Synchronization of Medical Data in a Mobile Provisioning Environment: mHealth Use Case

Richard K. Lomotey[*1] and Ralph Deters[2]

Department of Computer Science, University of Saskatchewan S7N 5C9, Saskatoon, Canada

[*1]richard.lomotey@usask.ca; [2]deters@cs.usask.ca

***Abstract*-The use of mobile devices such as smartphones and tablets, as well as other Information and Communications Technology (ICT) tools, to facilitate healthcare delivery in the medical landscape, known as mHealth, has risen phenomenally. In most mHealth systems, mobile devices are employed as services and health information client consumers. Meaning, physicians use these devices to access the Electronic Health Record (EHR) which resides on back-end platforms. However, in a research collaboration with the Geriatrics Ward of the City Hospital in Saskatoon, Canada, we have identified a huge potential for facilitating the mobile device as a medical data hosting node. Our approach is beneficial for collecting the EHR remotely and pushing it to the Health Information System (HIS). For instance, Geriatrics' patients who are home can be attended to outside of the health facility since their medical data is being retrieved from or pushed to the HIS remotely. However, mobile devices communicate over wireless mediums, which can experience intermittent loss of connectivity. It is also possible to visit the patient's home, and there not be any connectivity. These situations can cause an inconsistency between the data that is on the physician's mobile device and the data on the HIS. Our work therefore proposes a cloud-powered middleware architecture that facilitates the synchronization of the data between the mobile devices and the HIS in soft real-time. The middleware employs the propagation of deltas and timestamp to determine which of the data is the newest update in any direction and ensures that all the operations are in sync. The evaluation of our proposed system shows minimal latency.**

Keywords- REST; Mobile Cloud Computing; Middleware; Healthcare Information Systems; Resources State Change; Mobile Hosting/Provisioning

I. INTRODUCTION

Mobile devices such as smartphones and tablet devices are gaining popularity in the enterprise domain [1]. Less than a decade ago, these devices were just tools for exchanging data in the form of voice and simple SMS messages. Today, their productivity value (i.e., mobile commerce) is increasing, which is promising [2]. Furthermore, most of today's smartphones and tablet devices that are in production have good backings for various networks and protocols. Thus, connectivity can be established between mobile participants via 4G, Wi-Fi, and Bluetooth [4].

As the commerce field is witnessing a mobile boom, other non-commercial enterprises such as mHealth are experiencing equal growth in research conducted on the use of mobile devices and other ICT tools for better healthcare delivery [20, 21]. Relentless efforts have been made by researchers to utilize the potential provided by the ever-evolving Web in order to deliver mobile device services that offer better healthcare [21]. However, most of these researches focus on the mobile devices as client consumers, especially in cases where medical information is modeled as heterogeneous Web services.

In an ongoing research collaboration with the Geriatrics Ward of the City Hospital in Saskatoon, Canada, we have identified a huge potential for facilitating the mobile device as a host of medical information. Mobile hosting is a new research trend where the mobile device is modeled as a server or provider of Web services [4, 5, 7, 8]. Srirama et al. [5] noted that an advantage of mobile hosting is turning the mobile host into a multi-user node. From the view point of our medical partners, the mobile Web services hosting has improved upon the management and accessibility of medical records of their "aged" patients.

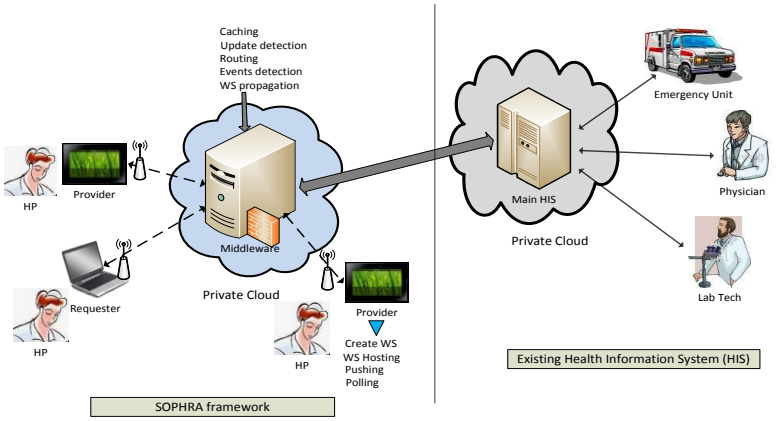

Fig. 1 The SOPHRA architecture [3]

DOI:10.5963/BER0301001

However, mobile devices primarily communicate via wireless channels that can be unstable due to user mobility and bandwidth fluctuation. In our previous research, which was reported in [3], we addressed the challenge of how to propagate messages reliably and in real-time to the healthcare mobile participants. The SOPHRA[1] architecture was proposed, which is a distributed mobile framework that aids in the medical services integration and messaging routing. The architecture is reproduced in Figure 1. The designed architecture achieved its goal through the deployment of a cloud-centric middleware platform that aids the healthcare professionals in securely sharing and accessing their mobile hosted health records reliably and in real time. The health records were modeled as Web Services (WS) which can be propagated between the system components. These WS, which are hosted on the mobile devices, are also replicated independently on the middleware to ensure high data and services availability. The SOPHRA framework supports both SOAP and RESTful Web Services protocols and currently facilitates message exchanges over Wi-Fi using HTTP.

