

CLOUD SERVICES BROKERAGE FOR MOBILE UBIQUITOUS  
COMPUTING

A Thesis Submitted to the  
College of Graduate Studies and Research  
in Partial Fulfillment of the Requirements  
for the degree of Doctor of Philosophy  
in the Department of Computer Science  
University of Saskatchewan  
Saskatoon

By  
Richard Kwadzo Lomotey

©Richard Kwadzo Lomotey, June/2015. All rights reserved.

## PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science  
176 Thorvaldson Building  
110 Science Place  
University of Saskatchewan  
Saskatoon, Saskatchewan  
Canada  
S7N 5C9

# ABSTRACT

Recently, companies are adopting Mobile Cloud Computing (MCC) to efficiently deliver enterprise services to users (or consumers) on their personalized devices. MCC is the facilitation of mobile devices (e.g., smartphones, tablets, notebooks, and smartwatches) to access virtualized services such as software applications, servers, storage, and network services over the Internet. With the advancement and diversity of the mobile landscape, there has been a growing trend in consumer attitude where a single user owns multiple mobile devices. This paradigm of supporting a single user or consumer to access multiple services from n-devices is referred to as the *Ubiquitous Cloud Computing (UCC)* or the *Personal Cloud Computing*.

In the UCC era, consumers expect to have application and data consistency across their multiple devices and in real time. However, this expectation can be hindered by the intermittent loss of connectivity in wireless networks, user mobility, and peak load demands.

Hence, this dissertation presents an architectural framework called, *Cloud Services Brokerage for Mobile Ubiquitous Cloud Computing (CSB-UCC)*, which ensures soft real-time and reliable services consumption on multiple devices of users. The CSB-UCC acts as an application middleware broker that connects the n-devices of users to the multi-cloud services. The designed system determines the multi-cloud services based on the user's subscriptions and the n-devices are determined through device registration on the broker. The preliminary evaluations of the designed system shows that the following are achieved: 1) high scalability through the adoption of a distributed architecture of the brokerage service, 2) providing soft real-time application synchronization for consistent user experience through an enhanced mobile-to-cloud proximity-based access technique, 3) reliable error recovery from system failure through transactional services re-assignment to active nodes, and 4) transparent audit trail through access-level and context-centric provenance.

# ACKNOWLEDGEMENTS

I would like to express my special appreciation and genuine thanks to Dr. Ralph Deters for his career advice, intuitive criticisms of my scholarly work, patience, and financial assistance throughout my duration of study – first as my Master’s supervisor, instructor, and now Ph.D. advisor. I feel highly honoured to have worked with one of the very few with depth and breadth of knowledge in the area of mobile distributed and cloud services computing. Beyond the academic arena, Dr. Deters treated me as family, always interested in the well-being of my immediate and extended relatives as well as my personal welfare. I will like to simply say, *Ralph! Thank you.*

My next appreciation goes to the members of my advisory committee: Dr. Luigi Benedicti, Dr. Chris Zhang, Dr. Derek Eager, Dr. Mark Keil, and Dr. Chanchal Roy. Your commitment to see me succeed in my studies was evidenced in your availability, and priceless suggestions on how to improve my work. To the following people who also influenced my scholarly work through advices and opportunities, I say a big thank you: Dr. Julita Vassileva (Co-Director, MADMUC Lab), Dr. Eric Neufeld (Head of Dept. Computer Science, U of S), and Dr. Dwight Makaroff (Grad Chair, Computer Science).

Also, my sincere thanks to the following people who supported me with logistics and administratively with open heart and kindness: Gwen Lancaster, Heather Webb, Shakiba Jalal, Linda Gesy, and Jeff Long.

Furthermore, I will like to thank all students of the Multi-Agent Distributed Mobile and Ubiquitous Computing (MADMUC) Lab for their friendship. A big thank you goes to the following friends who embraced me as family: Dr. Johnson Iyilade, Bunmi Adewoyin, Edgar Lelei, Dr. Rita Orji, Petra Scott Vychodilova, Tanner Scott, Peter Opoku, Noble Kuadey, Kwame Sepenu, Shirley Heidi Adu-Asare, Afshana Hussain, Shomoyita Jamal and Mohammad Asif.

On a more personal level, I will like to thank Christian, Vivian, Mama Beatrice, Mama Faustine, grandma Afinor Dada and entire family for the unconditional love and generosity. Thanks to Mr. Timothy Nyarku (Fo Sele) who always tells me to aim higher.

Funding for my degree was generously provided by MITACS, the Department of Computer Science, the Saskatchewan Bleeding Disorder Program, SAKINA Information Sciences, ZenFri Inc., IBM at TJ Watson Center, IEEE Services and BigData Congress, and Dr. Deters.

Finally, I say thank you God for protecting me.



This dissertation is dedicated to my mother Doris Dartey, my wife Emelia Lomotey, and my daughter Keziah Richard.

# CONTENTS

<b>Permission to Use</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iii</b>
<b>Contents</b>	<b>v</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Abbreviations</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Problem Statement . . . . .	3
1.3 Summary . . . . .	5
<b>2 Literature Review</b>	<b>6</b>
2.1 Cloud Computing (From MCC to UCC and the Personal Cloud) . . . . .	6
2.1.1 Mobile Cloud Computing (MCC) . . . . .	8
2.1.2 Ubiquitous Cloud Computing (UCC) . . . . .	14
2.1.3 The Personal Cloud . . . . .	20
2.1.4 Summary . . . . .	22
2.2 Brokerage Services . . . . .	23
2.2.1 Broker Frameworks . . . . .	23
2.2.2 Publish-Subscribe Brokers . . . . .	28
2.2.3 Summary . . . . .	29
2.3 Provenance . . . . .	30
2.3.1 Provenance-Based Frameworks . . . . .	30
2.3.2 Summary . . . . .	35
2.4 Data State Management and Synchronization . . . . .	35
2.4.1 Summary . . . . .	39
2.5 Web Services . . . . .	40
2.5.1 Service-Oriented Architecture (SOA) . . . . .	40
2.5.2 REpresentational State Transfer (REST) . . . . .	42
2.5.3 Semantic Web . . . . .	43
2.5.4 Graph and Linked Data . . . . .	44
2.5.5 Summary . . . . .	49
2.6 Conclusions and Research Questions . . . . .	49
<b>3 CSB-UCC: Cloud Services Brokerage for Mobile Ubiquitous Cloud Computing</b>	<b>53</b>
3.1 Overview . . . . .	53
3.2 The N-Consumer Device Space (Heterogeneous Consumer Devices) . . . . .	54
3.3 Multi-Public/Private Cloud Services . . . . .	55
3.3.1 The File Sharing IaaS Clouds . . . . .	56
3.4 Third Party Authentication and OAuth 2.0 . . . . .	57
3.5 The Broker Cloud . . . . .	59
3.6 Data Update Management in Group Sharing (File Sharing Use Case) . . . . .	64

3.7	Re-Visiting the Research Questions . . . . .	66
3.8	Designing a Distributed Broker Architecture for Scalability in the CSB-UCC . . . . .	67
3.8.1	The Controller . . . . .	68
3.8.2	The Sub-Proxies . . . . .	69
3.8.3	Assigning Jobs Based on the Best-Proximity . . . . .	71
3.9	Mobile and Sensor Data Sharing Based on the Best-proximity Use Case . . . . .	73
3.9.1	Mobile–Sensor Data Flow Architecture . . . . .	74
3.9.2	The Communication Flow . . . . .	76
3.10	Provenance in the CSB-UCC . . . . .	82
3.11	Some Applications that Adapted the CSB-UCC . . . . .	86
3.11.1	The ALILI Framework . . . . .	86
3.11.2	The Med App Framework . . . . .	86
3.11.3	Hemophilia Injury Recognition Tool (HIRT?) . . . . .	86
3.11.4	MUBaaS: Mobile Ubiquitous Brokerage as a Service . . . . .	89
3.11.5	SSCA Spray Quality . . . . .	89
3.11.6	SSCA Tank Mix . . . . .	90
3.12	Summary . . . . .	90
<b>4</b>	<b>Experiments/Evaluations</b>	<b>93</b>
4.1	System Requirements and Experiment Goals . . . . .	93
4.2	Evaluation of the Propagation Window . . . . .	96
4.2.1	Resource Synchronization in the Entire System . . . . .	97
4.2.2	Discussion of Experiments on the Propagation Window . . . . .	106
4.2.3	The Best-Proximity Based Job Assignment . . . . .	108
4.3	Testing the Sensor–Mobile Communication . . . . .	119
4.3.1	Linear Search Algorithms . . . . .	120
4.3.2	Latency Analysis with the Edge-Based Architecture . . . . .	121
4.3.3	Error and Request Re–Assignment . . . . .	125
4.3.4	Summary . . . . .	126
4.4	Testing for Scalability . . . . .	128
4.5	Fault-tolerance . . . . .	131
4.6	Summary . . . . .	132
<b>5</b>	<b>Conclusion and Future Work</b>	<b>134</b>
5.1	Summary and Contributions . . . . .	134
5.1.1	Designed the Cloud Services Brokerage for Mobile Ubiquitous Computing (CSB–UCC)	136
5.1.2	Sensor Data Sharing . . . . .	138
5.1.3	Provenance . . . . .	140
5.1.4	Best–Proximity Services Delivery . . . . .	141
5.1.5	Error Tracking and Request Re–assignment . . . . .	142
5.1.6	Best Paper Awards . . . . .	143
5.1.7	Some Real–World Applications Built as Part of this Dissertation . . . . .	143
5.2	Future Work . . . . .	143
5.2.1	Cybersecurity and Mobile Data Assurance . . . . .	143
5.2.2	Mobile Data Analytic-as-a-Service (AaaS) . . . . .	144
5.2.3	Autonomic Computing Architecture . . . . .	146
	<b>References</b>	<b>148</b>
	<b>A Results on the Propagation Window</b>	<b>167</b>
	<b>B Best-Proximity Experiments</b>	<b>169</b>
	<b>C Erlang Code Snippets of the CSB-UCC</b>	<b>174</b>

<b>D</b>	<b>The Haemophilia Injury Recognition Tool (HIRT?) App</b>	<b>182</b>
D.1	Overview of HIRT? . . . . .	182
D.2	Awards and Recognition . . . . .	182
D.3	Overall Scope of the Project . . . . .	182
<b>E</b>	<b>Components of the Mobile–Sensor Architecture</b>	<b>189</b>
E.1	Sensor Device . . . . .	189
E.2	The Mobile Hosts . . . . .	190
E.3	Cloud Services . . . . .	190
E.3.1	Applications and Personal Devices . . . . .	190
E.3.2	Push Notification . . . . .	191
E.3.3	Analysts . . . . .	191
<b>F</b>	<b>My Peer–Reviewed Publications with Contents from this Dissertation</b>	<b>192</b>

# LIST OF TABLES

2.1	Native Programming Environments . . . . .	17
2.2	Linked Data and Some Open Issues . . . . .	46
2.3	Some Existing Mobile Cloud Services . . . . .	50
3.1	Request-Response Scenario . . . . .	72
4.1	The Mobile Devices and their Capacity . . . . .	94
4.2	The Multi-Cloud Sources . . . . .	95
4.3	The Amazon EC2 Servers for Hosting the Distributed Broker . . . . .	95
4.4	Summary of the Dropbox Result . . . . .	99
4.5	Summary of the Amazon S3 Result . . . . .	100
4.6	Summary of the MEGA Result . . . . .	101
4.7	Summary of the Dropbox Result (Centralized Broker with Approx. 3200 Users) . . . . .	102
4.8	Summary of the Dropbox Result (Centralized Broker with Over 3200 Users) . . . . .	103
4.9	Summary of the Amazon S3 Result (Centralized Broker with Approx. 3214 Users) . . . . .	103
4.10	Summary of the Amazon S3 Result (Centralized Broker with Over 3214 Users) . . . . .	103
4.11	Summary of the MEGA Result (Centralized Broker with Approx. 3206 Users) . . . . .	104
4.12	Summary of the MEGA Result (Centralized Broker with Over 3206 Users) . . . . .	104
4.13	Summary of the Dropbox Result (Distributed Broker with Approx. 20026 Users) . . . . .	105
4.14	Summary of the Amazon S3 Result (Distributed Broker with Approx. 20098 Users) . . . . .	106
4.15	Summary of the MEGA Result (Distributed Broker with Approx. 20026 Users) . . . . .	106
4.16	Geographical Distance Between the Servers . . . . .	109
4.17	Breakdown of the Proximity Access (Time in ms) – Sydney . . . . .	110
4.18	Breakdown of the Proximity Access (Time in ms) – Sydney with Controller in North Carolina	114
4.19	Breakdown of the Proximity Access (Time in ms) – Sydney with Controller in Sydney . . . . .	116
4.20	Sensor-Mobile Latency Test . . . . .	122
4.21	Terminate Vrs Non-Terminate Parallelism Flows . . . . .	124
4.22	The Scalability Testing . . . . .	130
4.23	Fault Injection Analysis . . . . .	132
5.1	Features of the CSB-UCC . . . . .	137
A.1	Overall Summary of the Results on the Propagation Window . . . . .	167
B.1	Breakdown of the Proximity Access (Time in ms) – North Virginia . . . . .	169
B.2	Breakdown of the Proximity Access (Time in ms) – Tokyo . . . . .	170
B.3	Breakdown of the Proximity Access (Time in ms) – Singapore . . . . .	171
B.4	Breakdown of the Proximity Access (Time in ms) – North Virginia with Controller in North Carolina . . . . .	171
B.5	Breakdown of the Proximity Access (Time in ms) – Tokyo with Controller in North Carolina	172
B.6	Breakdown of the Proximity Access (Time in ms) – Singapore with Controller in North Carolina	172

# LIST OF FIGURES

1.1	Cloud Computing . . . . .	2
2.1	Cloud Computing Stack . . . . .	7
2.2	The MCC Design Concepts . . . . .	9
2.3	Architectural Overview of Mobius . . . . .	11
2.4	Data Flow and Services Delivery of Cloud2Bubble . . . . .	12
2.5	The Mobile Personal Grid Environment . . . . .	15
2.6	Intel’s Approach Towards a Unified Mobile Architecture . . . . .	16
2.7	The Web App as the Bridge Between the Traditional Web Site and Native Apps . . . . .	17
2.8	The Carmen Architecture . . . . .	18
2.9	The Envisioned Pocket Cloudlet Architecture . . . . .	19
2.10	The Hype Cycle for Cloud Services Brokerage (March 2012) . . . . .	21
2.11	Inter-Cloud Broker Architecture . . . . .	24
2.12	The Migration Service Broker . . . . .	27
2.13	The Architectural Design of Karma . . . . .	31
2.14	Synchronization Server in a Mobile Network . . . . .	36
2.15	Conflict Detection and Resolution Process Execution . . . . .	38
2.16	Services Oriented Architecture (SOA) . . . . .	41
2.17	Structure of the Data Web . . . . .	45
2.18	Sample RDF Graph . . . . .	45
2.19	Graph or Network Relationship Between Authors, the Prizes they have Won and where they are Born . . . . .	47
3.1	The Generic Architectural Design of the CSB-UCC . . . . .	54
3.2	The Anatomy of the Brokerage Server . . . . .	60
3.3	The Link Graph Between a User’s Accounts . . . . .	61
3.4	The Illustration of how a Single Request from the Consumer Device is Replicated by the Provider Interface . . . . .	64
3.5	The Distributed CSB-UCC Architecture . . . . .	67
3.6	The Composition of the Controller . . . . .	68
3.7	The Composition of a Sub-Proxy . . . . .	70
3.8	Determining the Best-Proximity Sub-Proxy . . . . .	72
3.9	Generic Architecture of the Mobile-Sensor Data Sharing Environment . . . . .	74
3.10	Main Activities of the Mobile Host . . . . .	76
3.11	Request Flow . . . . .	78
3.12	Parallel Request Flow from the Requester . . . . .	81
3.13	The Anatomy of the Provenance Engine . . . . .	83
3.14	The User Interface of the ALILI App . . . . .	87
3.15	The User Interface of Med App . . . . .	88
3.16	The User Interface of HIRT? . . . . .	88
3.17	Loadout Menu of Clandestine Anomaly . . . . .	89
4.1	Experimental Set-up . . . . .	94
4.2	Setup for Testing the File Propagation . . . . .	98
4.3	Four-Time Frames for the Total Propagation Time . . . . .	98
4.4	Update Propagation Window on Dropbox . . . . .	99
4.5	Update Propagation Window on Amazon S3 . . . . .	100
4.6	Update Propagation Window on MEGA . . . . .	101
4.7	Set-up for the Evaluation of the Proximity Accessibility . . . . .	109
4.8	Request-Response Payload from CouchDB . . . . .	110

4.9	The Proximity Test – Sydney . . . . .	111
4.10	The Sydney Proximity Test with Controller in North Carolina . . . . .	113
4.11	The Sydney Proximity Test with Controller in Sydney . . . . .	117
4.12	Search Duration of the Linear Algorithms . . . . .	120
4.13	Parallelism Approaches . . . . .	124
4.14	Correctness of Task Re-Assignment . . . . .	125
4.15	Tasks Assignment to Wrongly Identified Mobile Host . . . . .	126
4.16	Set-up for the Scalability Test . . . . .	128
4.17	The Scalability Test of the CSB-UCC . . . . .	130
B.1	The Proximity Test – North Virginia . . . . .	169
B.2	The Proximity Test – Tokyo . . . . .	170
B.3	The Proximity Test – Singapore . . . . .	170
B.4	The Proximity Test with Controller in North Carolina – North Virginia . . . . .	171
B.5	The Proximity Test with Controller in North Carolina – Tokyo . . . . .	172
B.6	The Proximity Test with Controller in North Carolina – Singapore . . . . .	173
C.1	Generic Server (Genserver) Module . . . . .	174
C.2	Processing the Request Line . . . . .	175
C.3	Get Connection Details . . . . .	176
C.4	Managing the Response Including Error . . . . .	177
C.5	REST URIs . . . . .	178
C.6	Sample Function Calls . . . . .	179
C.7	Connection States . . . . .	180
C.8	Processing the Requests by Splitting Paths . . . . .	181
D.1	Process Flow . . . . .	184
D.2	Language and Welcome Screen . . . . .	185
D.3	Initial Assessment and First Aid Screens . . . . .	186
D.4	Notice to be Reminded and Guide . . . . .	187
D.5	Survey Form and Re-Assessment Reminder . . . . .	188
E.1	Component Architecture of the Mobile-Sensor Environment . . . . .	189

## LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation, Durability
Amazon S3	Amazon Simple Storage Service
API	Application Programming Interface
EC2	Amazon Elastic Cloud Computing
EIS	Enterprise Information Systems
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IP	Internet Protocol
JSON	JavaScript Object Notation
MBaaS	Mobile Backend as a Service
MCC	Mobile Cloud Computing
OS	Operating System
PaaS	Platform as a Service
REST	Representational State Transfer
SaaS	Software as a Service
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
UCC	Ubiquitous Cloud Computing
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WS	Web Services
XML	Extensible Markup Language



# CHAPTER 1

## INTRODUCTION

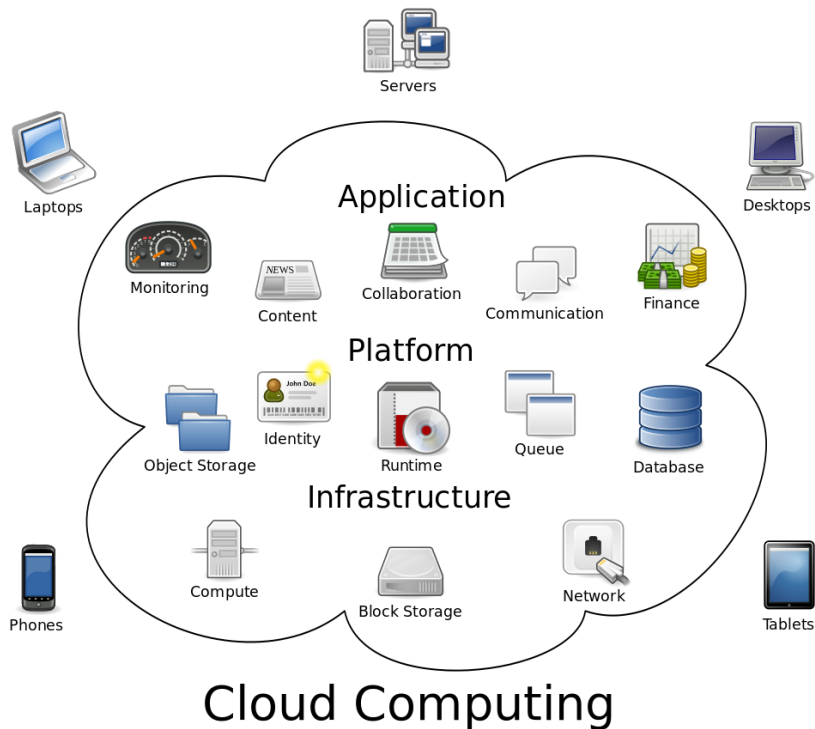
### 1.1 Background

Cloud computing which is a paradigm that facilitates the consumerization of virtualized applications, servers, storage, and network services from providers has become the backbone of most companies [1] [2]. Today, enterprises (i.e., services providers) expose multiple cloud services to their employees and other companies who act as services consumers. Moreover, cloud services providers especially from the commodity cloud suppliers are guaranteeing high services availability for anytime access [3] [4]. The commodity cloud (also known as, the utility cloud) is the cloud services delivery model that follows the pay-according-to-usage policy. This follows the business model of delivering the cloud services as a utility and billing the service consumer on pay-as-you-go bases. Thus, companies that do not have the financial muscle to deploy their own internal cloud architecture (i.e., private cloud) can purchase services at a reasonable price from providers. The importance of cloud computing for companies and consumers, as detailed in Chapter 2, is enormous. Some of these include: cost management as companies can cut down on their internal IT budget, improved maintenance culture as the task of infrastructure manageability is delegated to the cloud service provider, and soft-real time services delivery as downtime is low [5].

The architectural overview of cloud computing is diagrammatically illustrated in Fig. 1.1. The figure shows the three major categories of the cloud computing delivery model which are: Software-as-a-Service (SaaS) – services as applications, Platform-as-a-Service (PaaS) – services for development, testing, interfaces, etc. and Infrastructure-as-a-Service (IaaS) – services that support virtualization such as network, servers, etc.

As cloud computing is receiving much attention, there is one big question that draws urgent attention. That is, how do we compliment the anytime-access possibilities of the cloud with anywhere-access capabilities?

The answer to the above question has been the focus of most researchers who proposed the *Mobile Cloud Computing (MCC)* [7], which is the facilitation of mobile devices as the cloud services consumption node. The mobile devices complement cloud computing facilities by offering ubiquitous (i.e., anywhere) access to enterprise information [8]; a major revolution that Mohiuddin et al. [9] describe as 24x7x365 mobile access which is also promising for the mobile commerce space. Furthermore, most of today's smartphones and tablet devices in production have good backings for varied networks and protocols. Thus, connectivity can



Source: [6]

Figure 1.1: Cloud Computing

be established between the mobile devices and the cloud clusters via networks such as 3.5/4G and Wi-Fi [10] [11]. The success of mobile cloud computing is evidenced in some enterprise revolutions such as mHealth [12] [13], m/ubiquitousLearning [14], mCommerce [15], mobile emergency preparedness services [16], mobile in academic curriculum definitions [17] and so on.

But, what is more prevalent today is the consumer attitude where users own multiple mobile devices. It is a common phenomenon these days to see a single consumer who owns at least a smartphone and a tablet device; and these consumers expect to have application, data, and services consistency across these personalized devices. The era of supporting  $n$ -devices of a single user is dubbed *Ubiquitous Cloud Computing (UCC)* since consumers are no longer tied to a single client node but the options are wide for data access on multiple devices. Also, enterprises are exposing multiple cloud services which are relevant for consumers. This facilitates the provisioning of cloud services from multiple providers in a single workflow to support the multiple consumer devices of a single user. For instance, there is an increasing amount of standalone mobile apps that integrate email services, file storage and syncing services, and calendar and notification services. These services can be offered by different providers but they are presented to the consumer as a single service on the consumer's multiple devices. The paradigm where  $n$ -devices of a single user are enabled to access multi-cloud services is referred to as the *Personal Cloud* [18].

A reason the Personal Cloud has come to stay is evidenced in the rapid adoption of the Bring-Your-

Own-Device (BYOD) trend at the workplace. This means, employees as well as service consumers can use their personalized devices to consume corporate and enterprise services. This is to further fuel the enterprise workforce who need to access services on the go and cannot be tied to a single client platform. According to Intel, the presence of BYOD at the workplace is a major revolution that cut across industry and geographical boundaries [19]. The same company attests to the fact that the leap in personally owned device usage by employees between 2010 and 2011 from 3000 to 17000 has gained productivity by 1.6 million hours. Thus, the BYOD paradigm will be encouraged to facilitate ubiquitous access to services.

Currently, there is not enough research on the personal cloud (e.g., its architectural deployments, market analysis and impact, performance, etc.) apart from Cantara [18] who speculates in July 2012 that the successful deployment of the personal cloud will require the implementation of cloud services brokerage. Couple of years later, most of the research on brokerage services still focus on enabling interoperability and portability of applications across multiple cloud providers [4] but not offering mobile specific solutions. Also, there is a recent push for the deployment of a cloud model that is known as the *Mobile Back-end-as-a-Service (MBaaS)* [20]. While MBaaS is not directly linked to the personal cloud, the initial designs seek to enable third party services integration into mobile specific applications. This includes the design of back-end layers that can facilitate user authentication from third parties and so on. This again leads to the wide gap between research on the personal cloud and the existing up-and-coming solutions. But to offer any solution, there is the need to highlight the problem areas that this dissertation is focusing on.

## 1.2 Problem Statement

The cloud is a reliable back-end infrastructure. This means the cloud as a service guarantees anytime access (i.e., the facilitation of high availability with minimal downtime). The question therefore is how do we provide the same guarantee for anywhere-access? The answer to this question puts forward mobile cloud computing research where the mobile is expected to complement the cloud scenery with ubiquitous access. Ensuring seamless transactions in the mobile cloud ecosystem is a big challenge in itself [7]. Mobile computing has challenges that directly translate into mobile cloud architectures. Some of the challenges are: sporadic disconnections, fluctuating bandwidth, and tightly controlled energy budget that is dictated by the device's battery. The direct effects of these challenges are high communication latency, inability to propagate data and information within the mobile cloud ecosystem, and failed data and application state management. Also, the discussion on extending corporate services and data to mobile consumers (whether employees or customers) requires some level of audit to ensure confidentiality, integrity, privacy, and security especially now that there are several incidents of cybercrime [21].

The question that arises is how can enterprises enable the mobile consumers (i.e., the employees or other companies) to access cloud services seamlessly in the era of the personal cloud? To answer this broad question, there are some issues that must be explored such as:

### **A. How do we ensure consumer consistent experience?**

Since wireless communication channels can be unreliable, there is no guarantee that services or data will be available on the mobile all the time. While the consumer is in a disconnected state, updates can be pushed to any of the back-end services by other consumers. This phenomenon leads to other questions such as:

- (i) How do we detect and push new updates to the consumer irrespective of which service is updated?
- (ii) How do we propagate the updates to the active device of the user automatically without the user manually polling for the data?
- (iii) How do we ensure application consistency from a user's perspective?
- (iv) How do we ensure group data and file synchronization?

### **B. How do we ensure the aggregation of the services on a consumer's multiple devices?**

Since there are multiple services as well as multiple devices of a single user, there is the need to propose a services brokerage platform that can aggregate the services to the n-mobile devices. The brokerage platform can be a middleware or a Mobile Back-end as a Service (MBaaS); however, there are some questions that need to be answered regarding the brokerage service.

- (i) *How do we ensure the scalability of the broker?* Scalability is a key requirement in the area of distributed services and systems since it is paramount for systems to accommodate increasing throughputs (or workloads). The approaches that this work will adopt include the deployment of a centralized and distributed MBaaS. So, the question that will need serious attention is how can the MBaaS which will act as a middleware be deployed to accommodate high throughput requests from the users, as well as the increasing data processing from the cloud sources?
- (ii) *How do we ensure agility of the brokerage framework?* Most companies will like to change the structure, services, and the focus of their job process execution in order to maintain their businesses in an agile economy. Hence, most companies keep changing infrastructure or adding new ones. The question therefore is how can the broker be deployed in a way that changes to cloud services selection can be incorporated without challenges?
- (iii) *How do we support concurrency?* There is the need to support multiple connections of concurrent mobile devices when the brokerage platform is deployed.

### **C. How do we ensure fault tolerance/Error Recovery?**

Since the mobile environment experiences errors (e.g., termination of connection), there is the need to investigate the best approaches that can aid the mobile architecture to be resilient. For instance, how does the system recover from connection error? And how does the system determine faults and react accordingly to the error?

## 1.3 Summary

To address the research questions highlighted above, this dissertation puts forward an architectural design that seeks to ensure seamless enterprise data and application consumption in mobile cloud architectures, which are the main requirements of the personal cloud. The conceptualized architecture is multi-tier and enables n-devices of a user to access multi-cloud services through a designed brokerage platform - emulating a MBaaS. The dissertation further provides test cases using the public cloud services (such as Amazon Web services, DropBox and MEGA), several smartphones (Android and iOS), and some real world adaptation of the designed framework (e.g., the Clandestine Anomaly Augmented Reality Game by ZenFri Inc Canada). Preliminary evaluation results of the designed system, including real world test data, shows that the aforementioned research questions in the problem statement section are well addressed.

Further, key issues that are considered in the development of the architecture are expounded in the literature review section to know the current state-of-the-art approaches by the research community. These areas include:

- Cloud Computing
- Mobile Cloud Computing
- Ubiquitous Cloud Computing and the Personal Cloud
- Provenance
- Services Synchronization in Mobile Distributed Systems and Brokerage Services

The work is structured as follows. The next section details the existing works in the areas of cloud computing layers, mobile cloud computing, and the ubiquitous cloud computing. The same section highlights the unanswered questions in the area of the mobile ubiquitous cloud computing (personal cloud computing) and the specific research questions under consideration in this dissertation. Chapter 3 focuses on the deployment of the proposed cloud services brokerage for mobile ubiquitous cloud computing and the implementation of some real world and prototypic applications. Chapter 4 is dedicated to the evaluation of the designed architecture to determine whether the research goals are met or not. The work concludes in Chapter 5 with the research contributions and further outlook.

# CHAPTER 2

## LITERATURE REVIEW

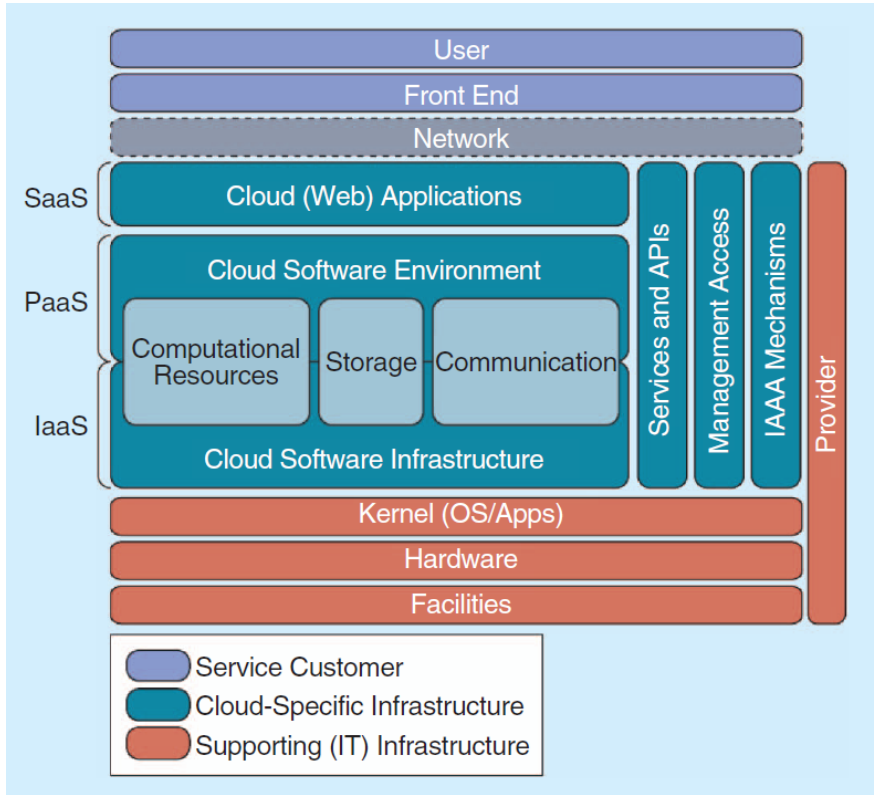
In order to understand mobile cloud computing platforms and paradigms, this chapter focuses on the existing works in the areas. The purpose of the chapter is to carry out detail investigation into the current deployments of Mobile Cloud Computing (MCC) services and how the MCC era is transforming into UCC as well as the new opportunities that the UCC era will provide enterprises to support consumers (e.g., employees) who own mobile devices. As well, major areas are explored that can potentially affect the deployment of the UCC services such as application and data state management, synchronization, etc. These macro issues will be explored with respect to the specific challenges that are present in mobile distributed architectures such as network instability, synchronization, scalability, and reliability.

