Facilitating Multi-Device Usage in mHealth

Richard K. Lomotey* and Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoonl, SK, Canada
{richard.lomotey@, deters@cs.}usask.ca

Abstract

Today, it is a common phenomenon for physicians to own multiple mobile devices such as smartphones and tablets in order to seamlessly access the Electronic Health Record (EHR). But, the over reliance on wireless communication channels by mobile devices limits the user expectation since consistent connectivity cannot be guaranteed due to the intermittent connectivity losses in mobile ecosystems. What is even challenging is the presence of the `CAP theorem` which states that though the following three properties are desired in a distributed environment: consistency, availability, and partition tolerance, only two of the properties can be guaranteed simultaneously. In this paper, we deployed a reliable mHealth architecture that enables healthcare practitioners to employ their multi-mobile devices to access the EHR. We proposed a middleware platform that synchronizes the medical data on the multi-devices of a single user with careful consideration to the CAP theorem. The architecture is based on mainstream technologies such as: the publish/subscribe technique for real-time data access, medical data encryption for security, and mobile-side caching.

Keywords: mobile devices, middleware, mHealth, CAP theorem, private cloud, Health Information System, ROA, SOA

1 Background

The medical terrain is witnessing significant adoption of mobile technology to advance on remote care delivery as well as enhancing physician-patient interaction. The employment of mobile devices and other ICT tools to facilitate healthcare delivery is known as mHealth [1]. Most health facilities are building their Health Information Systems (HIS) to support mobile requesters who need to access the Electronic Health Records (EHR) [2]. In this regard, the mobile devices are treated as consumers in a client-server interaction. Further, there have been attempts at aiding healthcare professionals to use mobile devices to input or update patient records in the HIS [2].

However, there is an emerging design paradigm within the mobile landscape which focuses on hosting enterprise services and data on the smartphone and tablet devices [3] [4] [5] [6] [7]. Mobile hosting in the health sector is evidenced in a few studies and implemented projects such as MobiHealth [8] and SOPHRA [9] which aim at facilitating the mobile device as providers of medical sensor data within a HIS.

This paper presents an ongoing research project which focuses on enabling the mobile devices as platforms for hosting medical data and application while ensuring application consistency on the medical practitioner's multi-mobile devices. The motivation for the project is in response to the call by the Geriatrics Ward at the City Hospital in Saskatoon, Canada to support the "Baby Boomer" generation who are approaching the retirement age. Statistics show that the Post-World War II era witnessed a rise in birth rate between 1946 and 1965 with Canada recording 15% increase in birth and America recorded

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 4, number: 2, pp. 77-96
*Corresponding Author: Department of Computer Science, Saskatoon SK, S7N 5C9 Canada, Tel: +13068818387

78 million births [10] [11]. The baby boomers are now reaching their retirement and the more reason governments have to start thinking about supporting them. However, it appears the baby boomers are not receiving the deserved medical attention. In a recent emotional article by the Seattle Times [11] titled *"Baby boomers retiring to rural areas struggle to find doctors,"* it is evident that the baby boomer generation will struggle with medicare considering their mobility. In response to support the baby boomers, the Geriatrics Ward at the City Hospital in Saskatoon, Canada in partnership with our research lab, MADMUC Lab at the University of Saskatchewan, Canada, is exploring ways to employ the mobile device to facilitate better healthcare delivery. We anticipate that if mobile hosting technologies can be enhanced, then healthcare practitioners can access live medical records which will enable them to offer remote support to patients. We realized that the healthcare professionals own multiple mobile devices and sometimes, these devices are forgotten at home or at the health facility. Hence, if n-screen applications can be deployed, the medical professionals can use any of their other available devices to access data. However, to deploy the multi-device application, there are some system challenges that are inherent in distributed architectures that must be addressed. In this work, we identified the presence of the CAP theorem [12] [13] (detailed in the next section), sporadic wireless disconnections, bandwidth fluctuations, and limited battery life. Part of the proposed solutions to address the aforementioned challenges is reported in [14] where we put forward a middleware-layer that coordinates the activities of the mobile participants as well as reaching the HIS. As an advancement on the previous work, we investigate the potential of adopting the publish/subscribe technique for real-time data dissemination, medical data encryption for security, and mobile health data hoarding techniques for the sustainability of medical business continuity in cases of network disruptions.

The remaining sections of the paper are structured as follows. The next section discusses the challenges and options for the CAP theorem in distributed mobile networks such as mHealth architectures. Section 3 and 4 detail works on mHealth and Web services resectively. Section 5 discusses our research focus. The architectural design of mAppGP is explained in Section 6 and 7. The evaluation of the framework is given in Section 8 and the paper concludes in Section 9 with a summary of our contribution and future work.

2 The CAP Theorem

It is surprising when we could not find any existing work within the E-Health and mHealth landscapes that exclusively focused on solving the CAP theorem challenge. Since the challenge is inherent in distributed systems, we decided to explain the concept to the reader and assert that the challenge applies to distributed mHealth architectures.

Services running in distributed scalable systems are expected to provide the following support: `Consistency` of the data/service, `Availability` of the system/data, and `Partition-tolerance` to failure. However, Eric Brewer [12] in an invited talk in 2000, made a conjecture which states that no distributed system can guarantee all three requirements at the same time. Brewer again noted that two of the three requirements can be guaranteed simultaneously if one requirement can be traded-off [12] [13]. After a decade, the CAP theorem still remains an issue to address especially in systems that desire the deployment of the three properties. The theorem is illustrated graphically in Figure 1 as reproduced from [15].

Gilbert et al. [13] employed a formal methodology based on read/write operations on dis-jointed nodes to prove Brewer's conjecture into a theorem. The formalism shows that for every associative write, it is impractical to read the same data if the write has to be available on all the dis-jointed nodes at the same time. Simply put, when a data is updated on a particular node in a distributed environment, the update can be seen by all requesters if and only if we lock the system from showing the old state of

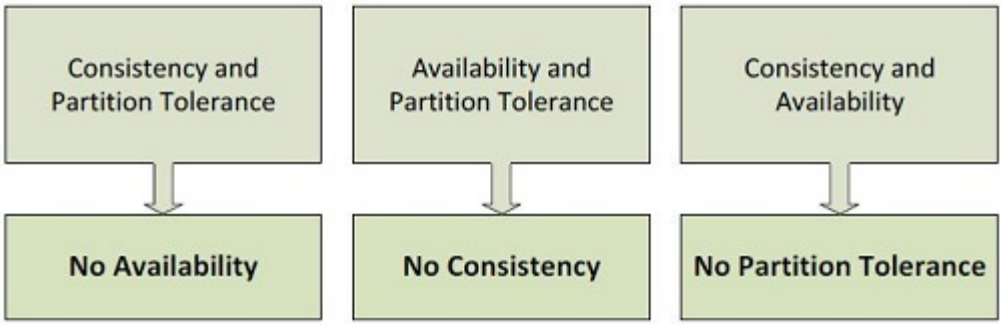

Figure 1: Conceptualization of the CAP Theorem

the data. Further, introducing locks means there will be times when the same data will not be available (accessible).

Consistency is the requirement that guarantees that states (i.e., data, application, or Web services) stored in a distributed system are seen the same at every node by the requesting clients [16]. Gilbert et al. [13] described consistency as "atomic" which means every system transaction is one and must be completed fully or not started at all. To ensure a consistent system, some developers and researchers have adopted the ***ACID*** (Atomicity, Consistency, Isolation, and Durability) [17] technique especially within the context of databases. `Atomicity`: Considering an operation in a distributed system, if a part fails then the entire transaction fails and the system must be rolled back to its original state. This means that each transaction is single and cannot be broken into parts. `Consistency`: This property ensures that anytime a request is issued to the database/server, the requester receives a valid and desirable data. According to Pritchett [17], "the database will be in a consistent state when the transaction begins and ends." `Isolation`: A distributed system should be built in a way that transactions being processed should be hidden from other processes until it comes to completion. `Durability`: After a successful transaction, the operation cannot be rolled back even if there are system failures [17].

Moreover, the use of database keys causes distributed transactions to become highly coupled [17]. It is also challenging to build long-running transactions in ACID since there is locking and serialization. Instead, some developers propose SAGAS [18] where long running transactions can be broken into smaller transactions but inter-linked. Further, the challenge with ensuring consistency is the high cost of bandwidth consumption due to the communication overhead since all the participating nodes have to commit to an atomic transaction [19]. Again, since mobile devices are not guaranteed reliable connectivity, it is impractical to ensure locking especially when a mobile participant is not available to commit at a beginning of a transaction.