However, a crucial concern that was understudied in the SOPHRA architecture is the data inconsistency that can arise as a result of the intermittent loss in connection [3]. The hosted medical data on the mobile can become outdated when updates are not timely and efficiently managed. This can lead to making wrong decisions based on the inconsistent (or outdated) information by the physician. This case can be further complicated if multiple users (i.e., physicians) are facilitated to make concurrent changes to the Electronic Health Record (EHR). The question that arises is, how to determine which of the changes are the latest that require propagation?

In this work, we have expanded on the SOPHRA architecture where we seek to address the challenges of medical data synchronization. We proposed a policy-based weak consistency model that allows the medical data to be synchronized within some time window between the mobile participants and the Health Information System (HIS). The data is not just synchronized, but it also follows a policy which prioritizes the data based on ranking. Assuming a particular data (say, deteriorating health condition) of a patient is recorded, that will have a higher priority and ranking over another update (say, list of vendors selling some particular drugs on discount). Hence, the highly prioritized update will be synchronized first in the system before the less prioritized. We have also proposed the delta-approach to aid the medical data synchronization process where only the changes to the data are propagated but not the entire data set.

The remainder of this paper is organized as follows: Section II reviews some works on Web services and data synchronization approaches in distributed systems. Section III details our proposed architecture and design choice justifications. An evaluation of the architecture is carried out in Section IV, and the paper concludes in Section V with our contribution and future outlook. However, a crucial concern that was understudied in the SOPHRA architecture is the data inconsistency that can arise as a result of the intermittent loss in connection [3]. The hosted medical data on the mobile can become outdated when updates are not timely and efficiently managed. This can lead to making wrong decisions based on the inconsistent (or outdated) information by the physician. This case can be further complicated if multiple users (i.e., physicians) are facilitated to make concurrent changes to the Electronic Health Record (EHR). The question that arises is, how to determine which of the changes are the latest that require propagation?

In this work, we have expanded on the SOPHRA architecture where we seek to address the challenges of medical data synchronization. We proposed a policy-based weak consistency model that allows the medical data to be synchronized within some time window between the mobile participants and the Health Information System (HIS). The data is not just synchronized, but it also follows a policy which prioritizes the data based on ranking. Assuming a particular data (say, deteriorating health condition) of a patient is recorded, that will have a higher priority and ranking over another update (say, list of vendors selling some particular drugs on discount). Hence, the highly prioritized update will be synchronized first in the system before the less prioritized. We have also proposed the delta-approach to aid the medical data synchronization process where only the changes to the data are propagated but not the entire data set.

The remainder of this paper is organized as follows: Section II reviews some works on Web services and data synchronization approaches in distributed systems. Section III details our proposed architecture and design choice justifications. An evaluation of the architecture is carried out in Section IV, and the paper concludes in Section V with our contribution and future outlook.

II. BACKGROUND WORKS

A. WS and REST*

Web Services (WS) [7, 13] are network oriented applications that serve information based on standards such as SOA and REST. Furthermore, Web services can be used to deploy business services and applications on many platforms since WS are mostly XML based and use HTTP as a communication protocol [11, 15]. Recent studies show that deploying web services in a cloud computing environment can guarantee scalability and good system performance [6, 10, 15]. Pautasso et al. [13] compared WS*, which they described as "Big" Web Services (or SOA framework), with RESTful Web Services. Their paper concluded that the best design architecture depends on the needs of the developer since the two standards have different

[1] This is not an acronym but a name decided by the project committee

DOI:10.5963/BER0301001

architectural design principles. In another finding, Beal [14] in his article, "Understanding Web Services," reports that Web services have shaped the paradigm of business communication between client nodes and servers. The Service-Oriented Architecture (SOA) framework provides support to various components of Web services to interoperate. According to Wicks et al. [11], SOA focuses on the reusability of software and integration. Additionally, SOA supports packaging, which makes changing older versions of software rapid and of minimal cost [11].

Apart from the SOA approach, other researches have explored the REST approach. *REpresentational State Transfer* (REST) is an architectural principle that uses the Web platform for distributed computing [9, 13, 15]. REST is better understood in the context of identifying everything as a resource, representation, and state. The design employs the HTTP methods, such as POST, to create a resource, and GET to retrieve a resource.

B. Mobile Provisioning of Web Services

The paradigm of mobile provisioning has received only a little research attention. Srirama et al. [7] are some of the innovative researchers that work on facilitating the mobile device as a provider of Web services. In their initial studies [7], they demonstrated the feasibility of hosting Web services on the mobile device by adopting the SOAP messaging approach. The paper resolved the IP address challenges in mobile peer-to-peer communications by adopting two approaches: High-Speed Circuit Switched Data (HSCSD) dial-up connection and General Packet Radio Services (GPRS) environments. Later, the authors extended their work in [5] to facilitate the mobile provider to handle SOAP requests over HTTP. In addition, the mobile provider was enabled to handle concurrent requests and support runtime services deployment.

Also, Meads et al. [4] employ the cloud based middleware technique to provide a communication interface for ubiquitous devices to communicate with mobile providers in heterogeneous networks. The paper concludes that mobile providers can be reached via Bluetooth or Wi-Fi, an approach that gives requesters the flexibility to explore different communication channels. Furthermore, Hassan et al [8] produced research on managing the limited resources of the mobile provider and proposed a mobile web service partitioning scheme. As a result, complex business processes can be executed by the mobile provider with a backend super computer doing most of the high demanding computations. Additionally, Aijaz et al. [19] demonstrated the high performance of RESTful mobile Web services provisioning compared to SOAP Web services.