### 2.1 Cloud Computing (From MCC to UCC and the Personal Cloud)

Though as of now there is no universally accepted definition for cloud computing, I agree with Buyya et al. [22] on the definition of the cloud as: *“a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers.”*

The current macro enterprise IT revolution in the areas of social networking services, Web 2.0, Internet, and so on is heavily dependent on cloud computing. This revolution has propelled the cloud computing paradigm to be placed into a three main taxonomy, as illustrated in Fig. 2.1, and further explained by Li et al. [1] as:

- **Software as a Service (SaaS):** These are application software deployed on remote servers by providers as services to be accessed by consumers over the Internet. Examples of SaaS are E-mail Services, Social media, Salesforce.com, etc.
- **Platform as a Service (PaaS):** These are cloud frameworks that are made available by services providers to be used for application development, testing, and deployment by consumers. Some examples of PaaS are Google App Engine, Microsoft Azure, etc.
- **Infrastructure as a Service (IaaS):** This is on-demand computing infrastructure made available



Source: [5]

**Figure 2.1:** Cloud Computing Stack

to consumers mostly in the form of virtualized servers, storage, and network. Some examples of IaaS are Amazon Elastic Compute Cloud, Rackspace, etc.

The cloud taxonomy has been employed at the enterprise level to provide business agility and better manage the cost of business infrastructural IT. When necessary, the three services (SaaS, IaaS, and PaaS) can be combined to provide composite cloud services to meet specific enterprise workflows. The benefits of cloud computing are enormous – ranging from enterprise business agility and system scalability to data maintainability. However, the services cost charged by the cloud services providers is deemed economical which gives enterprise stakeholders the breathing space to cut down on internal IT infrastructural budget. In general, the services which are made available by cloud services providers on a pay-according-to-usage policy are known as the *public cloud (a.k.a., commodity cloud or utility cloud)*. It is noteworthy to say that some of the public cloud services are also free. Moreover there are other advantages such as on-demand self-service, ubiquitous network access, and network pooling [5].

But as a challenge, stakeholders have raised questions regarding the safety, privacy, potential security risks, assurance, data ownership, and insurance of adopting cloud computing [1] [23] [24]. These questions are mostly asked regarding public cloud usage (this is not to say public clouds are not secured; rather, stakeholders want to know the safety guarantees). In cases where enterprises have huge economic muscles, they can deploy private cloud architectures (a.k.a., internal cloud) thereby enforcing internal security and

data protection [25]. Based on the physically distributed nature of the cloud services, enterprises are further aided to adopt *hybrid cloud* architecture mechanisms. Hybrid cloud is the employment of public clouds and a private cloud in a single architecture in order to achieve enterprise or business-driven goals [26]. With the current outlook of the cloud as well as the future projections, it is very clear that soon most IT related resources will be a service; an emerging cloud domain described as Everything as a Service (XaaS) [3]. The emerging macro trend within the cloud provisioning landscape shows that the consumers will be at the top of the cloud food chain according to the Gartner Report [27]. This creates the need to shift focus from the current enterprise-driven cloud approach to a consumer-driven approach. As far as the consumer-driven cloud is explored, the attention has mostly been on the agile data delivery node (client devices); and how these consumer devices can be facilitated to reliably access the back-end cloud services. While the personal computer (PC) has been at the forefront of consuming cloud hosted data, this assertion changed when the mobile device became the ubiquitous device at the center of human existence [28]. The increasing growth of the mobile landscape coupled with the unceasing agility and expansion of the cloud led to the exploration of “Mobile Cloud Computing” [7] [29]. Mobile cloud computing has witnessed different architectural designs and product delivery models to bring satisfaction to consumers in terms of data access. In the upcoming section, some paradigm defining works within mobile cloud computing are discussed.

### 2.1.1 Mobile Cloud Computing (MCC)

As the cloud computing field is evolving, the mobile landscape is equally growing rapidly. Nowadays, smartphones and tablet devices have established themselves as the dominant nodes to access and consume digital assets [30] [31]. The fact that the cloud is a back-end service and offers anytime access aligns very well with the mobile field which complements the cloud by providing anywhere access. However, there are certain challenges that have plagued the sustainable deployment of mobile cloud computing services. These challenges are summarized below after an exhaustive search and these challenges are the bases for most of the works in the area of mobile cloud computing.

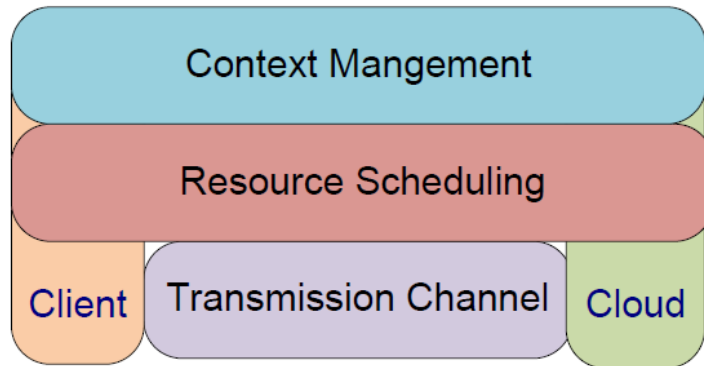
- ***Unstable Wireless networks:*** Mobile devices rely on wireless channels (e.g., Wi-Fi, Bluetooth, Infrared, NFC, 4/4.5/5 G, etc.) to communicate in distributed systems. Despite the advances made in providing these heterogeneous communication channels, these wireless channels experience sporadic disconnections which hampers the smooth transmission of data and communication services.
- ***Limited Bandwidth Availability:*** The available bandwidth in mobile networks is limited in comparison to those in LAN environments. Furthermore, the bandwidth in mobile networks has the tendency to experience fluctuations which leads directly to high communication latency and slow system responsiveness.
- ***Mobile Device Constraints:*** Recently, the resources of mobile devices (especially smartphones and tablet devices) have been improving in terms of processing, storage, and memory. However, the truth



still remains that mobile devices are “thin clients” with poor resources when compared to desktops. Thus, the high demanding workloads that are easily processed in desktop environments cannot be directly translated into mobile networks.

- **Energy Limitations:** The battery life of mobile devices is limited which has the tendency to cause the devices to shut down at times that these devices are needed most.

As researchers are exploring endless opportunities in the mobile cloud computing (MCC) field, their attention has always been on overcoming some of the above listed issues. Conceptually, the MCC architectures have been designed (as depicted in Fig. 2.2) with most of the debate focusing on mobile context management, resources allocation and management (scheduling), and the efficient utilization of the transmission channel [32] [33]. From a previous proposition by Huerta–Canepa and Lee [34] and later supported by Subpratatsavee et al. [35], the MCC architectures can also have components such as applications semantic – for the detection and differentiation of apps such as mutable apps (for barcode readers) and proprietary apps (for specific client tasks). In some systems, there can also be the introduction of Application Programming Interface (API) management components; where the API dictates how the mobile communicates with the cloud services [34] [36].



Source: [33]

**Figure 2.2:** The MCC Design Concepts

Notably, *computational offloading* from the mobile node to the cloud super nodes has been proposed to manage most of the aforementioned limitations [34] [37] [38] [39] [40]. Mobile cloud computing can also be extended to support Mobile Internet Devices (MID) [41] [42]. MID systems (or portable devices) have all the previously listed limitations and the possibility of offloading can be a solution. For instance, Bahl et al. [37] argue that offloading requires functional replication such as location awareness, mobile adaptability, and code block execution between the mobile and the cloud. Thus, the authors opined that current cloud architectures should be enhanced with middle-tier layers which can offer the following supports:

- Minimization of latency between the mobile and the cloud servers;
- Virtual machine (VM) migration into available public cloud in the event of disconnections;

- Uniformity in cloud platforms for resource augmentation; and
- Ability to store replica data.

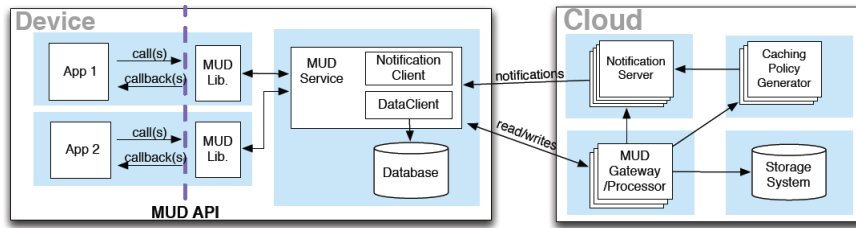
These supports can be facilitated easily now that public cloud providers are offering platform services, application services (e.g., Amazon Simple Notification Service), and context-aware services. The availability of these services can further aid the integration of mobile applications into social networks and sensor networks [37].

The *ECOS* project [40] featured prominently on the list of frameworks that support computation and workload offloading into the cloud from mobile nodes. Basically, offloading requires the delegation of some resource intensive activities to the rich cloud platforms as a way to manage the limited resources of the mobile. However, in multi-user domains such as enterprises, offloading can lead to challenges such as security and privacy breach, and computation and data sharing conflicts. The *ECOS* project seeks to address these challenges and further advanced on the offloading technique by achieving: 1) low latency transfer of computation, 2) offloading based on need with little penalty on energy exhaustion on the mobile, 3) encryption-based security control, and 4) multiplexing of offloaded task. Offloading aids in resources sharing and aids mobile developers to take advantage of the scalable cloud infrastructure.

Another way to enable mobile devices to share a common pool of computational resources is the deployment of virtual private mobile architectures [43] [44]. Primarily, the virtualization of the virtual private mobile environment is to offer mobility-as-a-service (allowing third parties to use mobile networks at low infrastructure cost), sandboxing (isolating mobile components in a network from malware attacks), and latency reduction (since point-to-point redirection can be facilitated). Equally, virtual private mobile networks can aid in reliable content delivery, gaming, and computational offloading [43]. Furthermore, Bifulco et al. [45] advanced on the topic of cloud services virtualization by proposing a more scalable architecture called *Follow-Me-Cloud (FMC)*. The whole concept of the *FMC* is to suspend the execution of application states on one mobile endpoint (due to loss of connectivity or application failure) and resume the application state later in a different environment where connectivity is stable. This approach overcomes the communication failure in a TCP/IP environment that hinders the mobile-server data exchanges when the user is in network blackout zones. At the implementation level, *FMC* is achieved by allowing the virtual machine migration to be handled by the network infrastructure; which aids every process of the data transmission to be transparent to any network within which the mobile is located [45]. Practically, assigned IP addresses to mobile devices in wireless networks change as consumers switch between available network services or change location but *FMCs* allow consistent configuration of IP addresses so the same address can be used across network nodes. Therefore, it is mandatory for the *FMC* to scale in such a way that many devices as well as multiple IP address sustainability can be achieved.

There is also a growing demand for the integration of data and notification services in mobile application systems [46] [47]. *Mobius* [46] has been proposed to offer unified messaging and data serving for Mobile Apps. The researchers opined that current client-server architectures which are directly emulated in the

mobile–cloud environment require consistent network interface. However, this requirement is a mirage since it can never be guaranteed in mobile networks. Hence, Mobius relies on client–side caching to offer offline services. The architectural overview of Mobius is shown in Fig. 2.3.



Source: [46]

**Figure 2.3:** Architectural Overview of Mobius

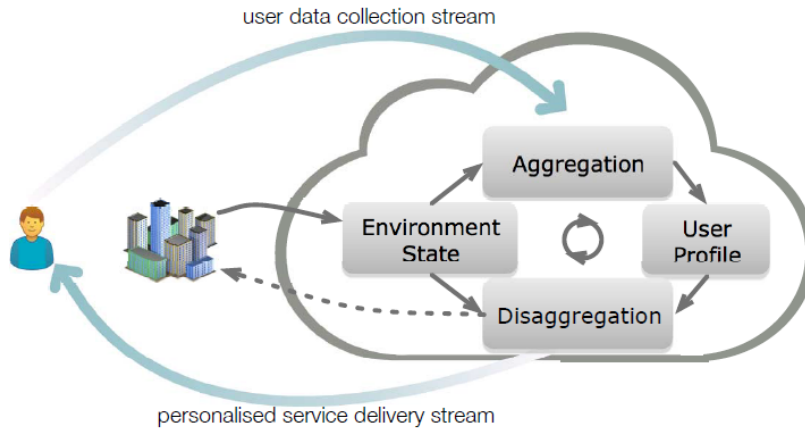
The component labelled MUD is the acronym for Messaging Unified with Data. The MUD is a table–oriented layer that supports read and write operations in soft–real time. Overall, Mobius achieves the following guarantees for the mobile developer community as reproduced from [46]:

- Reading/writing data and messaging
- Flexible real-time notification
- Disconnected operation handling for reading and writing
- Transparent, intelligent caching/pre–fetching
- Hosted, scalable back-end service

*RFS* [48] has also been designed for sharing files in a mobile–cloud ecosystem. Understanding the unstable connectivity challenges in mobile networks, RFS aims at providing a cache component on the mobile end that is synchronized with the back-end cloud. The uniqueness of RFS is that the file synchronization process is adaptable to the particular network protocol. So, if the mobile establishes connection to the cloud over a weak network, the synchronization process follows a pre–fetching methodology. Pre–fetching means the history of the user requests are kept so when similar requests are issued later, the system automatically adds other requests that might have been issued in previous attempts even if these requests are not explicitly called by the user. RFS also has pre–pushing services which work similar to pre–fetching but in this case, the server pushes the files to the client nodes once the server knows those requests may be needed by the user based on the user’s history without the user even making an attempt to request for the data. The combination of pre–fetching and pre–pushing techniques in RFS makes the system exchange files in real time with very minimal energy exhaustion. Before the deployment of RFS, *Wukong* was proposed by Mao et al. [49]. The Wukong platform equally relies on techniques such as: timestamp (to manage conflict between read/write operations applied to files in the mobile to cloud interaction), caching (to minimize the request between the mobile and cloud), file compression (to reduce the file size and manage the mobile storage space), and

optional data encryption mechanisms. Recently, some studies also put forward pre-fetching as a technique to efficiently achieve services offloading in mobile networks [50].

Moreover, Costa et al. [51] observed that there is an assimilation gap between the mobile user generated content and the services delivery models from the cloud providers. We can describe this situation as a conflict between enterprise cloud business models and the consumer demand and expectation. An attempt is made to bridge the two requirements by Costa et al. [51] who propose the *Cloud2Bubble* framework (Fig. 2.4). The framework aims at collecting both user generated content and the environment data (a.k.a., environment perception) based on sensors and offloads the collected data to the cloud platform for processing. In this case, the cloud platform performs the role of decision making, world modeling, and hardware layer. It is also important to state that Cloud2Bubble only provisions services to the user based on the preferences of the user; an attempt that prevents the mobile node from being overloaded with unwanted sensor data and services.



Source: [51]

**Figure 2.4:** Data Flow and Services Delivery of Cloud2Bubble

The collected data is also aggregated so it is presented to the user as a personalized data in a single-dimensional view. Identical to the Cloud2Bubble system is the Cognitive Engine (CE) [52] and context-aware services [53] which are designed for mobile services learning and adaptability to environmental changes. The expectation is that, CE systems are designed to facilitate the mobile to change its behavior based on the environmental data or database information. Thus, the cognitive learning capabilities are parts of the device rather than the user's responsibility. The CE service relies on local events and actions within the operating environment (i.e., network, or application domain). These events and actions are controlled activities by users which involves application requesting, request/response interactions, measurement data, configuration, and configuration feedback. The advancement in CE systems is under-studied but it has huge potentials to solving most recent challenges such as mobile data traffic routing, energy economization, and so on.

Further, the instability of wireless networks has led to questions of data and application state consistency on the mobile. It is important to emphasize that researchers such as Burckhardt et al. [54] and not long ago

Lee et al. [55] argue that eventual consistency methodologies should be adopted. For example, in order to facilitate access to application execution or data in offline mode (i.e., in the case of connectivity loss) mobile applications can be deployed to keep replicas or (mirrored data) in a local cache/storage. This locally cached data can be accessed in the offline mode and can support the user until connectivity is restored. Burckhardt et al. [54] further propose the employment of cloud types which is a way to ensure the same style of data and application formatting across the mobile and cloud components in order to enforce application level uniformity. The recognizable advantage of cloud types is that it ensures consistency at the schema level. This is a way to overcome the integration challenges of heterogeneous application services which are written in JSON, XML, HTML5, and so on. Also, the same authors put forward revision diagrams that can be followed to track changes to the application state. Revision diagrams aid in monitoring not just the stored application but the change set (or revision history) of the application when it was in offline mode. Thus, once connectivity is restored, the mobile application state will be synchronized with the server application state without any conflict.

Another major revolution in the design of mobile cloud services is the *CloneCloud* [56]. CloneCloud is a proposed approach that aids mobile application execution partitioning to rich cloud nodes. That is, instead of offloading all the computational requirements to the cloud node through virtual machines (VMs), the architecture distributes the application execution tasks and decides which states of the application to execute locally on the mobile and which states to offload to the cloud. Such partitioning allows for the synchronization of the application states after the various application endpoints have finished their executions. The partitioning is done in an offline mode where the decision to partition are based on set criteria such as network characteristics, processing power, energy consumption, and computational intensity of the application [56]. The partitioned services are executed in parallel and they can be suspended and captured during the life cycle of the application execution. Similar mobile frameworks to the CloneCloud are the *MAUI* platform designed by Cuervo et al. [57] and the application partitioning approach by Ferdous and Rahman [58].

Similarly, *Cloud-assisted mobile applications (CAM-apps)* perform resource allocation tasks to emulate computation intensive processing in static hardware (i.e., desktop) environments in mobile networks [59]. CAM-apps is a middleware-oriented framework that facilitates job distributions in such a manner that the minimum threshold for services quality is not compromised. The middleware itself on behalf of the mobile and the cloud server performs important roles such as remote service hosting, services management and allocation, and accounting and billing [59]. The idea of resources allocation has also been confirmed to extend the battery life of the mobile devices by Ge et al. [39].

Whether application partitioning, offloading, or virtual migration methodology is adopted, a crucial question to address is the performance of the system. Giurgiu [60] investigates the performance of workload transmission and distribution between mobile-cloud systems. The parameters that account for slowness in communication round-trip time is measured for such distributed mobile services. These parameters are the nature of the application setup, application demand, and workload intensity. The understanding of the

behaviour of these parameters can lead to efficient resource allocation (such as CPU and memory) as well as which services to offload.

Next, the dissertation explores mobile ubiquitous cloud computing.

### 2.1.2 Ubiquitous Cloud Computing (UCC)

In the genesis, the focus has been on mobile cloud computing (MCC) where questions were asked about how legacy services in cloud–desktops (static hardware ) systems can be translated to cloud–mobile architectures. But, recently there is a rapid shift in industrial and academic research toward ubiquitous cloud computing (UCC) largely because, the new consumer attitude in device ownership and services utilization expectations are changing [54] [61] [62]. It is a common phenomenon to come across consumers today who own multiple portable devices (e.g., smartphones, tablets, etc.) and they expect to have their services synchronized on all their devices. We refer to this paradigm of deploying cloud–centric applications that are accessible by a single consumer on n–devices as Ubiquitous Cloud Computing (UCC). Kim et al. [63] on the other hand referred to their seemingly UCC environment as *Mobile Personal Grid (MPG)* as illustrated in Fig. 2.5. While mobile cloud computing is enterprise–driven, ubiquitous cloud computing is consumer–driven; though, van der Merwe et al. [64] used the term ubiquitous cloud computing to refer to the increasing services consumption workload (cloudburst) on enterprise clouds. The motivation for n–device ownership in the modern era by consumers is linked to: preferential display and device manipulation choices, elimination of overhead time introduced for application set-ups, multitasking, backup (fallback) devices, personal ergonomics, and social acceptability [65] [66].

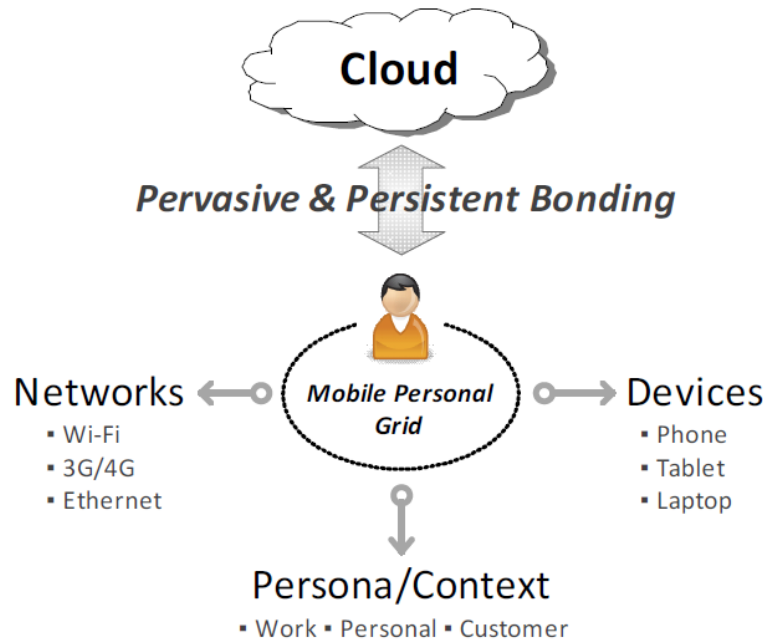
When van der Merwe et al. [64] outline their vision for the ubiquitous cloud computing, they were mainly focusing on how enterprises will adopt the “follow–the–sun” approach to support services infrastructure. For example, when a user moves from one geographical location to the other (e.g., New York to Tokyo), the cloud services of the user can also be migrated to a nearby infrastructure closer to the user’s current location. But, in this dissertation, the ubiquitous cloud computing paradigm is reviewed as a burst in consumer devices of a single user and how these devices can have access to cloud services by maintaining consistent consumer experience.

From the reviews, the UCC architecture is following three design paradigms:

- ***Multiple Consumer Devices to a Single IaaS Cloud Source:***

The employment of multi–consumer devices such as smartphones, tablets, and the personal computer to provide seamless access to data from a single provider can be seen recently. Some examples of products in this line are: the BlackBerry Synchronization Service (BSS), the Android Sync Software, Dropbox, and so on.

- ***Multiple Consumer Devices to a Single IaaS Cloud Integrated with Multiple SaaS:***



Source: [63]

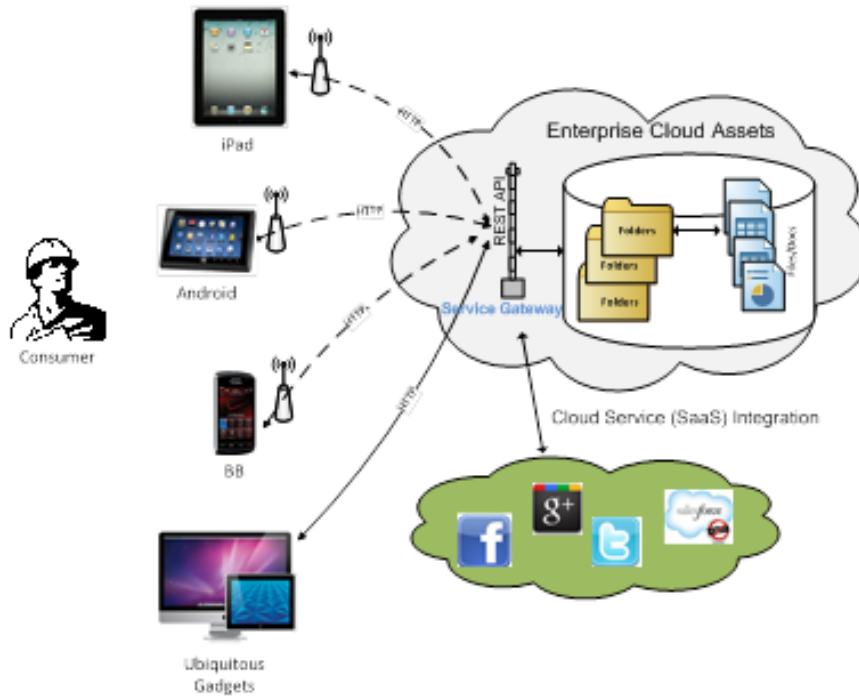
**Figure 2.5:** The Mobile Personal Grid Environment

Though much is not seen of this architectural design from the business to consumer (B2C) market, Intel is pushing in this direction [30] [67]. The company is moving towards mobile device unification due to the steady rise in the number of APIs for the mobile enterprise app store. Fig. 2.6 illustrates the architectural design that emulates the unified mobile architecture proposed by Intel. The approach adopted by Intel focuses on “all inclusive” consumer devices where APIs can be employed to break the barriers that are formed by the mobile vendors. The company also argues that there is the need to provide security gateway services in the form of brokers which should manage the safety of the data in the heterogeneous device ecosystem. The improvement in the architecture proposed by Intel is the integration of SaaS oriented cloud services. The iCloud (<https://www.icloud.com/>) by Apple Inc. also falls within this category.

- ***Multiple Consumer Devices to Multiple SaaS, IaaS, and PaaS Sources:***

The next architectural design that is worth highlighting is the support for multiple mobile devices and the convergence of multiple cloud layers. This is the model that best describes Gartner’s view of the ubiquitous cloud which they dubbed as the “Personal Cloud” [18] [27]. Currently, cloud services providers such as Rackspace (<http://www.rackspace.com/>) is facilitating consumers to integrate services from multiple IaaS cloud sources including private and hybrid clouds. Also, the Locker Project (<http://lockerproject.org/>) which works as a personal container is following this vision. But, before the discussion of the personal cloud, the dissertation discusses some issues pertaining to the UCC design from the research and developer perspectives.

The initial challenge that the developer community faced with the deployment of UCC services is the requirement for writing several codes in different languages for the different mobile platforms. The challenge



**Figure 2.6:** Intel's Approach Towards a Unified Mobile Architecture

arises due to the fact that the underlying operating systems of the mobile platforms are different; thus, their native development environments are also different. For example, iOS powered devices (e.g., iPad, iPhone, etc) support the Objective-C (now Swift) programming language for native app development while Android powered devices (e.g., Samsung Galaxy, Asus Transformer Prime, etc.) support Java. BlackBerry supports Java Micro Edition, and NOKIA and Windows Phone support C# or Python. The heterogeneity in native platform support means deploying UCC services will require knowledge of all these programming languages. Also, if the application is later upgraded, the upgrade has to be applied individually on every mobile platform. Even though a product such as Xamarin (<http://xamarin.com/>) is able to enable cross-platform app deployment natively, the product only supports the direct translation of the business logic of the app but the user interface design requirements vary. There may be future improvement by the company on the user interface integration. Table 2.1 highlights some mobile platforms and the native programming language they support. Though efforts are being made to offer unified native app development, there are still some barriers to cross especially with the mobile vendors.

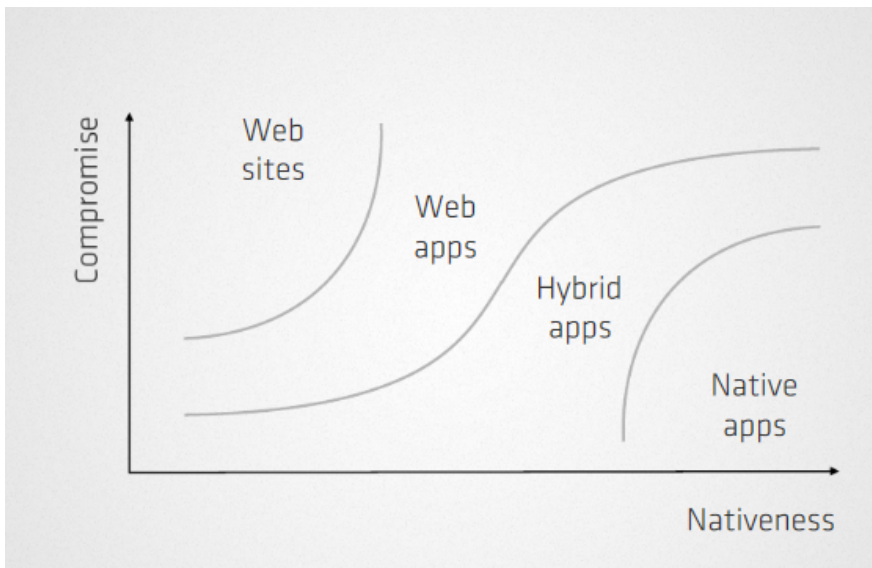
But recently, HTML5 and other Web-based frameworks (e.g., jQuerymobile, jQtouch, etc.) have come to leverage native app deployment on cross-platforms using a single code base. The single code base can be written following the Web standard and the code can be deployed as native app [68] [69]. Also, the fact is that mobile Web apps are different from traditional websites. Mobile Web apps just rely on Web technology services to provide application services. Furthermore, developers can now build hybrid apps where native codes can be combined with Web technologies to deploy standalone applications with full capabilities. The



**Table 2.1:** Native Programming Environments

<i>Device</i>	<i>Native Development / Programming Environment</i>
BlackBerry	Java Micro Edition (Java ME)
iOS (from Apple)	Objective-C and Swift (for Xcode) and C# (for Mono-Touch)
Android	Java
Symbian (deprecated)	C++ and Python
Windows Mobile	C# .NET

paradigm shift is illustrated in Fig. 2.7.



Source: [70]

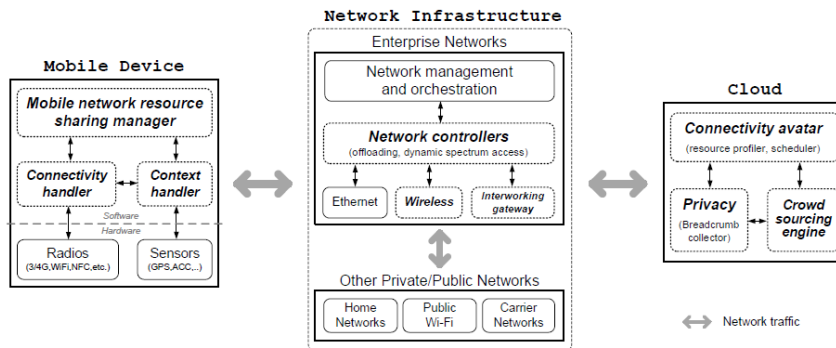
**Figure 2.7:** The Web App as the Bridge Between the Traditional Web Site and Native Apps

Aside the developer challenges, researchers are also proposing different architectural designs for the UCC services. Chen et al. [61] proposed *UbiCloud* which architecturally comprises of four components: the mobile client devices, services app pool, virtual machine pool, and back-end servers. In essence, the server hosts several distributed virtual machines for controlling specific services. Hence, the mobile client devices can communicate with any of the virtual machines over a universal plug-and-play (UPnP) protocol. UbiCloud further relies on Quality of Service (QoS) controls to determine which services to be pushed to which devices of the user. The presence of the QoS feature enables the resource limited devices to tap into the computing capacity of the nearby super nodes in the cloud.

Extending on the Cognitive Engine approached proposed by Hyty et al. [52] for the mobile cloud environment, Gad [14] also put forward a learning and adaptive framework that consists of consumer devices front-end, Web server back-end, and application cloud services. The focus of such learning systems is

to propagate different sensor data (environment data) to different consumer devices based on the devices capacity to propagate or consume that data. The type of data gathered is adapted towards which device is available for the user.

Kim et al. [63] argue that consumers often move between different networks and switch between several contextual environments (i.e., multiple personas such as business and personal). However, there is no guarantee for services and application consistency when the user is changing roles by either moving between public and private networks, or changing the contextual environment. Thus, the authors designed a ubiquitous cloud computing framework which they called *Carmen* (*Cloudcentric Architecture for Rich Mobile Experience Networking*). Primarily, the Carmen framework consists of a cloud-oriented controller (for determining the varied network interfaces that are available within the environment of the user and which devices are connected to which network interfaces), converged access networks (for context-aware interaction and cloud connectivity), and resource virtualization (where both the mobile device and the network are virtualized). The importance of the virtualization technique in the Carmen framework is to identify under-utilized networks and migrate intensive workloads to those environments while less intensive workloads or mobile devices that are in suspended mode can be moved into congested network environments. The technique further maintains connectivity balance and seamless user experience with services accessibility on different devices. The architectural overview of Carmen is shown in Fig. 2.8.

