Cloud Computing Architectural Framework towards Improving Availability of Service for Microfinance Institutions (MFIs) of Ethiopia

Yared Endale
HiLCoE, Computer Science Programme, Ethiopia
yaredcoreb@gmail.com

Abrehet Mohammed
HiLCoE, Ethiopia
abrehet@gmail.com

Abstract

The growth of the Internet as a vehicle for secure communication and electronic commerce has brought many technological innovations capable of providing quality of service, ease of solution implementations and, most of all, significant reduction of overall required cost. Among these compelling technologies in today's development, cloud computing is becoming the present and the future. Presently businesses and IT executives are confronted daily by conflicting and exaggerated claims of how the cloud will transform their industries, but the lure of transformative efficiency and agility is hard to ignore. One of the main concerns for financial institutions today is providing quality of service by ensuring system availability and ease off accessibility. Financial institutions, specifically the microfinance institutions (MFIs) in Ethiopia, face these challenges more critically than others due to inadequate network infrastructure and business orientations to reach the rural poor.

This research work used qualitative research method that explored and proposed an improved quality of service delivery and availability model, by adopting a general cloud computing security model and designing contextualized availability (i.e., system delivery) model along with its implementation model. This is done through a careful consideration of the unique characteristics and the common and basic requirements of microfinance institutions. This enables and makes sure all their branches (online or offline) can be empowered and a proper use out of their centralized financial systems can be ensured.

To address these problems, we adopted a theoretical secure cloud computing architectural model along with a data synchronization model for the offline branches to exchange data with and its overall deployment model.

Keywords: Cloud Computing; Cloud Security; Cloud Computing Architecture; Availability; Synchronization; Core Banking; Microfinance Institution

1. Introduction

Just as in the case of the invention of the personal computer or the advent of the Internet itself, what is today known as 'Cloud Computing' involves a major transformation of the way businesses and individuals access and use computing resources [1].

As there are multiple definitions of cloud computing the following two definitions are considered for this research.

"Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation" [2].

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [3].

The potential problems that delay the adoption of cloud computing, i.e., the mostly practiced secure cloud computing architecture, for some of the Small and Medium Enterprises (SMEs) and the large enterprises are: privacy and security of data, vendor lock-in and interoperability, service availability,

absence of proper Service Level Agreement (SLA), performance instability, latency on network, lack of scalable storage, and reputation fate sharing [5], among which this research will concentrate on service availability due to lack or latency of networks.

2. Related Work

The security framework proposed in [6] is claimed to provide secure connection and convenience to the user for accessing cloud service. The authors consider cloud orchestration environments and Single Sign-On Token to provide seamless experience to the user. Furthermore, they provide possible technologies for cloud collaboration.

The authors in [11] argue that understanding the relationships and interdependencies between the different cloud computing deployment models and service models is critical to understanding the security risks involved in cloud computing. In addition to that the authors 1) identified a core set of Security Components that can be implemented in a Cloud Ecosystem to secure the environment, the operations, and the data migrated to the cloud, 2) provided, for each Cloud Actor, the core set of Security Components that fall under their responsibilities depending on the deployment and service models, 3) defined a security-centric formal architectural model that adds a security layer to the current NIST SP 500-292: NIST Cloud Computing Reference Architecture, and 4) provided several approaches for analyzing the collected and aggregated data.

In [4] the authors claim that adopting cloud computing is a complex decision involving many factors. They hope that the guidance contained in their work will help better understand what questions to ask, the current recommended practices, and potential pitfalls to avoid. Through their focus on the central issues of Cloud Computing security, they have attempted to bring greater clarity to an otherwise complicated landscape, which is often filled with incomplete and oversimplified information. The authors also claim that not all cloud deployments need every possible security and risk control. Spending a little time up front evaluating the risk tolerance and potential exposures will provide the context needed to pick and choose the best options for an organization.