C. Data Synchronization

In highly distributed systems, such as enterprise information systems with cloud back-ends, the major concern is data state (update) propagation and management [12, 16]. Update management and synchronization of data is crucial for business continuity, real-time and accurate decision making, and so on. Services synchronization, which aims at ensuring consistency on every node in the distributed system, follows the following processes: database or data source access, data capture or retrieval, data conversion or transformation, and data transmission and security verification [16]. The mode of data exchanges during the synchronization process can be asynchronous (where one party sends a request to the other party and, without receiving a response, sends other requests) or synchronous (where a response has to be received for every request before another request can be made) [17].

Deploying middleware frameworks can advance the success of state management and synchronization from many sources. Lv and Zheng [16] put forward a middleware layer that takes files from multiple sources and applies a rule-based policy that combines those files into a single database. Rules are defined to capture changes that are occurring from different sources, and later, the changes can be synchronized in a single repository (specifically, XML flat file storage). Yang [18] also shares a similar opinion by proposing a middleware layer (the author called it a mediator) that performs mediation duties between front-end applications and back-end services. However, data synchronization issues in mobile networks are more challenging due to the limitations imposed on bandwidths in wireless networks and the intermittent loss in connectivity. These challenges lead to high communication latency during update managements, and sometimes updates are unsuccessfully propagated when the mobile is in a disconnected state [22, 23].

Latency reduction is a very important aspect of mobile services design. For example, in mobile multi-player game systems, latency can lead to inconsistency in the game state [24]; while, in mission critical systems such as mHealth [3], data propagation delays can lead to undesired circumstances. The concern for latency reduction and real-time data update has been our research focus over the past year. We have proposed a real-time data propagation middleware that aids in data routing among mobile providers and consumers using the health domain as a use case [3, 25, 26]. There are a few other works that also employed middleware to support end-to-end mobile communication in the health domain. For instance, Arunachalan and Light [27] proposed a mobile agent communication protocol that transmits medical data from the back-end server to the mobile clients. The agent communication protocol interfaces a middleware, which performs roles such as: data synchronization, data segregation, data distribution, power management, and geo-location detection. Further, the MUHIS (Middleware for Ubiquitous Healthcare Information System) framework, which is designed by Sain et al. [28], provides group collaboration and synchronization of data in the distributed medical domain.

D. Research Goal

In order to further clarify the research goal to the reader, let us consider the scenario below:

DOI:10.5963/BER0301001

The Geriatrics' patients are mostly the "aged" in society, and they receive medical attention either at home or at the health facility. With the power of mobile computing, the healthcare practitioners can visit the patients at their homes, examine them, record their medical information on a progress report, give medication, and send the report to the hospital facility. The mobile sends the medical data over wireless connections, but there is a chance that there could be a connectivity loss during the healthcare practitioner's visit. This can hamper the transmission of the medical data to the health information system. Assuming that later in the day another healthcare practitioner visits the same patient and tries to retrieve the medical data of the patient, the medical data will not reflect the immediate data collected by the previous healthcare practitioner since the data has not been synchronized yet (or on time). This can cause the second physician to give medication or recommendations based on outdated data.

In this work, our primary goal is to research how medical data synchronization can be achieved in mobile networks with great efficiency. By this, we aim at minimizing the latency between update propagation time as well as ensuring that the synchronization follows procedures that ensure reliability and transparency. This leads to the following research questions:

i. How do we accommodate the medical data on the mobile for efficient access?

ii. How do we synchronize the medical data in real time?

iii. Which data consistency model will best work for the medical domain?

iv. How do we ensure protection of the medical data?

The first three questions will be explored in the details of this paper, while the fourth question will be detailed in another research. We shall adapt some of the methodologies, which we reviewed to answer some of the questions. Especially, we shall employ the mobile hosting technique, cloud-hosted middleware methodology, and the services design.

III. THE PROPOSED ARCHITECTURE

To address the aforementioned questions, we proposed a mobile hosting environment that enables the healthcare practitioners to exchange the Electronic Health Record (EHR) reliably. The proposed architecture, which expands on the SOPHRA [3], as illustrated in Figure 2. The healthcare practitioners (labeled HP) can exchange the EHR between themselves as well as across the middleware. The architecture consists of three layers, namely: the mobile hosts, the middleware, and the main Health Information System (HIS).

The middleware and the main HIS are hosted on the cloud computing facility that is privately owned by the Hospital. As depicted in the diagram, the healthcare practitioners can send and receive updates related to health records remotely, even from patients' homes. And the healthcare workers at the hospital facility can also use the app on their devices as they move about visiting patients in the Ward.

The middleware is integrated into the HIS, and they both exchange the medical data which we model as JSON following the REST standard. The HIS also serve as the main datacenter where all the other medical units within the Hospital store their data. In this work, however, we shall overlook the details of the HIS since it is outside our scope. We shall, however, discuss the composition of the middleware and the design of the mobile hosts.

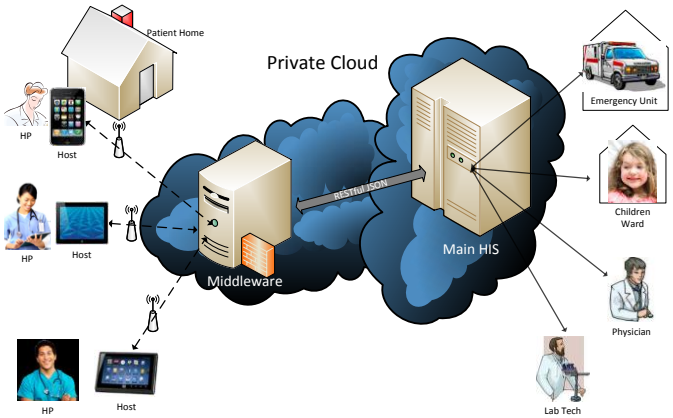

Fig. 2 The generic architecture

A. *The Mobile Hosts*

In the design of most mobile applications, caching has been the conventional approach to store data on the mobile for offline use. This means the cache has to be updated anytime the mobile connects to the backend facility. In this approach, the mobile is a primary data consumer, and the update transfer depends upon changing the entire dataset whenever updates happen. However, in the requirements for remote services accessibility and delivery, there is the need to support constant data update

DOI:10.5963/BER0301001

from the healthcare practitioners in both the online and offline modes. This means, the mobile node will have to be facilitated to support user interactivity at most times, which makes the mobile provisioning (hosting) technique a preferred methodology over the traditional consumption-only technique.