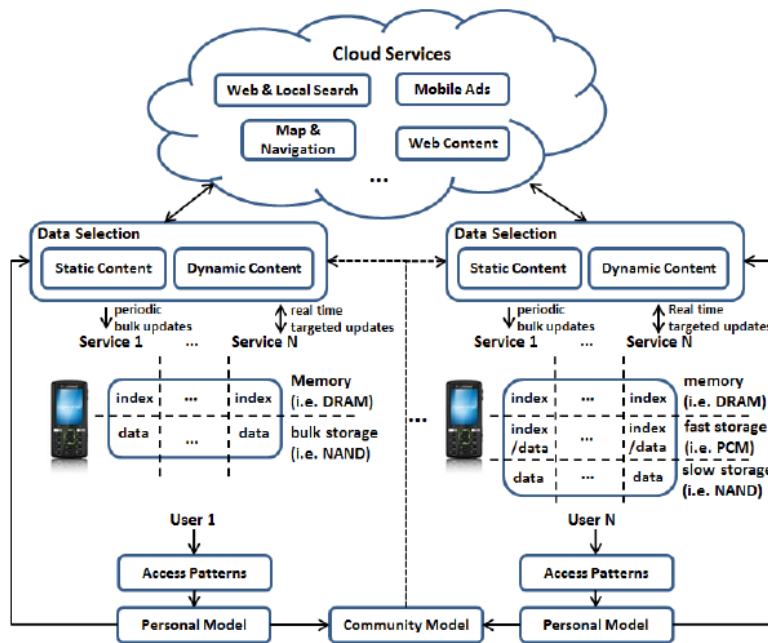


Source: [63]

**Figure 2.8:** The Carmen Architecture

Since the communication between the mobile and cloud nodes requires some guarantee of network availability, the question that arises is whether the n-device ownership trend in consumer attitude can change this network requirement. Koukoumidis et al. [71] have looked into this question by proposing *Pocket Cloudlets*. The idea of pocket cloudlets is to support multiple devices of a user to access cloud-hosted services but; copies of those cloud hosted-services will be kept on each mobile node. This idea according to the authors have some distinct benefits such as: 1) local access to parts or all of the services in the event of network loss, 2) reduce network traffic since the user is not always requesting to the cloud, 3) the fact that the services reside on the mobile means services can be personalized for the device, user, and context, and 4) personalized information can be stored securely without the worry of network attacks. In our opinion, the reason the

pocket cloudlet approach can be crucial for the next generation application deployments is that, mobile to mobile communication services can be deployed where services accessibility will be facilitated in an intra-mobile environment when access to the cloud is evasive. As already posited, mobile devices have wide array of network interfaces (such as Bluetooth, Wi-Fi, infra-red, NFC, etc.); but in a mobile to cloud interaction, communication exchanges occur over Wi-Fi, GSM or any HTTP protocol. So, mostly when we say there is connectivity loss, we are only referring to the fact that we don't have Wi-Fi or GSM access whereas other network interfaces such as Bluetooth can be available. Thus, the advancement of pocket cloudlets technology seems as a way to take advantage of other network interfaces that are supported by mobile devices to access services from other peer devices when services cannot be reached in the cloud. The architecture of pocket cloudlets is shown in Fig. 2.9. Pocket Cloudlets can further lead to new research questions, challenges, and behavioral studies. For instance, can we facilitate services accessibility from a colleague's device when the cloud is not accessible? How will people feel knowing other people are accessing services that they are hosted on their devices considering the fact that these devices are deemed to contain personal information? How do we ensure that in the course of accessing open services, personalized data is not stolen? And the questions go on.



Source: [71]

**Figure 2.9:** The Envisioned Pocket Cloudlet Architecture

Currently, the proponents of the pocket cloud have just envisioned the case of network loss and how that can be supplemented with a mobile services hosting. Additionally, *CIRRUS* clouds [72] extends on the pocket cloud architecture by designing services portability features that can be migrated between the cloudlets and public networks such as Wi-Fi in buses. Thus, such public environments can act as transient platforms for data transmission between the pocket cloudlets. Also, Miluzzo et al. [73] share the same views of the pocket

cloud but they referred to their proposal as *Vision* (to depict the future of mCloud). The hypothetical question that Vision seeks to answer is what will be the opportunities when mobile devices in the near future become as powerful as desktops? In that case, hosting cloud services directly on the mobile will become a very common style of app deployment.

The deployment of UCC services also opens up new dimensions for mobile collaborative services for businesses (e.g., between enterprises and their customers). For instance, research works in [62], [74], and [75] opined that enterprises can collaborate with their clients through exposing services that are consumable on n-devices. In such cases, a unified cross-platform has to be proposed that natively adapts the services to specific devices. In such collaborations, the authors also noted some salient factors that need consideration such as customer decision making and security. The extension on the UCC paradigm of services accessibility using the mobile as the services consumption node is the personal cloud.

### 2.1.3 The Personal Cloud

Soon, there will be a new digital universe which will require the need to have multi control access to devices. The new digital universe which was forecast in the Gartner Report [27] termed the era as the “*Personal Cloud*”, and will provide consumers with the flexibility to use n-devices to access data from multi-cloud providers in order to improve user satisfaction and productivity. Since today’s mobile devices and PCs are enabled to access the same cloud services (document sharing, social media and so on), the consumer-driven cloud is poised to be next revolution in the mobile cloud ecosystem.

The Gartner Report [27] supported the personal cloud claim by enumerating five macro trends that are shifting from traditional IT to form a new cloud ecosystem. These trends are:

- **Consumerization:** Users are becoming the deciding factors for technology innovations.
- **Virtualization:** Changing how IT enterprises are driven to implement applications.
- **“App-ification”:** Shifting focus to platform independent applications which can be consumed by end users.
- **Self-Service Cloud:** Supporting individual use of the cloud.
- **Mobility Shift:** Accessing consumer information from ubiquitous devices anytime-anywhere.

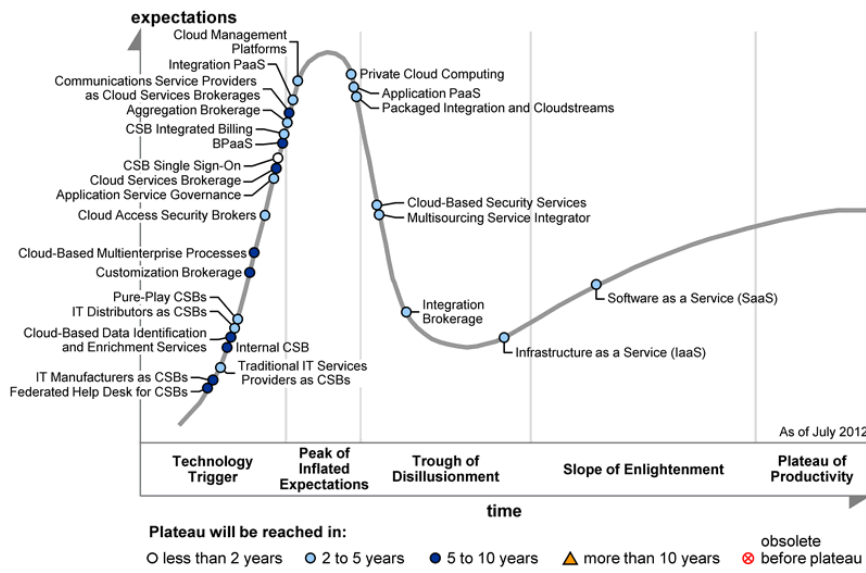
But, the bigger question is how prepared are the cloud stakeholders for the personal cloud? According to the Gartner Report [27], the personal cloud “will require enterprises to fundamentally rethink how they deliver applications and services to users”. While the consumers anticipated the coming personal cloud era, the enterprises did not see it coming. This according to the report is troubling.

As first step, Cantara [18] and other Gartner experts in their publication, “*Hype Cycle for Cloud Services Brokerage, 2012,*” present a theoretical view of a Cloud Services Brokerage (CSB). A role of the CSB is to

provide intermediary services between cloud providers and cloud consumers. The CSB acts as a hub; linking one or more external cloud service providers (e.g. public, hybrid, private) on behalf of one or more end users of those services. In general, the CSB assumes any of the following roles:

- *Aggregation brokerage*: bringing multiple services together to enable business workflow or enhance the consumer experience,
- *Integration brokerage*: using typically SaaS based cloud services to provide an integrated service, and
- *Customization brokerage*: bringing selective cloud services together and re-aligning them to deliver new set of services.

In March 2012, the Gartner experts also present graphically the CSB’s impact on consumers as reproduced in Fig. 2.10. From the graph, it can be inferred that there is a high expectation for technology triggers but the plateau of productivity is shifting towards Software as a Service and Infrastructure as a Service. The illustration from Fig. 2.10 shows that new technologies can be presented by combining existing technologies based on IaaS, PaaS, SaaS, and BPaaS (Business Process as a Service).



Source: [27]

**Figure 2.10:** The Hype Cycle for Cloud Services Brokerage (March 2012)

Proposing a generic architectural design for the personal cloud should be relevant to all the stakeholders (i.e., enterprises and personal users). Furthermore, cloud services delivery models should be considered in the architectural design. These models are: Business-to-Business (B2B) – adopted by cloud suppliers to support other providers and enterprises (e.g., IBM), and Business-to-Consumer (B2C) – adopted by cloud suppliers to support consumers directly (e.g., Amazon Web Services, Google App Engine, Microsoft Azure). Other factors which are equally important for consideration are: Cloud product divergence the possibility of enterprises coming up with unique cloud products that will be “walled garden” is looming, and cloud product

convergence cloud technology and products are converging; at least for now from research and developer perspectives. Today, new smartphones and tablet devices in the market from different vendors are sold with almost identical applications such as map services, weather services, social media apps, inbuilt GPS, etc. by the manufacturers.

Also, there is a growing attention for the deployment of Mobile Back-end as a Service (MBaaS) products [20]. The whole concept of MBaaS is to skew back-end services towards the mobile in order to fill the gap between the PaaS and IaaS at the abstract layer. This means the services will be conscious of the mobile specific challenges and address them rather than leaving it to the developers.

#### 2.1.4 Summary

The advent of paid for virtual computing resources, known as cloud computing, has become a defining moment for the research and enterprise communities. Cloud computing is facilitating high availability of services which otherwise will have been hosted on traditional IT environments. As the cloud field is advancing, the mobile domain is also advancing with the resources (features) of these devices being improved constantly. The mobile devices support consumer mobility and this has made services accessibility ubiquitous without tying the consumer to static hardware (e.g., desktops). As a result, mobile cloud computing has been gaining popularity from diverse stakeholders who are aiming at providing better services. Of great importance is the identification of the following challenges that hampers the successful deployment of mobile cloud computing services: intermittent disconnections, tight energy budget that is driven by the device battery life, fluctuating bandwidth, resource poverty nature of the mobile device features in comparison to their desktop counterparts, and high latency. In an attempt to overcome these challenges, various methodologies have been proposed in different works which can be categorized as follows:

- Virtual Machine (VM) Migration
- Data, Services, Resource, and Computation Offloading
- Partitioning and Parallel Execution
- Resource Allocation

In recent days however, there is a paradigm shift towards the use of n-devices to support cloud services consumption from a single user perspective. This new paradigm is referred to as ubiquitous cloud computing. The challenges in the mobile cloud computing domain apply to the ubiquitous cloud computing domain. However, the ubiquitous cloud computing era is driven by the expectation of consumers who wants to have services and application consistency across n-devices. The future direction of ubiquitous cloud computing is the personal cloud. The personal cloud provides an era of services and device convergence. The convergence of services can be achieved through the deployment cloud brokerage services. So, what are cloud brokerage

services, their challenges, benefits, and potential impact on the cloud and mobile space? These issues are discussed in the next section.

## 2.2 Brokerage Services

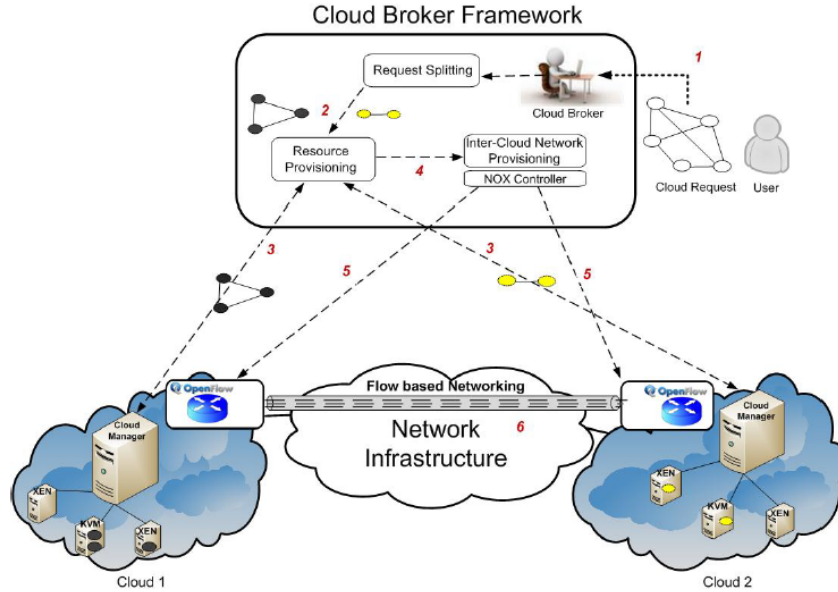
The cloud computing era has witnessed the deployment of cloud-based brokers especially with regards to data and storage aggregation. A possible way to achieve the integration of services goal is to adopt a Cloud Services Brokerage (CSB) [18]. The CSB represents "the market, model, and roles that support the intermediation between cloud services and cloud consumers" [18]. However, this dissertation narrows the scope of the rather broad definition of the CSB to the various models and aggregation frameworks that have been proposed for the integration of multiple consumer devices and cloud services. The brokerage idea is also supported by Intel [67] and Seo et al. [76] who are focusing more on the security issues that can arise from the integration of the consumer devices into the consumer cloud.

### 2.2.1 Broker Frameworks

The broker service offers a uniform interface for users to access cross-domain cloud services, provides monitoring services for both private and public clouds, performs failure and fault detection services, manages resources, migrates resources between cloud infrastructures, and manages interactions management [76] [77] [78]. Apart from the academic-oriented approaches to the service broker deployment, there are some ongoing enterprise oriented projects including well-established broker services for enterprise consumption. The Locker Project for example is an ongoing project that aims at integrating the data of individuals that are scattered across multiple sources. This type of integration is referred to as personal container. But the focus of the project is more on integrating data that are on SaaS-oriented clouds such as Online Social Networks (e.g., Flickr and Facebook). Rackspace also offers real-world broker service which aids customers to aggregate services from multiple IaaS-cloud providers. The service works with private clouds, public clouds, and hybrid clouds. Apart from integrating services, Rackspace also offers hosting services based on pay as you go policy that enables customers to dump their aggregated data on the platform. The Google Project Glass is another evidence of personalized aggregation service that performs brokerage functionalities. The idea of the project is to unify all the digital assets of a user from multiple sources and allow multiple consumer devices to be used in accessing the same data.

Houidi et al. [79] and Aazam and Huh [80] assert that cloud brokerage services are required for inter-cloud networking. The architectural overview of the broker proposed in [79] is illustrated in Fig. 2.11. In the view of the authors, cloud requests from the user can be split and provisioned on multiple cloud platforms. Also, the same user requests can be provisioned over inter-cloud networks that are connected to a network operating system (NOX).

Furthermore, Sundareswaran et al. [81] proposed a cloud brokerage framework that aids consumers in



Source: [79]

**Figure 2.11:** Inter-Cloud Broker Architecture

the selection of cloud products. In the view of the authors, it is challenging to select cloud services from different vendors based on Quality of Service (QoS) and Service Level Agreements (SLAs) so the proposed broker service does the cloud product selection using selection indices. The authors further proposed a cloud provider index which has the following parameters: service type of the provider, security of the cloud provider, pricing units, QoS, Operating System, and so on. But, the whole concept of the authors is based on data aggregation from the service providers. The *Scalia* [82] framework also aimed at aggregating storage facilities from multiple cloud vendors. *Scalia* optimizes the cost of engaging multiple cloud services by providing effective means for data placement. So, assuming it is too costly to keep data  $A$  on cloud  $C1$  in comparison to cloud  $C2$ , *Scalia* will recommend the movement of the data to  $C2$ . A further advantage of *Scalia* is the support for hybrid storage (i.e., private storage services and commercial storage services). This ensures adaptability to data movement and placement whereby services consumers are not over-charged when they can leverage the storage cost with local facilities.

The *ServBGP* framework proposed by Itani et al. [83] is aimed at building on QoS services to ensure services routing in a collaborative environment. Considering the high diversity in cloud provisioning and the increasing service consumer audience, it is no longer an option but a necessity to support services routing. The existing services selection models are based on manual selection by the consumers who can sometimes make mistakes in their choices. Hence, the *ServBGP* framework performs autonomic service selection and routing for the cloud consumer based on the QoS indications from the cloud service providers. The inter-service interaction is facilitated over the Border Gateway Protocol (BGP). The collaboration services that *ServBGP* offer in our view can be explored further to build learning and adaptable broker models that will



provide services recommendations based on the consumer demand trend. Currently, the ServBGP framework relies on advertisement from the service publishers so the broker sends the messages to the collaborators who are interested in those adverts. Also, the adverts serve as a basis to autonomously select the services that consumers may express interest in. Overall, the autonomous decision making regarding services selection is good but the current work is silent on cost analysis. In effect, quality services can be selected for the consumer but the consumer may not be in the position to afford the service. With this limitation, we can argue that consumers should select their own services or the framework should be improved to consider cost and affordability indexes.

There have also been proposals for broker models that aggregate multiple cloud services to provide a SLA-based tiered pricing [84] [85]. In the work of Nair et al. [85], their model which follows the enterprise requirement offers SLA-based services such as identity brokerage entitlement management, policy enforcement, usage monitoring reporting, and network security. Further, the functional overview of the broker model is given in terms of cost, security, and performance management. Regarding security, the authors advocate for the deployment of broker services that ensures confidentiality and protected data transfer between the cloud services and the consumers. The authors also justify the deployment of a storage service on the broker that employs a portal that interfaces the list of provider services. The portal securely guarantees confidential services selection in a manner that even services providers are not able to know the consumer's identity. Noreen et al. [86] have earlier also advocated for similar security requirements of the broker though they were more concerned with the applicability of brokerage layers in a database domain. The *MobiPass* [87] framework is also a local-based broker that facilitates mobile consumers to communicate in a trustworthy environment. For instance, though mobile consumer  $A$  and  $B$  are unknown to each other or they both don't know much about each other, the broker maintains a trust index for both  $A$  and  $B$  since the broker maintains record of their communication activities. So, based on the broker's trust for either  $A$  or  $B$ , the client consumers can use this centralized trust to either communicate with each other or deny the communication.

Furthermore, Li et al. [88] argue that there is the need to investigate the impact of the dynamics of cloud services parameters such as pricing schemes and virtual machines types on broker platforms. The authors further propose virtual machine migration which is based on cloud scheduling. In that case, the broker service has a scheduling optimizer that controls the execution plan between the cloud providers and the users.

There is also a growing concern for control plane integration for cloud services. Control plane is when services are completely composed before their usage. Basu et al. [89] opined that while data plane focuses on services brokerage and services transformation, control plane focuses on services configuration before it is used. Control planes have heterogeneous protocols and are domain specific such as user account creation, account configuration, data exchanges, mapping of user identities, etc. [89]. Control planes have service integration platforms that maps provisioned services that are coming from the providers.

Blum et al. [90] also explored the opportunity of propagating user generated content between services providers and consumers. The authors considered a user generated service workflow which consists of service

providers, mashup studio (for activities such as the design, creation, configuration, and monitoring), execution environment (for services description provisioning, deployment, and management), consumer environment (where services list is authenticated and consume by services consumers). The deployment of the services workflow is dependent on the implementation of a broker service which has functionalities such as policy evaluation and management.

In mobile networks however, challenges such as limited battery life and sporadic network disconnection exists so designing broker services comes with new challenges. For example, in order to achieve services composition (i.e., bringing fragments of software, data, and applications together to form a single workflow), considerations have to be given to failure of data and services state propagation. Chakraborty et al. [91] devoted their work to building a stable environment that can aid mobile consumers with services composition. The authors developed a broker-based composition protocol with the following features: decentralize control (in order to overcome single-point failure challenges in centralize systems), mobility-based dynamic composition model (to determine atomic services), and efficient utilization of mobile resources (to manage the poverty-features of the mobile). The authors further noted that the distributed broker framework offers flexibilities such as composition adaptability to services topology (i.e., the changes that happen to services request), load balancing on the broker, and error minimization for long-running composition tasks. The distributed broker approach is an improvement on the dynamic broker service that was earlier deployed by Chen et al. [92].

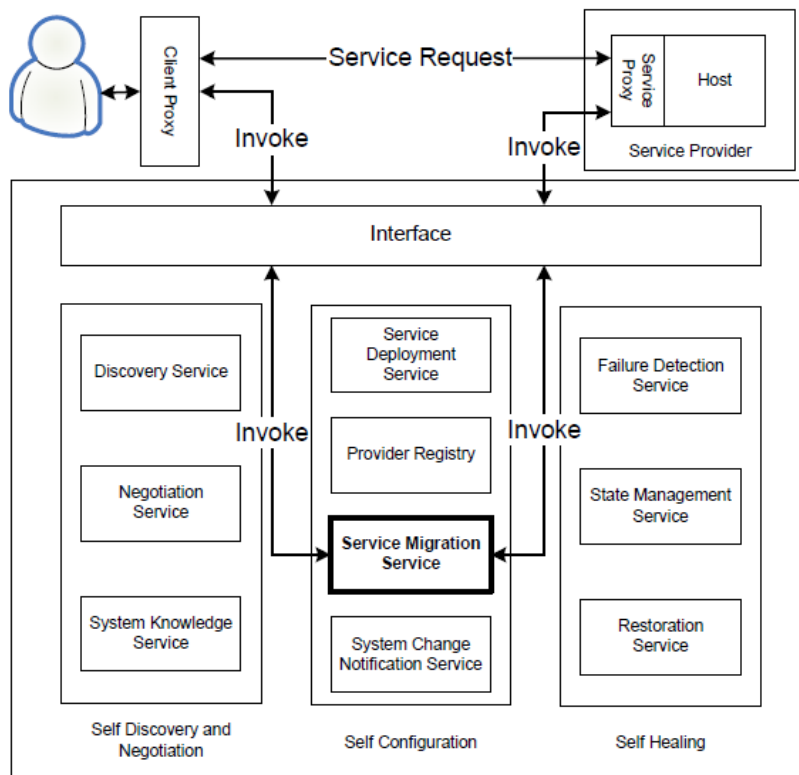
Mobile devices communicate by polling data from static servers. But in situations where the services are composite and the workload is high, the communication latency can be intolerable. The *Mobile Ajax Broker (MaJaB)* [93] is designed to improve on the optimization requirements for web request calls. The Ma-Jab framework incorporates techniques such as scheduling (load adaptability), multiplexing (heterogeneous communication interfaces are aggregated), piggybacking (prioritization of request), and priority-switching (synchronizing background and foreground processes).

Chauhan and Babar [94] further assert that sometimes QoS and SLAs are not standardized and this can lead to integration challenges. For instance, in developing tools for testing (which the authors called “Tools as a Service”), it is required that IDEs and APIs are integrated from multiple sources. These IDEs and APIs can be linked to the developer environment on Eclipse, Tomcat, MS Visual Studio, etc. through brokerage services. Primarily, the brokerage services can be designed to maintain proprietary standardization according to the developers’ specification.

The QoS of broker services has also been studied by Furtado and Santos [95], and Anastasi et al. [96]. These studies focused more on predicting the expressiveness of messages that are disseminated in publish-subscribe environments as a way to enhance QoS. Furtado and Santos [95] however, opined that integrated services should learn from the dynamisms of the changing environment rather than relying on static configurations. For instance, broker services can be configured to behave based on some static conditions from the services providers but once the conditions from those providers are changed, it means the broker will

be inefficient. In order to support the agility of the service providers in a pressurized business environment, the broker has to learn to adapt based on the changes from the service providers in order to enforce quality of service (QoS) aggregation. Furtado and Santos [95] supported their goal by deploying a Contract–Broker (C–Broker) which relies on the QoS indexes that are specified by the user (contracts). The C–Broker has a resource manager and monitors that check the changes that are being applied to the application services and whether those changes should be aggregated or ignored. In order to determine the changes at the application level, the C–Broker performs event sniffing tasks which are controlled by the monitors.

Broker platforms can also be deployed to migrate services between client applications and services providers. Messig and Goscinski [97] proposed a service migration broker that aids Web applications to be discoverable by service consumers (illustrated in Fig. 2.12).



Source: [97]

**Figure 2.12:** The Migration Service Broker

Key features of the broker platform are: transparency during services discovery, data synchronization and update management, QoS negotiation, automatic reconfiguration, and services publishing. The service request between the client consumers and the services providers is facilitated by client proxy and service proxy. The client proxy negotiates transparently with the broker on behalf of the consumer for all requests while the service proxy is maintained by every individual server hosts. The service proxy also negotiates with the broker to allow the utilization of services that are hosted on the back-end. This gives the service broker two roles, service management and service instantiation.

## 2.2.2 Publish-Subscribe Brokers

Publish-subscribe systems have been at the forefront of systems design that focus on information and data dissemination. The publish-subscribe (pub/sub) system is when the information recipients (subscribers) register for particular topics/channels and once those topics are sent by the information provider (publisher), the topics get pushed to the subscribers. The pub/sub system improves on the point-to-point communication by enforcing communication decoupling which improves on communication flow between the message providers and recipients [98]. In most pub/sub systems, brokerage platforms are deployed as centralized or decentralized platforms (also referred to as “hub-and-spoke”). In highly utilized environments where the workload is heavy, the centralized service lacks the capacity to scale. While the decentralized brokers on the other hand support high capacity workloads with good throughput, they also have challenges with synchronization and data communication overhead. Hence, the *PAPaS (Peer Assisted Publish and Subscribe)* framework is proposed by Ahmed et al. [99] who observed that P2P subscription services will scale better taking a cue from BitTorrent and Netflix. The P2P subscription service engages the publisher and subscriber in a direct communication so there is no channel overload on the broker. The P2P subscription scheme has also been proposed and tested by Oudenstad et al. [100] who observed that the scheme facilitates seamless data access in comparison to the conventional client-server scheme.

The discussion on the performance of publish-subscribe broker services is discussed by Farooq et al. [98] especially in the context of mobility message dissemination. The authors argue that publish-subscribe brokers are ideal for services and data exchanges in mobile environments which are habitually characterized by weak network signals and unstable connectivity. Since publish-subscribe systems enforces decoupling, messages which are published can be accessed by connected mobile devices in real time or can be accessed later when disconnected devices later got connected. In a point-to-point message communication however, once the device is not available at the time of message dissemination, the message will be lost. In the mobile environment, distributed brokers can be deployed in cellular networks or on a fixed location (server) that links the publisher’s messages (or events) to the registered message consumers. Since mobile users move from one location to another, they can disconnect from one broker and connect to another broker and still receive and publish messages [98]. But such distributed broker environments need to offer some performance thresholds such as maximum publisher throughput, subscriber throughput, and minimal/zero percentage message loss.

Gaddah and Kunz [101] have also advanced on the mobility broker services deployment by combing techniques such as pro-active caching and neighbor graph. Previously, the systems that support reliable message dissemination to mobile consumers in publish-subscribe environments are built on reactive techniques. Meaning, the mobile in a disconnected state will not receive the published message(s) from the publishers across the broker until the devices re-connect. The challenge also is that, there is no guarantee that the new broker will have the latest messages of the mobile consumer. Thus, the pro-active caching approach is based on studying the mobility pattern of the user and pushing the message state of the user to a hop ahead of the current broker. As a result, the message state is always available on the new broker before the user will connect

to it. The identification of which broker to send the message is based on a neighboring graph whereby the current broker contacts the immediate brokers and delivers the messages of the user before the user gets to those environments. One concern in our view with the pro-active methodology is the accuracy of identifying the next broker and the message throughput so that the wrong broker targets are not overloaded. But, the authors show the success of the approach in terms of measuring the publishing rate, subscriber population, and handoff frequency (i.e., the rate of moving from one broker to the next).

Lundquist and Oukel [102] also aimed at addressing the concern of overloading some brokers while other brokers may be under-utilized in mobility scenarios. The main idea is to introduce density-control functionalities so that the message dissemination workload will be balanced across the brokers. The load balancing is achieved by identifying parameters such as the collection of matching publications, the age of the subscription, the frequency of message replication on neighboring brokers, and the number of unique brokers that publish the message. With these parameters, the authors are able to determine the maximum and minimum utilities of each broker. Hence, once a broker is approaching its maximum utility, services are diverted to other brokers that have minimal utility. But, the diversion of services (including messages or events) is dependent on mutual reachability between the overloaded broker and the under-utilized broker. If an attempt to reach an under-utilized broker is not successful, another broker will be contacted. Also, the *Figaro* [103] framework aims to distribute the load in overloaded mobile and broker networks, where the authors model the mobile devices as agents. Figaro comprises of a middleware broker that maintains storage table for data routing. The uniqueness of Figaro is that, the developers treat the collection of agents accessing a channel as a colony; thus, policies such as banishment, reputation, and fairness can be enforced. As a result, agents can communicate with the broker about their successful and failure transactions with other agents and the broker will decide on which agents need to be updated. The importance of load distribution in pub/sub mobility networks is also emphasized by Salvador et al. [104] in their work which seeks to reduce message flooding. The work considered three parameters which are expressiveness of the message, the scalability of the pub/sub broker, and the degree of mobility support. The mobility support is facilitated by providing user migration capabilities to the broker. Hence, messages are sent as a tuple in the format:

*(message-type; source; destination; payload; timestamp)*

Based on this format, the broker maintains a global view of all activities including the destinations that are being flooded by messages. The use of time-stamp also maintains mobility safety whereby clients migrate from end to end with minimal/no denial of service.

### 2.2.3 Summary

The advent of cloud computing coupled with the elasticity of the Internet is changing the services landscape in terms of data and services accessibility. However, these wide array of services are spatial and scattered with multiple interfaces. The fact that the mobile terrain is also advancing towards ubiquitous services provisioning further leaves the consumer with divergent views on which data access node to employ and

which data to consume.

To manage the problem of services diversity, brokerage frameworks have been proposed to aggregate and integrate the distributed services. The brokerage frameworks provide services composition which aids consumers to consume multiple scattered services in a single workflow. These broker services predominantly are designed to offer the following macro services:

- Load balancing in consumer-provider scenarios with heavy processing demand
- Security and trustworthiness of multiple services consumption
- Resources management and utility control (especially in mobile networks)
- QoS preservation
- Standardization of SLAs
- Services composition

The employment of brokerage services to respond to the consumption of distributed services from multiple consumer nodes has only solved the issue of services control and management. The fact that the services are distributed further provides new opportunities such as i) the services consumption is extended to multiple consumers (e.g., employers, employees, customers, etc.), ii) the services are being published by multiple consumers, and iii) updates to services states are concurrent from multiple consumers. These factors lead to critical issues such as the preservation and trustworthiness of the services (especially data) as well as the credibility of the consumers. In view of this, the dissertation delve into the approaches that have been put forward in the area of data provenance and truth maintenance system.

## 2.3 Provenance

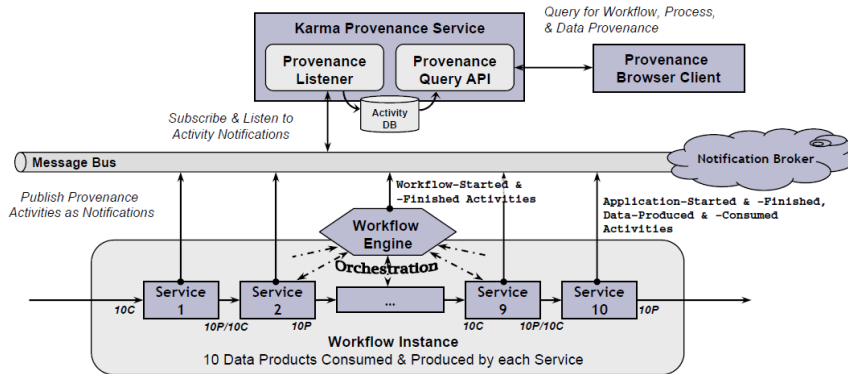
In this section, discussions are presented on provenance methodologies that the scientific community has proposed, adopted, adapted, and designed to enforce services and data assurance.