According to [7], data synchronization mechanism patterns address the question: “when should an application synchronize data between a device and a remote system (such as a cloud server)?” This problem is common yet often overlooked in mobile application design, but there is no one size fits all solution. Instead, mobile application developers must consider the constraints of many factors, including network availability, data freshness requirements, and user interface design. The authors also argued that data synchronization mechanism patterns are often architectural patterns. An architectural pattern expresses a fundamental structural organization schema for software systems. It provides a set of predefined subsystems, specifies their responsibilities, and includes rules and guidelines for organizing the relationships between them.

Oracle Database Lite synchronization is used and streamlined for multiple remote clients rather than a single user in [9]. With Oracle Database Lite, one can create an application where multiple users enter data on their client devices and the data is synchronized with a back-end Oracle database. In addition, only the data designated for each user is downloaded from the server to the client’s device. For example, if there is a sales force, each sales person retrieves only his/her data on the client device. Any modifications made on either the client device by the sales person in regards to his/her accounts or modified on the server by the office can be synchronized. One can even create applications that collect statistics and can be either synchronized automatically, when an event occurs, at defined intervals, or when polled. This type of implementation could be used on a vending machine, where remote synchronization can keep the office continually informed of how many items are left.

3. The Proposed Solution

3.1 The Proposed General Secure Cloud Architecture

After a careful study, comparison and analysis, among the best secure architectural frameworks available, we have selected the secure cloud architecture proposed by Munir and Palaniappan in [6]. The reason for the selection is that it is built on the three main aspects of cloud computing environment: SaaS, PaaS, and IaaS. These main aspects are the crucial elements of private cloud deployment model that we are basing our proposed solution in this research.

As shown in Figure 1 [6], in addition to its simple and easily understandable presentation of the architecture, it basically can address many of the security concerns generally fundamental to the financial software solutions and the gaps identified that are unique to our research interest in this paper.

In the proposed security model, a user can be certified by a 3rd party Certificate Authority and then can be issued a token for service by End User Service Portal [6]. After joining the service portal, a user can purchase and use cloud services which are provided by a single service provider. The End User Service Portal is composed of access control, security policy, key management, service configuration, auditing management, and virtual environments that provide secure access control using Virtual Private Network (VPN) and cloud service managing and configuration.

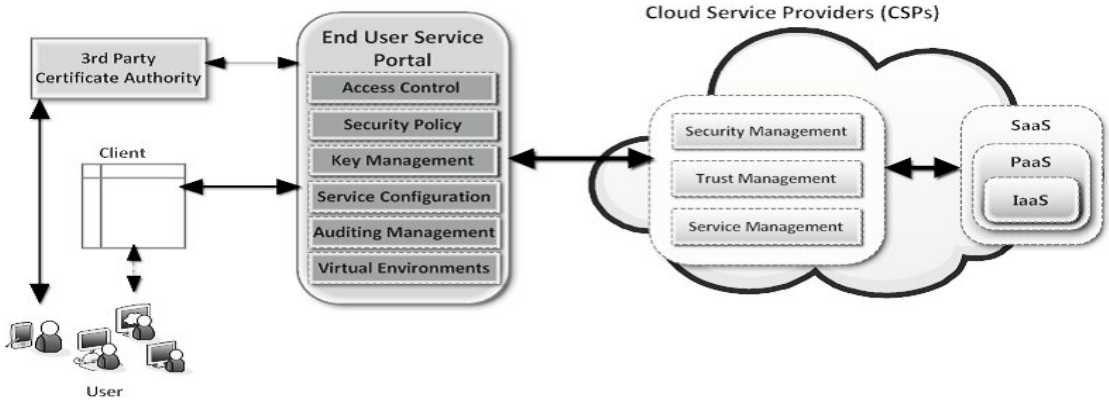

Figure 1: Security Model for Cloud Computing

The framework for secure cloud computing as shown in Figure 1 is based on the security model that describes the details of each component and apply the needed security technologies for implementation between components in cloud computing [6].

Our security framework proposed below provides secure connection and convenience to the user for accessing a cloud service. We consider cloud orchestration environments and Single Sign-On Token to provide seamless experience to the user. Furthermore, we provide possible technologies for cloud collaboration.

