure. Therefore, any other kind of communication other than SMS gateway can communicate with the system. For example Twitter has a lot of similarities with SMS and it is accessible through APIs for programmers over the Internet.

We presented a wide variety of examples for health-related applications that can be implemented leveraging the proposed framework. It would be ideal to provide means for customers to adopt the proposed framework and easily create their own specific user interaction using this platform. For example, our presented platform enables a laboratory to setup a system to send out lab results to the patients using web user interface without the need for any technical knowledge of the system and it allows the lab to only focus on the user experience and interaction. Alternatively plug-ins for the platform provide needed services for customers. Using cloud computing, our proposed framework offers great scalability and flexibility in that the system can efficiently grow in terms of the number of users and features of the system. In addition, cloud computing enhances availability and fault tolerance which are vital for a robust mobile health application. Moreover, the proposed multi-layered architecture makes the design modular and less error-prone and the separation of concerns results in enhanced system scalability.

The proposed light-weight mobile-health framework in this thesis defines an affordable, practical and scalable platform for a wide range of future health-care applications. Although the usage of smart-phones is growing fast among the global population and more complex technologies based on data communication platforms such as push notifications and modern social media communication applications are gradually replacing basic text messaging, there is still a signifi-

cant portion of the world's population who are in need of simple and efficient health applications compatible with basic cellphones. Therefore our proposed SMS-based application offers a viable solution for cheap and efficient communication with people in need of health-care who do not access to the Internet or high computation power on their mobile phones.

Although the focus of this thesis is to create an application framework that mostly provides individual assistance in the areas related to health-care, as an outcome, we gather considerable amounts of health related data and the behavior of our users. The users in this context include both individuals using the features of the application, and devices that provide input data to the system. As a result, we have a great opportunity to have a holistic view on all of the gathered data, and infer useful information by analyzing the data to find meaningful patterns exploiting data science and machine learning techniques. For example an increase in the search about flu related topics to our system from a specific geographical area, implies a flu epidemic emerging in that region. This information can result in better preparation for the situation by taking quick action in informing the community and the health providers that could be involved such as hospitals, drug-store chains, and better preparation for emergency services. Data analytics on the huge amounts of collected data in the studied mobile-health applications in this thesis is a promising future research direction.

The mobile-health is still in its infancy and we anticipate that in the years to come the mobile phones will have an important role in improving the global health. As mobile and smart hand-held devices are becoming commodity, they will be used heavily in health-related daily life applications including diagnostic, prevention, and intervention. This thesis lays the foundation for practical and lightweight mobile-health applications targeted for simplicity and scalability, especially for parts of the world that older and simpler mobile phones are still in use. It can be efficiently expanded as newer health applications emerge.

REFERENCES

[1] Text messaging-a new way for delaware physicians care to help its members. `http://www.businesswire.com`, June 2008.

[2] Model-view-controller (mvc) software design pattern definition. `https://en.wikipedia.org/wiki/Model-view-controller`, April 2016.

[3] Erin Allday. Health department answers questions via text messages. `http://www.sfgate.com/health`, April 2006.

[4] Marco Altini, Julien Penders, and Herman Roebbers. An android-based body area network gateway for mobile health applications. In *Wireless Health 2010*, WH '10, pages 188–189, New York, NY, USA, 2010. ACM.

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, Apr. 2010.

[6] Gladys Block, Barbara Sternfeld, Clifford H Block, Torin J Block, Jean Norris, Donald Hopkins, Charles P Quesenberry Jr, Gail Husson, and Heather Anne Clancy. Development of alive!(a lifestyle intervention via email), and its effect on health-related quality of life, presenteeism, and other behavioral outcomes: randomized controlled trial. *Journal of medical Internet research*, 10(4), 2008.

[7] Dale Bramley, Tania Riddell, Robyn Whittaker, Tim Corbett, R-B Lin, Mary Wills, Mark Jones, and Anthony Rodgers. Smoking cessation using mobile phone text messaging is as effective in maori as non-maori. *The New Zealand Medical Journal*, 118(1216), 2005.