Hence, we designed the mobile node following the medical data hosting and provisioning approach in order to answer research question 1. This means that the mobile can function as an application server, and this can facilitate the healthcare professionals to work constantly, including the capturing of new records, reading existing medical records, and updating existing records. The medical data in this work involves patient-specific information, such as demographics, allergies, vitals, visits, medications, etc. The architectural design of the mobile host is illustrated in Figure 3. The mobile host has two layers, the *Application Layer* and the *Server Layer*.

The server layer is the engine that causes the mobile to behave as a server. The layer is divided into two major components that are responsible for incoming requests and outgoing requests. The server layer listens to incoming requests over a *port* just as any application server. This port is the entry point for all incoming requests and transactions. Once the request is received, the *Processing Logic* component is activated, which determines the nature of the incoming request. There are three major operations that can be supported, which are the CRU (i.e., the Create, Read, Update) operations. The delete operation is not supported. Depending on which operation is required (which can be determined based on the HTTP method), the processing logic takes the appropriate action. HTTP GET is a read operations, the HTTP POST method supports the create operation, and HTTP PUT supports the update operation.

The *Data Formatter* ensures that the incoming data follows some specific structure, which is JSON. If this is not the case, the data formatter will transform the data into the supported JSON format. Then, the data is then parsed to the *Data Transformer* component that ensures that the data is transformed into a Web service. This means that we are able to assign meta-data to the data, including identifier and E-tag. The latter attribute (E-tag) is essential to determine whether changes have been applied to the data. Any changes to the data will cause the E-tag to change. Thus, when we compare the E-tag of the value on the HIS to the version on the mobile, we are able to know if there is an update when there is an inconsistency in the E-tag values. There is also a *Data Layer* component that is responsible for ensuring that duplicate web services are not created. Meaning, the data layer ensures that no two web services have the same identifier, which would violate the REST principle. For instance, the data layer ensures that two patients will not have the same identification number since that can lead to conflicts, so, if an identifier exists, it can only be updated but not create a duplicate. This layer uses system lock to ensure that the changes will not happen that will cause issues with identity.

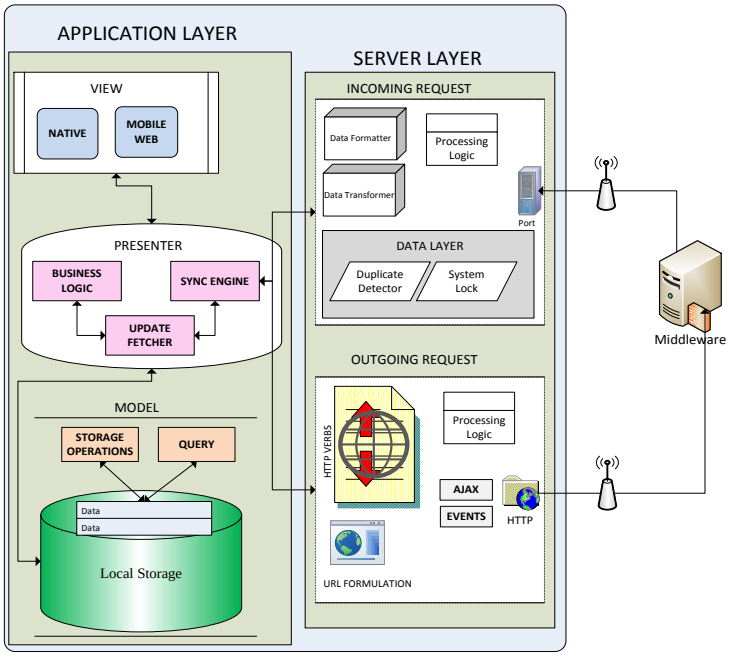

Fig. 3 The mobile host architecture

The second part of the server layer is the outgoing request engine. This is where the mobile processes the requests that are being sent to the middleware and subsequently to the HIS. The operations of the healthcare professionals must also be supported so that they are able to send updates to the HIS and subsequently sync the data states. In this case, the *processing logic* receives what operation the user wants to take and assigns the most appropriate *HTTP Verb*. Here, the requests that are being sent are also constructed into a valid URL, an operation that is handled by the *URL Formulation* engine. This request is issued over HTTP to the middleware by using the AJAX protocol. Also, the mobile can just send events, such as a notification of available connectivity, to the middleware.

DOI:10.5963/BER0301001

The next layer of the mobile host is the application layer, which is designed following the *Model-View-Presenter* approach. The *View* is the user interface that allows the healthcare practitioners to interact with the system. This view can be designed following both the *Native* and *Mobile Web* approaches. The latter approach requires that the development is done in a platform-supported specific environment such as Java for Android device, C# for Windows Phone, Object-C for iOS, and so on. The mobile web approach involves the use of web technologies such as HTML5, CSS, and JavaScript. The web ensures single code deployment on multiple platforms. An example of a user interface of the app is shown in Figure 4.