3.2 The Proposed Availability Model

a) Offline Functionality with Data Synchronization

For any information system to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly. High availability systems aim to remain available at all times, preventing service disruptions due to power outages, hardware failures, and system upgrades. Ensuring availability also involves preventing denial of service attacks, such as flooding messages to the target system forcing it to shut down [12].

Due to the network infrastructure and quality of Internet connection in Ethiopia, financial institutions suffer greatly on availability of service in most of their branches. It is a common phenomenon for major banks to disrupt some of the operations of their branches due to network unavailability during working hours and as a result clients are not getting

the services they request. Therefore, most MFIs who have data centers for their core banking applications are not able to reach their rural clients to be served using their highly invested technology. It is, therefore, critical for these financial institutions to look for a solution to maximize utilizing the expensive core banking systems they acquire.

During requirement gathering in this work, all of the major MFIs interviewed operating outside of the capital city claimed to have requested their technology providers to give them an option for offline functionality that can be synchronized with their central database.

The lasting solution for this problem would be to have a reliable network infrastructure regardless of location, which is not practical at the moment. Therefore, the MFIs with most of their branches operating in rural parts of the country today, cannot afford to wait until the network problem is taken care off. Hence, they usually crawl back to their software providers for alternative solutions.

Based on the requirements gathered in this work, we propose a modified availability model using the base architecture shown in Figure 1. We introduced an additional system functionality to ensure availability of services for offline branches of MFIs. Figure 2 shows the additional data synchronization functionalities that can be initiated by both the central and offline servers when connection is established. The proposed availability model consists of the offline client (i.e., the offline application server as a client during the process of synchronization), the cloud from the service provider (i.e., used to access the central application and database servers) and the end user service portal integrated into the application software to ensure secure communication. Availability of service due to network latency unreliability can be addressed using offline services using data synchronization when connection is available depending on the MFI's preferred arrangements.

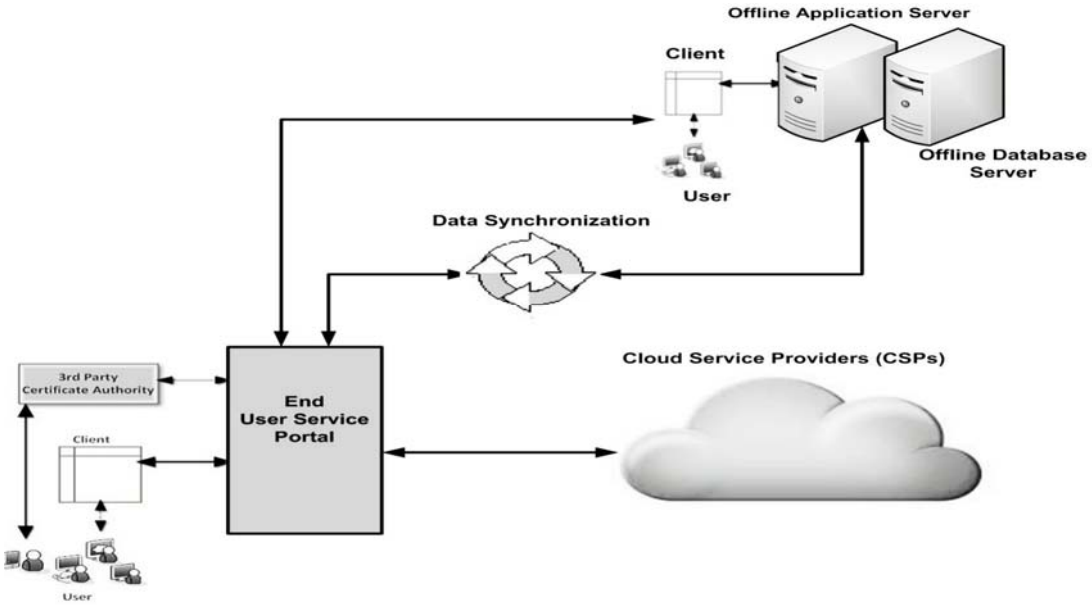

Figure 2: The Proposed Availability Model

As shown in Figure 2, data synchronization is part of the services or functionalities run on the central and offline application servers. An offline branch or a mobile branch from rural areas, using their light weight locally installed software, can run the services in the application after working offline and manually synchronize with the central server after connection is established. Therefore, the application server in both offline and in the data center should run the synchronization services after connection is established by the offline branch during bank closing hours. This can be achieved by implementing mobile