Availability ensures that when parts of the nodes in a distributed system become inaccessible as a result of failures, the other nodes should continue to operate and support every read and write operation [19] [20]. It is important that intended responses are received for each request [17]. Gilbert et al. [13] report that systems tend to fail during peak performance, therefore making availability very difficult to achieve when it is needed most. Availability can be achieved if consistency is compromised when the ***BASE*** (Basically Available Soft-state Eventual-consistency) [17] technique is adopted. BASE is applied to minimize the level of coupling in ACID systems. Decoupling transactions is achieved by introducing persistent message queues. This process ignores two-phase commit and ensures latency. Further, eventual consistency is achieved if the system allows for a time lag between operations. According to Vogels [21], considering N_h hosts that store replicas, N_w successful writes to a replica, and N_r replicas that can be read from hosts, then $N_h \geq N_w + N_r$ ensures eventual consistency and partition tolerance within a distributed system. This means that outdated replicas can be seen on some hosts when a read operation

is invoked. There are different types of eventual consistency models such as the `Causal consistency`: This model assumes that if a replica is updated by a process X and duly notifies process Y about the update, then all read operations of Y should return the updated replica [21] [22]. `Read-your-writes consistency`: This is a special case of causal consistency where after process X has updated a replica, durability is ensured where by the older version of the replica cannot be read again but only the updated version [21]. From the parameters used above to explain Vogel's concept of replicas, if $N_r + N_w > N_h$, then read-your-writes consistency is guaranteed [23] [20]. `Session consistency`: Read-your-writes consistency is implemented to work only within a context of a session [21]. This is very practical because every session is guaranteed read-your-write consistency but when a session fails, a new session has to be started with no guarantee of consistency from the previous session [21].

Partition-tolerance is achieved when a distributed system is built to "allow arbitrarily loss of messages sent from one node to another" [13]. The current demand from web services consumers makes it impractical to keep all data at one source. This is because when the source fails, it means the entire system becomes unavailable. Partition-tolerance therefore allows for system states to be kept in different locations. This property is a given in a mobile distributed environment since mobility and offline access to data can only be achieved through partitioning.

Browne [24] noted that the CAP theorem is most applicable in scalable systems as transactional load increases. The best pair of guarantees normally preferred by users is *Availability* and *Partition-tolerance* while trading-off consistency because it is better to give some data rather than access denied in most workflows [24]. In summary, how is the CAP theorem phenomenon applicable to the mHealth architecture deployment? From the explanations of the various system requirements and the illustration in Fig. 1, it is not possible to guarantee all the three requirements; so which pairs of the three requirements are possible guarantees? Since the mobile node is required to access medical data which resides on some server-backend, partition-tolerance is automatically present. This means it is impractical (if not impossible) for a distributed mHealth system to guarantee consistency and availability at the same time. However, with the presence of partition-tolerance, either consistency or availability is feasible. The challenge is, which one of the two requirements should be traded off. From the discussion with our medical partners on the project, we opt for the availability option because the medical practitioners want to have offline access to medical data (which is the foundation for supporting Geriatrics patients in their homes outside the health facility). However, mHealth systems also require consistency of the medical data so we have to come up with a reasonable eventual consistency model that can guarantee real time data synchronization. This goal is explored further in the paper when we introduce our mHealth architecture. But before then, we discuss Web services and how they have become integral part of Health Information Systems (HIS) deployment.

3 Related Works on mHealth Systems and Applications

E-health is defined by Eysenbach [25] as a study that employs information and communication technology mostly with the aid of the Internet; to support the safe delivery of healthcare services. This field is useful from the perspective of patients, healthcare delivery personnel, and healthcare administrators. Eysenbach also stated that e-health is more beneficial in terms of reducing cost of diagnosis and greater improvement in the communication between patients and doctors. The exponential rise in the use of mobile phones as reported by Ranck [1] motivated us to look into how the use of mobile devices such as smartphones and tablets will better enhance the delivery of E-Health. This is described in [26] and [9] as ***mHealth***.

A lot of mHealth applications are being deployed in recent times but little is known about their architectural design. Some of these applications as highlighted in [1] include: the Radiology Assistant,

Epocrates Rx, iAnasthesia, Unbound Medicine, Medequations, Skyscape's PDR and the Human Atlas. The BlackBerry smartphone has also seen some mHealth app developments like the one created by the University of Pittsburgh Medical Center for health practitioners to use in accessing patients records [6].

Minutolo et al. [27] showcase the importance of employing mobile technology as a means of supporting patients with heart failure conditions. The authors designed a rule-based system that detects abnormality in a patient's posture and physical activity. Further, the medical data is modeled based on knowledge representation which encompasses a patient's heart rate variability (HRV). The HRV is designed to trigger an alert if the patient is in precarious situations. Knowledge reasoning is also put forward where available factual data is supplied to the reasoning engine for new inference. Overall, the proposed architecture comprises of three layers: data layer, decisional layer, and action layer. The data layer represents the user interface and the interactive interface that accepts input that is processed by the decisional layer. The decisional layer therefore is responsible for the execution of the rules. The action layer executes instructions that are received from the decisional layer such as triggering alarms.

The SOPHRA [9] framework focuses on mobile health record hosting in a highly distributed wireless environment. Considering undesired situations such as the intermittent loss in connectivity and bandwidth fluctuation issues, a middleware layer is proposed which is based on the following macro technologies: caching, medical data routing, system events monitoring, and pushing. The middleware relies on the publish/subscribe mechanism to synchronize the mobile-hosted data with the server-end data. The concept of cloud-powered middleware adoption in the design of mHealth applications is also evidenced in the works by Ji et al. [28] [29]. The major reason for the implementation of the middleware is to enable "Always Best Connected and best Served (ABC&S) communication mode." The middleware achieves this goal based on content adaptation. Thus, the middleware reads the HTTP requests headers of the mobile requester and detects features of the requester such as device resolution. This information aids the middleware to deliver content that can be easily assimilated on the device. Similar to [27], the authors also designed a rule-based system that performs the reasoning on which content to deliver to which device.

Huang et al. [30] also designed a location-based mobile health system that facilitates remote health monitoring. The idea of the authors is to build a system where mobile data is collected and sent to a centralize health facility and based on the incoming data, the location and position of the mobile user can be identified. The system is prototyped and the design is a three layer with the following components: mobile monitoring terminal for the identification of position information using GPS/Wi-Fi, service center platform which is the centralize location with super-nodes such as databases, communication server, web server, etc., and browser client which is responsible for capturing reviewing information. Apart from the academic highlighted studies on mHealth, a growing number of enterprise players are also deploying and supporting mobile health services deployment based on cloud computing technologies. Some notable providers are Cerner [31], Qualcomm [32], and Dell [33].

4 Adopting Web Services in mHealth Applications

The main architectural design that we are focusing on in our work is the adoption of Web services. In our work, we adopt the Web services approach where the medical data accessibility is facilitated over the network. We found the approach as being efficient for mobile devices consumption of the electronic health record. But first, we highlight the two major paradigms for building web services in modern systems; the SOA and ROA.

4.1 SOA

The Service-Oriented Architecture (SOA) is an architecture that is used to create het-erogeneous web services. The SOA framework provides support to various compo-nents of web services to interoperate. According to Wicks et al. [34], SOA focuses on reusability of software and integration. Another key thing they report about SOA is packaging, which makes changing of legacy software very fast and at minimal cost.

Even though SOA helped web services developers to overcome interoperability problems, it has limitations. SOA passes a lot of XML since it uses the Simple Object Access Protocol (SOAP) and this makes simple communication between components difficult. In most cases, to achieve cross platform interoperability between web ser-vices, a lot of standards have to be followed [35]. Lee et al. [36] report on the chal-lenges faced by Telco with issues on scalability due to earlier SOA design to build transactions. Other challenges identified with SOA design were poor system perfor-mance under heavy workload and adaptability. As a solution to address the challenges in SOA, they provided ROA approach.

CardioNet [37] [38] is a mobile app developed for heart patients to monitor their heart beats through wired sensors which are worn by the patient. With the help of sensors and mobile devices, the data is sent from the patient to the health service providers. The data is analyzed and then recommendations are sent back to the patient based on the results. Rusu et al. [2] report on the performance of CardioNet. It is a distributed system implemented using Service Oriented Architecture (SOA) where everything (i.e. hardware, software, and medical activities) is modeled as "services". The system also uses the high level SOAP. The system guarantees interoperability and permits heterogeneous systems' integration. However, one will still argue that building a ROA web service can be more reliable and scalable since the authors did not mention the performance of the system regarding these parameters.