The *Presenter* coordinates the activities of the view and the model. The presenter has a *Business Logic* unit that responds to the actions of the user. This unit ensures that the user gets the appropriate responses for every action taken. The presenter also has a unit dubbed *Update Fetcher* that ensures that we receive only the changes that are made to a particular record rather than the entire data set. The *Sync Engine* ensures that the updates coming from the middleware are pushed to the model, and the updates by the user are pushed out.

The *Model* is the main storage area on the mobile. Both the server layer and the application layer share the model. The model contains a local storage and supports the several queries. The local storage also stores the medical data as web services, and the data can be updated, read, and new records can be created.

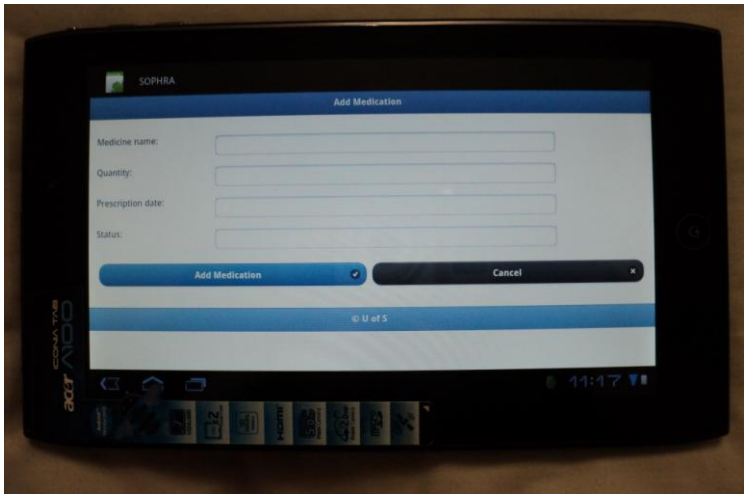

Fig. 4 A sample user interface of the app.

B. *The Middleware*

The middleware offers a Software-as-a-Service (SaaS) capability that aids the transactions that are to be processed on behalf of the mobile and the HIS. The middleware shields the HIS from the remote users of the system. This is an added advantage for the medical data protection at the back-end. The mobile establishes communication with the middleware over HTTP in a Wi-Fi or 3.5/4G environment. The middleware has the following components (illustrated in Figure 5), which have been designed to strategically answer our research questions:

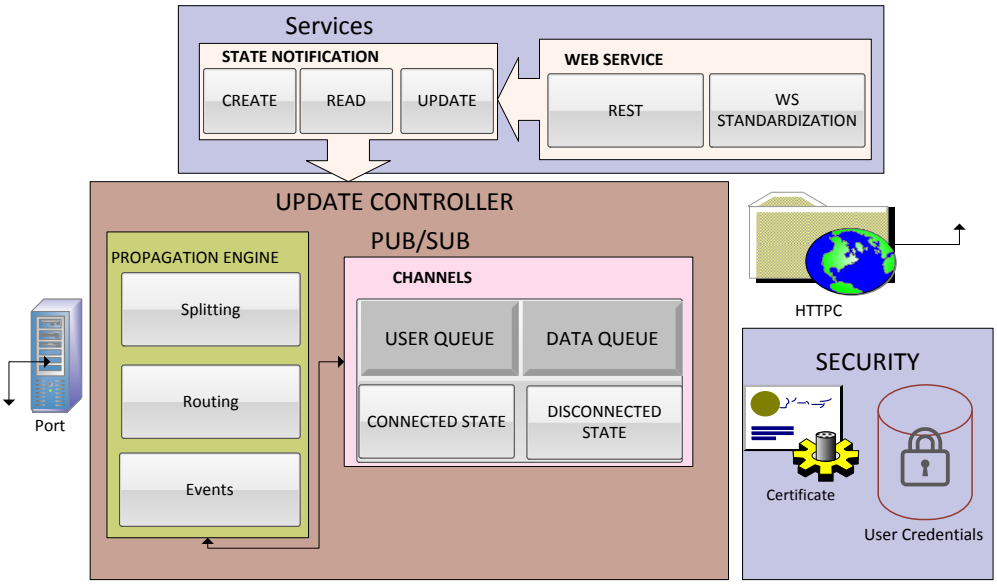

Fig. 5 The middleware Layer

DOI:10.5963/BER0301001

Services: This layer ensures that the medical data from the HIS follows the REST web services standard and protocol. There is a *web service* component that ensures all medical data is a Web service with a meta-data and a message body. Further, the WS standardization ensures that all the medical data can be identified based on a URL call. The next functionality of the services layer is to ensure that notifications are published based on the actions of the healthcare professionals. Actions, such as the creation of new records and updated medical records, are published to the healthcare professionals of interest. A *Read* request, however, is required to fetch, so it does not change the state of the data. However, the *Create* and *Update* operations can change the state of the data, so it requires additional attention so that the data propagation can be effected. When the create operation is invoked, the mobile hosts must be contacted so that the newly created data will be pushed to the mobile. Until this is done, the data is inconsistent because the healthcare professionals do not have the same view of the data in the entire system.