branches with a high end machine like modern laptops or any easily portable machine and moving to the nearest network coverage area to make the connection. This obviously will require the institution to have other security policies to be implemented out of the system for the implementation model it selects.

Manual synchronization is selected instead of automatic synchronization because:

- The services are user initiated. Most of the business units in our case are mainly permanently offline and have to initiate the synchronization services after getting a successful connection service to their VPN during business closing hours. Hence, data cannot be transferred asynchronously to the central server.
- Data synchronization and complete storage is selected for availability of a complete and latest central data which is required for the next working day. Manual synchronization is more appropriate, ideal, and more practical considering the business scenario in question.

During manual synchronization, services in both the online and offline branches should be run to ensure secure data transfer for data integrity. After these services are run, both the central and the offline branches should have the same copy of the data. We suggest some operational services that could be run on either side of the domains to do the job of synchronization as shown in Table 1.

We propose the services required for secure data synchronization from the offline and central server. Every service is listed and its purpose for the synchronization process is described for each site.

Offline Branch Services: The offline branch services are run on the offline server and what each job is responsible for is shown in Table 1.

Online Branch Services: All the online branch services are run on the online central server. Table 2 shows what each job is responsible for.

Table 1: Operational Services Proposed for Offline Branches

Service	*Purposes and Description*
Connection Detection Updating Service	This service is designed to run only at remotely deployed *Offline Branch* and ping the central server. It should update the table created to record the online/offline status at central server which is used as the reference to check whether the branch is online.
Branch Transaction Forwarder Service	The responsibility of this service is to copy the transactions which were originated at remotely deployed offline branch to the central server (central database).
Branch Update Synchronizer Service	The responsibility of this service is to update and synchronize the branch database with the central server (central database).
Branch Sync Export Service	This service is responsible for creating text files containing data captured at a branch while offline for delivery to the head office to be uploaded into the bank wide server. The system fetches the list of records from the sync tables whose status is ‘Active’.

Table 2: Operational Services Proposed for Central Online Branch

Service	*Purposes and Description*
Proxy Posting Service	The responsibility of this service is to update the central database with the transactions which originate at a remote branch. It is activated only at the central server.
Central Sync Import Service	It is responsible for uploading data received from an offline branch into the central server. The data to be uploaded is decided from the sync table. Each record of this table represents the entity that needs to be synchronized to the central server.
Central Sync Export Service	It is responsible for creating text files containing data captured at a central server while a branch is offline for onward delivery to the branch server.

b) The Proposed Conceptual Synchronization Model

Due to the service behavior of financial institutions, we propose the synchronization mechanism as proposed by McCormick and Schmidt [7] for availability pattern. Data Storage and Availability patterns address the questions:

- How much data should be stored?
- How much data should be available without further transfer of data?

These questions arise in both the design of mobile applications and remote systems with which they interact. Often, mobile application projects have constraints (such as network capacity) both remotely and locally. Likewise, the capacity of local storage is often limited relative to the whole dataset needed for a computation [7].

This is also true for the offline branch setup we propose as a solution. Institutions cannot afford to have a server capable of storing the whole dataset from the central server in all mobile or offline branches. Connection speed is also a problem we are trying to address.

During running the end of day system processes in offline application servers, synchronization services can also be run as proposed in our availability model. The mobile offline branch is assumed to have created a connection to the cloud to access the central server. Since MFIs will have multiple offline sites, the proposed architecture should support concurrent users during synchronization.

The following demonstration for Oracle Lite synchronization architecture will provide a model for components working together to ensure seamless synchronization. It is modified according to the security architectural model we proposed and the entities involved in our cloud architecture. Mobile clients are the same as offline clients and mobile server as offline server in our scenario.