4.2 ROA

Resource-Oriented Architecture (ROA) as presented by Overdick [39] follows the guidelines of the REp-resentational Sate Transfer (REST). Xu et al. [40] in their work used REST to build ROA system and compared the results with Service-based web services. ROA is more feasible in building business pro-cesses because of the use of hyperlinks that enable processes to communicate. The use of uniform interface (HTTP methods) exposes resources to be managed easily than in SOA where service compo-nents have to be created at every operation. Xu et al. also proved that ROA business processes allow process visibility. This is good especially in the sense that the client can talk to the server periodically for updates and the vice versa. Selonen et al. [41] report on building a lightweight architecture style for building ROA as a solution to their Mixed Reality project. This helped them to achieve their aim of storing, retrieving, and managing interlinked content which otherwise couldn't be successful with SOA.

5 Use Case and Research Focus

The fundamental goal of this project is to support the deployment of a mHealth application on multiple devices of the physicians. The purpose of the application is to host the Geriatrics patient's medical data such as visits, vitals, demographics, allergies, and attending doctors on the mobile device. The reason for the multi-device app deployment is summarized in the use cases that we present below. These use cases are interviews and concerns that the medical partners raised.

5.1 Use Cases

Concern from User A: "Our patients are the aged and some require home care atten-tion from our health personnel outside the health facility. Is there a way to access the medical records of these patients on my tablet device outside the hospital when some of these patients have no Internet at home to guarantee connection to the hospital server?"

Concern B: "Last week Monday I had to visit three patients outside the hospital and by the time I got to the home of the second patient, my BlackBerry Playbook tablet run out of battery. I forgot my charger at the hospital so I have to drive back to the hospital to pick it (about 35 minutes drive) and charge the tablet before going back to visit the patients. Meanwhile, I had my iPhone on me which was fully charged." This concern is a case the project team anticipates.

Concern from User C: "I want a device-independent application. I will like to easily access patients' records on any of my mobile devices even if I forgot the other one at home.

Concern from User D: "Sometimes the Wi-Fi out there, even here in the Ward is unstable so using my iPad to access the data is challenging; but, I can use data plan if I can have the application on my Android Galaxy.

5.2 Research Aim and Questions

From the concerns raised by our medical partners, it is clear that we have to deploy a cross-device application. That is, the application has to be supported by multiple mobile platforms and users should be supported to switch between devices when the need arises. It is also important to note that the mobile devices are important for the medical practitioners in this project because they are able to record new medical information of the patient (e.g., recovery progress, new developments, visits to other health facilities, etc) using the mobile and this information is sent to the health information system at the City Hospital. Considering the requirements and the vast background issues, we formulate the following research questions.

1. How do we synchronize the mobile-hosted medical record with the main HIS? Hosting the medical record on the mobile ensures offline accessibility (or availability) but can lead to inconsistent records from the CAP theorem.
2. How do we ensure medical data accessibility on the medical physicians' multi-devices?
3. How do we enforce medical data privacy and security?

To answer the research questions, we proposed a multi-device application called mAppGP (short for ***m***Health ***App***lication for ***G***eriatrics ***P***atients), and it is pronounced *"map"*. The architectural overview is described in Figure 2.

6 The mAppGP Framework

We put forward a three-tier architectural design that can meet our goals. The mAppGP framework consists of the mobile devices, a cloud-hosted application middleware, and the main HIS. The generic architectural design of mAppGP is shown in Figure 2. The primary functionality of the application is to support the physicians to access the following features on the go: patients' demographics, vitals, medical history, visits, attending doctors, and recovery progress report.

Following the REST Web services architectural design, the Electronic Health Records (EHR) are exchanged as network-oriented applications across the system components. The Web services are supported in the following ways:

- The `create` operation: using the HTTP POST method to create new medical records such as: new ailment, new visit, newly assigned healthcare profes-sional, and new allergies
- The `read` operation: using the HTTP GET method to fetch existing medical records which are already created
- The `update` operation using the HTTP PUT method to modify existing records

The mAppGP framework does not support the `delete` operation because the EHR is not supposed to be deleted rather, the main HIS tracks all modifications through *audit trail*. So assuming an entry is erroneous, the error will just be corrected through an update operation. In the upcoming discussions, we focus on the various system components and our proposed methodologies that aid in addressing the research questions.

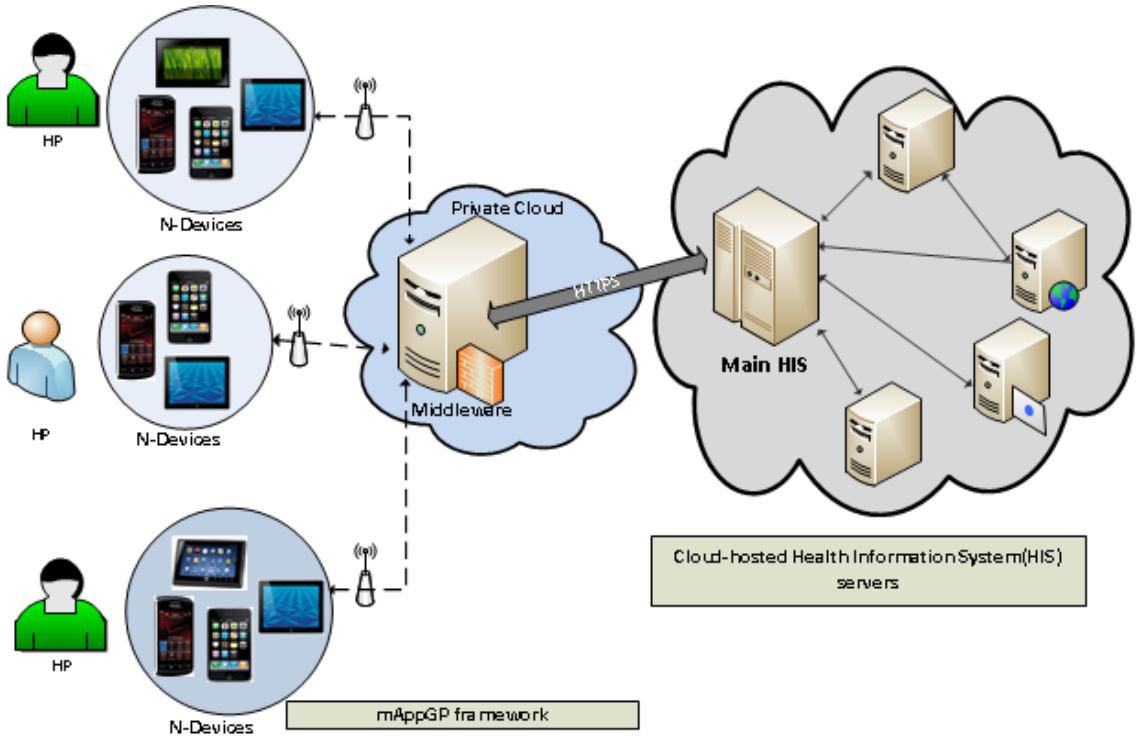

Figure 2: The mAppGP Architecture

6.1 The Existing Health Information System (HIS)

The medical units such as the Children ward, Emergency Unit, Laboratory, etc. all have their medical job execution workflow but the final processed job is stored on the centralized main HIS. The hospital has its own private cloud infrastructure where all the servers are running the high demanding medical process.

Also, the HIS acts a services brokerage platform that allows other units to access the medical data from other services publishers (or providers). For example, though the laboratory is a separate unit with their own management and are responsible for their medical data (e.g., blood test results), the HIS brokerage facilitates the physicians to access the data. The successful deployment of such a brokerage service is based on centralized security control and the fact that all the data from the entire units must meet the

services standardization of the HIS. The distributed medical services and job execution workflow are enabled to form a single workflow if the need arises. However, this is achieved by authenticating the users from a centralized source that ensures the credibility of the user. The benefit of a centralized security access also includes the enforcement of transparency and policy-based privacy.

Further, all communication with the main HIS is over HTTPS which is more secured than HTTP. All the data is in JSON [42] format. We found JSON as light-weight and easy to manipulate on the mobile side, middleware, and on the high capacity HIS clusters.