To answer research questions 2 and 3, the *Update Controller* is proposed. This is where we determine how to enforce real-time synchronization as well as which consistency model to follow. The controller is responsible for the medical data propagation and the publish-subscribe (pub/sub) methods. The pub/sub is proposed to ensure real-time data synchronization. The pub/sub component has *channels*, which are groups (known as topics of interest) of medical data that the healthcare professionals are interested in receiving. For instance, healthcare professionals will prefer to have quick updates on the status of their assigned patients than an updated list of pharmacies. The channel has a *user queue* which represents the list of healthcare professionals that must receive updates. The *data queue* represents the messages/medical data that must be delivered to each healthcare professional. Further, the channel keeps two states for the device of the users, the connected state and the disconnected state. The *connected state* means the healthcare professional's device (i.e., the mobile host) is available to receive updates, and the middleware can propagate the update. The *disconnected state* means the mobile host is not reachable. This can be because the device is turned off, or there is no wireless connectivity in the environment of the healthcare professional. In that case, the updates cannot be delivered to the mobile host, so the channel will keep the update in the data queue to be delivered when the state of the device changes to connected. This makes our system adopt the eventual consistency approach. This means there is a time window within which there are inconsistencies, but it only requires connectivity for the update to be pushed.

When the device is in a connected state, the propagation engine can perform tasks such as data *routing*, which involves the transfer of messages between the mobile and the HIS based on the activities of the healthcare professionals. The propagation treats all the activities, especially with the channels, as *events*. Further, the middleware does request *splitting* to differentiate/separate messages going to the mobile host from the messages going to the HIS. The mobile hosts reach the middleware through a port number, and the middleware communicates to the HIS over HTTPC (i.e., the http client) interface.

The middleware also has a security layer that is responsible for two major tasks, the storage of the certificate and user access credentials. Since the communication between the middleware and the HIS is HTTPS, both platforms share a *certificate* that aids them to communicate securely. The *user credentials* stores the username and password pairs of each user. When the healthcare professionals pass the authentication test, they can then have access to the system.

Further, it is also important to state how modifications are propagated. Unlike the create operation, the update (i.e., modification) operation requires that the state of an existing data changes. In this case, the middleware determines the exact changes that have been made to the medical data. Then, only the change is propagated to the mobile host. For example, the visit record of the patient will be updated when the patient comes to the hospital or when the healthcare professional visits the patient. In this case, a new visit is only appended to the previous records, so the middleware only propagates the new record, and the mobile host will append it to its record. This is to reduce bandwidth usage.

IV. SYSTEM EVALUATION

We evaluate the proposed system with devices of the following specifications: *iPad 3* — OS: Apple iOS 5.1.0, Resolution: 2048x1536, Processor: A5X (dual-core, w/ quad-core graphics), and Storage: 16GB, RAM: 1GB. The middleware is deployed on a privately owned cloud with the following specifications: *Processor: Intel Core i-5, CPU 2400@ 3.10 GHz 3.10 GHz, RAM: 16 GB, and System 64-bit operating system.* The mobile devices connect to the middleware through 802.11g Wi-Fi 54Mbps connection.

A. Concurrent Request Support by a Mobile Host

In this experiment, we tested the ability of the mobile host to support concurrent requests from the healthcare practitioners. A similar experiment was conducted in [3], so our aim is to compare the results in both cases. Table I and Figure 6 highlight the result of this experiment.

The old SOPHRA system, reported in [3], shows a very poor result, even though the results were encouraging at the time we originally built the system. In the earlier version, the REST requests that were supported ranged between 1 and 40. In the current advancement, as much as 540 concurrent requests can be processed. The request response time of the new system is shown in Figure 7. To be rational, we can say that today's device capacity, in terms of processing and storage, has improved more than the devices of the last two years. However, we observed that is not the only reason for the huge performance leap. The separation of concern design in the current mobile host dictates that only user-specific tasks are processed in a request-

DOI:10.5963/BER0301001

response call. The server-based tasks are processed separately in the background and do not interfere in the request-response interaction. This is now particular encouraging for the healthcare practitioners since latency is greatly reduced. Also, this serves as a great advantage for the synchronization process since the time to propagate the updates is reasonably small.

TABLE I: COMPARISON OF RESPONSE TIMES OF THE REST WEB SERVICE

Systems	Average response time (ms)	Standard deviation	Maximum request rate (request/s)
SOPHRA	584.47 (for 30 requests)	174.94	51.33
NEW SYSTEM	565.44 (for 540 requests)	65.78	175.53

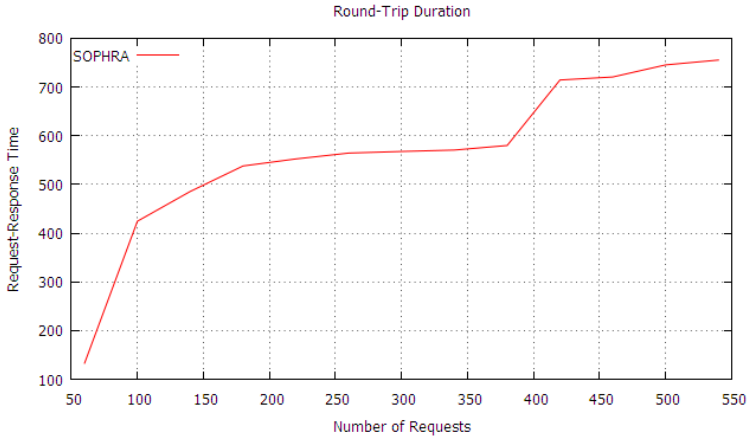

Fig. 6 The Round Trip Duration from a mobile host

B. *Scalability*

The second experiment is to determine how the middleware scales with respect to increasing workloads. The architecture is centralized, which means all the communications are routed through the middleware. Although the initial system currently scales well, there is need for improvement. The performance of the middleware is a crucial factor since it can be a source for latency. It has a lot of transactions to handle, as described in previous sections.