On the client device, data is stored in a special type of relational table, called a *snapshot table*. A snapshot table behaves identical to a regular relational table, but has functionality that enables tracking changes made to the table.

An SQL query, called the *publication* item, can determine the record set that is downloaded to the snapshot table. It is executed on the database server. The result of the query defines the structure (columns) of the snapshot table on the client device as well as its rows. A collection of publication items is called a *publication*, which corresponds to a single Oracle Lite database on a client device. All snapshot tables that are based on publication items and are part of a single publication are stored in the same Oracle Lite database [8].

The following steps describe how all components work together to create a seamless synchronization from multiple clients. For synchronization purpose among the offline server and the central server, Advanced Queuing (AQ) can be used. According to [10], AQ is a sort of Message Oriented Middleware developed by Oracle and integrated into its Oracle database. AQ uses database structures as repository for asynchronous queuing as an element in various Oracle-oriented and heterogeneous operations.

Synchronization is initiated on the Offline/Mobile client by the user.

1. Offline/Mobile client software (service) gathers all of the client modifications and the Sync Client service uploads them to the Sync Server on the Offline/Mobile Server.
2. The Sync Server places the modifications into the In-Queue.
3. The Sync Server gathers all modifications destined for the Offline/Mobile client from the Out-Queue.
4. The Sync Client downloads all modifications for the client local database (also known as the ODB file).
5. The Offline/Mobile Client applies all changes for client local database. If this is the first synchronization, the local database is created.

6. All modifications uploaded by all Offline/Mobile clients are gathered by the MGP (Message Generator and Processor) out of the In-Queue.
7. The MGP executes the apply phase by applying all modifications for the Offline/Mobile clients to their respective application tables to the back-end central database.
8. MGP executes the compose phase by gathering the client data for each Offline/Mobile client.
9. MGP places the composed data for Offline/Mobile clients into the Out-Queue, waiting for the next client synchronization for the Sync Server to gather the updates to the client.

3.3 The Proposed Offline Deployment Model

After ensuring the availability of offline functionality services in their financial applications, MFIs can adopt the online/offline deployment model as proposed in Figure 3. After working offline for a given time, offline or mobile branches using their agents on the field, the system admin can make the synchronization using available connection options in the area.