6.2 The Middleware Platform

The mAppGP application is a middleware-oriented application. Since the project is focusing on a distributed mobile network, we need to aggregate the distributed mobile components. The middleware is a server application that is built using the Erlang programming language that acts as a hub for all the requests from the healthcare professionals (HP as shown in Figure 2). The middleware is proposed to achieve the following functional requirements:

- *Shielding users from directly accessing medical data from the HIS*: This is because the middleware acts as the router for all the communication.
- *Maintaining services compliance and standardization*: All the data must be complaint with the hospital regulation so the middleware ensures the follow-ing standards are met: the data is JSON and the users are properly authenti-cated.
- *To enforce data transformation*: The raw data from the mobile end is in var-ying formats such as textual, video, and images so all the data is transformed to binary terms.
- *Enforcing security and privacy*: Providing security for the entire system. This requirement is detailed in the later part of the paper.

The process execution flow of the middleware is explained and graphically represented in the Figure 3.

The mobile requesters communicate with the middleware through the HTTP Interface. Every data is encrypted from the mobile-end before being transmitted over the network. The responses from the middleware are also encrypted so both the mobile nodes and the middleware performs data encryption and decryption tasks. When the middleware receives a request from the mobile, the data is first decrypted and then the user credentials are retrieved. The user credentials are the username and password pairs. The middleware uses the credentials to authenticate the user with the main HIS. The successful authentication attempt paves way for dual communication between the middleware and the main HIS. If the authentication process is failed, the request is not sent to the main HIS. In case of successful authentication, the application state on the mobile can be synchronized with the main HIS. The physicians can view, create, and update medical data by directly interacting with the HIS. In this work, two different data encryption and decryption algorithms are implemented from the Erlang crypto module [43] which we discuss below:

- ***AES in Cipher Feedback mode (CFB)***: This encryption algorithm requires the submission of three items: Key, IVec (i.e., initialization vector), and data. In the Erlang programming language, the code snippet for the encryption is written as:

  `Crypt = crypto:aes_cfb_128_encrypt(Key, IVec, Data)`, where *Crypt* is the encrypted text that is transmitted over the network.

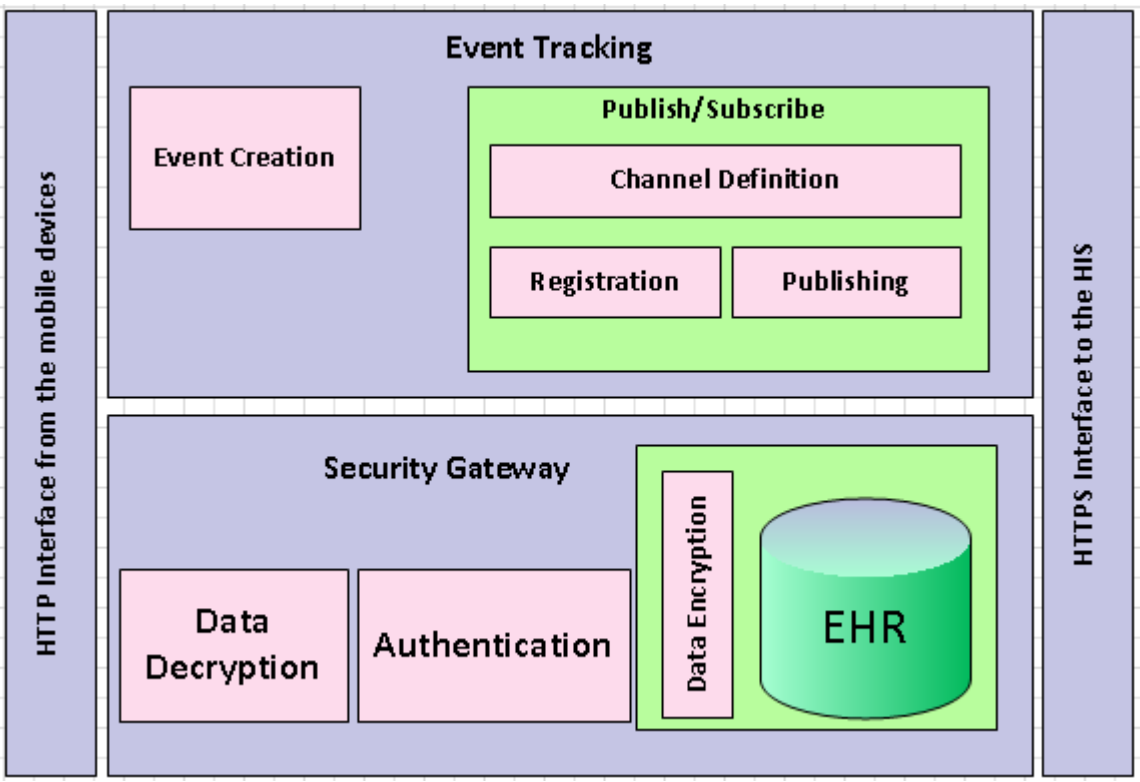

Figure 3: The macro components of the middleware

The decryption process is accomplished through the code snippet below:

```
Data = crypto:aes_cfb_128_decrypt(Key, IVec, Crypt),
```

This line of code will retrieve the encrypted code that is coming from the mobile end. In this regard, the Key and IVec are shared between the mobile and the middleware. For security reasons, we shall eliminate details of how the key is shared and character encoding details.

- ***DES in 8-bit CFB mode***: This encryption algorithm also uses three parameters: Key, initialization vector, and the data. The code snippet is shown below:

  ```
  Crypt = crypto:des_cfb_encrypt(Key, IVec, Data),
  ```

  The decryption can be achieved through:

  ```
  Decrypt = crypto:des_cfb_decrypt(Key, IVec, Crypt),
  ```

The second major function of the middleware is event-tracking. As already posited, we are looking into cases of intermittent disconnections which means, receiving responses is not always guaranteed for every request. Also, updates to medical data from other physicians may not be available to some intended physicians which can lead to undesired situations. Hence, the event-tracking module is designed to monitor all activities of the system. The physician can create an event which is for example, medical record that should be accessible by some other members of the medical team on their mobile devices. The middleware relies on the publish/subscribe methodology for the event dissemination. We have a channel definition module where the specific channels are created. The channels represent the specific information that a physician wants to share with others. In this case, the physician who is creating the channel is known as the provider or publisher. The intended physicians have to register to receive the

updates when they are published to a channel. The application on the mobile establishes a tcp socket connection with the middleware so we can determine nodes that are connected and disconnected. The channel information and updates are stored securely in a database on the middleware. The channel data is encrypted following the encryption algorithms above. When the disconnected mobile nodes re-connect, the middleware will push the updated data to the mobile.

6.3 The Mobile Devices

The user is expected to freely use n-devices to access the medical records as and when it is needed. The first challenge we face as the research developers is how to write the code for the heterogeneous platforms. Mobile devices such as smartphones and tablets have different underlying operating systems. Further, the native programming environments are different for the various devices. For example, iOS supports Objective C, Windows Phone supports C, BlackBerry supports Java, and the list goes on. The challenge here is that, we have to write independent codes for all of these platforms in order to achieve our goal. To further complicate the problem, if in the future there is the need to update/upgrade the application, we have to rewrite the code all over again in the various programming languages.

To overcome the problem with the heterogeneity, we decided to adapt the HTML5 environment for the mobile development. HTML5 enables the deployment of mobile web applications that can be deployed on multiple platforms. We developed the mobile side of the application in the Xcode environment for iOS and we used PhoneGap Build [44] to compile the code into different formats. This approach aided us to build a single code base that is deployed on multiple mobile platforms. We list the mobile platforms in the evaluation section. An interface of the application is shown in Figure 4.

In order to maintain a consistent look and feel across the different mobile displays, we used the JQuerymobile framework together with our own proprietary CSS to maintain styling and layout. The HTML5 component aided us to also have access to the core device features such as the video, camera, and audio. Further, we built a JavaScript library that links the user interface to the underlying layers. The mobile establishes connectivity using HTTP (ie., XMLHttpRequest), Ajax, and Socket over Wi-Fi. Further, we rely on the client-side publish/subscribe mechanism to push updates to the middleware as well as retrieving notification of updates to the medical data from the middleware.