Thus, we evaluate the newly designed middleware using the parameters similar to the previous scalability test in [3]. We configured a load tester that sends concurrent requests from 600 to 40000. These requests include the request to read medical records such as vitals, visits, allergies, and so on. In Table II and Figure 7, we report the results of the experiment. The report includes the reproduction of the previous results in [3].

TABLE II: OUTCOME OF THE MIDDLEWARE SCALABILITY TEST

System	Mean Throughput (req/s)	Maximum Throughput (req/s)	Minimum Throughput (req/s)
SOPHRA	331.40	387.77	284.67
NEW SYSTEM	1282.50	3270.78	390.00

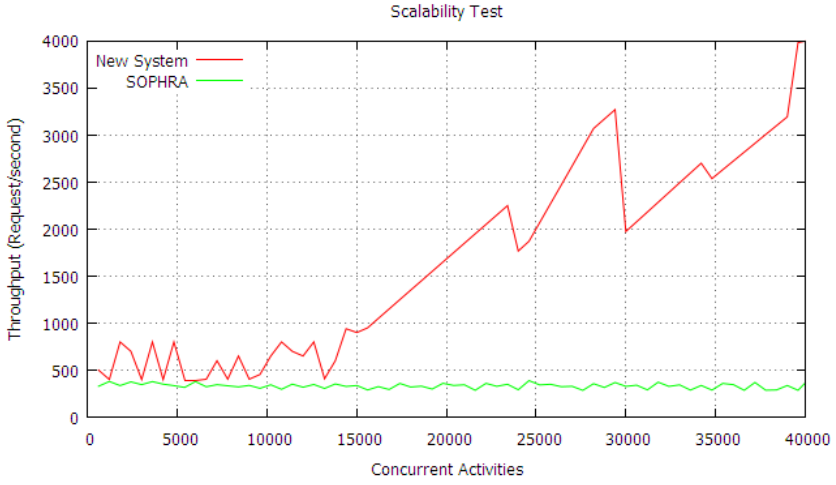

Fig.7 Scalability of the middleware

DOI:10.5963/BER0301001

When we compare the throughput, the new system has a higher scalability in terms of the amount of requests processed per second. The SOPHRA system, though, followed the exponential distribution and shows almost identical throughput with increasing number of requests. However, the new system has shown the ability to handle increasing requests with an increasing throughput. In the new system, messages which are not delivered are queued, and the middleware gets to do something else. This always causes the system to have sleep time and wake up time when the load increases.

V. CONCLUSIONS

In this work, we describe a mobile provisioning environment that can enable healthcare practitioners to access the electronic health record in real time. Mobile technology is advancing the marriage between mobile and other ICT tools, known as mHealth, and, therefore, is receiving significant research attention. The opportunities presented are remote healthcare delivery, easy access to information, as well as ubiquitous healthcare. However, mobile devices rely on wireless channels to communicate, so there are inherent challenges, such as latency, access denied to medical data, inconsistent data, and so on.

In this work, we collaborate with the medical practitioners from the Geriatrics Ward at the City Hospital in Saskatoon where we seek to answer the following research questions: (i) how to store the data for use on the mobile, (ii) how to ensure real time sync, (iii) which consistency model is appropriate, and (iv) medical data privacy. We answered the questions by proposing a middleware-oriented architecture that aids in the synchronization process. The pub/sub methodology is proposed to ensure updates are propagated as soon as they are available. The experimental results also show a high performance boost with respect to the old system, which was described in our previous work [3].

However, we have not done much in answering the fourth question. This is because it requires independent research that will be explored in the future.

ACKNOWLEDGMENT

Thanks to Shomoyita Jamal and Sunny Sharma who are the graduate students who developed part of the User Interface of SOPHRA.

Thanks to the Geriatrics Ward at the City Hospital in Saskatoon, Canada for their commitment to the success of this project and funding.

Further thanks to the Editorial Team of Biomedical Engineering Research, and the Reviewers of this work for their invaluable critique.

REFERENCES

[1] C. Gibbs, “The Rise of Tablets in the Enterprise,” GigaOM Pro, June 2011.

[2] C. Warren, “Native App vs. Web App: Which Is Better for Mobile Commerce?,” http://mashable.com/2011/05/23/mobile-commerce-apps/.

[3] Lomotey, R. K., Jamal, S., and Deters, R. 2012. SOPHRA: A Mobile Web Services Hosting Infrastructure in mHealth. Mobile Services (MS), 2012 IEEE First International Conference on Mobile Services, vol., no., pp.88-95, 24-29 June 2012, Honolulu, Hawaii, USA, doi: 10.1109/MobServ.2012.14.

[4] A. Meads, A. Roughton , I. Warren, and T. Weerasinghe, “Mobile Service Provisioning Middleware for Multihomed Devices,” Proceeding WIMOB '09, Networking and Communications IEEE Computer Society Washington, DC, USA 2009.

[5] S.N. Srirama, M. Jarke, and W. Prinz, “Mobile Web Service Provisioning. Telecommunications,” (AICT-ICIW '06), International Conference on Internet and Web Applications and Services/Advanced International Conference on 19-25 Feb. 2006.

[6] C. Barnatt, “Explaining Cloud Computing” [Online], 10th May 2009, Available: http://www.explainingcomputers.com./cloud.html.