*Data provenance (a.k.a., Data Lineage)* is the ability to track the life-cycle of data from its creation through to its present stage. According to Chao-Fan et al. [105], “No provenance, no trust”. The data provenance mechanism facilitates derivation of data history which aids enterprises to enforce data transparency, accountability, security, privacy, audit trail, confidentiality, services authenticity, and so on [106] [107] [108] [109] [110] [111].

### 2.3.1 Provenance-Based Frameworks

According to the works shown by Davidson and Freire [112], Simmhan et al. [113], Tylissanakis and Cotronis [114], and Woodman et al. [115], provenance implementation in scientific workflows can lead to reproducibility

and services re-use; while Ikeda et al. [116] posit that provenance can enhance workflow refresh. A typical framework that aims at combining services provenance, workflow, and a broker is the *Karma* [113] system which is illustrated in Fig. 2.13. Further, Knutov et al. [117] noted that provenance can aid in efficient deployment of Adaptive Hypermedia System (i.e., environments that support link traversal) and Lawabni et al. [118] explained that provenance can be used to determine services dependencies and relationships in heterogeneous systems.



Source: [113]

**Figure 2.13:** The Architectural Design of Karma

In enterprise information systems (EIS), data provenance can be implemented at the applications level, workflow coordination and control level, and system level [111]. The adoption of cloud services further fuel the debate on understanding the granularity of services security. By adopting provenance as a security measure, Zhang et al. [119] argue that cloud data and information leakage can be prevented through transparent and accountable event management. Hence, the authors focused on tracking the atomic operation on files in a cloud ecosystem such as: creating a file, deleting a file, writing to a file, reading from a file, moving a file, and renaming a file. But, the main contribution of the authors work is the rule-based adoption for the provenance data traceability. The authors opined that data leakage can happen within the same domain (locally) or across domains. The local rule identifies activities such as file copying, file renaming, and file movements. The cross-domain rules however, track activities such as sender-side logic on the atomic operations, receiver-side logic on atomic operations, and email attachments.

Specific to relational database query aggregation, Amsterdamer et al. [120] [121] propose rules that govern atomic query operations. The authors argue that rules can be applied to operations such as single queries, multiple queries, and composite queries in relational databases. These rules can be defined to address the security issues and the complexities in query generation. Relational database provenance also proves to be efficient for ensuring the quality of the queried data. However, provenance can lead to storage complexity, time complexity, and storage size increment. Also, storing atomic provenance information can lead to data growth where the size of the provenance data can overtake the actual data size. These challenges have been the guiding principles for Bao et al. [122] who seek to reduce the challenges that provenance can introduce in relational storages. The authors noted that previous studies have proposed provenance

tree techniques (as a means of tracing data derivations) but it can be time consuming to traverse the tree later. So, at first, the authors propose rules that govern whether similar provenance data should be copied for identical tuples or referenced; which the authors favor the latter option. Second, the authors propose dynamic programming methodology that optimizes the generation of a provenance tree. Additionally, the *Perm* (Provenance Extension of the Relational Model) framework which is developed by Glavic and Alonso [123] is meant to optimize the query pattern and storage space of provenance data in relational databases. For every data generated, Perm automatically creates a provenance data which is also relational so once a data is queried, the provenance record of that data can equally be fetched without explicitly querying the provenance data for history derivation.

Furthermore, provenance is not just based on policies and rules; but, the roles that the individual system components play. She et al. [124] proposed a role-based provenance mechanism for services aggregation and trustworthiness. In distributed systems, actors such as hardware, software, human operators are all considered as independent services. Thus, enterprises rely on the Services-oriented Architecture (SOA) [125] model to integrate these independent actors into a single business workflow; an approach known as services composition. According to She et al. [124], provenance should be based on the roles that these independent actors play especially, when the actors are from different application domains. Services composition from different application domains is challenging considering the fact that there is no centralize authority. So, the approach proposed by the authors is distributed protocol access control. In this case, the authors considered the physical resources (e.g., files, directories, etc.), data resource (e.g., data and meta-data), and security authority that controls access. Further, the authors explore inter-domain role-based access control (RBAC) which permits users in a domain to have access to resources in another domain through role mapping. In essence, the role that an actor is playing in domain  $A$  is translated to domain  $B$  when the actor moves to that domain. The data quality index in a RBAC is represented as a tuple of the format:

$$Quality_{Data} = (Reliability_{Data}, Trustability_{Data}, Reinforcement_{Data})$$

Where  $Reliability_{Data}$  is the reliability of the data from its source of origin,  $Trustability_{Data}$  is the trustworthiness of the originators and contributors to the data, and  $Reinforcement_{Data}$  is the reinforcement factor which is based on the frequency with which individual system components repeat the generation of similar information.

There are also data provenance mechanisms that are based on group collaborations; which is an extension on the role-based provenance [126] [127]. Enterprises keep local copies of provenance data but there are cases where data tracking is required between multiple enterprises. In that case, while local copies of provenance data can be kept at the individual participating enterprises, there is also the need to keep a global provenance record that aids in determining the activities of other users. In such group collaborations, it is important that enterprises keep sight of internal confidentiality of data so that privacy is not breached. The works by Park et al. [126] and Nguyen et al. [127] propose mechanisms that can aid in the local confidentiality maintenance as well as global provenance tracking. The authors describe a uni-provenance mechanism where all entities



have access to the environmental data and a multi-provenance mechanism where entities keep local copies of their provenance data and only make it available to collaborators from other domains on request and trustworthiness.

Once provenance is enforced, it leads to the tracking of user activities, and recording application procedures. In this regard, we can argue that provenance can potentially risk data and user privacy. Alharbi and Lin [128] propose a privacy-preserving data provenance (PDP) mechanism that aids users to securely access services remotely without risking the exposure of their identities. In such environments, data provenance focuses more on the prevention of unauthorized user activities while relying on group signature to enforce user privacy. But, the PDP approach is feasible when there is a central authority (or centralized trusted authority) that controls the security on the distributed trusted servers (the trusted servers in this case are the hosts and providers of services) and a huge user base ( $U_1, U_2, \dots, U_n$ ). In the system, the provenance data which are kept on the trusted servers, do not contain the information of the individual users but just the operations that have been performed on the data for the benefit of keeping audit trails; but, the information on individual users are kept by the trusted authority. So, user privacy is preserved from the masses and only administrators who are allowed access to the TA can view the user's information. The same concern for securing provenance data has been the focus of Hasan et al. [129] who argue that provenance data can be forged or tampered with when it is passed through insecure networks. In this regard, the authors argue that provenance data should be encoded with cryptographic hash keys before they are transmitted.

While provenance services have been built to track data derivation and services as a user triggered event, there is also the need to look into automatic provenance mechanisms. Braun et al. [130] propose Provenance-Aware Storage System (*PASS*) that automatically collects and records provenance data without any intervention at the user and application levels. *PASS* records provenance data by observing the sequence of execution of processes (or events) at the operating system and kernel level in the Linux environment. Braun et al. [130] further noted that there exist Observed-Provenance Systems (where automatic provenance is enforced through observation of process execution sequence) and Disclosed-Provenance Systems (where users manually expound the workflow composition for provenance tractability). While the two types of provenance services are complimentary, the major challenge is that, the identification of provenance granularity can appear differently at the process execution level from the user expectation. The question that arises then is how to synchronize provenance consistency between the user level and the system level. The authors then proposed versioning control techniques where changes at every level are recorded and once new updates are applied, a new version of provenance records is created. The only critique of the *PASS* framework is that, the conflict resolution in the versions is not discussed. But, the authors rely on provenance pruning to control the provenance record from growing out of proportion.

Provenance storage is very crucial and every effort must be made to curb the provenance data from over-growing. Provenance record is also an extension of the semantic of the actual data [117]. Let's consider a practical scenario: Assuming we want to record the individual read requests to the Google homepage

(<http://www.google.ca>), the size of the page itself is less than 1MB but the number of visitors to the site daily is around 10 billion hits. This means if we want to record only the IP address of the machines that are browsing the site, the provenance file will be approximately 540GB daily. The concern for the minimization of provenance record has been the focus of the work presented by Chapman et al. [131]. The authors propose algorithms that eliminate duplicates of provenance records. First, the authors keep a single hashtable that contains all provenance records as a single long string. Secondly, new provenance records which are found in the string are not stored again but when the provenance record is not on record, then a new file is created that stores the new record. The authors further propose node factorization methodology. Since system nodes record individual provenance records, there is the tendency to have duplications. So, node factorization aids common provenance records to be placed in a single source where pointers are used to refer to them by each node; while the node-specific provenance records are kept by each individual node. Xie et al. [132] also agree that duplicated provenance records should be eliminated since the relationship between provenance record and the actual data sets is a graph. The authors further propose the adoption of compression algorithms for the provenance data in order to maintain the provenance storage. Cheney [133] also argues that provenance record can be down-sided if we can carefully identify what is provenance and what is not.

The question of what should be a provenance record and what should not is detailed in the work of Reilly and Naughton [134]. The authors posit that provenance should be considered at two levels; logical provenance and infrastructure provenance. The former level focuses on the transformation that is occurring to the actual data sets within the system while the latter level focuses on the environment where the data transformation is happening (where the environment provenance can include operating system, date, processor, and so on). This notion is the underlying policy for the *Garm* [135] framework which combines provenance between the data process and the system environment and the work presented by Lim et al. [136] who argue that data derivation history should be linked to the system environment.

Another mega trend in enterprise services deployment and consumption nowadays is collaborative services. The advancement in cloud technologies is facilitating the deployment of collaborative services that aid in workflow composition; however, the begging question is how tracking can be achieved in such environments [137]. This question is answered by providing a three-level provenance tracking namely business level (using product lifecycle through product life status diagram), process level (using workflow instances through workflow action diagrams), and data level (using workflow instances through input, output, and conversion logic) [138]. Also, in a collaborative services environment, process documentation can be employed as a means to ensure trailing analysis [139]. The proposal of trailing analysis is to guarantee services tracking in a dynamic environment at runtime. The process documentation keeps the mapping relationship between the actors of the system, the messages that they are receiving, and the messages that they are sending. Further, with the advancement in Web technologies, collaborative services that are Web-based can rely on provenance to determine the semantic structure of Web data [140]. Provenance record tracking in distributed services can be a daunting task due to the fact that the individual services and data are of different semantics, structure,

and formats. So, to be able to deploy provenance technique for such collaborative system, a mediator (or a broker) can be implemented that converts the services heterogeneity into a common domain model [141].

Also, *Truth Maintenance Systems (TMS)* aid in the reasoning deductions of data changes from its input (creation stage) to its output (usable/processed stage) [142] [143]. Apart from TMS, Assumption-based Truth Maintenance System (ATMS) have also been employed to ensure consistency of data entered into an expert system [144], and provide interface uniformity for services binding [145]. The deployment of TMS is to facilitate the deployment of reasoning services. The combination of provenance and TMS/ATMS can produce highly trustable and transparent services composition.

### 2.3.2 Summary

The advancement in cloud technology coupled with the dynamism of the services landscape is facilitating the deployment of next generation services. The high availability of heterogeneous services and data is further aiding consumers to access collaborative services. The issue therefore is how services and data assurance can be enforced in the era of services composition. To answer this question, data provenance methodology has been proposed by researchers and practitioners as a measure to enforce trustworthiness of services composition.

In this regard, provenance has proved to be an effective means to enforce data and services reliability, tracking, quality, and so on in a community. From the available works, provenance can be enforced in a:

- Group Collaboration,
- Workflow Composition,
- Role-based Service Composition, and
- Rule-based Services Composition.

It is surprising to note that as of now, not many works can be found that enforces provenance in distributed mobile networks. For instance, how do we enforce audit in a system that has mobile consumers and providers who are geographically dispersed? How do we ensure transparency and services consistency in the mobile environment? When updates are propagated to disconnected mobile nodes, how do we ensure synchronization? In the view of this dissertation, some of the provenance mechanisms can be adapted to answer these questions if researchers understand data management techniques in distributed systems.

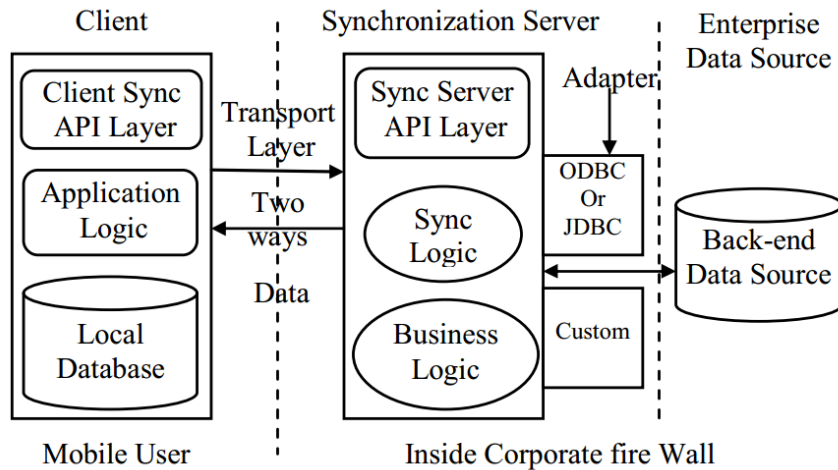
## 2.4 Data State Management and Synchronization

In highly distributed systems such as enterprise information systems with cloud back-ends, the major concern is data state (update) propagation and management [158] [159] [160]. Update management and synchronization of data is crucial for business continuity, real-time and accurate decision making, and responsive system

design. 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 [161] [162]. 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) [163].

The success of state management and synchronization from many sources can be advanced by deploying middleware frameworks. The authors in [161] 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 [164] also shares 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 [165] [166] [167].

Xue [168] therefore proposed a middleware framework (called synchronization server as shown in Fig. 2.14) that aids in reliable and real-time state (data) synchronization in mobile networks. The mobile/client-side application stores the application logic and a cache of the replica data from the back-end data source. The middleware then stores the business logic and the synchronization logic that facilitates the update propagation between the mobile and the back-end. The main objective of the mobile cache is to provide offline data access. Further, the middleware aids in pushing only updates from the backend to the front-end; an attempt that is aimed at reducing network overload. The middleware further facilitates the detection of conflicting updates.



Source: [168]

Figure 2.14: Synchronization Server in a Mobile Network

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 [169] [170]; while, in mission critical systems such as mHealth [171], data propagation delays can lead to undesired circumstances. The concern for latency reduction and real-time data update has been my research focus over the past year. A proposal is put forward for a soft real-time data propagation middleware that aids in data routing among mobile providers and consumers using the health domain as a use case [172]. There are few other works that also employed middleware to support end-to-end mobile communication in the health domain. For example, Arunachalan and Light [173] proposed a mobile agent communication protocol that transmits medical data from the back-end server to the mobile clients. The agent communication protocol interfaces the 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 in [174] [175] provides group collaboration and synchronization of data in the distributed medical domain.

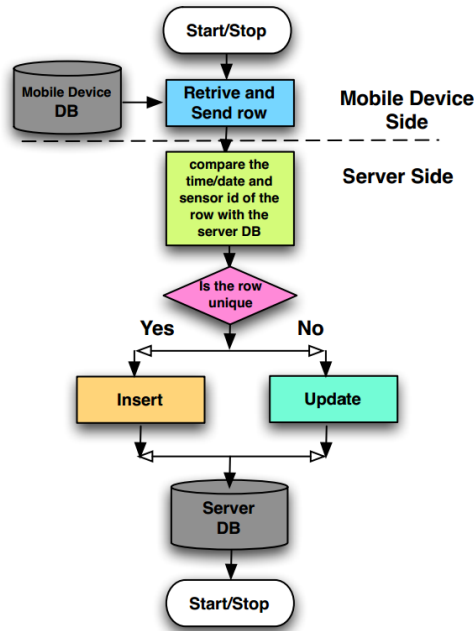
The data stream in distributed services can be in different formats (files, multimedia, etc.) and mostly, these data is transmitted as packets [176]. Boukerche and Owens [177] investigated packet transmission algorithms that enforce quality of service (QoS) in mobile distributed systems that transmit multimedia packets. In such environments, multiple servers are facilitated to send multimedia data to multiple mobile clients as streams where a base station is introduced in-between as a middleware. The question is how to ensure proper arrangement of the fragmented data when they arrive on the client node. The authors therefore focused on the QoS by investigating the impact of the following algorithms: *First-In-First-Out (FIFO)*– the arrival of packet streams are stored in a single queue as and when they arrive on the mobile node in a buffer, *Priority Queuing (PQ)*– priorities are set based on which packet stream should be sent in the order of importance, *Round Robin (RR)*– the buffer is based on FIFO scheduling and queuing and reading from the buffer is also based on FIFO, and *Weighted fair queuing (WFQ)*– the combination of PQ and RR.

But, the introduction of a middleware layer is not the only approach to ensure data synchronization in mobile networks. Choi et al. [178] proposed an ad-hoc communication environment where a central server is eliminated so that mobile devices can communicate directly with peers. The mobile application consists of ad-hoc agent sync for synchronizing application state changes, user authentication layer for authorization, services discovery protocol, and network layer. The synchronization of services is facilitated by implementing timestamps on services so once the timestamp is altered, the application synchronization is issued to an adjacent device.

The advancement in mobile technology has called for client side data hoarding as a means of enhancing high availability of data, transaction, and application in mobile networks. Availability has been a challenge in mobile ad hoc networks (MANETS) a decade ago due to the extreme limitation imposed on the mobile device features [179]. According to Choi et al. [180] [181], mobile hoarding is effective for caching data that is required for runtime transaction. In the event that the mobile is disconnected from the back-end super

nodes, the mobile can rely on the hoarded data to keep the application running. Without the hoarded data, disconnection in the wireless network can result directly into the termination of the runtime transaction. The hoarded data on the mobile has the same schema as the data on the back-end so the system keeps three commit states. There is local commit state where transactions and data are committed on the mobile in its disconnected state, the global commit which gives a universal overview of the transaction in the entire mobile networks so that all transactions appear same on all nodes, and pre-commit state for ensuring availability.

Once data hoarding is adopted, the next problem that is presented is how to ensure data consistency and conflict resolution in the files that are maintained on the various nodes. Marforio et al. [182] referred to application conflicts as “colluding”. Gad [183] put forward a mobile learning environment that enables data pattern detection as the data state changes. The back-end consists of a data structure that is mirrored on the mobile so in case the data on the mobile changes, it directly changes the pattern on the server which means the data has to be synchronized to ensure consistency. The pattern changes are determined by comparing the timestamp, the identifier, and the date of change but the schema is never affected. If these values are different from what is stored on the server, then the incoming data will override the existing data on the server. In this case, there is no versioning. The workflow pattern for conflict resolution in the mobile-server environment is reproduced in Fig. 2.15.



Source: [183]

**Figure 2.15:** Conflict Detection and Resolution Process Execution

Apart from pattern detection or keeping same schema of relational storage on the mobile and the server side, data can be communicated and updated within a context. The *UrbanWeb* [184] framework stores contextual data on a server and the mobile and relies on tagging to determine the inconsistency/consistency in the data. If contextual data is tagged on one node, the information is synchronized on the other node

with the tagged word so that the context will be the same. This type of approach is referred to as *context matching*.

Another key issue in synchronizing data in mobile distributed systems is the security and trustworthiness of the participating nodes. How should we securely transmit data (especially, Personal Information Management)? And how sure are we that the recipient whose resources states have to be updated is genuine? In an effort to answer these questions, Klingelhuber and Mayrhofer [185] opined that the mobile end of the data should be encrypted before the data is sent to the untrusted server. The cost of encryption on the mobile can be expensive since the process requires computation with complexities. The authors show that encryption keys (based on passwords and public keys) can be shared between the mobile and the data sources so that only intended nodes can synchronize the data. In order to manage the limited bandwidth in the mobile environment, the authors propose that instead of the whole state of a resource which is updated be moved to other nodes, only the deltas (i.e., the changes) should be transmitted.

The transmission of deltas instead of the whole data state has been studied by Koskimies [186] and Ratner et al. [187] as a means to enable adaptive synchronization. Since synchronization is needed but can be costly, adaptive synchronization ensures that update management is done based on the availability of bandwidth, and the priority of the data that have to be synchronized. Adaptive synchronization can lead to cost optimization and efficient system manageability [187]. This is the approach that the Open Mobile Alliance (OMA) group is following to deploy the SyncML project [188]. SyncML (Available: <http://technical.openmobilealliance.org/Technical/technical-information/material-from-affiliates/syncml>) is an open standard in the mobile domain that is aimed at synchronizing services in distributed environments. Additionally, Lam et al. [189] proposed the exchange of deltas based on path expression merging to efficiently manage the data transmission cost. Also, Veeraraghavan et al. [190] [191] noted that deltas (which they referred to as fidelity) can aid in the deployment of platform specific data models. That is, high capacity environments such as the desktop can be facilitated to run workload intensive applications while low capacity systems such as smartphones and tablets should have a minimal (low-fidelity) version of the same application. This approach is referred to as “fidelity replication.”

### 2.4.1 Summary

The increasing capacity of the mobile (a.k.a., “pocket supercomputer”) is becoming a defining point for next generation application deployment. Prior to the modern era, enterprise-oriented applications for the mobile were built to access backend services through Ajax requests over HTTP. Such means of communication typically follows the request–response fashion. However, the advancement in mobile technology and increasing consumer demand for services accessibility with the mobile has propelled the developer community to start hosting application states on the mobile. Hosting services on the mobile requires mobile data and transaction hoarding/caching. In this regard, the new form of communication between the mobile and the back-end service is synchronization. But, the process of synching services and application state can be time consuming

and unreliable due to the instability of wireless connectivity.

As a means to enhance the deployment of real-time and mission critical mobile applications, the three-layered architecture has been strongly preferred such that the mobile distributed environment can comprise of mobile consumers, a middleware/mediator, and the back-end services. The introduction of middleware platforms can greatly lead to:

- Services Composition
- Security Enforcement
- Real-time Synchronization of Application State, and
- Data and Application Level Consistency.

Synchronization of data, application, and services is paramount and cannot be avoided in the modern era of mobile system deployments. However, there is the need for us to understand the new dimension of data storage so that we can propose efficient synchronization algorithms. Next Web services design principles and protocols are discussed.

## 2.5 Web Services

Web Services (WS) [192] [193] are network-oriented applications that can be deployed to convey data and information following standards such as SOA and REST. The last decade has witnessed other standards such as Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), Universal Description Discovery and Integration (UDDI), and XML Schema Definition (XSD) in order to ensure data availability and access at real-time. Even with the introduction of the semantic web [194] [195] [196], the above mentioned standards remain the dominant underlying protocols. However, cloud services providers are mostly adopting SOAP and REST web services in today's cloud delivery models since the focus of reaching the consumers is through Application Programming Interfaces (APIs).

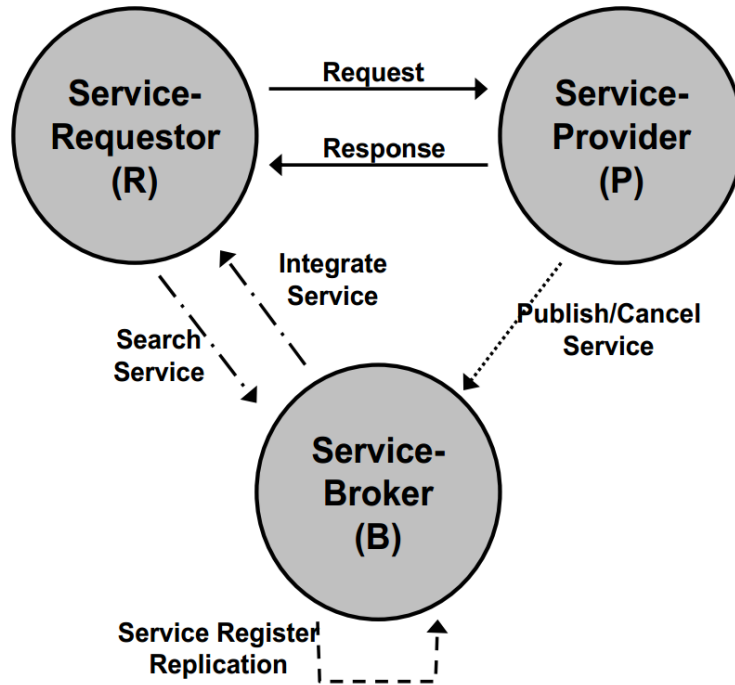
Further, the fact that Web services are distributed aligns very well with mobile architectures. Thus, the consumption of data in the form of Web services in distributed mobile networks, including Web services accessibility in mobile cloud computing, has witnessed significant research. The question however is which of the Web services is more affordable, agile, and lightweight for mobile environments? It is important to state that the Web services protocols were/are not designed specifically for the mobile; but, as the mobile landscape evolves and advances, researchers have seen the need to link the two paradigms. In the next section, some of the works on the various Web services paradigms and their applicability to the mobile domain is expounded.

### 2.5.1 Service-Oriented Architecture (SOA)

The Service-Oriented Architecture (SOA) provides support to various components of Web services to interoperate. SOA focuses on reusability of software and integration [197]. Another key feature of SOA is packaging,



which makes upgrading of legacy services (applications) very fast and at minimal cost. Primarily, the SOA model engages the client (service requester) who sends a request to the service registry (service broker) in search for a particular service. The broker keeps the list of all published services from the available service providers so that when the client's request arrives, the broker only makes the service provider discoverable. The communication within the SOA is shown in Fig. 2.16 as reproduced from [198].



Source: [198]

**Figure 2.16:** Services Oriented Architecture (SOA)

In most cases, the SOA relies on the Simple Object Access Protocol (SOAP) which follows the following messaging structure according to Mizouni et al. [199].

- An envelope that acts as a container and provides description for the message (body) and the instructions (headers) on how the body should be treated.
- Encoding rules for data type expression.
- Routing standards for remote procedure calls and responses
- Standardization for exchanged communication using neutral protocols (e.g., SMTP, HTTP, TCP).

Though the SOA aids Web services developers to overcome interoperability challenges [200], it imposes a lot of limitations when implemented in mobile distributed systems. SOA mostly generates the Extensible Markup Language (XML) since it uses SOAP; and this makes simple communication between system components in mobile distributed systems challenging. The difficulty is due to the fact that SOAP passes large XML data; thus the consumption of data becomes a problem on mobile clients which have limitations of

processing and storage [201]. Another hurdle in the SOA framework is that, to achieve cross platform interoperability between Web services, a lot of standards have to be followed. Though standards such as security, integration and management have been projected for SOA, there is no common platform that integrates all the standards [197]. Hence, today's researchers from both industry and academia have opted for the REST architecture as a preferred methodology for consuming services in mobile networks.

### 2.5.2 REpresentational State Transfer (REST)

The REpresentational State Transfer (REST) has clear semantic structure that aids in resources identification and composition. RESTful Web Service (REST-WS) is a Web service framework that is built on the architectural principles of REST and communicates over HTTP [202]. The REST protocol enables developers to use HTTP methods to interact with the resources. The following HTTP methods [203] [204] currently exist: *GET*– for resource retrieval (fetching/reading), *HEAD*– similar to GET request but the requesting resource only receives the response headers without the entire message body, *POST*– invoked to push or create a new resource, *PUT*– alters (updates) the state of a resource, *DELETE*– for removing the specified resource, *PATCH*– to update parts of a resource without changing the state of the entire resource.

Overall, the REST design follows the following principles:

- Every identifiable entity is a resource and should be assigned unique identifier (ID).
- The key resources should be given Universal Resource Identifiers (URIs) which will facilitate interactions within the system. The URIs provide a global namespace for resource and service identification.
- Resources can be manipulated through representations using uniform HTTP methods.
- Since resources are decoupled from their representation, it makes content accessibility very simple regardless of the format of resource content.
- While resources have states, their interactions should be kept stateless. At the end of every transaction, resources should have information about themselves but not how the last interaction was done.
- Hypermedia as the engine of application state (HATEOAS): In order to navigate between resources, URIs such as hypertext can be used in a resource representation. HATEOAS aids the client to know the next steps to take since the returned URI contains links to available options.

All the above principles are not mandatory to be considered in a single design. Only parts can be employed and such systems are described as low-REST. However, adhering strictly to all the above listed principles is known as high-REST.

In mobile cloud computing and services, previous researchers and works reported in [202], [204], [205], [206], [207], [208], [209], [210], [211], and [212] all proved that adopting the REST design leads to higher scalability, reliability, and decoupling in mobile networks.

On the issue of provisioning services from the mobile node, AlShahwan and Moessner [213] and AlShahwan et al. [214] deployed both the SOAP-based and REST Web services and investigated the performance of each. The results show that the REST architecture facilitates services delivery in shorter time with less error rate. Due to the verbosity of the SOAP protocol with a non-uniform interface, there were significant message losses in the mobile network when the protocol is employed. On the other hand, the REST architecture shows better performance since it supports loose-coupling and has uniform interface that explicitly defines the operation of every HTTP method invocation. The result reported by the authors is also confirmed in our previous work in [201].

Additionally, Mizouni et al. [199] investigate the cost of employing the two Web services paradigm in a mobile context. The authors used the following indices as factors of cost: disconnections during service execution, scalability, and bandwidth consumption. It is found that the REST approach is more scalable since it supports higher message throughput. Also, REST shows minimal request-response time as well as flexibility of allowing phone calls during services execution. Further investigation by Aijaz et al. [215] [216] also confirmed the flexibility of REST as a technique to ensure services transparency.

### 2.5.3 Semantic Web

Another area that is offering value to enterprises today is the Semantic Web which is birthed from the Web 2.0 technology [217] [218]. The semantic web is a vision of Sir Tim Berners-Lee who proposes the idea as an extension to the WWW for more generic data and knowledge exchange [219]. The semantic web aids machines (e.g., computers) to interpret and understand Web contents intelligently which facilitates the accomplishments of tasks such as finding, sharing, and information aggregation on the Web without human intervention [220]. The semantic web technique offers smart ways to extract patterns and build knowledge which has helped in social network analysis, and web structure and content usage [220]. Previously, the web was only understood by humans, which leads to poor content mining at the machine level; but, the advent of the semantic web allows hyperlinks to be described explicitly following a formal semantic for richer mining. The practicality of the semantic web is evidenced in Linked Data which will be described later. Also, the semantic web technology has great potential due to the advent of the modern data economy, popularly referred to as “big data”. Big data is described by Akerkar et al. [220] as:

- *Big transaction data*: Exponential increase and diversity in the volume of transaction data.
- *Big interaction data*: Increase in open data such as social media and device data.
- *Big data processing*: Increasing processing demand on high-dimensional data.

The marriage between the semantic web and big data is what is seen today in most enterprise services deployment such as: Facebook and Google including DBpedia [220] [221] [222].

The works by Eljinini [223], and Lopes and Oliveira [224] show the applicability of the Semantic Web in the health and bioinformatics domains respectively. The Electronic Health Record (EHR) plus clinical

data is growing rapidly without any universal standards. Eljinini [223] realized that most of the EHR and clinical data which are online are in the HTML format which is meant for human readability but not at the machine level. Thus, the author organized numerous websites and built ontology between them that can take the advantage of the semantic web for more accurate knowledge discovery. The ontology is defined as: determining the purpose of each site (i.e., products or services), determine the main entities (classes) to be considered within the site, provide metadata for the classes (e.g., attributes, properties, etc.), establish object-based hierarchy between the classes, transform the ontology into a formal module based on the Web Ontology Language (OWL), and provide documentation. Frber and Hepp [225] provide further details on creating vocabularies for semantic web architectures which is identical to the ontology proposed by Eljinini [223].

Apart from enhancing the knowledge discovery process, the semantic web has also been linked to the improvement on the Web's Quality of Service (QoS). For instance, Garcia and Felgar de Toledo [226] tabulate the QoS of the Web which includes properties such as: Response Time, Latency, Throughput, Scalability, Capacity, Availability, and Robustness. The authors proposed a QoS policy that translates into the semantic web paradigm with higher performance over the traditional web. Moreover, Ketter et al. [227] identified that the combination of semantic web with existing technologies such as Web agents can greatly enhance the decision making process on the Web.