Actually, what we have done is to build a Model-View-Presenter (MVP) architecture on the mobile. The view is the graphical user interface that is displayed for the user to interact with the system. Now is the time to mention what we have been silent on: the n-screen support. Supporting n-screen usage means that multiple mobile devices can be used by the same healthcare practitioners. This means that all the components on the mobile are device specific. Device specific features are not shared with other devices or application processes. So, when we want to support offline data accessibility by storing the application state (or the current state of the mobile health data), it only means that those application states are specific to each device. As already stated, mobile devices experience bandwidth fluctuation and sporadic disconnections so we build the mAppGP application to support usage in both offline and online modes. In the online mode, all communications go directly to the middleware which handles the user authentication procedures as explained in the previous section. But, in the offline mode, the centralized security mechanism becomes a challenge since the middleware server cannot be reached. Hence, we built a security layer on the mobile that stores the security tokens of the user. The user has to supply password and username pairs on the mobile before the medical data that is stored on the mobile can be accessed. In this regard, even if the device gets into wrong hands, the data is safe and privacy is preserved.

All the communication between the mobile system components are handled by the presenter. We treat the storage as the model in the architecture. The fact is that, when the device is in offline mode, all the data that is being created or modified are just on the mobile side and will be stored until connectivity

Figure 4: Sample interface of the application

is restored. In this disconnect state (due to partition-tolerance), we run clearly into the CAP theorem situation. The data on the mobile is inconsistent with the HIS while there is availability. So, we employ the eventual consistency model following the session consistency methodology. We treat the data collection period remotely as a session and the publish/subscribe component keep tracking the restoration of connectivity. We have heterogeneous server interfaces, so whichever is available becomes the channel for the mobile storage to synchronize with the main HIS through the middleware. The pub/sub is also an additional attempt to reduce the latency barrier in the mobile environment so that the medical records will be propagated in real time.

7 The Model-View-Controller Deployment

So far we have discussed the physical composition of the mAppGP framework as well as the functionalities and the development environments. However, it is important that we also explain the flow of the business logic. The entire mAppGP architecture follows the Model-View-Controller (MVC) methodology. The methodology is adopted to ensure clear separation of concern (loose coupling) between the system components.

We treat the view layer as the component that is deployed on the mobile devices as explained earlier. Primarily, all the activities on the mobile are designated to the view. Each view has a presenter that coordinates the mobile-specific tasks such as the management of the application state, mobile storage,

and client side security. The flow execution of the MVC is illustrated in Figure 5.

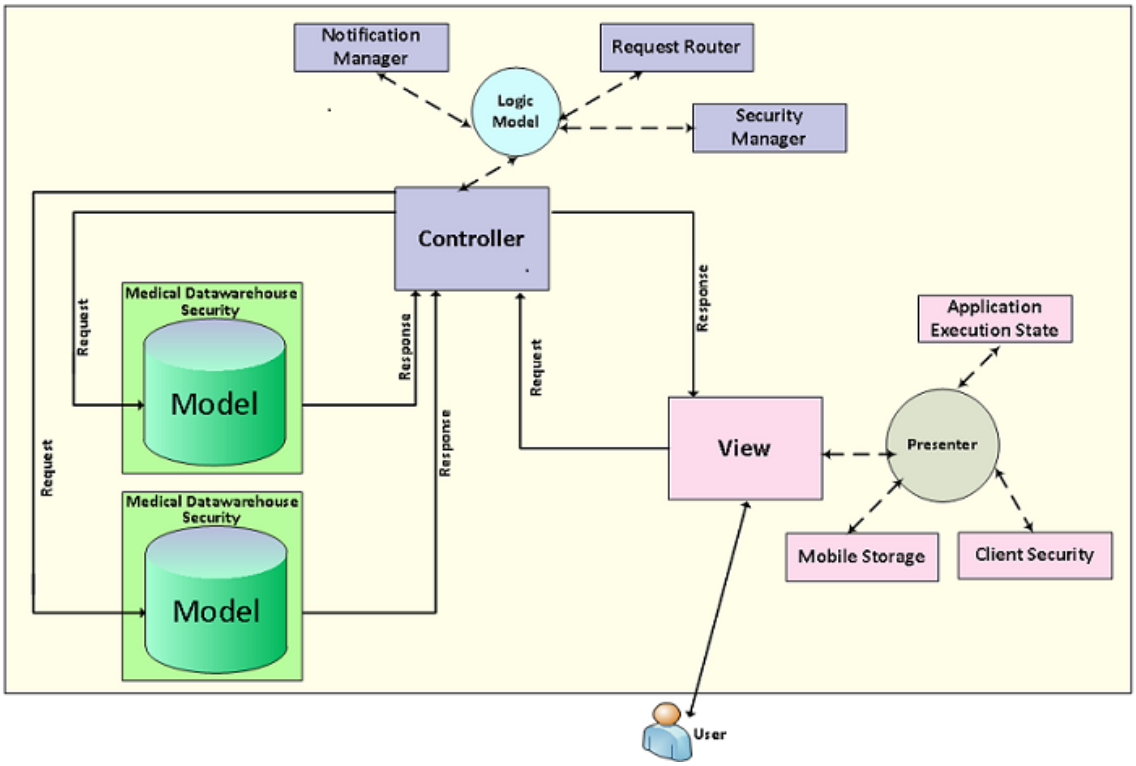

Figure 5: The MVC design

The controller is the server interface that the middleware exposes to all the mobile requesters. The controller facilitates the communication between the views and the model. The controller resides on the middleware and it has a logic model that per-forms tasks such as request routing, update notification management, and security compliance enforcement. The model is the storage area on the HIS that the data that is coming from the mobile sides can reside. The combination of MVC/P in the deployment of the mAppGP framework aided us to switch between offline and online modes. Also, it becomes easier to perform offline specific tasks using the MVP as well as online specific tasks using (MVC) without violating the standards.

8 Experiments

Currently, the mAppGP application is deployed on the following devices: BlackBerry PlayBook — OS: BB Tablet OS, Resolution: 1024x600, Processor: 1GHz, Number of Cores: Dual-Core, Storage: 16GB, RAM: 1GB, iPad 3 — OS: Apple iOS 5.1.0, Reso-lution: 2048x1536, Processor: A5X (dual-core, w/ quad-core graphics), Storage: 16GB, RAM: 1GB, NOKIA Lumia 900 — OS: Windows Phone OS, Resolution: 800x480, Processor: 1.4GHz, Number of Cores: Single-Core, Storage: 16GB, RAM: 512MB, Transformer Prime TF201 — OS: Android 4.0, Resolution: 1280x800, Processor: NVIDIA Tegra3, Number of Cores: Quad-core, Storage: 32GB, RAM: 1GB, Personal Computer (PC) — OS: Windows 7 System 32-bit, Processor: Intel Core i5, CPU 650 @ 3.20GHz 3.19GHz, Storage: 500GB, RAM: 4GB (2.99GB usable). The middleware is hosted on one of the private cloud servers of the City Hospital with the following specifications: OS: Windows System 64-bit OS, Processor: Intel Xeon, CPU E5410 @ 3.67GHz, 3.55GHz, RAM: 16 GB. The available Wi-Fi is 75Mbps Ethernet for the mobile

testing. The mobile storage capacity is evaluated to determine the maximum storage of medical records. The results show that we can store medical records up to 4.5GB on the BlackBerry Playbook, 5GB on the iPad3, 4.5GB on the Android Transformer Prime, and 2.6GB on the NOKIA Lumia 900.

8.1 Read and Write Request Monitoring

Hosting the mobile medical record requires some processing capacity. We investigate the impact on latency regarding the reading of medical records from the back-end as well as writing. From our current records, the medical history has the largest data size (approximately 80kb) and it is also the fastest growing component in size. Thus, we evaluate the request-response time for varying number of requests from 100 to 1000 for the medical history. The requests were issued sequentially in a closed loop using the asynchronous polling technique. So, we receive response for every preceding request before a subsequent request is issued. We conducted the experiment for all the devices that we put forward but for brevity, we report the result that we got using the iPad3 which is graphed in Figure 6.