[8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6):599–616, June 2009.

[9] Wesley Chun. What is cloud computing? `https://developers.google.com`, June 2012.

[10] Sunny Consolvo, David W. McDonald, Tammy Toscos, Mike Y. Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, Ian Smith, and James A. Landay. Activity sensing in the wild: A field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1797–1806, New York, NY, USA, 2008. ACM.

[11] Judith B Cornelius and Janet S St Lawrence. Receptivity of african american adolescents to an hiv-prevention curriculum enhanced by text messaging. *Journal for Specialists in Pediatric Nursing*, 14(2):123–131, 2009.

[12] ITU World Telecommunication/ICT Indicators database. Mobile-cellular subscriptions. `http://www.itu.int/en/ITU-D/Statistics/`, July 2013.

[13] BJ Fogg and Enrique Allen. 10 uses of texting to improve health. In *Proceedings of the 4th International Conference on Persuasive Technology*, Persuasive '09, pages 38:1–38:6, New York, NY, USA, 2009. ACM.

[14] I. Foster, Yong Zhao, I. Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1–10, 2008.

[15] R.S.H. Istepanian, E. Jovanov, and Y.T. Zhang. Guest editorial introduction to the special section on m-health: Beyond seamless mobility and global wireless health-care connectivity. *Information Technology in Biomedicine, IEEE Transactions on*, 8(4):405–414, 2004.

[16] E. Jovanov, A. O'Donnell Lords, D. Raskovic, P.G. Cox, R. Adhami, and F. Andrasik. Stress monitoring using a distributed wireless intelligent sensor system. *Engineering in Medicine and Biology Magazine, IEEE*, 22(3):49–55, 2003.

[17] David Joyce and Stephan Weibelzahl. Text-messaging as a means to lowering barriers to help seeking in students with depression. In *Proceedings of IADIS International Conference e-Society, Dublin, Ireland*, pages 211–214. Citeseer, 2006.

[18] James G Kahn, Joshua S Yang, and James S Kahn. Mobile health needs and opportunities in developing countries. *Health Affairs*, 29(2):252–258, 2010.

[19] Arvind Kumar, Amey Purandare, Jay Chen, Arthur Meacham, and Lakshminarayanan Subramanian. Elmr: Lightweight mobile health records. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 1035–1038, New York, NY, USA, 2009. ACM.

[20] Chinmoy Mukherjee, Komal Gupta, Rajarathnam Nallusamy, and Sumit Kalra. A system to provide primary healthcare services to rural india more efficiently and transparently. In *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*, ACWR '11, pages 379–384, New York, NY, USA, 2011. ACM.

[21] Jami L Obermayer, William T Riley, Ofer Asif, and Jersino Jean-Mary. College smoking-cessation using cell phone text messaging. *Journal of American College Health*, 53(2):71–78, 2004.

[22] Barbara Sternfeld, Clifford Block, Charles P Quesenberry Jr, Torin J Block, Gail Husson, Jean C Norris, Melissa Nelson, and Gladys Block. Improving diet and physical activity with alive: a worksite randomized trial. *American journal of preventive medicine*, 36(6):475–483, 2009.

[23] Narjes Torabi and Victor C. M. Leung. Robust access for wireless body area networks in public m-health. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, pages 170–176, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[24] Woodrow W. Winchester, III. Cover story: Catalyzing a perfect storm: Mobile phone-based hiv-prevention behavioral interventions. *interactions*, 16(6):6–12, Nov. 2009.

[25] Mudasser F. Wyne, Vamsi K. Vitla, Praneethkar R. Raougari, and Abdul G. Syed. Remote patient monitoring using gsm and gps technologies. *J. Comput. Sci. Coll.*, 24(4):189–195, Apr. 2009.

[26] Tae-Jung Yun, Hee Young Jeong, Tanisha D Hill, Burt Lesnick, Randall Brown, Gregory D Abowd, and Rosa I Arriaga. Using sms to provide continuous assessment and improve health

outcomes for children with asthma. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 621–630. ACM, 2012.