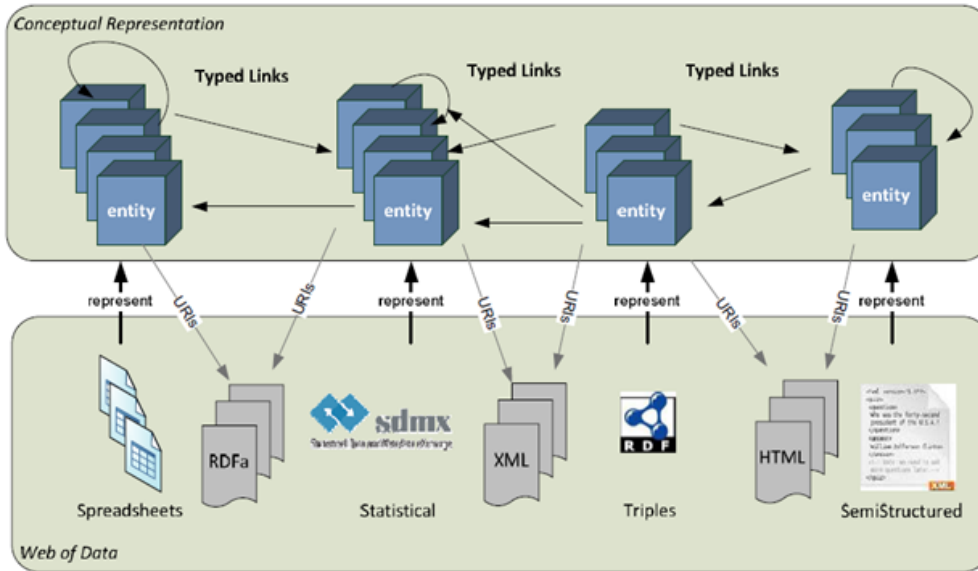
In order to explain how the semantic web works in practice, the concept of linked data and graph-storage is discussed in the next section.

#### 2.5.4 Graph and Linked Data

The Linked Data, also referred to as Linked Open Data (LOD) [228] is semantic web in practice. The paradigm shift in the current structure of the Web from static document to data web where data from diverse domains are inter-linked is changing the requirements for retrieving facts. The structure of the data web is illustrated in Fig. 2.17.

In order to make the accessibility of inter-linked heterogeneous resources at the machine-readable level, *Linked Data or Linked Open Data* is proposed by the World Wide Web Consortium (W3C) working group [228] [229] [230]. The current composition of the data web is in varying formats (e.g., relational databases, XML, CSV, APIs, etc.) which leads to lack of standardization of queries on the Web. Further, the diversity of the web data creates problem for data integration when already, the vast majority of the datasets are in silos (i.e., they are not linked). For instance, there is no linkage between the data of Richard Lomotey on Facebook and Richard Lomotey on Google+. The concept of Linked Data is to bypass the strict requirements of data structure and semantic as well as the identification of the data source; and rather, rely on HTTP Universal Resource Identifiers (URIs) to identify the data entities [228].

Linked data relies on the Resource Description Framework (RDF) to model the data as graph. An example of RDF graph is shown in Fig. 2.18:



Source: [228]

Figure 2.17: Structure of the Data Web

```

<RichardLomotey><firstName> "Richard"
<RichardLomotey><lastName> "Lomotey"
<RichardLomotey><schoolsAt> "UniversityOfSaskatchewan"
<RichardLomotey><supervisedBy> "RalphDeters"
...
<RalphDeters ><firstName> "Ralph"
<RalphDeters ><lastName> "Deters"
<RalphDeters ><profAt> "UniversityOfSaskatchewan"

```

Figure 2.18: Sample RDF Graph

The RDF is a triplet of the structure:

**Subject:** a URI in the namespace of a server (e.g., “RichardLomotey”)

**Object:** a URI in the namespace of another server (e.g., “RalphDeters”)

**Predicate:** the type of link between the subject and the object (e.g., “supervisedBy”)

When the server of the subject successfully links the server of the object using a predicate, the latter responds with a RDF Schema or Web Ontology Language (OWL) definition [228]. The response can also contain other links which leads to higher degree of interoperability.

To query linked data, SPARQL (<http://www.w3.org/TR/rdf-sparql-query/>) can be used.

But what are some of the challenges with Linked Data? In order to answer this question in its simplistic way, Table 2.2 is provided.

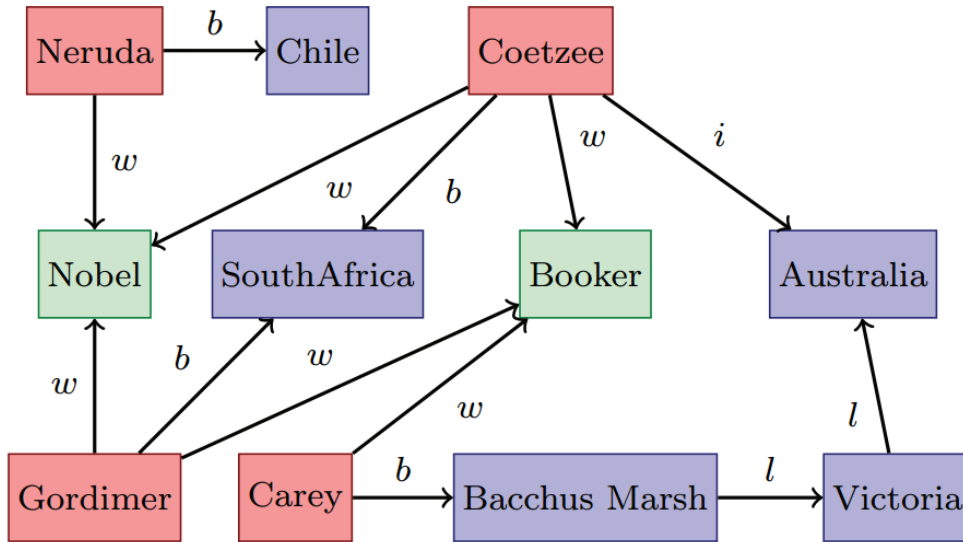
**Table 2.2:** Linked Data and Some Open Issues.

<b>Challenge</b>	<b>Proposed Solutions</b>	<b>Author(s)</b>
How do we trust the data or the quality of the data which is coming from multiple sources	Provenance (for trust and quality control) is proposed.	Sequeda and Miranker [231]
How is the evolving inter-linked structured data preserved?	Provide distributed, service-based infrastructure called DIACHRON, for curation and preservation. Macro techniques such as provenance, change detection, multi-version archiving, etc. are also proposed.	Auer et al. [228]
How do we identify good and useful data sources in the ocean of linked data?	Apply feedback-based technique to identify subsets of the data set; and find the useful domain specific information within the subset.	Rodrigues de Oliveira et al. [232]
How can the enterprise stakeholders take advantage of Linked Data?	Offer Linked data provisioning through multimedia support for content enrichment.	Halb et al. [233]
What are the theoretical foundations for Linked Data traversal that can lead to the identification of data sources during the search period?	Proposed semantic query that is based on the computability of queries.	Hartig and Freytag [234]
How is data published into the Linked Open Data cloud?	Querying using SPARQL	Hausenblas [235]
How is quality data assessed in Linked Open Data (LOD)?	A framework called, <i>Sieve</i> , is deployed for flexibly expressing quality assessment methods as well as fusion methods.	Mendes et al. [236]
Is REST and Linked Data complimentary styles?	The similarities and divergence are discussed.	Page et al. [237]
Continued on next page		

Table 2.2 – continued from previous page

Challenges	Proposed Solutions	Author(s)
How is data integrated at the user interface level with linked data offering uniformity?	Visualization tool is deployed.	Paulheim [238]
How do we ensure the scalability of Linked Stream Data (i.e., data coming from streaming sources such as sensors) integrated with Linked Data?	Continuous Query Evaluation over Linked Streams (CQELS) is proposed.	Le-Phuoc et al. [239]
How are changes tracked in a Linked Data environment?	Events are proposed for tracking changes.	Urdiales-Nieto and Aldana-Montes [240]

Furthermore, as already posited, Linked Data follows the graph model. Graphs are natural ways of building relationships between arbitrary real-world entities and it appears every aspect of our digital lives follows some form of graph. The inherent graph phenomenon can be seen in online collaboration services, crowd-sourcing, online social networks, online purchases, and multi-player gamification platforms.. A typical graph which establishes a relationship between authors and the prizes they have won (*w*) and where they are born (*b*) is shown in Fig. 2.19.



Source: [241]

Figure 2.19: Graph or Network Relationship Between Authors, the Prizes they have Won and where they are Born

Most recently, the enterprises that are leading adopters of graph techniques are:

**Facebook Open Graph (FOG):** Facebook Open graph facilitates the establishment of real-world relationship of a Facebook user through her virtual storyline. For example, the graph API allows the user's activities to be tracked through likes, downloaded applications, purchases, timeline posts, and so on. The tracked activities can then be shared with friends and this allows for a translation of virtual world community into real-world activities. (More is available on the FOG at <https://developers.facebook.com/docs/concepts/opengraph/overview/>)

**Google Knowledge Graph (GKG):** Google relies on semantic-base techniques to enhance the user search experience by implementing the knowledge graph. The idea of the knowledge graph is to pick a user's search term and based on the other similar searches by other users, Google is able to recommend other searches that the user can do without the user explicitly asking for those new search terms. The advantage is that, the user is presented with a single view of distributed search which aids in minimizing the search time that requires the user to jump from one website to the other in search of related materials. Further, the knowledge graph has aided Google to improve on the page ranking. (More is available on the GKG at <http://www.google.ca/insidesearch/features/search/knowledge.html>)

**Bing One-ups Knowledge Graph (BOKG):** This service relies on the Encyclopedia Britannica to provide results to user search queries on Bing. However, if the search term is not found inside the Britannica, the query is extended to other online-based thesaurus such as Wikipedia and Freebase. The whole concept of the knowledge graph deployment is to enhance data organization and presentation of query results. (More is available on the BOKG at <http://thenextweb.com/microsoft/2012/06/07/bing-challenges-googles-knowledge-graph-with-new-britannica-encyclopedia-partnership/>)

**Twitter Interest Graph (TIG):** Twitter interest graph isolates users based on their likes and dislikes which form their interests. This concept encapsulates the user from the generic social graph which is dependent on user nodes (or your relationship to others). Rather, interest graph facilitates a type of relationship where users of common interest are linked. Equally, the dislikes or the things that are not of interest to individual members within the Twitter community can also be extended or explored to discover other knowledge. (More is available on the TIG at <http://blogs.ischool.berkeley.edu/i290-abdt-s12/2012/11/25/analyzing-the-twitter-social-graph/>)

In view of the fact that graphing has achieved significant success, emerging NoSQL databases have adapted the idea. In essence, graph NoSQL databases (a.k.a., Networked databases by Ritter [242]) can be employed to build data storages in a way that can aid in simple queries as well as establishing relationship between the data without writing relational database queries. Relational databases rely on keys and join operations which makes searching tedious and time consuming in data silos. Thus, some enterprises today prefer to store their data in graph databases so that they can improve on the query time [243] [244]. Graph databases support edge-centric query and analysis which aids in the simplification of complex queries [241] [245] [246]. Moreover, graph databases are transactional and are optimized for cross-record connections. At



the moment, examples of NoSQL graph databases as listed in <http://nosql-database.org/> are Neo4J, Infinite Graph, InfoGrid, HyperGraphDB, and GraphBase.

### 2.5.5 Summary

Web services which are network-oriented applications have become inseparable from mobile cloud computing. Most of the mobile applications that are deployed from enterprise applications, social media, mobile collaboration apps, and so on all rely on Web services to exchange data between the various system components. Though a lot of standards have been proposed for the deployment of Web services, the dominant standards are SOA and REST. Hence, cloud services providers expose their services APIs using either of the two standards which aids developers to interact with the cloud product from mobile applications. However, studies on the topic area show that the REST architectural design is more cost effective in distributed mobile networks though the SOA also has its advantages (e.g., interoperability). The REST protocol is found to be lightweight, loosely-coupled, and can lead to higher scalability.

Also, the semantic web, which is originally proposed for better knowledge discovery at the machine-level, is gaining widespread enterprise adoption. More importantly, linked data which is the practical expression of the semantic web follows the graph model. This phenomenon makes the design of architectures a natural fit. Furthermore, accessing data in a graphed storage environment reduces latency.

However, there is a huge research gap between the performance of linked data when the primary consumers are mobile devices. For instance, how are updates in nodes in the graph propagated using the optimal path? How can mobile devices reliably propagate updates to nodes in a graph database?

The questions can be endless and requires a lot of research dedication.

## 2.6 Conclusions and Research Questions

In the near future, the enterprise landscape will adopt mobile technology to empower more than a third of its workforce to support ubiquitous access to services according to the LANDest White Paper published under the title *“Mobility Tipping Points”*. This expectation is influenced by the evolvability and advancement in two macro fields: mobile computing and cloud computing. Cloud computing is the era where services are exposed by providers to consumers over the network as applications, servers, etc. on pay-as-you-go bases. While there are several reasons why companies are embracing cloud computing (including cost optimization on IT infrastructural budget), the quality of service indices are promising. For instance, cloud computing platforms offer guarantees such as high scalability, availability, fault-tolerance, and. These services have been classified as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

In order to compliment the anytime-accessibility feature of the cloud with anywhere-accessibility, some researchers proposed the adoption of Mobile Cloud Computing (MCC). This is to facilitate mobile devices such as smartphones and tablets to consume cloud-hosted services. Despite the advances in mobile technology

in terms of connectivity, storage, and processing power, the mobile technology has challenges that directly translates into mobile cloud computing. Some of the challenges are: sporadic disconnections, fluctuating bandwidth, and tightly controlled energy budget that is dictated by the device's battery. The direct effect of these challenges are high communication latency, inability to propagate data and information within the mobile network, and challenges with data and application state management.

Also, the attitude of consumers and enterprises recently calls for a new research approach. It is a common phenomenon to see users who own multiple mobile devices and they expect to have application and data consistency across the various devices. This expectation is further fueled by the enterprise workforce who need to access services on the go and cannot be tied to a single platform. Moreover, enterprises are exposing multiple cloud services which are relevant for consumers. The era of supporting n-devices to access cloud services is referred to as Ubiquitous Cloud Computing (UCC). The extension on the UCC paradigm to facilitate the accessibility of m-cloud services is called the Personal Cloud.

Based on the architectural designs to meet the expectations of the UCC, the mobile landscape has witnessed several application deployments. In Table 2.3, some services with their features are provided.

**Table 2.3:** Some Existing Mobile Cloud Services

Platform/Service	Feature	Architecture
<i>ownCloud</i>	Privately owned by the University of Saskatchewan. The service allows personalized file management as well as file sharing with others.	Supports multiple mobile platforms and desktops but the back-end is a single IaaS provider.
<i>DropBox</i>	Public cloud service for file sharing. It can also be used for personal file management	Supports multiple mobile platforms and desktops but the back-end is a single IaaS provider.
<i>SkyDrive</i>	Public cloud service for file sharing. It can also be used for personal file management	Supports multiple mobile platforms and desktops but the back-end is a single IaaS provider.
Continued on next page		

**Table 2.3 – continued from previous page**

<b>Platform/Service</b>	<b>Feature</b>	<b>Architecture</b>
<i>Google Drive</i>	Public cloud service for file sharing. It can also be used for personal file management. Also, the service integrates other SaaS platforms offered by Google such as reminders.	Though the service supports only Google services, it is designed to integrate multiple IaaS and SaaS offerings by the company.
<i>Amazon S3</i>	Public cloud service for file storage. The mobile API is available for developers and the company has its own mobile Apps.	The back-end is a single IaaS provider from Amazon.
<i>BlackBerry Synchronization Service</i>	Specifically designed for users of BlackBerry devices to synchronize their information.	Offered to/for only BlackBerry device users.
<i>Android Sync Software</i>	Specifically designed for users of Android devices to synchronize their information.	Offered to/for only Android device users.
<i>iCloud</i>	Specifically designed for users of iOS devices to synchronize their information.	Offered to/for only iOS device users. The company (i.e., Apple) may offer other platform support in the future
<i>Locker Project</i>	This is a personal container for sharing files.	Synchronizes files from different social media platforms.
<i>MEGA</i>	Similar to DropBox and Amazon S3, MEGA can be used for file storage, management, and sharing.	This is offered by a single IaaS service provider.
<i>RackSpace</i>	This is an aggregation service that can be employed to organize data from other IaaS providers	This is offered by a single IaaS service provider but the idea is to support services integration to other services providers.

Looking at the above services (in Table 2.3), it can be seen that the business-to-consumer (B2C) services model is adopted by providers to deliver content but there is no real support for services integration with

other providers. For instance, users are not enabled to access DropBox files from Amazon S3 and the vice versa. The way the existing services are deployed means users have to live with the chaotic divergent of services and decide on which services to use based on their quality of service (QoS) offerings.

While this chapter has extensively reviewed the area, the question on how to enable enterprises to facilitate mobile consumers (i.e., the employees or other companies) to access cloud services in the era of the UCC is not answered adequately. Specifically, there are some open questions which are considered as the research questions to be addressed in this dissertation. The questions are outlined below:

**A. How can consumer consistent experience be ensured?**

There is the need to reduce the latency in mobile networks especially when consuming cloud services. Previous works attempted to address the issue of resources state synchronization in mobile networks in real time. However, the personal cloud in its infancy requires further outlook on answering the following questions:

- (i) How can new updates be detected and pushed to the consumer irrespective of which service is updated?
- (ii) How can application consistency be ensured from the user's perspective?
- (iii) How can group data and file synchronization be ensured?
- (iv) How can soft real-time data accessibility in the mobile network be enforced?

**B. How can the aggregation of the services and subsequently deliver them to the consumer's multiple devices be ensured?**

The reviewed works show strong approval for the adoption of cloud services brokerage to integrate multiple services as well as multiple devices of a single user. However, there are some questions that need to be answered regarding the brokerage service.

- (i) How can the scalability of the broker be improved?
- (ii) How can concurrency be supported?
- (iii) How can fault tolerance be ensured?
- (iv) What is the efficient way to authenticate the user/s from the m-cloud sources and audit?

In conclusion, these questions will be addressed by proposing a personal cloud computing architecture (in chapter 3) that will be evaluated based on the following quality of service (QoS) factors:

- Scalability,
- Soft-real time synchronization,
- Error tracking and recovery, and
- Audit trail through provenance enforcement.

## CHAPTER 3

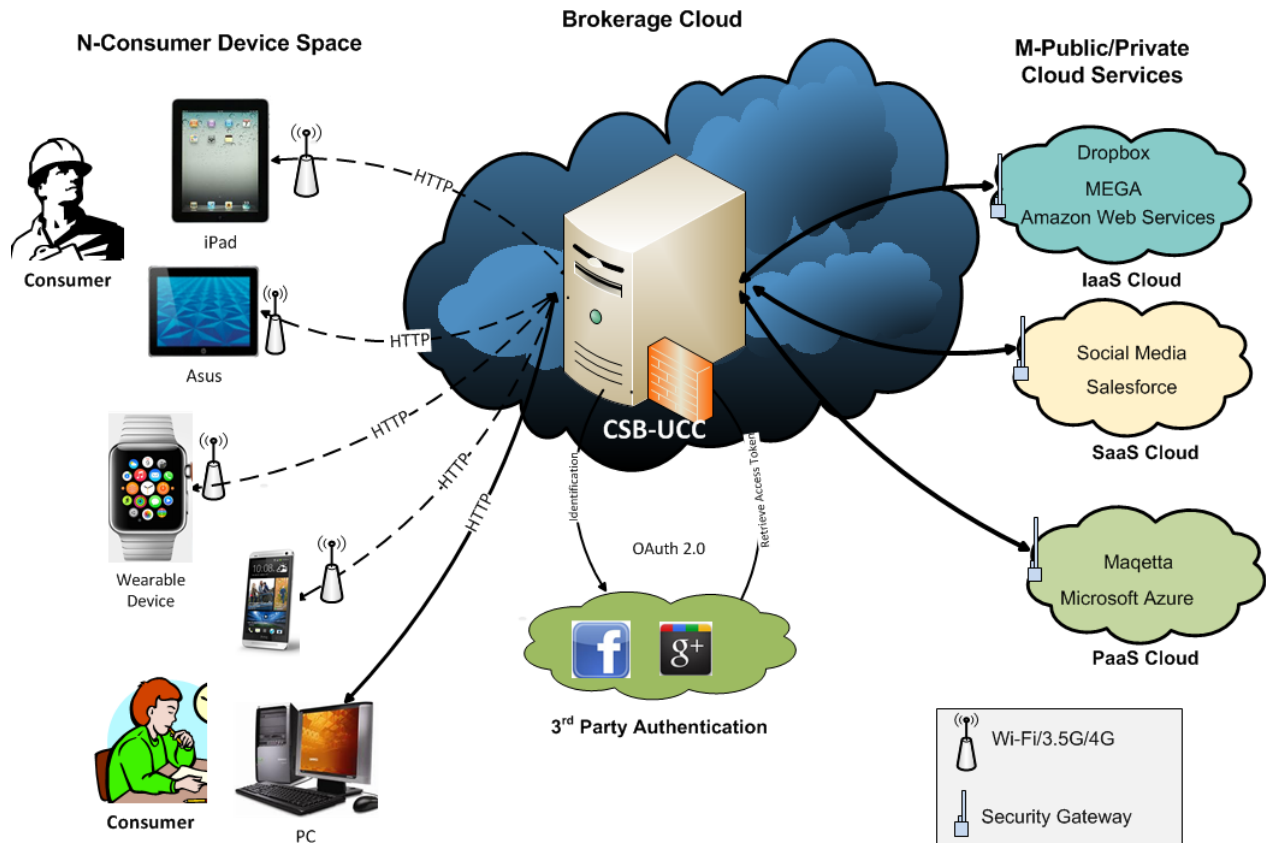
# CSB-UCC: CLOUD SERVICES BROKERAGE FOR MOBILE UBIQUITOUS CLOUD COMPUTING

### 3.1 Overview

The objective of this dissertation is to explore efficient ways enterprises can extend multiple cloud services offerings to consumers who own n-mobile devices (e.g., smartphones, tablets, notebooks, and wearable devices). To achieve this goal, the dissertation proposes a brokerage service called, *Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC)*, that acts as an aggregator that links the devices of a user to the multi-cloud providers, thereby forming an ecosystem of the personal cloud as suggested by Cantara [18]. The CSB-UCC is a 4-tier architectural design as illustrated in Fig. 3.1 with multiple sub-layers. The architecture consists of the consumer devices, an IaaS cloud-hosted broker layer (i.e., broker cloud), 3rd Party Authentication SaaS-oriented cloud layer, and the various multi-cloud providers. The upcoming discussions address each of the layers and their sub-layers. The CSB-UCC focuses on identifying the various consumer devices of a single user and mapping them to the cloud services providers of the same user.

As of now, the existing brokerage frameworks that have been proposed either integrate multiple devices to a single cloud provider (e.g. Dropbox) or aggregate multiple cloud sources to a single user. The CSB-UCC is not aggregating the data on the broker as seen in some frameworks such as RackSpace; rather, the former aims at ensuring services selection on the broker and delivering the integrated multi-cloud services directly to the n-devices of the consumer. First, the high-level deployment is discussed and secondly, the architectural composition of the brokerage framework. A detail justification of the design choices is also provided to answer the research questions raised in Chapter 2.

To address the question of ensuring consistent user experience as described in the previous two chapters, the designed architecture extends on some earlier research. The architecture adapted techniques such as ubiquitous cloud computing, and the publish-subscribe broker for real-time synchronization based on the research by Chen et al. [61] and Ahmed et al. [146] respectively. To address the question of reliability of the broker, the CSB-UCC is designed as a centralized, and distributed broker. This will enable the work to evaluate which of the approaches is ideal for higher scalability and concurrency support. The provenance technique is also investigated by extending on the work of She et al. [124] to ensure audit trail in the entire



**Figure 3.1:** The Generic Architectural Design of the CSB-UCC

system. However, for ease of readability, the main components in the generic architecture (Fig. 3.1) are discussed first. Then, the sub-components are discussed based on the research questions.

### 3.2 The N-Consumer Device Space (Heterogeneous Consumer Devices)

In order to move away from the “walled garden” app deployment where applications or certain services are enabled by vendors only on certain mobile platforms, I decided to open the accessibility of the CSB-UCC to all modern smartphone and tablet platforms including the PC. This approach requires studies on data consumption formats that can enforce platform independent app development. Since today’s cloud services are exposed mainly as REST API [67], the dissertation adopts the REST Web services mechanism. This means the mobile device can access data as JSON or XML and file sharing can be facilitated as well based on the HTTP methods.

Moreover, in today’s heterogeneous networks that consist of Wi-Fi, 3G, and 4/4.5G networks, most of the consumer devices in client-server and events-driven systems are smartphones and tablets, running native apps or mobile Web apps. Mobile Web apps provide the platform for a single code base to be deployed on

variant mobile platforms. The advancement in HTML5 makes the Web app design approach deliver similar and more improved user experience as native (resident) apps. The mobile browser pattern has become the de facto standard for mobile applications since the Web is everywhere. One key benefit of adopting the mobile web methodology is the use of the latest HTML5-oriented web technology frameworks. Web frameworks such as PhoneGap (<http://phonegap.com/>), Sencha Touch (<http://www.sencha.com/>), and jQuerymobile (<http://jquerymobile.com/>) support diverse mobile operating systems and allow mobile web developers to leverage their web technology skills in creating appealing applications. Moreover, these frameworks facilitate dynamic access capabilities to the device native features. However, there are challenges when the mobile Web app approach is employed. Due to browser diversity, the applications deployed tend to have different look and feel and some JavaScript functionalities failed to work on certain platforms (e.g., `alert()`).

Hence, the first task is to build the mobile side of the application in a way to overcome the issues of browser diversity. To address this issue, the client side application is built as a *Platform Independent Model* that has the specific features of each of the client devices. So, depending on the platform on which the application is deployed, the application uses the features of that platform to render the same look and feel. This aids the deployment of the application on the BlackBerry Playbook, Android powered tablets, iPad, iPhone, Windows 7 Phone, and the PC.

Apart from the Web design approach, the native design using development environments such as Xamarin can be adopted. More importantly, the aim of the work is to support any type of mobile design regardless of the underlying design environment. Depending on what the need of the enterprise is, the mobile application can be designed in any language and only has to interface the CSB-UCC layer through HTTP. Services in the form of data exchanges and file sharing can all be facilitated.

### 3.3 Multi-Public/Private Cloud Services

Though the CSB-UCC is a hybrid cloud architecture, each component is serving a distinctive purpose. First, I focus on the Infrastructure as a Service (IaaS) oriented cloud services. Specific to this work and to define a clear focus, the IaaS cloud services under consideration will be the Amazon S3, Dropbox, and MEGA facilities. The facilities are employed primarily as repositories where documents in different formats are stored. The motivation for using the three facilities is necessitated by the use case where I want to validate the employment of the CSB-UCC architecture in a group file sharing scenario. For instance, let's consider the use case below where I assume the Department of Computer Science at the University of Saskatchewan wants to adopt the CSB-UCC:

*I expect that the students have their own Dropbox and MEGA accounts and service to keep their personalized data. On the other hand, we keep a pool of shared materials on the Amazon S3 which can be accessed by the members of the lab. The assumption is that, materials which are on the S3 facility can be seen by all including the tracking of changes to the various data. It is impractical to allow the students to keep the shared*

*data which comprises of huge application codes in their Dropbox account since most of them subscribed only for the 2GB free space. However, there is lab funding to pay for more space on the S3 facility to facilitate data sharing.*

The above scenario applies to corporate entities that want to facilitate their employees to access enterprise data and personalized data via their smart mobile devices. Equally, I expect that students in the department who want to share resources from their private Dropbox account can do so by just sending the data to the Amazon S3 facility by a click of a button on the consumer device. However, the main issue to discuss is the identical security workflow of the three cloud providers. This discussion will further explain the authentication procedure of the CSB-UCC.

Before discussing the solutions proposed in this dissertation, it is important to state that the University of Saskatchewan has deployed a cloud computing service for file sharing called *ownCloud* (<https://owncloud.usask.ca/>). This service is similar to DropBox and SkyDrive but it is privately owned and accessible only by members of the University of Saskatchewan. The CSB-UCC to be discussed in this work varies from *ownCloud*. The latter is a file storage while the former is an aggregation service that allows synchronization of data, services, and applications across multiple cloud services. A cloud service can be aimed at file storage, application state management, services migration, and information sharing.

### 3.3.1 The File Sharing IaaS Clouds

The three IaaS cloud services are all utilized as file and documents storages. Firstly, the dissertation discusses the Amazon S3 service for brevity and show how the security workflow is adopted by the other two IaaS providers. A data container which is called a “bucket” has to be created initially within which the file contents are deposited on Amazon S3. The service also allows the data owner to specify customized metadata of every file uploaded, a feature that encourages the integration of Amazon S3 with other Amazon services. Furthermore, due to its flexibility, Amazon S3 can be used in a composite enterprise architecture that has other cloud framework components such as Google App Engine (GAE) and Microsoft Azure. However, it is important to highlight the peculiar way Amazon S3 enforces security access for data consumption.

The Amazon S3 framework follows the strict security flow defined within the Amazon data protection policy. The service permits hierarchical access to files including usage permissions (read, create, and modify operations) which fit well into enterprise-oriented workflows. Based on the AWS Identity and Access Management (IAM), user policies can be defined. For example, a policy can be defined that grants all access to a user based on the JSON statement below:

```
{ "Statement": [ { "Effect": "Allow", "Action": "*", "Resource": "*" } ] }
```

Amazon also provides an AWS Management Console which allows users to set group policies and create user access levels using a graphical user interface. While the security policies promise consumer satisfaction in terms of data safety and protection, it poses other challenges too especially in mobile networks. Every user (requester) needs to have an *Access Key Id* and a *Secret Access Key* which are assigned by the Amazon IAM



service when the users of the resources are created. Then, a Hash Message Authentication Code (HMAC) signature has to be generated with these credentials, which has to be added to the HTTP request headers for Amazon S3 to authenticate the requester. Based on the signature, Amazon S3 is also able to determine the access level privileges of the requester. Now, it is interesting to state that the Dropbox service and the MEGA facility all have the same security workflow as that of Amazon S3.

In an enterprise where the primary consumers of the business information are mobile clients, the security policy becomes a challenge. For instance, while the data hosted on these IaaS layers have a metadata feature for easy data manipulation, the same feature contributes to HTTP traffic in wireless networks. So, adding additional HMAC signature string to the already verbose HTTP header request is a recipe for increasing latency in a mobile distributed environment which experiences sporadic disconnections. Furthermore, storing an Access Key Id, Secret Access Key, and a HMAC signature in an application domain on the mobile device raises other security concerns since these devices can easily find themselves in the wrong hands. Also, calculating the HMAC signature on the mobile device increases the processing workload of these devices, and high processing workload directly leads to high energy (battery) consumption.

However, there is a bigger security challenge which has to do with the issuance of the key. That is:

*How can multiple students or users be authorized to access a data from a common source (e.g., Amazon S3) on their devices without giving every user the security keys?*

The importance of the above question is the avoidance of storing the security keys within a client application domain and denying the user any knowledge of what the keys are. Also, in case the mobile device gets into the wrong hands, the data cannot be accessed if the security keys are not stored within the mobile application domain.

The answer to this question is one of the motivations for the design of the CSB-UCC and to allow third party security integration from the consumer devices. In the next section the dissertation discusses the details of the SaaS cloud layer which offers OAuth 2.0 security from social media services; specifically, Facebook and Google+.

### **3.4 Third Party Authentication and OAuth 2.0**

The purpose of the third (3rd) party authentication is to enable security authorization from a user's social media subscriptions. Using third party social networking Application Programming Interface (API) in personalized and enterprise oriented applications is becoming very popular for different reasons. But for this work, the motivation for the integration with the social networking services is to facilitate business-to-business (B2B) as well as business-to-consumer (B2C) support for the CSB-UCC framework. The B2B model facilitates enterprise adoption of the framework to provide an authentication layer for sharing documents with other enterprises while the B2C model supports direct end-user (i.e. employees or external customers) access to the enterprise document.