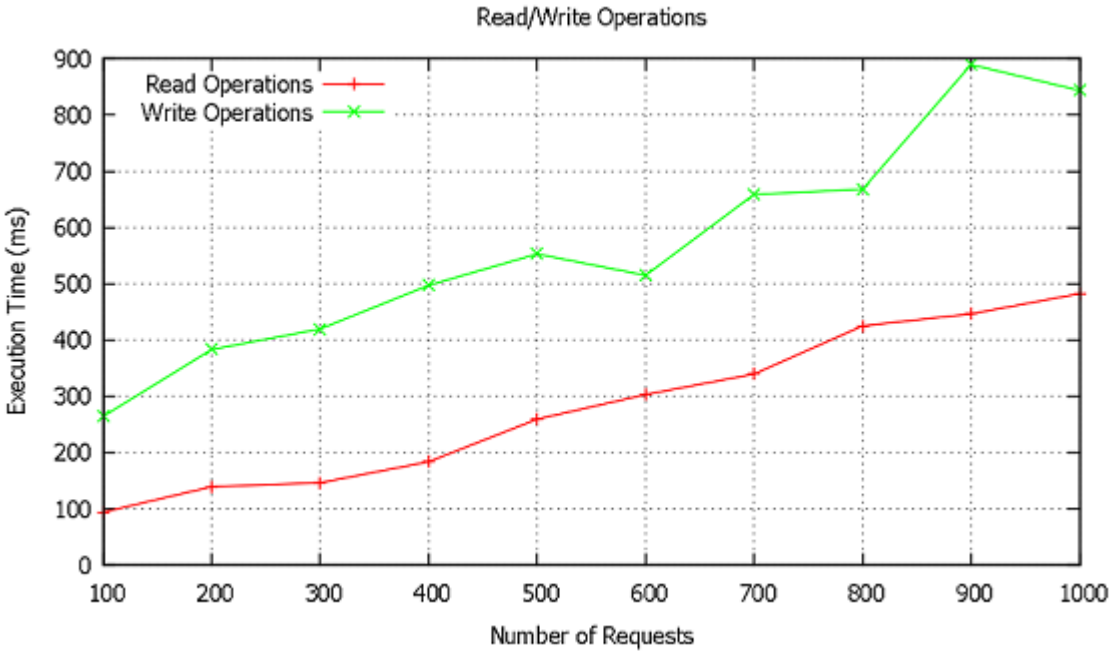

Figure 6: Read and Write Request Tracking

Regarding the read operation, the minimum request duration is 93.75ms for the 100 sequential requests and the maximum request time is 482.26ms for the 1000 sequential requests. Overall, the distribution is linear and we experience an error rate of zero percent. The write request on the other hand shows an increased duration which is typical of any information system. However, the degree of increase is tolerable. The minimum duration to complete the sequential write operation for 100 requests is 264.11ms and the maximum duration is 889.67ms. In the real world usage, not this many requests will be issued at the same time so we are comfortable with the results. Our aim is to determine the extremes so recording operational duration of less than a second is considered acceptable by the project team.

8.2 The Publish-Subscribe

Our next experiment focuses on the synchronization and update management within the multi-devices. The goal here is to invetigate the impact of the data transfer size on the update management. We tried to determine how the proposed publish-subscribe technique can aid with faster update propagation contrary to the othordox polling requests through the read operation. Considering different medical data sizes from 100MB to 1000MB, we graph the result in Figure 7 for all the devices that we put forward.

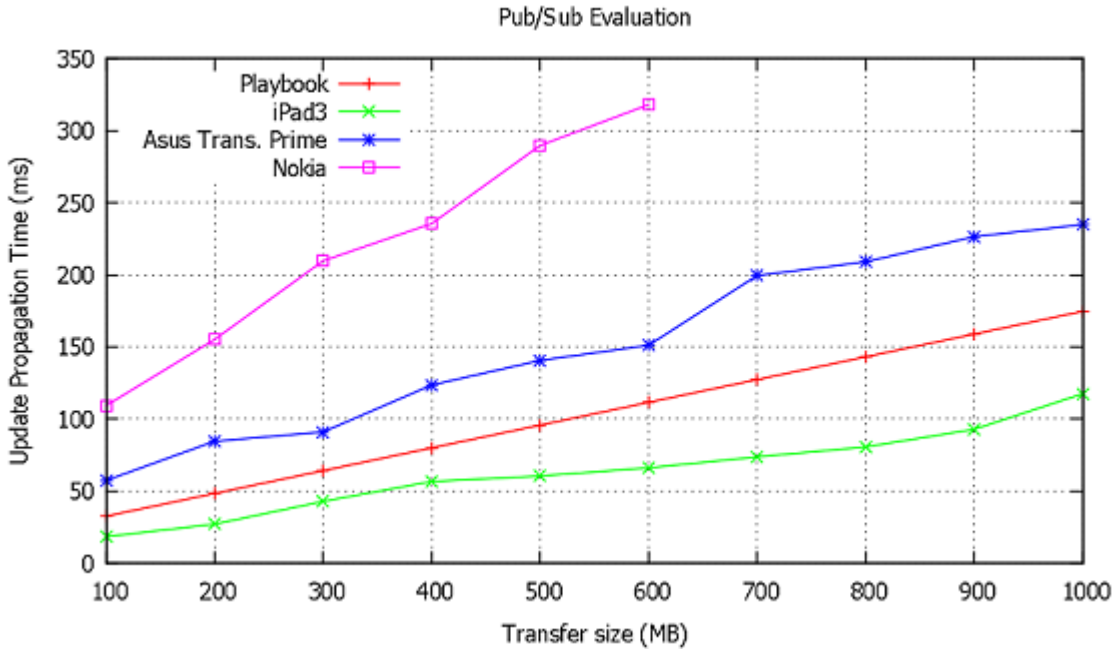

Figure 7: Publish-Subscribe Testing

With the publish-subscribe testing, we created channels and registered the multi-devices of the user for a single user. Then we created numerous medical records on the backend so that the middleware can push (publish) the newly created records to the mobile devices. The assumption made in the experiment is that, the devices are all connected so there is no need to queue the updates for later time before they are pushed to the devices. Hence, once the updates are published, they are pushed to the subscribers. The approach has shown significant improvement on minimizing latency in comparison to the read operation. The difference in update time can be visibly seen in Figure 6 and Figure 7. The error rate in the case of the publish-suscribe is also zero percent and in cases where updates are not immediately delivered to the specific device, the middleware will queue the data to be delivered later. Another observation that is worth noting is that, we receive errors for medical data transfer sizes beyond 600MB on the Windows Phone. We attribute the error scenario in this case to the device capacity limitations.

8.3 Data Encryption and Decryption Cost

Security is paramount as highlighted throughout this work. With the proposed encryption algorithms (i.e., DES and AES), we evalaute the cost on the mobile processing. We report the result in Figure 8 for the iPad3. All the medical records that is being transmitted from the middleware to the mobile is encrypted so the data needs decryption. Similarly, raw data is encrypted on the mobile before the data is transfered to the middleware. We considered a number of medical data sets from 2 to 100 and evaluate

the encryption and decryption cost. In the experiments, we report only the time duration to complete the encryption and decryption process without adding the latency or request-response time.

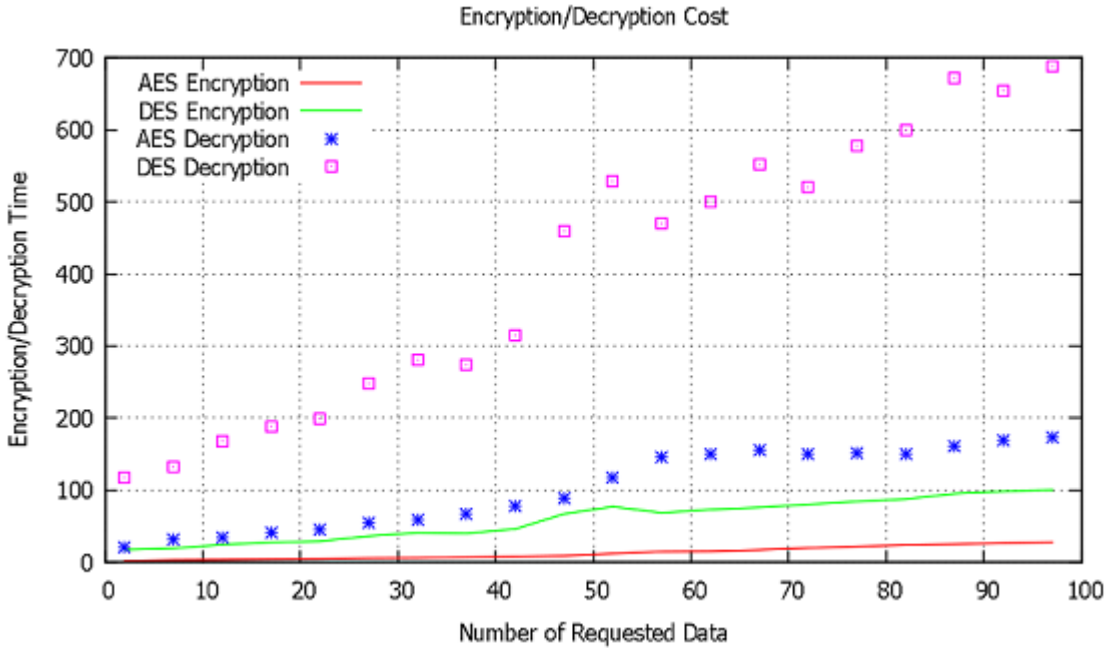

Figure 8: Medical Data Encryption and Decryption

Initially, the DES algorithm is considered and the encrption and decryption costs are measured. The approximate time for the encryption is 59.33ms for 97 records with a single record having approximate size of 47kb. When the same data is encrypted and sent from the middleware, it takes averagely 407.17ms for the data decryption. These rates for the encryption and decryption are alarming because we have not added the round-trip time yet which means the total request-response duration for an encrypted data to be decrypted can run into some intolerable seconds. Further, we are dealing with huge records in thousands so the approximate time for just 97 records reaching almost half a second to decrypt is not good for a mission critical application which requires speed. Another challenge presented by the DES algorithm is high processing cost which leads to high energy consumption on the mobile.