[7] S.N. Srirama, M. Jarke, and W. Prinz, “Mobile Host: A feasibility analysis of mobile Web Service provisioning,” 4th International Workshop on Ubiquitous Mobile Information and Collaboration Systems, UMICS 2006, a CAiSE'06 workshop, June 5-6th, 2006.

[8] M. Hassan, Z. Weiliang, and Y. Yang, “Provisioning Web Services from Resource Constrained Mobile Devices,” Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on 5-10 July 2010.

[9] X. Feng, J. Shen, and Y. Fan, “REST : An Alternative to RPC for Web Services Architecture,” First International Conference on Future Information Networks, ICFIN 2009, p 7-10, 2009.

[10] J. Christensen, “Using RESTful web-services and cloud computing to create next generation mobile applications,” Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, p 627-633, 2009, OOPSLA 2009.

[11] G. Wicks, E. Van Aerschot, O. Badreddin, K. Kubein, K. Lo, and D. Steele, “Powering SOA Solutions with IMS," Pg. 9 Publisher: IBM Redbooks Pub., Date: March 30, 2009, Part Number: SG24-7662-00, Pages in Print Edition: 410.

[12] Lindholm, T., Kangasharju, J., and Tarkoma, S. 2009. Syxaw: Data Synchronization Middleware for the Mobile Web. Mob. Netw. Appl. 14, 5 (October 2009), 661-676. DOI=10.1007/s11036-008-0146-1 http://dx.doi.org/10.1007/s11036-008-0146-1

[13] C. Pautasso, Z. Olaf, and F. Leymann, “RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision,” Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08, p 805-814, 2008, Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08.

DOI:10.5963/BER0301001

[14] V. Beal, "Understanding Web Services" September 2010. Available: http://www.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp

[15] Q. Wang, R. Deters, "SOA's Last Mile-Connecting Smartphones to the Service Cloud," cloud, pp.80-87, 2009 IEEE International Conference on Cloud Computing, 2009.

[16] Lv, J., and Zheng, X. Y. 2009. Research for a Data Synchronization Model Based on Middleware and Rule Base. Information Science and Engineering (ICISE), 2009 1st International Conference on , vol., no., pp.2998-3001, 26-28 Dec. 2009, doi: 10.1109/ICISE.2009.904.

[17] Mokdad, L., and Sene, M. 2006. Performance measures of a call admission control in mobile networks using SWN. In Proceedings of the 1st international conference on Performance evaluation methodolgies and tools (valuetools '06). ACM, New York, NY, USA, , Article 64 . DOI=10.1145/1190095.1190177 http://doi.acm.org/10.1145/1190095.1190177.

[18] Yang, G. 2010. Data synchronization for integration systems based on trigger. Signal Processing Systems (ICSPS), 2010 2nd International Conference on , vol.3, no., pp.V3-310-V3-312, 5-7 July 2010, doi: 10.1109/ICSPS.2010.5555804

[19] F. Aijaz, S. Z. Ali, M. A. Chaudhary, and B. Walke, "Enabling High Performance Mobile Web Services Provisioning" , Published in: Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th, 20-23 Sept. 2009.

[20] J. Ranck, "The Rise of Mobile Health Apps," GigaOM Pro, October 2010.

[21] 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. 3/2010, pp: 1-14.

[22] Soon, C. J., and Roe, P. 2008. Annotation architecture for mobile collaborative mapping. In Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia (MoMM '08), Gabriele Kotsis, David Taniar, Eric Pardede, and Ismail Khalil (Eds.). ACM, New York, NY, USA, 211-218. DOI=10.1145/1497185.1497230 http://doi.acm.org/10.1145/1497185.1497230.

[23] Xue, Y. 2008. The Research on Data Synchronization of Distributed Real-Time Mobile Network. Computer Science and Software Engineering, 2008 International Conference on , vol.3, no., pp.1104-1107, 12-14 Dec. 2008, doi: 10.1109/CSSE.2008.1296.

[24] Khan, A. M., Chabridon, S., and Beugnard, A. 2007. Synchronization medium: a consistency maintenance component for mobile multiplayer games. In Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (NetGames '07). ACM, New York, NY, USA, 99-104. DOI=10.1145/1326257.1326275 http://doi.acm.org/10.1145/1326257.1326275.

[25] Lomotey, R. K., and Deters, R. 2012. Using a Cloud-Centric Middleware to Enable Mobile Hosting of Web Services. Procedia Computer Science, Volume 10, 2012, Pages 634-641, ISSN 1877-0509, 10.1016/j.procs.2012.06.081., http://www.sciencedirect.com/science/article/pii/S1877050912004383.

[26] Lomotey, R., Kazi, R., and Deters. R. 2012. Near real-time medical data dissemination in m-Health. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES '12). ACM, New York, NY, USA, 67-74. DOI=10.1145/2457276.2457290 http://doi.acm.org/10.1145/2457276.2457290.

[27] Arunachalan, B., and Light, J. 2008. Agent-Based Mobile Middleware Architecture (AMMA) for Patient-Care Clinical Data Messaging Using Wireless Networks. In Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '08). IEEE Computer Society, Washington, DC, USA, 326-329. DOI=10.1109/DS-RT.2008.50 http://dx.doi.org/10.1109/DS-RT.2008.50.

[28] Sain, M., Lee, H., and Chung, W. 2009. Middleware in ubiquitous computing system with MedRec architecture. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS '09). ACM, New York, NY, USA, 149-154. DOI=10.1145/1655925.1655953 http://doi.acm.org/10.1145/1655925.1655953.

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. Over a couple years, most of his works are industry collaboration with the Saskatoon Health Region.

Prof. 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.

DOI:10.5963/BER0301001