Furthermore, the social networking feature makes the CSB-UCC usable for different enterprises which may have varying business workflow as well as business focus. For instance, an enterprise that wants to support only their internal staff (employees) to access Amazon S3 hosted data can employ the simple login approach. This approach requires that the user provides a username and a password from the mobile device to the broker; and the CSB-UCC checks from its repository of users to determine whether the supplied credentials are valid and matching. Provided the user's credentials match, the CSB-UCC then fetches the user's Amazon S3 credentials and forwards the request to Amazon. In this process of authentication, the enterprise' data accessibility is "sandbox" and only legitimate employees can have access to the data. Also, there is nothing like data sharing; and hierarchical privileges can be enforced. The same simple login can be used to access personalized data from Dropbox and MEGA. But, another point of call is what happens when the enterprise is an advertising or marketing company and will like to target multiple customers outside their facility.

The social networking authentication component allows users of the CSB-UCC to login through Facebook, or Google+. The authentication through the social networking sites is based on OAuth 2.0 technology. Thus, when a user opts to login through a social media say Facebook, the broker forwards the request to the Facebook authentication system where the user will be presented with the option to provide Facebook ID and password. If the user is a valid Facebook account holder, the broker fetches the user's credentials including the Facebook security access token and stores these credentials in the user token repository (a storage area on the broker). But, in an enterprise where security is paramount and only employees are required to access the data on the Amazon S3 facility, the system administrators are strongly advised to pre-populate the user's accounts including the access tokens from the social media as well. In that case, if a user logs into the system and the credentials are not already stored in the broker's repository, the user will be denied access to the data. The pre-population of the social media access token is also to prevent the creation of fake identities for malicious purposes on the Online Social Networks since Jin et al. [147] report on the increasing menace.

Another issue that is observed is the possibility of having one user with multiple social media accounts. To solve this issue with multiple accounts, a graph database is created that links (or maps) the accounts of each user. Since the work relies on the security tokens which are unique for every user account, the user only need to specify his/her accounts and the tokens are linked from the various social media forums. Also, following the OAuth 2.0 technique, an end-user does not need to provide security credentials all the time until the browser session ends.

A third way that the CSB-UCC is useful in the context of social media is the facilitation of an authentication approach that this dissertation describes as the *hybrid authentication mechanism* (i.e. either by social networking or proprietary personal login). Since enterprises have different needs, it is likely to find some that may require the hybrid authentication approach in order to meet their business execution workflow. The proposed framework can deal with such a situation since all the features are available for combinatorial usage. The social media authentication does not influence the Dropbox and MEGA services as much as Amazon

S3 since in most cases, the latter facility is employed by enterprises for group access while the former frameworks are mostly personal. The only advantage it offers Dropbox and MEGA users is the flexibility of using any account and the broker will determine the exact account that the user is registered with on those IaaS providers and issue the request with that account.

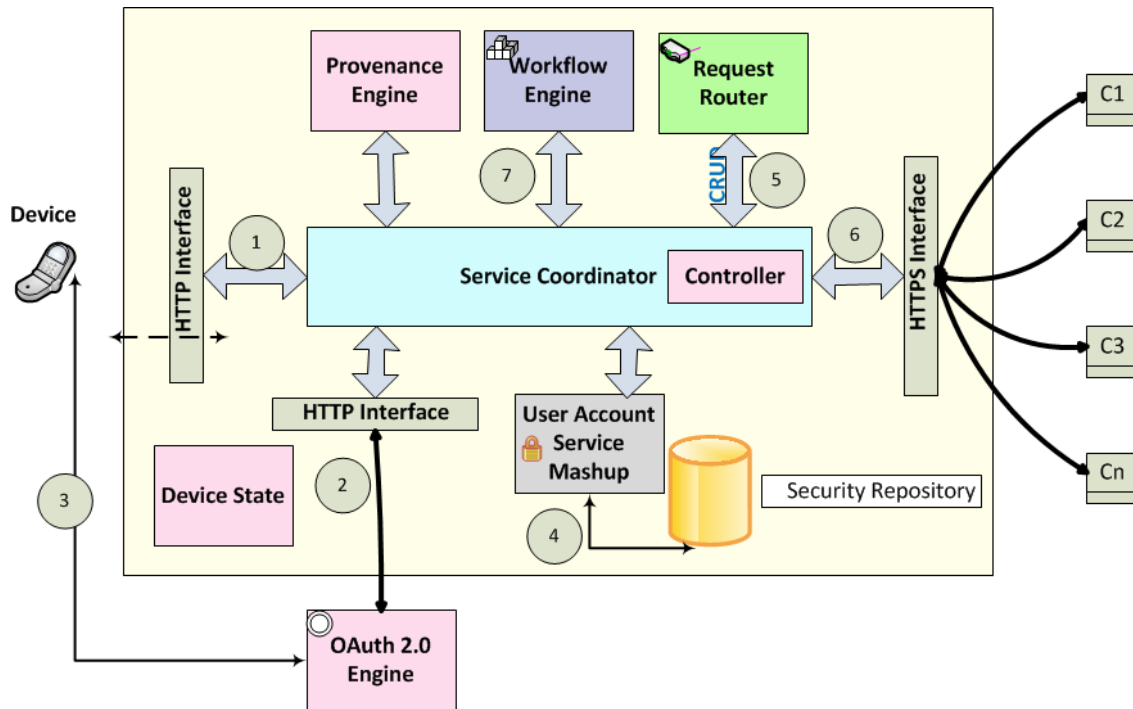
But, there is a security risk that is worth noting. Since, the OAuth 2.0 technique is employed for logging through social media, the mobile embedded browser keeps a session of the login details. Hence, when a user logs into the system on the first occasion through social media say Facebook and the “keep me logged in” option is checked, the subsequent login (within some time frame) takes the user straight to the application interface without the option to provide user credentials. Hence, end-users are cautioned to logout of the application all the time rather than just closing the application. Also, this caution should be a guiding principle for the high security enterprises that will adopt the CSB-UCC framework. To overcome this risk, it is recommend that the browser cache and cookies be programmatically cleared each time the application closes. Also, the user should be signed out automatically after the application is idle for some pre-define time.

### 3.5 The Broker Cloud

The cloud services brokerage (or broker) is an application server that can be hosted on any IaaS compute cloud layer, be it a public cloud (e.g., Amazon EC2) or a private cloud but this work puts forward the latter platform since I wanted to have more control over the security. The broker is the hub that links all the actors (i.e., the consumer devices, the 3rd Party Authentication cloud, and the multi-cloud services) in the entire CSB-UCC architecture. In Fig. 3.2, the anatomy of the broker layer is illustrated with the three HTTP/S interfaces that are exposed to the actors which are external services. The numbers show the initial order of activity flow. Next, I discuss the flow execution of the broker.

First of all, when users launch an application on the device, the client side HTTP interface is called (as shown in **step 1**) which is primarily the Universal Resource Identifier (URI) of the brokerage server. The URI call which is treated as a request is immediately intercepted by the Service Coordinator which is the component that is responsible for the coordination of all the system flow activities. The Service Coordinator routes the request call through the social media HTTP interface in **step 2** to the OAuth 2.0 Engine which is an external authentication system. Assuming it's Facebook, the request is sent to the Facebook endpoint with the credentials.

Typically, when the request is issued to the Facebook OAuth 2.0 system, Facebook provides a login interface that requires the user to input the username and password; this process is captured in **step 3** in Fig. 3.2. After the successful authentication with Facebook, a response in JSON format is returned to the broker through the HTTP interface in step 2. The service coordinator sends the response to the User Account Service Marshup in **step 4**. The security marshup service has a key/value pair repository that stores the



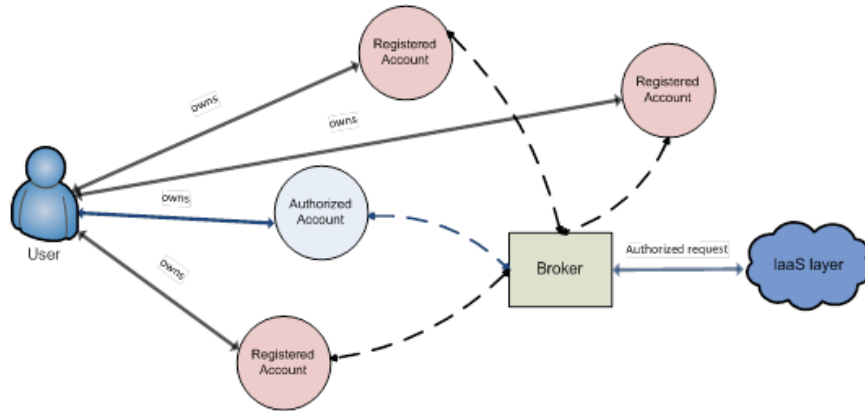
**Figure 3.2:** The Anatomy of the Brokerage Server

responses from the OAuth 2.0 Engine. Since the framework supports several logins, the access tokens of the same user are mapped in the repository using a hash map. So, the system is able to determine the same user even if the user has multiple accounts.

Since the CSB-UCC supports multi-cloud services providers, the broker has a component called service selector which facilitates the user to choose the preferred service. It is important to note that the CSB-UCC is an application middleware that aggregates the user's accounts from social media through graphs but it is not an IaaS or data aggregator. Rather, it is a software layer in front of all the multi-cloud services. So, assuming a user selects the Amazon S3 option, the broker will then employ its Amazon S3 interface and send the request to the Amazon security gateway over HTTPS. After the request has passed Amazon's security test, the broker fetches the requested object and sends it to the mobile requester. The same case applies to the Dropbox, MEGA, or any enterprise file or database services if they are selected by the user. Thus, the broker component also acts as a message *router*. Assuming the request does not pass the security requirements of the IaaS providers, the request will be denied.

It is important to note that each IaaS layer has only one account credentials (i.e., one email account as username and a corresponding password) for every customer's account. Therefore, though mobile users are offered the flexibility of using multiple accounts, the broker can only issue a request to the IaaS facilities with the registered and authorized user account. It is based on this authorized account that the IaaS providers will assign the Access Key Id and the Secret Access Key. Using these credentials, the brokerage then calculates the HMAC signature which is SHA-1 Base 64 encoded that is sent as part of the HTTP headers over the

Internet to the IaaS facilities because it enforces the integrity of the request. For clarity, Fig. 3.3 illustrates the overview of how the broker graphs the user accounts.



**Figure 3.3:** The Link Graph Between a User's Accounts

From the illustration, a user can have an authorized account with which the cloud services can identify the user. However, the user may own other accounts such as emails and social media services (e.g., Facebook, Yahoo ID, Google+ etc.). The other accounts are labelled as registered accounts since the user has stored these accounts on the broker as well. So, what the broker does is anytime the user logs in with any account, the broker uses the linked data (graph) to map the registered account to the authorized account and issues the request with the authorized account. Considering a simple use case: assuming **User A** has a Dropbox account that is bound to the account name *a1.example@usecase.com*, then this email is the authorized account. Now, **User A** may have other accounts such as *a2.example@testcase.ca* which when the user tells the broker about it becomes a registered account. Moreover, the second account can be a social media id. So in case **User A** later logs into the system with the second account, the broker can map the second account to the authorized account and issue the request to the cloud with the authorized account.

Further, it is possible for the registered account to be the authorized account for another cloud layer say MEGA and in that case, the authorized account of Dropbox becomes a registered account. This scenario arises when a user employs multiple accounts for multiple cloud services registration. The graph relationship then becomes advantageous because the user is facilitated to provide any account and the broker will handle the routing of the request with the appropriate authorized account. But first, the user has to specify during the initial configurations which accounts are the authorized ones for which services.

At this stage, this work has successfully answered the main research question on how the user/s of the system can be authenticated from the multi-cloud services. The approach adopted hides the entire existence of the cloud services from the mobile domain. A mobile requester only interacts with the broker or with the social networking service. Also, the mobile consumers (both users and applications) have no knowledge of the security tokens of the cloud services. This can minimize the risk of unauthorized use of the system by unintended users.

Moreover, the entire decentralized authentication system enforces multi-level security throughout the framework. Firstly, the user and the consumer application domain are shielded from the multi-cloud sources (e.g., Dropbox and Amazon S3) and rather, the user is authorized through a third party (SaaS) security layer using the OAuth 2.0. Secondly, the SaaS security service is also shielded from acquiring any knowledge of the existence of the multi-cloud storages/services because the information gathered from the 3rd party cloud layer is just stored in a repository, and the decision to use the stored data to do the actual authentication to the m-cloud sources is handled by a different component on the broker.

In the design of the CSB-UCC, two modes are supported: *offline and online* modes. Assuming the authentication procedure fails in either steps 1, 2, 3, or 4 in Fig 3.2, the application will switch to the offline mode where the users are able to access their documents and modify them locally on their devices but they will not be able to update their files with any of the external cloud storages/services. Further, the users are not permitted to access shared documents/data which are sent to the cloud sources by other colleagues in group sharing scenarios in the offline mode. The users also cannot synchronize their personalized Dropbox files if the application is in the offline mode.

However, the successful authentication process flow automatically switches the application to the online mode. In that case, all the limitations present in the offline mode are overcome and the users can do “live synching”, meaning they can be receiving updates from the cloud storages as well as pushing updates to the backend from the consumer devices in real time. At this point, it is important that we discuss the remaining components of the broker which are actually functionalities that are activated after the successful authentication process.

When a user is successful with the OAuth 2.0 security authentication, the application becomes accessible in the online mode which means the users can perform the **CRUD** (Create, Read, Update, Delete) operations within the system. Currently, I summarized these four operations in two verbs in order to support the REST APIs from the multi-cloud sources. Hence, the system uses the HTTP POST method to either store a new file/data or update an existing file/data. We employ the HTTP GET method to fetch the files/data from the multi-cloud sources to the consumer device storage. When a file is deleted (removed) from the system, we equally treat it as a write operation so we use the POST method to ensure that the file is deleted from the other end.

Furthermore, when an online user issues an HTTP request, the Service Coordinator redirects the request to the Data Router component in **step 5**.

**Data Router:** The data router pushes the data from a consumer device to any/all of the multi-cloud services and polling the data from the cloud services to the mobile consumers as shown in **step 6** in Fig. 3.2. Since I am dealing with actual data (document files, text, etc.) which are persistently stored on the cloud services, I decided to build a mapping relationship between the consumer, the cloud sources, and the files. After the successful creation of a user account, the user is given a unique identifier (id) and the id is employed to create and name the bucket on the Amazon S3 facility or a folder on the Dropbox or other IaaS

cloud service. Equally, all the cloud sources are assigned unique identifiers (these IDs are not dependent on the consumer but rather based on the number of cloud sources that the CSB-UCC can support). The use of the ID in creating the bucket and the folder is to avoid the duplication of names creation which may hamper security.

I anticipate that though brokerage services may support multi-cloud providers, the consumers may choose to use only some services but not all. For instance, the CSB-UCC can be built to support five cloud service providers but some of the users may opt to use only two or even one while some may use all five. Hence, each user is tracked by the number of cloud services that the user has subscribed for. Based on the graph relationship, the data router component identifies the user's account id (Account) and searches for all the cloud provider services of the consumer. After the identification of the consumer's cloud services, the data router then searches for the users' folder and data on each of the identified cloud providers.

Further, the uniqueness of the CSB-UCC from other related frameworks is that, the consumer can also register n-devices (e.g. mobile, tablets, PC) to which data from the cloud sources can be pushed to automatically. In this regard, auto-sync is performed without explicitly calling for the data.

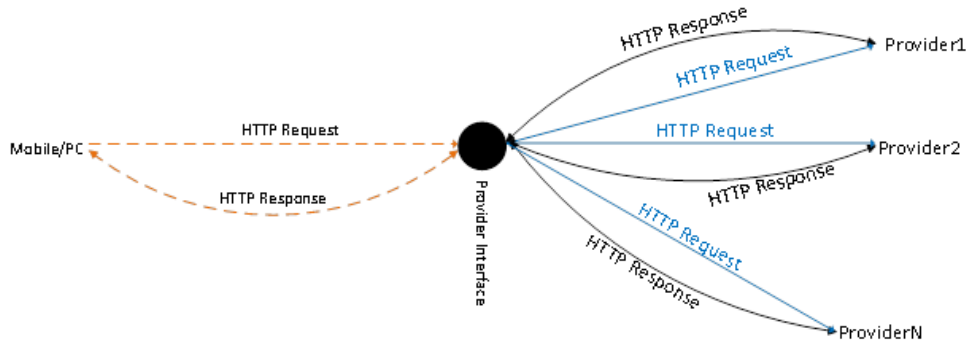
**The Workflow Engine:** The workflow engine is responsible for the business activity flow of the CSB-UCC including error tracking/detection. It is often said there is no "error free" system; so the workflow engine is designed to handle the error cases as well as the error recovery process.

The resources (data) stored on the cloud sources have states. Since, the cloud providers host the resources as web services, the data are assigned ETag values. The ETag is a unique string of the web service which changes whenever there is an update (state change) to that service. The workflow engine uses the ETag of the client side resources and compares it to the same resource on the cloud and determines whether the cloud side data is updated if the ETags are not equal. An update on the cloud requires the workflow engine to notify the data router component which in turn sends the data to the registered mobile devices when the consumer requests for the data. The ETag usage aids the entire architecture to manage the issues with distributed stale cache especially in the mobile network.

Furthermore, an error scenario is defined that is handled by the workflow engine. Since heterogeneous networks (i.e. wireless and wired networks) are being considered, there are potential cases where requests will not be served before there will be loss of connectivity especially with the mobile consumers. Hence, when the mobile re-connect, the workflow engine re-issues the request for the resource from the cloud providers.

**Device States:** Apart from the resources (data) states, the client devices are also assigned states which are defined simply as: *Connected* and *Disconnected*. The client devices that connect are in the Connected state and are ready to receive data from the cloud providers or push data to the cloud source. The client devices that loss connectivity or turned off are in the Disconnect state and data is not sent to them by the data router. Rather, the workflow engine records the disconnect state and waits for the device to reconnect. So, when the device reconnects, the workflow engine moves the device state to "connected" and the data is re-sent.

**The HTTPS Interface:** The HTTPS interface (also dubbed as the Provider interface in Figure 3.4) forwards the HTTP requests to the cloud services providers on behalf of the consumer. Write requests (e.g. POST and PUT for creating and updating resources respectively) following the REST protocol are straightforward since these methods are supported by most IaaS layers (e.g. Amazon WS, Dropbox, etc).



**Figure 3.4:** The Illustration of how a Single Request from the Consumer Device is Replicated by the Provider Interface

However, the read request (e.g. GET) requires another consideration. Since the work is based on HTTP polling, there is the need to design the HTTP request call in a way that does not consume so much bandwidth with unnecessary requests. Hence, when the consumer makes a single HTTP request from the mobile device, the provider interface replicates the requests to the various cloud services providers that the consumer owns and waits for all the responses. The responses from the multi-cloud sources are then returned to the consumer as a single response. Further, the CSB-UCC framework is extensible with REST API-oriented IaaS cloud sources. The framework supports REST because the syntactic structure is identical across multiple cloud providers; which requires single programming effort.

The other components such as Provenance Engine and the Controller will be discussed in detail when we re-visit the research question.

### 3.6 Data Update Management in Group Sharing (File Sharing Use Case)

There are different facets of mobile data management and the issue is an entire research field that requires independent research. However, for the purpose of discussion, a brief overview of how mobile file sharing in group usage is facilitated by the CSB-UCC and the back-end is provided regarding the IaaS cloud only. For other use cases involving data state management, data synchronization, data integrity, etc., the framework has to be adapted to meet those goals.

Clearly, the transaction of file management and sharing in the CSB-UCC falls within the *CAP theorem* as discussed in detail by Lomotey and Deters [148]. The theorem simply states that in a highly distributed



environment, three desired properties are **C**onsistency of the service, high **A**vailability of the service/system, and **P**artition–tolerance to faults; however, only two out of the three properties can be simultaneously guaranteed. In a group file sharing, there are chances of not having connectivity to the IaaS cloud services and the fact that the users data are stored on distributed layers (i.e., Dropbox, Amazon S3 and n–consumer devices) means that partition tolerance is inherent in the architecture. This means the system can only achieve consistency of data within the system or ensure high availability as the complementary guarantee. Conventionally, the Dropbox service opts for the availability option which means, the system allows for some time lag within which the data is synchronized across the multiple mobile platforms. This also means that there are times when users view outdated data in their Dropbox account on the mobile especially when the synchronization process is not complete. Moreover, the Dropbox service enforces session consistency. In view of this, the CSB–UCC is aligned to offer the availability guarantee and compromise on data consistency.

The system can equally guarantee consistency by compromising on availability. To do this, locks can be introduced on the broker following the ACID approach which ensures that the moment a user is authenticated, every transaction of that user has to be complete before any other transaction can be initialized by the same user. However, from the lessons learned in a previously reviewed literature, Lomotey and Deters [148], the consistency guarantee is less preferred by users. Users want to have access to data and keep working and later synchronize their data if the need be.

So, availability of the system is guaranteed by providing the offline mode as mentioned earlier in the previous section. The question now is how to ensure soft real time data synchronization. The answer to this question leads to the discussion of the last component of the broker which is the Workflow Engine.

**The Workflow Engine** in in Fig. 3.2 is initialized when the user completes the authentication process through the OAuth 2.0 Engine. The role of the workflow engine is to issue asynchronous HTTP HEAD request to the two cloud providers concurrently. The reason requests are issued concurrently is to minimize the waiting queue if the sequential request technique is adopted. Also, the HTTP HEAD request is issued in this case because the header information is lighter which is good for the bandwidth utilization. The workflow engine issues the request to retrieve only the Etag value of the stored data. This value is compared with all the stored data of the user on the n–consumer devices and mismatch Etag pairs call for synchronization of the data from either end.

Further, when the user is in the offline mode, the Workflow Engine still polls updates from the cloud sources and puts the updates in notification mode. So, as soon as a user switches to the online mode, the workflow engine informs the Service Coordinator which also directs the Resource Router to issue the requests for the updated files. The proposed workflow engine aids the broker to push the updated resource (files) states to the consumer devices faster.

## 3.7 Re-Visiting the Research Questions

At this point, it is important to re-visit the research questions and highlight those questions that have been solved and those that are yet to be answered. For brevity, the questions will be repeated and brief overview of how each is solved is provided.

### A. How can consumer consistent experience be ensured?

- (i) *How can new updates be detected and pushed to the consumer irrespective of which service is updated?:* The users can register their devices on the broker and updates available on the multi-cloud sources are pushed to them. There are provisions for offline and online modes that detect which of a user's n-devices are connected and disconnected.
- (ii) *How can application consistency be ensured from the user's perspective?:* The user has to subscribe to m-cloud services on the broker. Updates are then pushed to the connected devices of the user. In the disconnect state, the broker keeps the updates through notification and when connectivity is restored, the updates are pushed to the user's device.
- (iii) *How can group data and file synchronization be ensured?:* From the lessons learned in the CAP theorem, we propose the weak consistency approach where files are exchanged with changes over a time window. This gives users high availability to work on files in an offline mode and later synchronize them when there is connectivity.
- (iv) *How can soft real-time data accessibility in the mobile network be enforced?:* The idea of notification, and concurrent request issuance to the m-cloud sources are all aimed at ensuring soft-real time data accessibility. A lot more will be done on this area when we discuss scalability.

### B. How can the aggregation of the services and subsequently deliver them to the consumer's multiple devices be ensured?

- (i) *How can the scalability of the broker be improved?:* Will be addressed in latter discussions.
- (ii) *How can concurrency be supported?:* This is discussed in section 3.5. Preference is given to the proposed approach over sequential indexing because the latter can introduce higher latency.
- (iii) *How can fault tolerance be ensured?:* This will be discussed along with scalability.
- (iv) *What is the efficient way to authenticate the user/s from the m-cloud sources and audit?:* There are three authentication modes provided through username/password credentials, 3rd Party authentication based on OAuth 2.0, and hybrid authentication that combines the other two. The concept of provenance will be discussed later.

### 3.8 Designing a Distributed Broker Architecture for Scalability in the CSB-UCC

As argued by Chen et al. [92], building a distributed broker platform can enhance scalability. In view of this argument, this dissertation explores the distributed broker approach. In Fig. 3.1, the CSB-UCC appears to be a centralized broker but actually, it is a distributed architecture with hidden components. For clarity, the architecture is re-designed in Fig. 3.5 and this illustration will guide the discussion in this section. In Fig. 3.5, the HP represents users with n-devices but the discussion will be more on the CSB-UCC and its components.

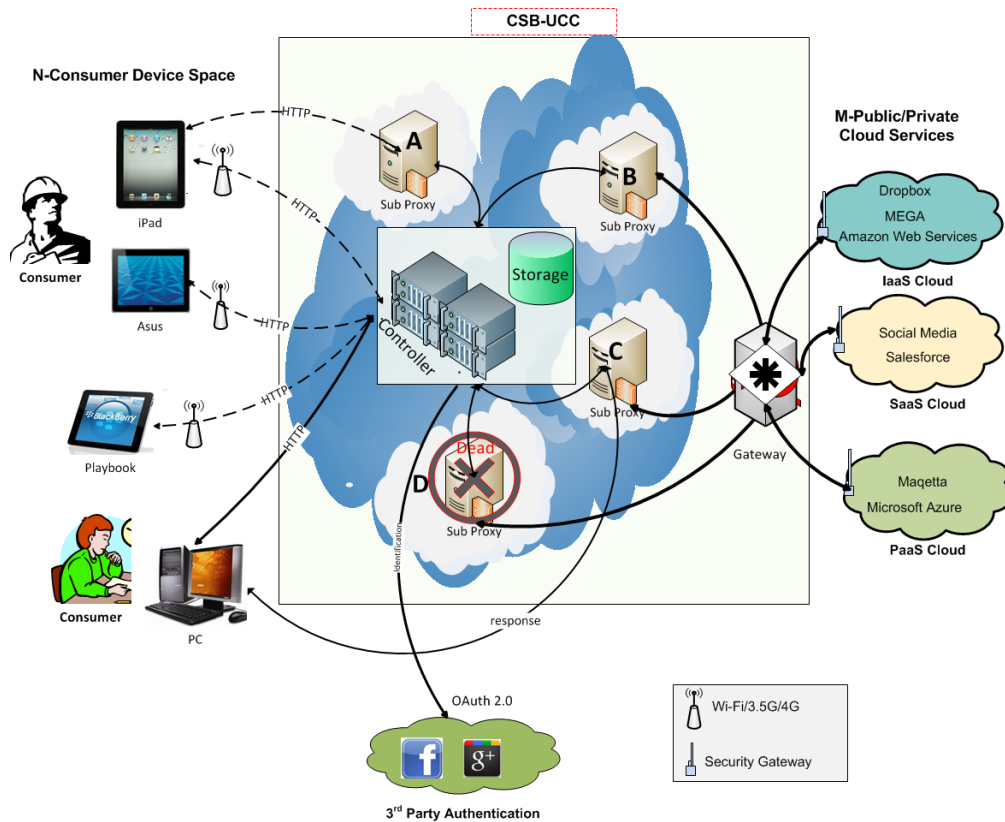


Figure 3.5: The Distributed CSB-UCC Architecture

The CSB-UCC as shown in Fig 3.5 has two components that emanate from the previous design in Fig. 3.1: the *Controller* and the *Sub-proxies*. The previous architecture in Fig. 3.1 presents the CSB-UCC as a centralized broker layer that processes all the incoming and out-going requests from the n-devices and the multi-cloud sources. But, the enhanced architecture (as shown in Fig 3.5) has sub-proxies, each with the capability of the previously described broker.

### 3.8.1 The Controller

The controller is the main coordinator of the activities between the users plus their n-devices and the multi-cloud sources. Especially, the mobile nodes issue a request to this layer and the layer determines the best sub-proxy that can handle the request; and routes the request to that sub-proxy. The response to the mobile is then sent back directly by the sub-proxy. Fig. 3.6 highlights the sub-components of the controller.

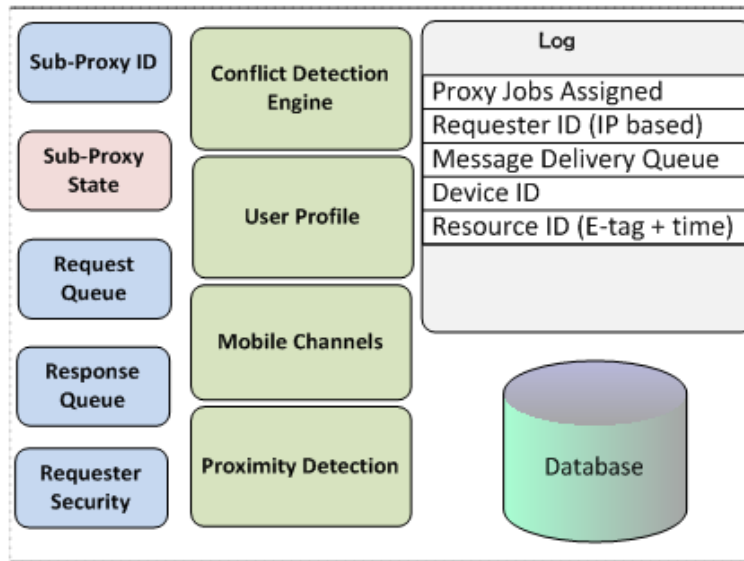


Figure 3.6: The Composition of the Controller

**Sub-Proxy ID:** Each sub-proxy is an application server that can be hosted on a separate IaaS cloud, and has a unique identifier. This is to facilitate efficient communication between the controller and the sub-proxies. This is beneficial to access specific processes as laid-down in the REST design principle.

**Sub-Proxy State:** The requests which are sent by the mobile participants (i.e., the users) are not processed by the controller. Rather, they are processed by the Sub-Proxies. So, when the requests arrive, the controller determines which sub-proxies are in a state that can process a request. There are five states for every sub-proxy which are:

- *Dead State:* This means the sub-proxy (or, the application server) is dead and is no longer reachable. This can be due to fatal errors, failures, or crashes. Sub-proxies in this state are not issued requests by the controller.
- *Alive State:* This means that the sub-proxy is reachable.
- *Responsive State:* This means the sub-proxy is responding to the communication of the controller. There are cases where the sub-proxies are alive but non-responsive. This can be due to intensive transaction processing.

- *Busy State*: This is when a sub-proxy notifies the controller and other sub-proxies that it does not want to receive any further requests.
- *Available State*: This means the sub-proxy is ready to receive new tasks from the broker-layer.

**Request Queue**: This is a channel where the controller stores the incoming requests as queue based on a First-Come-First-Serve bases. The queue increases when most of the sub-proxies are dead, busy, and non-responsive.

**Response Queue**: The responses are also queued when being delivered to the various mobile nodes. This queue is specifically for situations where a sub-proxy is not able to deliver a response to the mobile because of communication failure.

**Requester Security**: The controller is responsible for the authentication and authorization of the user activities. This has been described in detail in previous sections.

**Conflict Detection Engine**: This is where the requests are accessed to see whether they are repeated. Also, the controller checks that the same task is not sent to two different sub-proxies. Especially, when the sub-proxies communicate outside the broker layer, there are chances of issuing repeated requests to other proxies by the controller.

**User Profile**: This is where the broker stores the activities of the user. This is crucial for audit rail as will be discussed later when provenance is discussed.

**Mobile Channels**: This is where the controller delivers the requests to the mobile devices. This channel is for the delivery of messages that were undelivered directly by the sub-proxies. When the mobile is ready to receive a response, the controller uses this channel to deliver the response but, when the mobile is not available, the response is stored in the database.

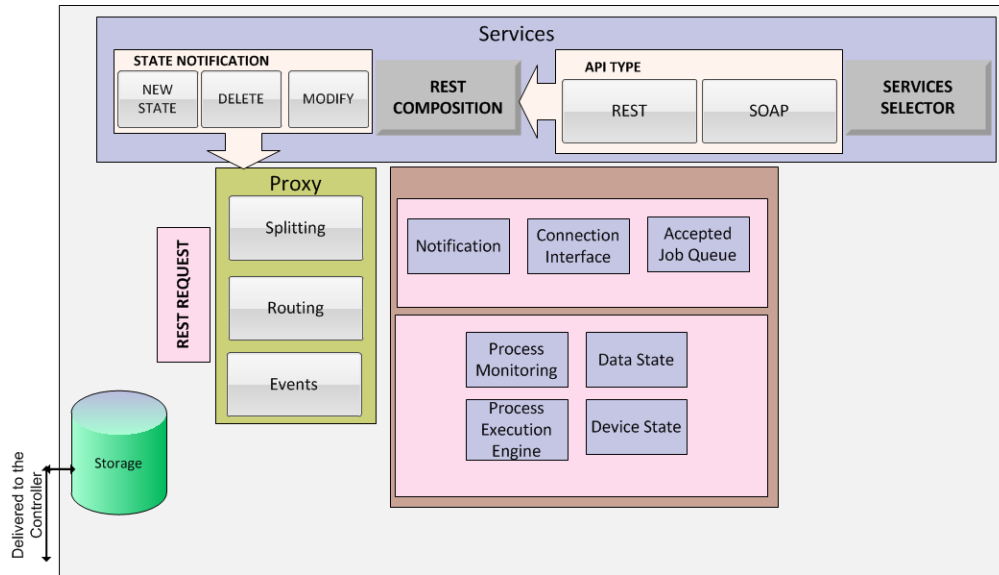
**Proximity Detection**: This component determines the location of the requester and directs the mobile request to the proxy within the best proximity. The details will be discussed in later sections.