However, when we replicate the same experiment (i.e., the DES algorithm) with the AES algorithm, the result has improved greatly. Averagely, the 97 records are encrypted in 12.72ms while the decryption duration is approximately 102.23 ms. The better performance by the AES algorithm means it will be adopted at the expense of the DES algorithm. The only limitation we have at the moment is that, we are not able to get the AES working on the Windows Phone but we shall fix this limitation as soon as possible before the next release of the mAppGP application.

8.4 System Scalability

Finally, we evalaute the scalability of the middleware where we want to determine how the middleware will behave when the system users are increasing. In an mHealth environment, the capacity to process requests is crucial and especially in our case, where the middleware acts as a hub, scalability cannot be overlooked. The number of users involved in scalability testing is large so we simulate the activities of users from 1000 to 25000. A HTTP load generator is configured on a server (that is identical to the one hosting the middleware) which sends the concurrent HTTP requests to the middleware as actual mobile requesters; asking to perform different roles as supported by the middleware. A single request contains

the instructions to be executed which can be a read or write opeartion. Also, the experiment considers the ability of the middleware to encrypt and decrypt the medical data that is being trasnfered between the mobile participants and the main HIS. The result is recorded in Figure 9. The load generator sends requests following the exponential distribution of mean, 0.1 requests/second. It is observed that the error rate is zero percent for the concurrent requests and there was no corruption of data or message losses which shows the reliability of the middleware. The middleware further demonstrates high throughput support for increasing requests because as the concurrent users are increasing, the throughput is exponentially increasing accordingly. Averagely, the throughput is 1035.191196 requests/second.

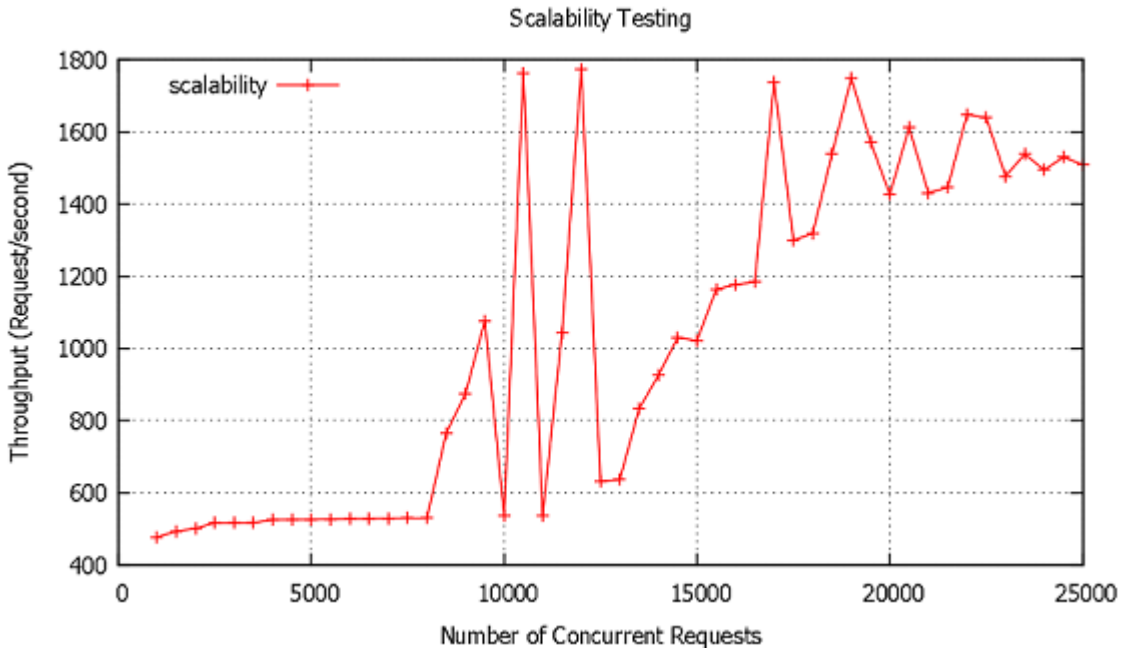

Figure 9: Scalaility of the middleware

The high concurrency support is due to the Erlang programming language which has no garbage collection penalties or shared memory corruption challenges. Furthermore, we have so many processes that we spawn when the load is increasing so that the workload can be distributed among the processes equally. The workload distibution is done for only the read requests which will not lead to inconstencies.

9 Conclusion

There is a growing need to support the mobile workforce within the medical domain to deliver healthcare remotely. What is even important is the fact that the ageing class of society will need medical attention outside of the hospital facility. In this regard, supporting the healthcare practitioners to employ their smartphone and tablet devices to interact with the Health Information System (HIS) in a centralized location is witnessing active research.

In this work, we presented a distributed mobile infrastructure called mHealth Application for Geriatrics Patients (*mAppGP*). The aim of our work is to support healthcare professionals to employ any of their n-devices to access the medical records on the HIS. To accomplish this task, we identified the presence of the `CAP theorem` as a fundermental issue to consider. Since partition-tolerance is a given in mobile distributed systems, we can only add one more guarantee out of data availability and data consistency. We opted for the availability option which we achieved through mobile-side data caching. In order

to minimize the inconsistency windows, other mainstream technologies such as the publish-subscribe are explored. A reasonable amount of time is also spent on security issues in order to maintain data privacy. The implementation of the framework comprises of the mobile requesters, middleware, and the HIS. The middleware is proposed as a router and hub for the mobile participants.

9.1 Future Work

The future extension on this work will consider services aggregation on the middleware-layer in order to further reduce the transfer cost between the HIS and the middleware.

9.2 Acknowledgments

- The Geriatrics Ward at the City Hospital in Saskatoon, Canada.
- The reviewers of the paper and the editors of JoWUA.

References

[1] J. Ranck, "Report: The rise of mobile health apps," GigaOM Pro, Tech. Rep., 2010.

[2] M. Rusu, G. Saplacan, G. Sebestyen, N. Todor, L. Krucz, and C. Lelutiu, "ehealth: Towards a healthcare service-oriented boundary-less infrastructure," *Original Research: Applied Medical Informatics*, vol. 27, no. 3, pp. 1–14, 2010.

[3] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Proc. of the 2006 Services/Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications (AICT-ICIW'06), Guadeloupe, French Caribbean*. IEEE, February 2006, pp. 120–125.

[4] F. Aijaz, S. Z. Ali, M. A. Chaudhary, and B. Walke, "Enabling high performance mobile web services provisioning," in *Proc. of the 70th Vehicular Technology Conference Fall (VTC 2009-Fall), Anchorage, Alaska, USA*. IEEE, September 2009, pp. 1–6.

[5] A. van Halteren and P. Pawar, "Mobile service platform: A middleware for nomadic mobile service provisioning," in *Proc. of the 2nd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2006), Montreal, Quebec, Canada*. IEEE, June 2006, pp. 292–299.

[6] A. Meads, A. Roughton, I. Warren, and T. Weerasinghe, "Mobile service provisioning middleware for multihomed devices," in *Proc. of the 5th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMOB'09), Marrakech, Morocco*. IEEE, June 2009, pp. 67–72.

[7] M. Hassan, W. Zhao, and J. Yang, "Provisioning web services from resource constrained mobile devices," in *Proc. of the 3rd IEEE International Conference on Cloud Computing (CLOUD'10), Miami, Florida, USA*. IEEE, July 2010, pp. 490–497.