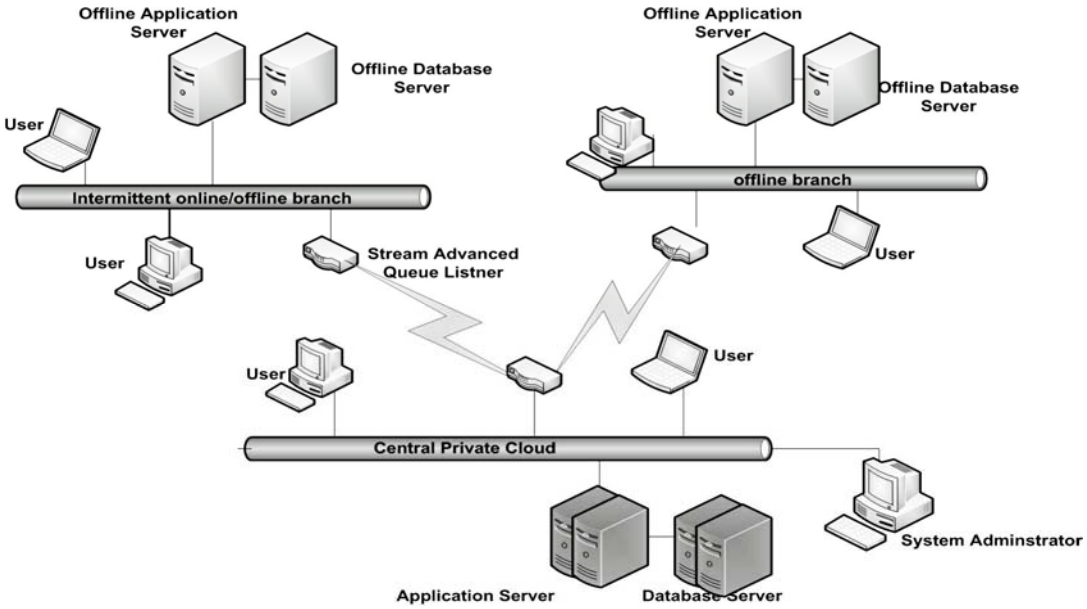

Figure 3: Intermittent Online/Offline Deployment Model

To set up the above deployment model, the following are key factors concerning configurations to be made on both the offline and online (i.e., central) application servers:

- Offline enabled set up consists of a Central Server Installation and a Branch Installation. Central Server consists of at least an Application Server instance and a Database Server instance. The Branch Server consists of another Application Server instance and a separate Database Server.
- Connection setups with the addresses of both the central and offline should be configured in either side accordingly.
- In the system configuration of both sides of the application server, there should be a setup to enable offline functionality through configuration files.
- Enabling Cross Domain Security (same-origin policy) in the Application Servers: In order to enable a transaction API at domain D1 to be invoked from domain D2, we need to enable cross domain security in both D1 and D2.

4. Conclusion and Future Work

The proposed cloud computing architectural framework, in addition to the general security architecture adopted, addresses the problem of availability of service for offline branches through its

availability model and data synchronization using the underlining security features integrated in the secure cloud architectural model. Our Availability Model uses application or operational services integrated into the offline and online branch application deployments, designed to run all the database operations required, which eventually ensures the same copy of required data available in both sides of the client and central servers.

Therefore, we recommend financial institutions, MFIs in particular, when implementing their automated centralized financial systems, should evaluate if the crucial general security model available and the service availability mechanisms integrated regardless of their business orientation, are in place along with its implementation strategy in their technological solution implementations. This will ensure all their functional business units (online or offline branches) can make the most use out of their cloud services (i.e., SaaS, PaaS, and IaaS).

This research did not test the proposed solution by designing a prototype API for the operational services proposed. Therefore, as a future work, the proposed solutions will be designed as an API for the operational services required to be run on both offline and online servers using a web based application development model. We will further test and evaluate the Availability Model and its deployment mechanism and explore the efficiency of the synchronization techniques used from data security perspective.

References

[1] Brayan Barnett, “Cloud Computing & Financial Services for the Poor, Primer & Procurement Guidance”, USAID under the Knowledge-Driven Microenterprise Development (KDMD) project, implemented by the QED Group, LLC, October 2011.

[2] http://en.wikipedia.org/wiki/cloud_computing, Last Accessed on July 2014.

[3] Peter Mell and Timothy Grance “The NIST Definition of Cloud Computing”, Recommendations NIST Special Publication 800-145, September 2011.

[4] Cloud Security Alliance (CSA), Version 3, 2011.

[5] Mohammad Manzurul Islam, Sarwar Morshed, and Parijat Goswami, “Cloud Computing: A Survey on its Limitations and Potential Solutions”, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia.

[6] Kashif Munir and Sellapan Palaniappan, “Framework for Secure Cloud Computing, Services and Architecture”, International Journal on Cloud Computing: Vol. 3, No. 2, April 2013.

[7] Zach McCormick and Douglas C. Schmidt, “Data Synchronization Patterns in Mobile Application Design”, In proceedings of the Pattern Languages of Programs (PLoP) 2012 conference, October 19-21, Tucson, Arizona.

[8] Oracle Database Lite, Automatic Synchronization White Paper, August 2008.

[9] Oracle Database Lite, Synchronizing Data between a Device and an Oracle Database, August 2007.

[10] Hevner, Alan R., et al. “Design Science in Information Systems Research”, MIS quarterly 28.1 (2004): 75-105.

[11] NIST National Institute of Standards and Technology, "NIST Cloud Computing Security Reference Architecture", NIST Special Publication 500-299, June 2013.

[12] https://en.wikipedia.org/wiki/Information _security#Availability, Last Accessed on June 2015.