**Log**: The log is the reference repository for the broker to track its internal activities.

**Database**: This is disk storage where the requested transactions of the mobile users are stored to be accessed later. When a request is processed and not delivered and the same request is issued, the conflict detection engine pulls the result from the storage; and this enables the controller not to assign this new request to any sub-proxy.

### 3.8.2 The Sub-Proxies

It is important to state that the sub-proxies are application servers that can communicate and exchange jobs directly without communicating through the controller. The sub-proxies can delegate their tasks to other peers when they are burdened with a lot of tasks. However, when a sub-proxy dies, even though all the other sub-proxies are aware of the situation, only the controller can re-assign the job of the dead sub-proxy's tasks. The sub-components of the sub-proxy are illustrated in Figure 3.7.



**Figure 3.7:** The Composition of a Sub-Proxy

**Notification:** The notification component is responsible for informing the controller about its status. The various status of the sub-proxy is discussed in the previous section. The information is delivered over HTTP (i.e., the Connection Interface).

**Process Monitoring:** This component checks which transactions are being processed and their status. The status of a transaction can be *Complete* or *Incomplete*.

**Process Execution Engine:** This is where the transactions are processed. Transactions can be requests which include: fetching files/data, updating data, creating new data, and so on.

**Accepted Job Queue:** This component keeps track of the number of jobs that are assigned to the sub-proxy. The component is crucial for the determination of job assignment by the controller, as we shall explain later.

**Data State:** This is to determine whether an existing data has been updated or not. The state of the data is determined based on the state of the ETag value of each data, which is a REST resource.

**Device State:** This is to know whether a device is connected or disconnected. Connected devices are in a state to receive responses directly from the sub-proxy. However, due to the intermittent loss in connectivity in mobile networks, a device in a disconnect state will not be able to receive a response. In this case, the response is sent to the controller to store the response in the database.

**Storage:** This is where all records including the data state, device state, the job queue, etc. are stored.

**REST Request:** The mobile devices can only interact with the CSB-UCC strictly following the REST standard.

**Proxy:** The proxy component is responsible for the request routing, request splitting, and event monitoring. *Routing*– The CSB-UCC ensures that responses are sent back to the mobile devices for every request. The updates are also pushed to the appropriate devices as needed. *Splitting*– The proxy splits the incoming

requests to the appropriate service providers (i.e., the multi-cloud sources). Since the user can subscribe to multiple cloud sources, splitting is required to issue the request to the specific service provider. Also, updates from the cloud providers are split to the specific devices of the user. *Events*– The event component interfaces the notification layer and the queue layer to determine the availability of updates or the state of devices respectively.

**Services:** The services component is responsible for macro activities such as services selection, API type detection, REST composition and issuance of appropriate notification. *Services Selector*– This component enables the users to choose and select the specific cloud service that the user wants to communicate with. The services can be IaaS, SaaS, and PaaS. *API Type*– This component is responsible for the choice between the REST and SOAP API. *REST Composition*– Though will not be implemented in the CSB-UCC, the SOAP API can be supported through protocol transformation. This means, the indicated REST composition layer will transform the SOAP API to REST when the need arises before delivering it to the mobile end-point.

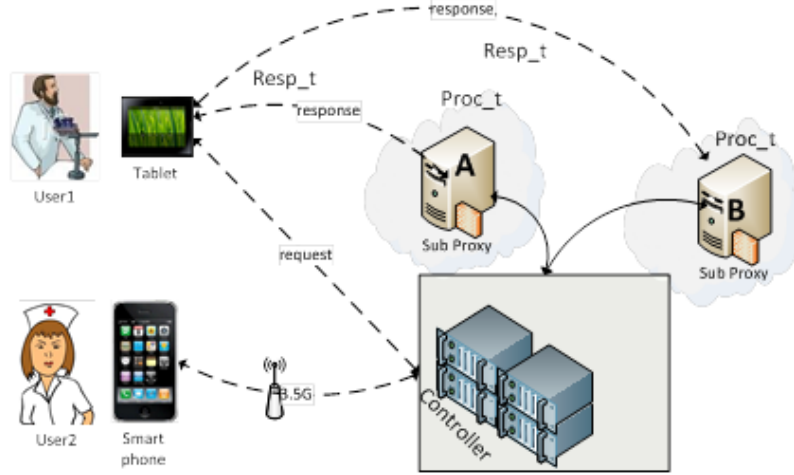
### 3.8.3 Assigning Jobs Based on the Best-Proximity

Recently, several methodologies are being proposed by system designers to achieve soft-real time data and content exchanges. *Edge Computing* is one of the techniques that is recently dominating where the computing power is distributed to logical nodes near client-endpoints; an approach that deviates from centralized processing [149] [150]. Content providers such as Netflix adopts such techniques to deliver data streams to their users in perceived real-time. Another major architecture is the *Fog Computing* which is introduced by CISCO [151] [152]. Just as cloud computing, Fog Computing extends services to the edge of the network but its characteristics are proximity to end-users, dense geographical distribution, and support for mobility.

However, this dissertation proposes a communication flow called “best-proximity”. The best-proximity is not the same as the closest proximity. Since there are multiple sub-proxies, the controller has to determine which of the sub-proxies can serve a request in the shortest time. In order to explain the concept better to the reader, Fig. 3.8 is adapted from the generic architecture for our next discussions.

The controller component determines the location of the user when a request is issued and assigns the task to the sub-proxy with the best option. From Fig. 3.8, we can assume that sub-proxy **A** is the closest to User1 hence, it is appropriate to route the request of User1 to that sub-proxy and not sub-proxy **B**. This assumption is based on the closest proximity policy. However, this approach does not mean that the request-response time will always be the best minimal. To explain this further, I consider the scenario below.

For every request, the controller accepts the request and assigns the request to a particular sub-proxy. The sub-proxy has a processing time (Proc.t) to handle the request and response time (Resp.t) for which the response travels back to the mobile node. Assuming the processing time is the same for sub-proxy **A** and **B**, then it will be best to assign the task (i.e., the request) to sub-proxy **A** which is the closest that guarantees the shortest response time (or soft real time). However, if the processing time varies, then the job assignment has to be re-considered. Let us consider the parameters in Table 3.1. In Scenario 1, the best



**Figure 3.8:** Determining the Best-Proximity Sub-Proxy

case for the reduction in latency in order to ensure soft-real time request-response is to assign the incoming request to Proxy **A** since the total time is 155ms. The total time is calculated as:

$$Total = Req\_t + Proc\_t + Resp\_t$$

where  $Req\_t$  is the request time.

**Table 3.1:** Request-Response Scenario

<i>Scenario</i>	<i>Proxy</i>	<i>Req_t (ms)</i>	<i>Proc_t (ms)</i>	<i>Resp_t (ms)</i>	<i>Total (ms)</i>
Scenario 1	Proxy A	15	120	20	155
	Proxy B	15	120	70	205
Scenario 2	Proxy A	15	120	20	155
	Proxy B	15	20	70	105

Furthermore, from Scenario 1, the situation will always favor sub-proxy **A** if it has lower  $Req\_t$ , and  $Proc\_t$  or if the sum of all the time components produces the least time. In Scenario 2 however, the best case for latency reduction is to route the request to sub-proxy **B** because it has the least total time. Even though it has a higher  $Resp\_t$  (because it is far from the user), it has a significantly low  $Proc\_t$  probably because its accepted job queue is shorter.

So, even though closeness is vital for job assignments (as in the case of edge computing and fog computing), it is not the ultimate factor. Though the proposed best-proximity methodology is previously employed by Content Delivery Networks (CDNs) frequently [268], this dissertation has shown that the methodology can be employed in ubiquitous cloud computing systems. The evaluation section will further discuss experiments to validate this claim.

The next section discusses the employment of the proposed best-proximity technique to ensure sensor-



mobile communication.

### 3.9 Mobile and Sensor Data Sharing Based on the Best-proximity Use Case

Gamma ray is an electromagnetic radiation with a very high frequency that can be biologically hazardous. Most workers in the mining, manufacturing, security, and other industries find themselves in such hazardous environments and governments are trying to contain this issue. While traditionally, high gamma radiation detection sensors have been manufactured to be carried along, they are not good access point for actual dosage readings. With the recent advancement in mobile technology, this section proposes a mobile hosting architecture to enable mobile-to-sensor communication following the best-proximity technique. This means the sensor can detect the radiation and send readings to a smartphone device of the user. All other near-by mobile devices (which are authorized) will receive the notification to alert the people in the hazard zone. But before the discussions, there is the need to explain some distributed mobile computing concepts on services hosting.

The distributed mobile architecture enables mobile devices to communicate with other autonomous system components to achieve a common application goal. *Mobile hosting* (a.k.a., mobile provisioning) [265] is when the mobile device is facilitated to serve as the service provider node. An advantage of mobile hosting is turning the mobile host into a multi-user node. Several other enterprises, including the health domain, find the concept useful since it enhances the management and accessibility of medical data particularly in remote mobile health data accessibility use cases [201]. More importantly, the recent advancement in sensor technologies is pushing researchers to develop mobile hosting techniques that will allow direct communication between sensor devices and mobile hosts.

This section will discuss the area of mobile services hosting especially in a group-sharing scenario and how it adapts the CSB-UCC. Previous works mainly focus on a single mobile hosting node and how it can behave as a server regarding security, data management, message synchronization, and load bearing. This dissertation however focuses on multiple providers of similar service of interest. In a group scenario, the ability to engage multiple mobile services provisioning nodes will lead to better load balancing. However, the bigger challenge is how to reduce communication latency and manage the state of the data across the several nodes. This work therefore proposes an edge-based connection in an attempt to determine the optimal path between the adjacent mobile hosts. Different modes of mobile-to-mobile service communication are designed by adapting the services flow patterns that include sequential, parallelism, loop, and choice approaches. The proposed approaches will be evaluated to determine the best methodology for achieving low-latency communication, efficient job re-assignment, and error management when communications between the mobile host and the consumer fails.

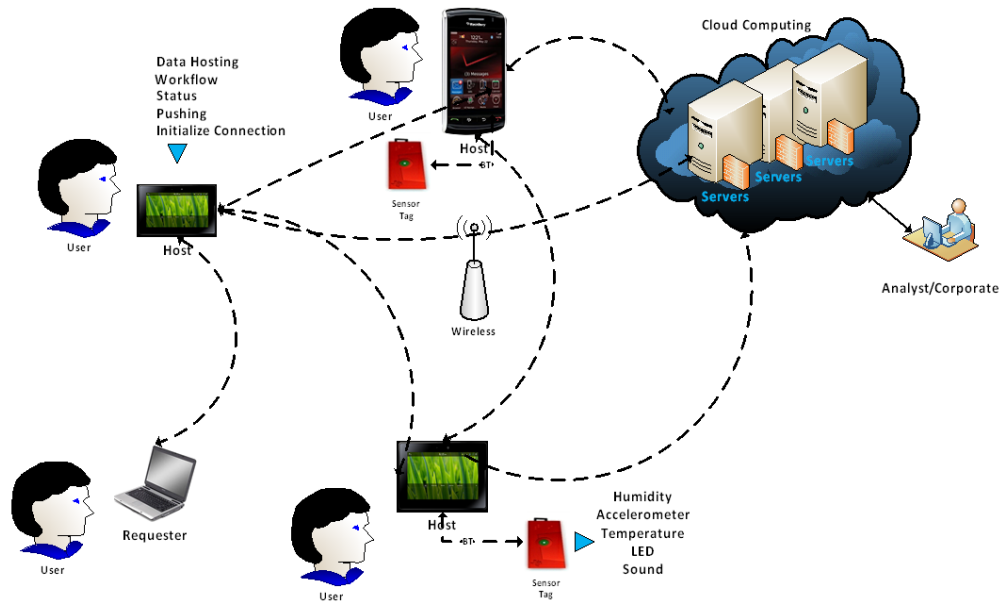
However, before moving forward, here are the specific questions that will guide the proposed solution:

- How is latency reduction addressed in the mobile–sensor group sharing scenario?
- How can task distribution in the group sharing environment be handled in order not to over load one host?
- How is the situation with failures addressed?

These questions are explored in research collaboration with the Environmental Instruments Canada Inc. located in Saskatoon, Saskatchewan, Canada.

### 3.9.1 Mobile–Sensor Data Flow Architecture

The generic architecture of the mobile hosting environment for group sharing is illustrated in Fig. 3.9. The architecture comprises mobile hosts, sensor tags, and cloud based servers. This is a typical adaptation of the CSB–UCC. The inclusion of sensor tag support in the architecture further distinguishes it from other existing works. The use of sensor devices for the purpose of personal radiation measurement, body workout tracking, etc. is on the increase. More importantly, readings from the sensor can be sent to the mobile hosts via Bluetooth. The mobile hosts can share this data with adjacent hosts on the edge or push it to the cloud server for corporate analysis.



**Figure 3.9:** Generic Architecture of the Mobile–Sensor Data Sharing Environment

#### Sensor Tag

For clarity, the T1 Sensor Tag (<http://processors.wiki.ti.com/index.php/SensorTag-User-Guide>) is employed as the sensor device in this section. This is supported by the CC2541 SensorTag Development Kit. The

main features of the sensor as highlighted in Fig. 3.9 includes: humidity check, accelerometer detection, temperature reading, LED notification, and sound notification. In Fig. E.1 (refer to Appendix E), the component architecture highlights some of the key features.

As research on cyber-physical systems (a.k.a., Internet of Things, IoT) is growing, the mobile hosting architecture is developed beyond just the smartphone ecosystem. The integration with sensor devices further proves the need to explore group sharing mobile hosting of services rather than a single node access. These sensors are wearable and can communicate with the mobile in real time. Since the sensors do not have the capacity to deliver content directly to the cloud servers because they support only Bluetooth, the primary source for content delivery is the near-by mobile host. The mobile device that receives the content can deliver it to the cloud servers through Wi-Fi or 3.5/4G network. The same mobile host can also deliver the content to other mobile hosts in the same environment.

## The Mobile Hosts

The mobile hosts are the smartphone endpoints for the services provisioning. Unlike previous works that design a single mobile host with multiple consumer nodes, this dissertation focuses on treating all the mobile nodes as hosts. This means, in a group sharing scenario, every available mobile endpoint can either be a consumer or a provider. For example, if mobile host  $A_H$  has the latest reading of dosage from the sensor device that the other mobile devices do not have,  $A_H$  becomes a host and all the other mobile devices from say  $A_{H+1}$ , ...,  $A_N$  become consumers. The main activities of the mobile hosts as highlighted in Fig. 3.10 are discussed as follows.

### *Services/Data Hosting*

The application state is hosted on the mobile. This includes data in storage and the entire business logic of the application host. The storage contains the sensor data as well as any other data that is of interest to the application state. Updates can be made to the data in storage by replacing the entire data state. This approach is the simplest form of data management in mobile networks. For a more rigorous approach on synchronization of data in mobile hosting environment, we have published an independent research in [266].

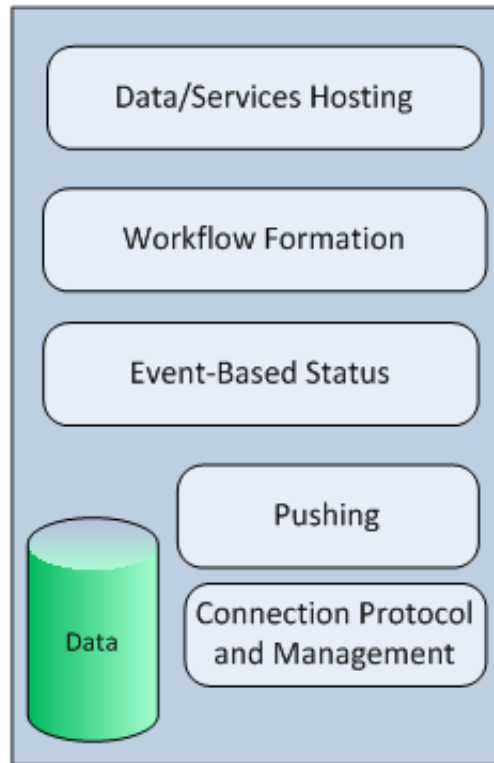
### *Workflow Formation*

The workflow formation is the underlying business logic of the application on the mobile host. The workflow determines which device to connect to, how the device should connect to other devices, how to process a request, whether a request should be declined or accepted and so on. More on the workflow will be discussed in the next section when the flow patterns are introduced.

### *Event-Based Status*

This is to inform the other mobile participants about the status of the mobile host. The status is event-based so the state of the mobile host determines which status message is sent. The following states are implemented to inform the mobile requester.

- Available



**Figure 3.10:** Main Activities of the Mobile Host

- Unavailable
- Busy

Furthermore, customized HTTP status codes are designed that are returned to the client requesters on the state of a mobile host. Available is 200, Unavailable comes with no code rather, the connection terminates or timed out, and Busy status code is 500.

#### *Pushing*

The services that are received on the mobile host are pushed to the other hosts that need that service or to the cloud backend.

#### *Connection Protocol and Management*

The mobile host uses the Bluetooth protocol to connect to the sensor devices but 3.5/4G and/or Wi-Fi to connect to the other external components. The connection to a mobile host can be initialized by the host or by the client requester (i.e., the service consumer).

### **3.9.2 The Communication Flow**

The major issue surrounding latency and request re-routing depends on the communication flow. The communication depends on the number of devices within a P2P sharing zone, the number of concurrent

requests being received by a host (mobile or sensor), and the number of requests being issued by a client (this includes sensors as well).

First, there is the need to understand the possible number of communication flows as not to overburden a single device node, and enforce some level of load distribution. In this case, there is the need to understand the series of the communication flow.

When there is only one mobile host in an area, there is only one communication flow (or loop to itself). If there are two (2) hosts, there are potential two communication flows, one communication from Host<sub>1</sub> to Host<sub>2</sub>, and from Host<sub>2</sub> to Host<sub>1</sub>. Since at this stage the focus is only on the communication flow, it is also assumed the sensors and the clients are hosts. Another way to perceive this is to consider all the devices as part of the flow. This model then extends as follows:

$$3 \text{ Hosts or Devices} = 6 \text{ Com}_{Flow}$$

$$4 \text{ Hosts or Devices} = 12 \text{ Com}_{Flow}$$

$$5 \text{ Hosts or Devices} = 20 \text{ Com}_{Flow}$$

where Com<sub>Flow</sub> is the communication flow.

To find the series for the communication flow, assuming N is the number of devices, then the communication flow will be:

$$Com_{Flow} = N * (N - 1)$$

Based on the series we set a maximum threshold for connections that can be handled by a single mobile host. This threshold is set as a fraction of the communication flow where  $\frac{2}{3}$  Com<sub>Flow</sub> works best in terms of not overburdening the mobile device. This threshold is very important because this is what enables us to determine whether a mobile host should move to the “Busy” state or remain in the “Available” state as discussed in the previous sections.

Though the proposed communication flow above appears simple, it is the adaptation of the Ford–Fulkerson algorithm [267] from the field of graph theory. In this algorithm, for each edge from say host H1 to H2, let c(H1, H2) be the capacity and f(H1, H2) be the flow. If we want to find the maximum flow from source S to K, then the following algorithm can be employed:

*Equation (i)*

$$\forall (H1, H2) \in E \quad f(H1, H2) \leq c(H1, H2)$$

*Equation (ii)*

$$\forall (H1, H2) \in E \quad f(H1, H2) = -f(H2, H1)$$

Equation (iii)

$$\forall H1 \in V : H1 \neq S \text{ and } H2 \neq K \Rightarrow \sum_{w \in V} f(H1, H2)$$

Equation (iv)

$$\sum_{(S, H1) \in E} f(S, H1) = \sum_{(H2, K) \in E} f(H2, K)$$

Equation (i) is the *capacity constraint* which means the flow along an edge cannot exceed its capacity. Equation (ii) is the *skew symmetry* which is the net flow from mobile host H1 to H2 must be the opposite of the net flow from H2 to H1. Equation (iii) is the *flow conservation*– the net flow to a node is zero, except for S, which generates the flow, and the K which is the flow consumer. Otherwise, H1 is S or K. Equation (iv) is the *Value (f)* which is the flow leaving from S or arriving at K.

Now that the communication flow and the maximum threshold are explained, the best-proximity concept is explained.

### Edge-based Connection but not Geographical Proximity-based

As stated earlier, most of the on-going works on edge computing focuses on enabling adjacent devices within the closest-proximity to communicate. This approach is also evidenced in the proposed cloud-edge architecture by Netflix. Even though the proximity-based approach is perceived to be efficient for latency reduction in clusters, an interesting observation is made in this work that show that geographical proximity between the adjacent devices should not be the sole determining factor for latency reduction in mobile-sensor communication. To explain our observations, the illustration in Fig. 3.11 is used.

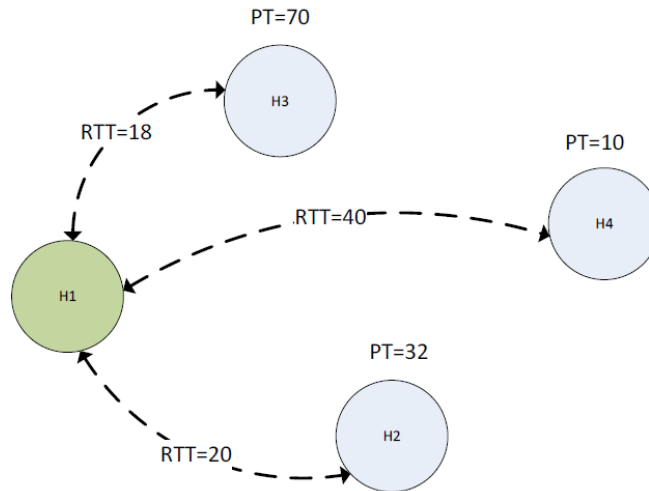


Figure 3.11: Request Flow

This work defines proximity within the concept of Round-Trip Time (RTT) where RTT is the sum of the time taken for the request from one host to reach the other and the response travel time. In this case, we are not considering proximity as the distance between the adjacent devices but as time. In most practical cases, distance is directly proportional to time so the assumption can also be made that the closer the devices, the shorter the travel time. However, we shall use time in the discussions throughout the section since the goal is to explore time optimality and not the shortest distance.

From Fig. 3.11, the closest host from H1 is H3 since the RTT is 18ms. If we assume that H3 is unavailable and/or busy, then the request will be sent to H2 since the RTT between the two hosts is 20ms. The third option is to connect to H4 in case the rest are unavailable. This description is the typical case with edge connection that is based on close proximity.

When we consider other factors such as the busy state of a particular host, closeness alone cannot guarantee latency reduction. From Fig. 3.11, let us denote the entire time required to make a request and receive the response as  $T$ . Then, we can obtain the value of  $T$  by summing the RTT and the processing time (PT) of the request by a host as:

$$(a) T \text{ for } H1 \text{ to } H2 = 20 + 32 = 52ms$$

$$(b) T \text{ for } H1 \text{ to } H3 = 18 + 70 = 88ms$$

$$(c) T \text{ for } H1 \text{ to } H4 = 40 + 10 = 50ms$$

Now, we can see that even though the distance  $|H1H4|$  is double that of  $|H1H2|$  and  $|H1H3|$ , it is best the request starts between H1 and H4 if we want to reduce latency. If the system experiences failure (e.g., loss of message or connection in the middle of a transaction), then the next option is to connect to H2 before H3. Clearly, this shows that the processing time (PT) of a mobile host is very crucial as we shall see later in the evaluation and plays a crucial role in the determination of the adjacent edge to connect to in a group sharing scenario.

We have therefore proposed based on our finding in this work that latency reduction should be viewed as the minimization of  $T$  which is  $RTT + PT$ . Determining the RTT is fairly straightforward since modern mobile devices have GPS and accelerometer. The network signal strength can also be measured so these features can be combined to determine the approximate RTT. The difficulty is the determination of the PT. There are several processes running on users' devices including running background apps and so on. This makes it challenging to determine the exact busy state of the device. However, we prefer to determine the PT within an application domain on the mobile host. So, the mobile host application has its processing threshold as explained in the previous section. In this case, when the mobile host broadcast its name within a discoverable area, it also sends its threshold value which we consider as the PT. If the mobile host is not busy, then requests can be sent to it based on the requester's view of  $T$ . It is important to state that the determination of the entire time ( $T$ ) is the responsibility of the mobile requester and not the mobile host.

Now that the proposed edge connection is explained, the paper discusses the various flow patterns that

we explore. The flow patterns are crucial for the way the requests are issued to discoverable mobile hosts as well as the re-assignment of failed requests (tasks).

### **Sequential Flow**

In the sequential flow, the mobile requesters are enabled to connect to one host at a time. When the connection fails, another host with the smallest RTT + PT can be contacted. Each requester has its list of discoverable devices that is implemented as an array that facilitates the requester to know which host to connect to.

In this approach, the complexity is  $O(n)$  so it is much simpler to design. The concern however is going over the list every time to determine the next available host. Following the linear search, the expected number of comparisons to make in the list to identify an object is

$$n \text{ if } h = 0, \text{ and}$$

$$\frac{n+1}{h+1} \text{ if } 1 \leq h \leq n$$

Where  $n$  is the number of iterations and  $h$  is the number of times an object appears in the list. Since the available mobile hosts are unique, only one is stored in a list (no duplications). Thus, the expected comparisons in the list to find a host is

$$\frac{(n+2)(n-1)}{2n}$$

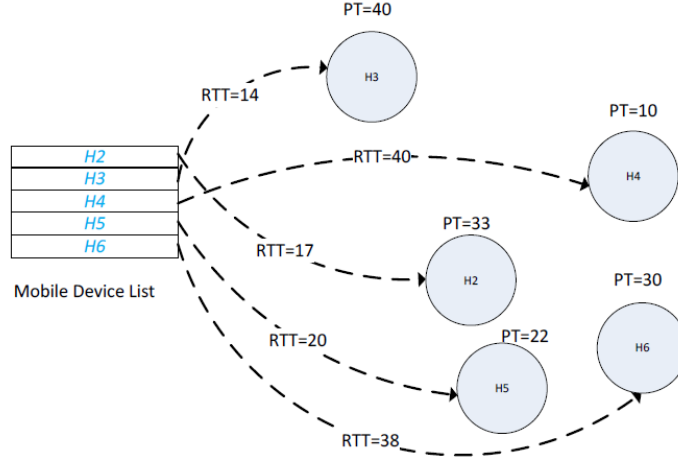
In the sequential flow, the mobile requester uses the linear search to find the preferred mobile host in the list of its discoverable devices. The request is made and when a response is received the flow communication for that transaction is terminated. However, if the system fails, then the mobile requester has to make the request to the second most preferred from the list. This can continue until the requester receives the required or expected response.

### **Parallelism**

The problem with the sequential flow is that, the requester has to send a request and wait for feedback to determine whether it should make another request to a different host or not. This means if there is system failure, the requester will take longer time to get a result. The time involves request issuance, travel time, and the processing time at both the requester and the recipients ends. The sequential flow works best when in one attempt to an intended mobile host (that has the minimal RTT + PT), a response is received with no error.

In practical scenarios, this is not the case since failures are bound in mobile wireless networks due to sporadic disconnections and user mobility. Also, new devices join the community while some people leave the





**Figure 3.12:** Parallel Request Flow from the Requester

geographical area. Hence, the parallelism workflow is designed to issue concurrent requests to the available hosts but with a priority. For clarity, Fig. 3.12 is presented to illustrate the point.

The parallelism approach we propose extends on the sequential flow pattern. Instead of making one request at a time to the host with the least total time,  $T$ , the parallelism approach makes concurrent requests to all the mobile hosts in the list of available devices. However, the responses are prioritized based on the value of  $T$ .

In Fig. 3.12 for example, the mobile client (i.e., H1) will have the following calculated  $T$  values (i.e.,  $RTT + PT$ ) in the list before issuing the requests concurrently:

$$(a) T \text{ for } H1 \text{ to } H2 = 17 + 33 = 50ms$$

$$(b) T \text{ for } H1 \text{ to } H3 = 14 + 40 = 54ms$$

$$(c) T \text{ for } H1 \text{ to } H4 = 10 + 40 = 50ms$$

$$(d) T \text{ for } H1 \text{ to } H5 = 22 + 20 = 42ms$$

$$(e) T \text{ for } H1 \text{ to } H6 = 30 + 38 = 68ms$$

From the above, the requester will create a priority list ( $L_p$ ) as an array such that:

$$string[] L_p = \{ "H5", "H2", "H4", "H3", "H6" \};$$

Where H5 is of higher priority than H2 in that order. H2 and H4 have the same priority so the device only puts one ahead of the other. This list facilitates the mobile client, H1, to focus on the order of receiving the response from the edges. When the response from say H5 is successful, the client will terminate all the transactions with the other devices. This is to save the communication overhead.

In the event that the transaction with H5 fails, then the client will focus on the next device in the priority list until the entire  $L_p$  list is exhausted.

## Loop

The loop flow basically means repeating a transaction. This can be seen in two phases; repeating a transaction on the device itself or to an external host. When a transaction fails between a client and a host, the process can be repeated if the reason for the failure is known.

It is important to state that the loop flow is different from doing the same transaction but with a different host. For example, a transaction between H1 and H2 is not the same as a transaction between H1 and H3.

In this dissertation, the flow will be considered a loop if and only if the same transaction is repeated between the same client and host in the previous transaction. The loop flow however is frequent when a device repeats its own transactions. This includes re-starting a request, re-ordering of a priority list, re-arranging responses and so on. The transactions on the mobile itself are loops since they are repeated workflows in the same environment.

## Choice

The choice flow is akin to the existing works on how devices communicate with edge nodes. The approach facilitates a device to connect randomly to any mobile host that is detected. In the best case scenario, the randomly selected edge is the node with the least total time  $T$ .

However, the proposed edge connection based on minimal  $RTT + PT$  is not applicable to this flow. This flow pattern just connects to any edge that is available and at random. To be fair, we did a bit of prioritization where the list of randomly selected hosts is in the order of the least  $RTT$ . In this case, the selection is random but considers the least travel time between the request and the response. The choice flow pattern typically emulates the existing works on edge computing that proposes distance based proximity as a sole factor for latency reduction.

The choice flow pattern is purposefully designed in this work to enable us evaluate the proposed methodology against a benchmark of an existing approach. Thus, in the evaluation, more discussions will be presented on the performance of our approach (i.e., minimal  $RTT + PT$ ) which is adapted from [268] versus the general approach (i.e., minimal  $RTT$ ).

In the next section, the dissertation discuss another major topic, which is provenance with respect to the CSB-UCC.

## 3.10 Provenance in the CSB-UCC

The proposed provenance methodology is to answer the question of how to enforce data/services reliability and audit. For the purpose of discussion and to narrow this very broad topic to the reader, the dissertation shall consider a healthcare scenario as summarized below.