[8] D. Konstantas, V. Jones, and R. Herzog, "Mobihealth - innovative 2.5 / 3g mobile services and applications for healthcare," in *Proc. of the 11th Information Society Technologies (IST) Mobile and Wireless Telecommunications, Thessaloniki, Greece*, June 2002.

[9] R. K. Lomotey, S. Jamal, and R. Deters, "Sophra: A mobile web services hosting infrastructure in mhealth," in *Proc. of the 1st IEEE International Conference on Mobile Services (MS'12), Honolulu, Hawaii, USA*. IEEE, June 2012, pp. 88–95.

[10] S. Canada, "Generations in canada," in *Statistics Canada*, 2011, http://www12.statcan.gc.ca/census-recensement/index-eng.cfm.

[11] J. Bernard, "Baby boomers retiring to rural areas struggle to find doctors," The Seattle Times, September 2012, http://seattletimes.com/html/nationworld/2019048802_ruraldoctors02.html.

[12] E. Brewer, "Towards robust distributed systems," Invited Talk, Principles of Distributed Computing, Portland, Oregon, USA, July 2000, http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf.

[13] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web service," *ACM Special Interest Group on Algorithms and Computation Theory (SIGACT) News*, vol. 33, no. 2, pp. 51–59, June 2002.

[14] R. K. Lomotey and R. Deters, "Supporting n-screen medical data access in mhealth," in *Proc. of the IEEE International Conference on Healthcare Informatics 2013 (ICHI'13), Philadelphia, PA, USA*. IEEE, September 2013.

[15] ——, "Reliable consumption of web services in a mobile-cloud ecosystem using rest," in *Proc. of the 7th IEEE International Symposium on Service Oriented System Engineering (SOSE'13), San Francisco, California, USA*. IEEE, March 2013, pp. 13–24.

[16] R. K. Tharakan, "Brewers CAP theorem on distributed systems," February 2010, http://www.royans.net/arch/brewers-cap-theorem-on-distributed-systems/.

[17] D. Pritchett, "Base: An acid alternative," *ACM Queue*, vol. 6, no. 3, pp. 48–55, May 2008.

[18] C. Young, "Biztalk server 2006: The compensation model, sagas," 2006, http://geekswithblogs.net/cyoung/articles/100424.aspx.

[19] Z. Kessin, *Building Web Applications with Erlang*. O'Reilly Media, Inc., 2012.

[20] O. T. Network, "20. Read-Your-Writes Consistency: Getting Started with Replicated Berkeley DB Applications," June 2011, http://download.oracle.com/docs/cd/E17076-02/html/gsg-db-rep/C/rywc.html.

[21] W. Vogels, "Scalable web services: Eventually consistent," *ACM Queue*, vol. 6, no. 6, pp. 14–19, October 2008.

[22] C. Notes, "Consistency model - a survey: Part i - what's data consistency model and why should we care?" 2009, http://xcybercloud.blogspot.com/2009/05/data-consistency-model-survey.html.

[23] M. Research, "DBMS2: Read-your-writes (RYW), aka immediate, consistency," May 2010, http://www.dbms2.com/2010/05/01/ryw-read-your-writes-consistency/.

[24] J. Browne, "Brewer's CAP theorem," January 2009, http://www.julianbrowne.com/article/viewer/brewers-cap-theorem.

[25] G. Eysenbach, "What is e-health?" *Journal of Medical Internet Research*, vol. 3, no. 2, October 2001.

[26] V. W. Consulting, "mHealth for Development: The opportunity of mobile technology for healthcare in the developing world," The United Nations Foundation and Vodafone Foundation Technology Partnership, February 2009, http://www.globalproblems-globalsolutions-files.org/unf_website/assets/publications/technology/mhealth/mHealth_for_Development_full.pdf.

[27] A. Minutolo, G. Sannino, M. Esposito, and D. P. Giuseppe, "A rule-based mhealth system for cardiac monitoring," in *Proc. of the IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES'10), Kuala Lumpur, Malaysia*. IEEE, November-December 2010, pp. 144–149.

[28] Z. Ji, X. Zhang, I. Ganchev, and M. O'Droma, "A content adaptation middleware for use in a mhealth system," in *Proc. of the 14th International Conference on e-Health Networking, Applications and Services (Healthcom'12), Beijing, China*. IEEE, October 2012, pp. 455–457.

[29] ——, "A personalized middleware for ubiquitous mhealth services," in *Proc. of the 14th International Conference on e-Health Networking, Applications and Services (Healthcom'12), Beijing, China*. IEEE, October 2012, pp. 474–476.

[30] L. Huang, Y. Xu, X. Chen, H. Li, and Y. Wu, "Design and implementation of location based mobile health system," in *Proc. of the 4th International Conference on Computational and Information Sciences (ICCIS'12), Chongqing, China*. IEEE, August 2012, pp. 919–922.

[31] Cerner, "Cerner," http://www.cerner.com/solutions/Physician-Practice/Hardware-and-Technologies/Mobility/.

[32] Qualcomm, "Qualcomm," http://www.qualcomm.com/solutions/healthcare.

[33] Dell, "Dell mobile clinical computing," http://www.dell.com/Learn/us/en/70/healthcare-mobile-clinical-computingc=usl=ens=hea.

[34] G. Wicks, E. V. Aerschot, O. Badreddin, K. Kubein, K. Lo, and D. Steele, *Powering SOA Solutions with IMS*. IBM Redbooks, 2009.

[35] E. Inc, "Soa disadvantages," SOA Tutorial, http://www.exforsys.com/tutorials/soa/soa-disadvantages.html.

[36] L. WonSeok, L. C. Min, L. J. Won, and S. Jin-Soo, "Roa based web service provisioning methodology for telco and its implementation," in *Proc. of the 12th Asia-Pacific network operations and management conference on Management enabling the future internet for changing business and new computing services (APNOMS'09), Jeju, Korea, LNCS*, vol. 5787. Springer-Verlag, September 2009, pp. 511–514.

[37] A. Saenz, "Cardionet, concept = win, strategy = epic fail," August 2010, http://singularityhub.com/2010/08/31/cardionet-concept-win-strategy-epic-fail/.

[38] CARDIONET, "CARDIONET," http://www.cardionet.com/.

[39] H. Overdick, "The resource-oriented architecture," in *Proc. of the 2007 IEEE Congress on Services (SERVICES'07), Salt Lake City, Utah, USA*. IEEE, July 2007, pp. 340–347.

[40] X. Xu, L. Zhu, Y. Liu, , and S. Mark, "Resource-oriented architecture for business processes," in *Proc. of the 15th Asia-Pacific Software Engineering Conference (APSEC'08), Beijing, China*. IEEE, December 2008, pp. 395–402.

[41] S. Petri, B. Petros, and Y. Yu, "Experiences in building a restful mixed reality web service platform," in *Proc. of the 10th International Conference on Web Engineering, (ICWE'10), Vienna, Austria, LNCS*, vol. 6189. Springer-Verlag, July 2010, pp. 400–414.

[42] JSON, "JSON schema," http://json-schema.org/.

[43] E. P. Language, "crypto," http://erlang.org/doc/man/crypto.html.

[44] P. Build, "Phonegap build," https://build.phonegap.com/.

Richard K. Lomotey is currently pursuing his PhD in Computer Science at the University of Saskatchewan, Canada, under the supervision of Dr. Ralph Deters where his main work focuses on mobile cloud computing. He has been actively researching topics relating to: ubiquitous cloud computing and the paradigm shift in enterprise mobility workforce support, real-time mobile-cloud services delivery for business continuity, services composition and provenance for reliable mobile consumption and provisioning, and steps towards faster mobile query and search in today's vast data economy (big data). Richard is also passionate about providing solutions to the challenges pertaining to the deployment of mobile enterprise systems and has been exploring architectural designs that can enhance System Performance, Scalability and Reliability of Heterogeneous Web Services, Fault Tolerance and Resilient Systems, and Distributed NoSQL Databases queries. He holds MSc Computer Science from the University of Saskatchewan and BSc Computer Science from the University of Cape Coast, Ghana. As a recipient of the Department of Computer Science Scholarship at the University of Saskatchewan, Faculty Scholarship, and MITACS Internship Funding, Richard has worked with ZenFri Inc, SAKINA Information Sciences, and MarkeTel.

Ralph Deters obtained his Ph.D. in Computer Science (1998) from the Federal Armed Forces University (Munich). He is currently a professor in the department of Computer Science at the university of Saskatchewan (Canada). His research focusses on mobile and cloud computing.