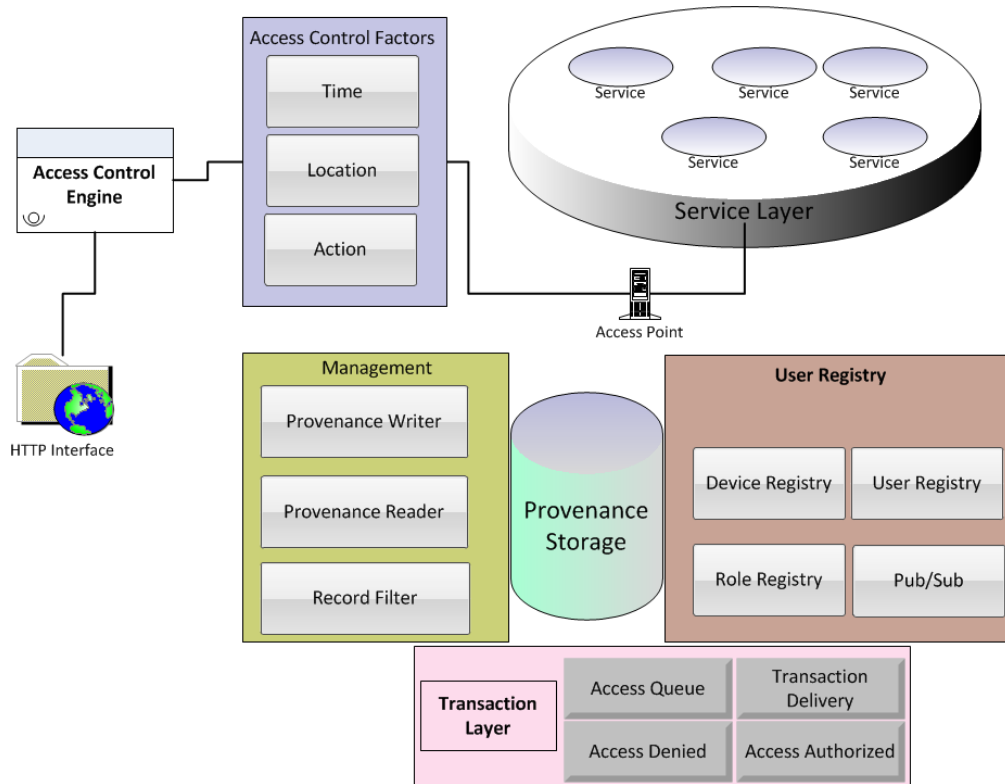
*Assuming the healthcare sector adopts the CSB-UCC to enable healthcare practitioners to use n-devices for medical data accessibility. That means, a single healthcare practitioner can own multiple devices (e.g.,*

smartphones and tablets) and will like to access any medical information on the go. The healthcare practitioners can also share medical data/files with colleagues. This is a typical personal cloud computing ecosystem and the advantage here is to promote remote healthcare delivery as well as timely medical information access.

The above scenario helps to formulate some questions that better explain why provenance is crucial for audit trail. These questions are:

- (i) How do we detect unusual request from a physician who own n-devices?.
- (ii) How do we know which devices belong to the physicians, and therefore should authorize medical data accessibility on those devices?.
- (iii) How should detected attackers be treated? Outside the scope of this work.

The provenance engine which is proposed on the broker is meant to answer the research questions on audit trail. As already posited, the physicians can use multiple devices when accessing the medical records. This creates the need to investigate the best approaches that can enforce protection for the medical data accessibility. The anatomy of the proposed provenance engine is shown in Fig. 3.13.



**Figure 3.13:** The Anatomy of the Provenance Engine

The main reason for adopting the provenance approach in the era of n-screen apps is to ensure services tracking. The provenance methodology as seen from the literature review facilitates tracking of user actions

within a distributed system. In the proposed system, an HTTP Interface is defined which allows the mobile participants to reach the broker. Here are the explanations of some of the proposed components:

**Access Control Engine:** This component is responsible for the determination of whether a request should be served or not. A request can be in the form of retrieving a medical data, updating an existing record, or creating a new record. In the event that the system is hacked, attackers can send fake write requests such as create new records or update existing records, and this must be prevented. Hence, policies are defined based on some specified factors.

**Access Control Factors:** There has been the successful deployment of context-aware systems that defines system accessibility within a context. Context can be time-based, location-based, and role-based. In this dissertation however, the policy is defined based on the combination of all three factors. The access control policy is based on an algorithm that the broker runs as shown below:

$$Access = Time \cap Location \cap Action$$

Even though the user is expected to provide username and password pair before logging into the system, this is not enough to enforce security since that requirement can be compromised. So, when a request is sent by users, the broker determines the *time* of the request, the *current location* of the user, and what *action* the user wants to take. The combination of these factors is good enough to determine a genuine or suspicious request. To understand this further, let's consider the scenarios below:

1. A user sends a request at 1.00 am (i.e., time) from a drinking bar (i.e., location) to edit the allergy of a patient (i.e., action)
2. A user sends a request at any time from a different location other than Saskatoon to do anything
3. A user sends a request at any time from any location to delete a medical record

In *scenario 1*, the request is suspicious when we combine the time and location of the user and the action to be taken. Why will the allergy of a hospitalized patient be modified that late from a drinking bar? In this case, the broker will present the user with a set of security questions that must be answered that justifies the change. Further, the broker will not allow the modification to take effect on the persistent cloud database server but the modification will be stored as a provenance record on the broker until a supervisor/another colleague approves the changes. In the event that the request is sent by a hacker, the user will probably not be able to answer the set of security questions. Better still, the changes will be discarded later when the physician and the supervisor reject the changes from the provenance record.

In *scenario 2*, the application can be designed for say physician at the City Hospital in Saskatoon Canada and their patients within a certain geographical boundary. It is also logical that the physicians can travel and carry their devices with them. The suspicious question however is, why will a request by a physician come from say Mexico? In this case, the request is stored as a provenance record and a communication is established between the physician and a colleague. The physician will have to justify to the colleague why

the request is necessary; and this colleague can then approve the request. Until that is done, the request will be stored as a provenance request.

In *scenario 3*, the delete request is not supported at all since a health information system is being considered.

**The Service Layer:** The service layer is the multi-cloud sources.

**Management:** The management layer determines how the provenance data is controlled. This includes the *provenance writer*– which controls all write operations on the provenance record, *provenance reader*– which controls how the provenance record is fetched, and the *record filter*– which aids in organizing the provenance data.

**User Registry:** This has been explained in previous sections. The *Role Registry* is where access roles of the system users are defined. For instance, some users can only view the medical records and as such, should not be allowed to make modifications.

**Transaction Layer:** This is explained in previous sections.

**Granularity Control:** This component on the broker is responsible for the determination of the depth of changes that is taking place from the users. Here is where we establish the actual changes that are happening in the system. One of the major concerns in mobile networks is bandwidth usage and management. Besides, when the data to be transmitted is small, limited bandwidth can also transmit the data. Hence, I propose the transmission of *delta only* technique. This means, I can transmit only the changes of the medical data across the system components rather than the entire resource state. Let us consider the medical data states below that describe medications given to a patient:

### 1. Initial State

{*Medication*: {*Tylenol*: {*Quantity*: 5mg, *Prescription\_Date*: October 14 2013, *Status*: active }}, {*Acteomine*: {*Quantity*: 10mg, *Prescription\_Date*: November 02 2013, *Status*: active}}}

### 2. Updated State

{*Medication*: {*Tylenol*: {*Quantity*: 5mg, *Prescription\_Date*: October 14 2013, *Status*: active }}, {*Acteomine*: {*Quantity*: 10mg, *Prescription\_Date*: November 02 2013, *Status*: active}}, {*Acteomine*: {*Quantity*: 10mg, *Prescription\_Date*: November 10 2013, *Status*: active}}}

From the initial state and the updated state, the medication list is the same except the addition of new medication in the updated state on November 10 2013. So, to synchronize this update with the database on the cloud, previous researchers just move the entire updated state to the database so that the previous state will be overwritten (or replaced) with the new state. The approach is easier since the determination of deltas on the mobile can be a daunting task. In this work however, I am able to move only the delta from the mobile to the database. Hence, instead of moving the entire updated state, only the last medication record that is applied is moved, and appended to the existing version on the database so that consistency can be attained.

This is advantageous for the transfer of lightweight data across the fluctuating bandwidth. On the broker

(specifically the sub-proxies), there is a *modification tracker* that checks the delta being added and propagates these changes throughout the system. Since delete is not allowed in the CSB-UCC, the modifications will only involve an update or a creation of new records. This makes the processing tolerable for the broker. It is also important to state that, in some rare circumstances, a write operation will not change the state of a medical record. For instance, if a healthcare professional erroneously hit the submit button when no new information is added to the system, the new state of the data will be the same as the previous. However, this will activate the write behavior that requires transfer. This is unnecessary and just consuming bandwidth that should have been conserved. So, the modification tracker is able to detect this type of actions and prevents the propagation of such needless transfers. However, the action will be stored in the *provenance storage*, which keeps the records on the activities of the healthcare professionals.

### 3.11 Some Applications that Adapted the CSB-UCC

Currently, there are several applications that have been deployed successfully that adapt the CSB-UCC framework to meet their specific goals. Some are mentioned and discussed briefly.

#### 3.11.1 The ALILI Framework

The *ALILI* application by Lomotey and Deters [148] is designed to support personalized and enterprise data sharing in a groupware. The mobile service supports file sharing from multiple IaaS end-points (Amazon S3 and Dropbox) and aids the updates to be pushed to all of the user's n-devices. The homepage of the mobile service is shown in Fig. 3.14. An update by a single user is propagated to all other users who are entitled to see those changes.

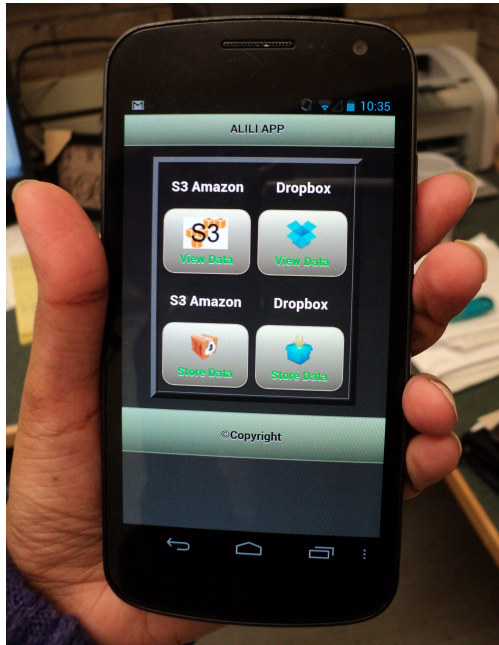
#### 3.11.2 The Med App Framework

The *Med App* by Lomotey and Deters [153] is a mobile health (mHealth) application that enables the healthcare practitioners to remotely access the medical data. The data includes patient demographics, vitals, allergies, and so on. The Med App framework adapts the CSB-UCC framework on the distributed broker methodology as well as the enforcement of the best-proximity use case. The work was presented with the best paper awards at the ACM/IEEE ASONAM 2013, FOSINT-SI 2013, and HI-BI-BI 2013.

The user interface of Med App on a mobile platform is shown below in Fig. 3.15.

#### 3.11.3 Hemophilia Injury Recognition Tool (HIRT?)

In this work by Lomotey et al. [154], the *Hemophilia Injury Recognition Tool (HIRT?)* was developed for, and with help from, young men with MILD hemophilia. Many people with milder forms of hemophilia are able to live a normal and active life and have few episodes of serious bleeding. Minor bumps and bruises that occur with daily life or sports often heal by themselves.



**Figure 3.14:** The User Interface of the ALILI App

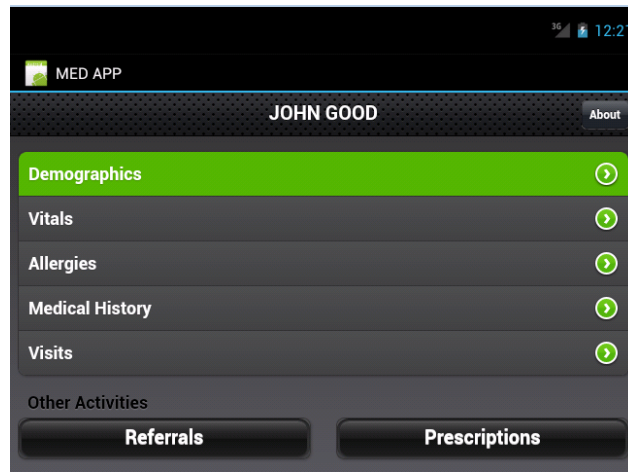
But once in a while, an injury may be more serious, and it is important to be able to tell the difference. HIRT? which is a mobile App was developed: 1) as a self-management tool, to help young men with mild hemophilia assess their symptoms and decide if an injury needs medical attention, and 2) to help them contact a hemophilia treatment center (HTC).

This self-management App does not replace professional advice. It is designed to prepare users to discuss their decisions with their health care team. This evidence-based self-management tool helps young men with mild hemophilia assess an injury and decide when to seek medical attention. It supports a person with mild hemophilia to make decisions based on his own assessment of physical signs and symptoms. It also suggests signs that indicate that the injury is getting worse and that he should contact the HTC to prevent long-term problems. Apart from the App being developed as n-screen service for multiple platforms, the application also employs the provenance technique proposed to ensure audit trail. The purpose is to aid the research team of HIRT? keep track of how the designed self-assessment guide is being used across Canada and the world at large. The application can be downloaded from the App Store. Further details on this work is provided in Appendix D.

A screen shot of the interface of HIRT? is shown in Figure 3.16.

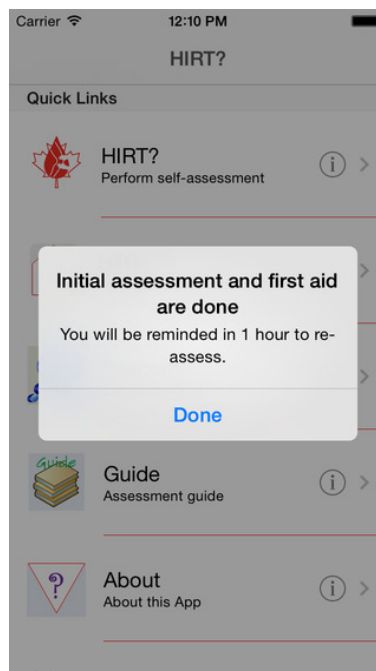
The project team comprises:

- *Richard K. Lomotey* – Department of Computer Science, University of Saskatchewan, Canada
- *JoAnn Nilson* – Saskatchewan Bleeding Disorder Program, Saskatoon, Canada
- *Kathy Mulder* – Children’s Hospital, Health Sciences Center, Winnipeg, Canada



**Figure 3.15:** The User Interface of Med App

- *Kristy Wittmeier* – Dept. of Paediatrics and Child Health, Univ. of Manitoba, Canada
- *Candice Schachter* – School of Physical Therapy, University of Saskatchewan, Canada
- *Sarah Oosman* – School of Physical Therapy, University of Saskatchewan, Canada
- *Cathy Arnold* – School of Physical Therapy, University of Saskatchewan, Canada
- *Ralph Deters* – Department of Computer Science, University of Saskatchewan, Canada



**Figure 3.16:** The User Interface of HIRT?



### 3.11.4 MUBaaS: Mobile Ubiquitous Brokerage as a Service

This work proposes a cloud-based application middleware, called *Mobile Ubiquitous Brokerage as a Service (MUBaaS)* [155], which enables n-devices of a user to access multiple cloud services end-points in soft-real time. This is achieved by proposing distributed brokers that coordinates the transactions of the user while taking load balancing into account. The work is part of the *Clandestine Anomaly Game Project* by ZenFri. Inc. Canada, in Manitoba. The loadout menu in Fig. 3.17 is reproduced from <http://zenfri.com/loadout-the-units-and-upgrades-of-clandestine-anomaly/>.



**Figure 3.17:** Loadout Menu of Clandestine Anomaly

Epic sci-fi adventure AR game for mobile devices.

Clandestine: Anomaly is a tower defense game leveraging Augmented Reality (AR) and GPS technology to create a groundbreaking, unique and immersive mobile gaming experience. Through their mobile device, players will detect and assemble technology, coordinate campaigns, defend against attacks and neutralize enemies. Clandestine: Anomaly includes an engaging story where the user's gameplay choices determine how the game unfolds.

The project team comprises:

- *Richard K. Lomotey* – Department of Computer Science, University of Saskatchewan, Canada
- *Ralph Deters* – Department of Computer Science, University of Saskatchewan, Canada
- *Corey King* – Zenfri Inc. Manitoba, Canada
- *R & D, ZenFri Inc.* – Zenfri Inc. Manitoba, Canada

### 3.11.5 SSCA Spray Quality

This project, that translates into an app as final product, identifies the ASABE spray quality of nozzles available in North America, within their recommended pressure range. Data are supplied by nozzle manufacturers, and are also available from manufacturer catalogues. Only nozzles for which spray quality information is known are included in this app. Manufacturers bear responsibility for following the ASABE S572 standard for the determination of spray quality.

Manufacturers represented are Albuz, Billericay Farm Services (Air Bubble Jet), Greenleaf Technologies (TurboDrop), Hardi, Hypro, John Deere, Lechler, TeeJet, and Wilger (ComboJet). The SSCA Spray Quality app is intended to assist spray applicators making the spray application using the spray quality recommended on product labels, or in accordance with other Best Management Practices related to spray coverage or drift prevention. It produces tables that show the spray quality of any selected nozzle within the manufacturer recommended pressure range.

The app has two basic features: The first is to identify the nozzle currently installed on a sprayer, either by entering the nozzle model number in its search engine or by narrowing the search according to the best information available. The second feature is to identify a nozzle that meets certain spray quality and flow rate criteria, by manufacturer and nozzle type. The product can be downloaded from the Apple store.

The App is available for download in the Apple Store

The project team comprises:

- *Richard K. Lomotey* – Department of Computer Science, University of Saskatchewan, Canada
- *Ralph Deters* – Department of Computer Science, University of Saskatchewan, Canada
- *Tom Wolf* – Agrimetrix Research & Training

### 3.11.6 SSCA Tank Mix

This App is developed for pesticide applicators to help them calculate the water and product requirements for a spray operation. The purpose of this App is to provide product amount calculations necessary for a spray operation. The applicator selects application volumes and product rates, and also enters sprayer and field information to allow estimation of number of tanks required for the field, as well as the number of rounds possible with a load. A large combination of measurement units is available to meet diverse of users. The product is available on Apple store and Google Play

The project team comprises:

- *Richard K. Lomotey* – Department of Computer Science, University of Saskatchewan, Canada
- *Ralph Deters* – Department of Computer Science, University of Saskatchewan, Canada
- *Tom Wolf* – Agrimetrix Research & Training

## 3.12 Summary

This chapter details the architectural design of the Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC). This is a framework that is designed to answer the specific research questions on the personal cloud computing which seeks to support mobile ubiquity. The CSB-UCC is a 4-tier architecture

that comprises n-device mobile endpoints, a broker platform, 3rd party authentication layer, and multi-cloud services. The concept of the CSB-UCC is to allow n-device usage to access multi-cloud services. Some applications that are deployed on the architecture are the ALILI framework which supports file sharing in a groupware, and the Med App which supports n-screen access in mobile health (mHealth). The specific research questions in this work are highlighted below with a short description of how they are solved.

#### **A. How can consumer consistent experience be ensured?**

- (i) *How can new updates be detected and pushed to the consumer irrespective of which service is updated?:* The users can register their devices on the broker and updates available on the multi-cloud sources are pushed to them. There are provisions for offline and online modes that detect which of a user's n-devices are connected and disconnected.
- (ii) *How can application consistency be ensured from the user's perspective?:* The user has to subscribe to multi-cloud services on the broker. Updates are then pushed to the connected devices of the user. In the disconnect state, the broker keeps the updates through notification and when connectivity is restored, the updates are pushed to the user's device.
- (iii) *How can group data and file synchronization be ensured?:* From the lessons learned in the CAP theorem, we propose the weak consistency approach where files are exchanged with changes over a time window. This gives users high availability to work on files in an offline mode and later synchronize them when there is connectivity. Also, the exchange of deltas in a data is proposed in the system rather than propagating the entire data state.
- (iv) *How can soft real-time data accessibility in the mobile network be enforced?:* The idea of notification, and concurrent request issuance to the multi-cloud sources are all aimed at ensuring soft-real time data accessibility. The best-proximity access is proposed which ensures requests are responded to in the shortest time.

#### **B. How can the aggregation of the services and subsequently deliver them to the consumer's multiple devices be ensured?**

- (i) *How can the scalability of the broker be improved?:* The distributed broker architecture is proposed rather than the centralized broker technique. This is to ensure efficient load balancing.
- (ii) *How can concurrency be supported?:* The broker determines how many multi-cloud sources are available and replicates the requests to those cloud sources simultaneously. This approach is expected to outperform the sequential indexing which can introduce higher latency.
- (iii) *How can fault tolerance be ensured?:* The distributed broker ensures that when a single node fails, the other nodes can continue to serve the mobile requests.

(iv) *What is the efficient way to authenticate the user/s from the m-cloud sources and audit?:* There are three authentication modes provided through username/password credentials, 3rd Party authentication based on OAuth 2.0, and hybrid authentication that combines the other two. Furthermore, a provenance methodology is proposed that combines policies with user roles to ensure services tracking and audit trail. The provenance relies on the location, action and time of access to decide whether users can perform some roles or not.

# CHAPTER 4

## EXPERIMENTS/EVALUATIONS

The designed architecture, the CSB-UCC: Cloud Services Brokerage for Ubiquitous Cloud Computing, is evaluated in this chapter to determine whether the quality of service (QoS) factors that were set are achieved. In the concluding sections of chapter 2 (i.e., section 2.4), the following QoS factors are listed to be evaluated:

- Scalability,
- Soft-real time synchronization,
- Error tracking and recovery, and
- Sensor and mobile data flow.

The QoS factors are set to determine whether the major research goals are answered. If not, what are the challenges and limitations that prevent the realization of those goals. For the purpose of evaluation, the architecture will inherit the features of the ALILI framework, the Med App, and the Clandestine Anomaly system as mentioned in section 3.10. This is to enable a practical setup of n-device to multi-cloud services ecosystem. The idea is that, files/data will be stored on various cloud sources and the work will evaluate the time window within which the files are propagated to the various devices. The same approach can be adapted to evaluate the scalability and error tracking. When the need arises in cases where heavy usage is required, simulation techniques are put forward. Further, the broker is built in the Erlang programming language (refer to Appendix C for some code snippets) while the security interface to the 3rd Party authentication cloud is built in C#.

In the next sections, the experimental goals are highlighted. These goals are skewed towards the specific research questions under discussion in this dissertation.

### 4.1 System Requirements and Experiment Goals

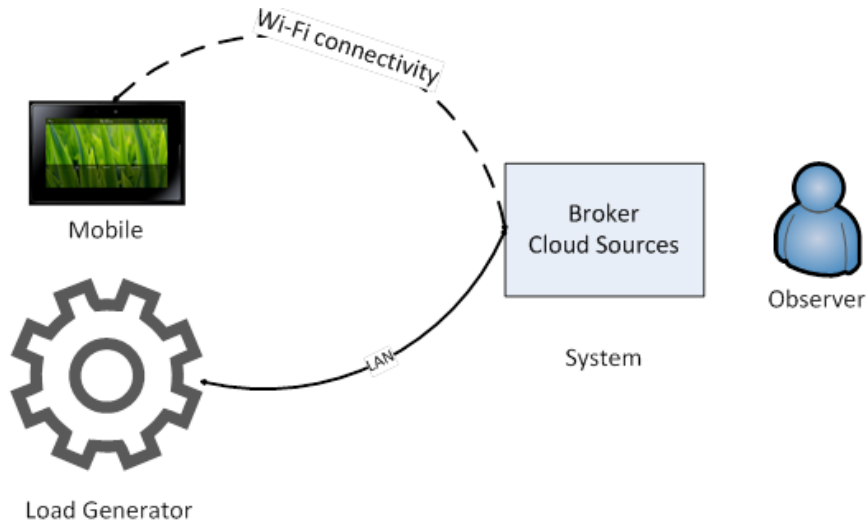
In the experiment, the following mobile devices in Table 4.1 are employed. The mobile devices under consideration are outside the IaaS cloud environment. These devices are purchased specifically for our projects and have less than 5% resource utilization.

The factors that influence client-server interaction such as communication latency, scalability and reliability in a Wi-Fi network are measured and the empirical data is recorded by an observer (Fig. 4.1). In

**Table 4.1:** The Mobile Devices and their Capacity

<i>Device Type</i>	<i>OS</i>	<i>Processor Speed (GHz)</i>	<i>Cores</i>	<i>RAM (GB)</i>	<i>Storage (GB)</i>
BlackBerry Playbook	BB Tablet OS	1.07	Dual-core	1.00	16.00
iPad3	Apple iOS	1.40	A5X (dual-core)	1.00	16.00
Asus Transformer Prime	Android 4.0	NVIDIA Tegra3, 1.30	Quad-core	1.00	32.00
NOKIA Lumia 900	Windows Phone OS	1.40	Single-core	0.512	16.00
PC	Windows 7 System 32	Intel Core i5, CPU 650 @ 3.20GHz 3.19GHz	Dual-core	4.0	500.00

a situation where heavy workloads are needed, simulation is done with a load generator which is built in Erlang/OTP 17.1 (<http://www.erlang.org/>).



**Figure 4.1:** Experimental Set-up

The broker component is hosted on Amazon EC2 instances. To validate the argument of the distributed architecture and the feasibility of the proposed best-proximity policy, several Amazon instances hosted in different regions are required. The multi-cloud sources (mostly IaaS) services under consideration are outlined in Table 4.2. I obtained six (6) different Amazon instances each at two locations, North Carolina in the USA and Asia Pacific (Sydney) in Australia. The system capacities are outlined in Table 4.3. This work also puts

forward the *fault-injection* technique to validate the error tracking claim.

**Table 4.2:** The Multi-Cloud Sources

<i>Cloud Service</i>	<i>OS</i>	<i>Data Storage Capacity</i>	<i>Data Type Stored</i>	<i>File Size (Payload)</i>
DropBox	N/A	5GB	Files	300 MB
MEGA	N/A	50GB	Files	700 MB
Amazon S3	N/A	Unlimited	Files	3 GB
Internal Server	Windows 7 Enterprise	500GB	JSON in CouchDB	100 MB

**Table 4.3:** The Amazon EC2 Servers for Hosting the Distributed Broker

<i>CPU</i>	<i>RAM (GB)</i>	<i>Storage (GB)</i>	<i>Region</i>
Intel Xeon E5410@2.33 GHz	1.70	250	North Carolina
Intel Xeon 5506@2.16 GHz	1.70	250	North Carolina
Intel Xeon 5506@2.12 GHz	1.70	250	North Carolina
Intel Xeon 5506@2.80 GHz	1.70	250	North Carolina
Intel Xeon 5506@2.40 GHz	1.70	250	North Carolina
Intel Xeon 5506@2.33 GHz	1.70	250	North Carolina
Intel Xeon 5506@2.17 GHz	1.63	250	Asia Pacific (Sydney)
Intel Xeon 5506@2.82 GHz	1.63	250	Asia Pacific (Sydney)
Intel Xeon 5506@2.15 GHz	1.73	250	Asia Pacific (Sydney)
Intel Xeon 5506@2.17 GHz	1.44	250	Asia Pacific (Sydney)
Intel Xeon 5506@2.64 GHz	1.45	250	Asia Pacific (Sydney)
Intel Xeon 5506@2.11 GHz	1.64	250	Asia Pacific (Sydney)
Intel Xeon @2.12, 2.33 GHz	16.00	1000	Saskatoon (Internal Server)

Though Amazon Web Services (AWS) offer infrastructure instances that support Linux OS, the windows environment is selected for all instances so that the operating system will not be an influential factor in the determination of results. During the purchase, Amazon recommends the North Carolina region as the closest to my location in Saskatoon. In my estimation, the Asia Pacific (Sydney) region is farther away. This concept will be valuable later when the best-proximity use-case is evaluated. Further, all the Amazon instances support moderate network performance.

The following is an outline of the experiments to be conducted to determine how the implemented architecture deals with the QoS factors in line with the major research goals.

### **Goal 1: Soft-real time synchronization**

In the view of this work, latency can be a hindrance for the facilitation of consistent user experience. Thus, the experimental set up focuses on how the architecture addresses the minimization of the window between updates and their propagation.

*Experiment 1:* Testing the performance of the benchmark platforms (i.e., DropBox, Amazon S3, and MEGA).

*Experiment 2:* Propagation of the files/data which are updated on the IaaS cloud services to the various mobile devices where the broker is centralized.

*Experiment 3:* Propagation of files/data where the broker is centralized and overloaded.

*Experiment 4:* Propagation of the files/data which are updated on the IaaS cloud to the mobile nodes where the broker is distributed.

*Experiment 5:* Evaluation of the proposed load context proximity access.

*Experiment 6:* Data Propagation in the mobile–sensor system.

### **Goal 2: Scalability testing of the brokerage platforms**

In today’s services computing economy, there is the surge in system user base (known as “Big Users”). This calls for systems to support high throughputs and sustain heavy workloads during peak periods. Thus, the scalability of the designed architecture will be tested.

*Experiment 7:* Scalability of the centralized broker.

*Experiment 8:* Scalability of the distributed broker.

### **Goal 3: Error tracking and fault recovery**

Systems can fail, and certainly when a centralized broker fails, the entire system can become inaccessible. However, as the distributed broker is proposed in this work, there is the need to evaluate how the system reacts when a node fails. This creates the need to investigate how the controller re–assigns the task of dead sub–proxies to other proxies.

The error tracking will also be tested in the mobile–sensor network.

### **Goal 4: Audit trail through provenance enforcement**

This experiment will evaluate the proposed provenance policy that is based on system assess following the three context information such as location, time, and action to be taken.

## **4.2 Evaluation of the Propagation Window**

In latency sensitive scenarios, fractions of a second will appear insignificant but when the consumers plus their devices increase within the system, few fractions can turn into intolerable seconds. The update propagation window of the implemented CSB–UCC architecture is evaluated to determine the total time to receive updates on the participating consumer devices. The evaluation at this stage focuses on the propagation window within which the broker is able to push updates to the client devices. The window is measured based on the response



times in a mobile–cloud interaction. In reality, users can update their files on the IaaS directly without the involvement of the broker; so, the experiment is setup in a way that updates are pushed to the three IaaS clouds (i.e., Dropbox, MEGA, Amazon S3) directly from the consumer devices outside the broker. The reason for this experiment is to test how slow or fast it takes to propagate updates to the mobile nodes in groupware environment. For instance, when user **A** updates a file service on MEGA, how long does it take for user **B** to receive this update? The experiment is conducted with the mobile devices in an 802.11g Wi-Fi environment. The results are actual experimental values from the ALILI mobile application.

The experiments are conducted using the client devices that are put forward, and considering every IaaS cloud, the update propagation time is observed . The experimental setup also involves separate experiment for the centralized broker and the distributed broker. The upcoming sections discuss the detail experimental setups and the results.

#### 4.2.1 Resource Synchronization in the Entire System

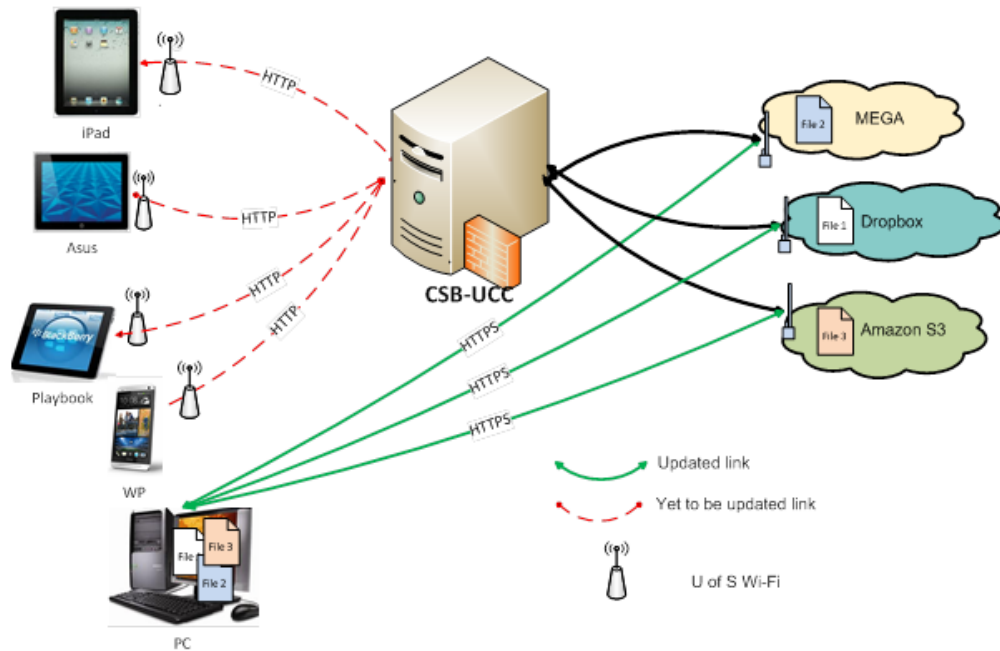
The experiment here is on the determination of the update propagation window for each of the IaaS using the proposed broker. The assumption in this experiment is that, the mobile devices are available and reachable over Wi-Fi. Furthermore, there are several pre-conditions that can be set for this test, but, I focus on the case where updates take place outside the broker as shown in Fig. 4.2. This test case is employed because in reality, cloud services can be interacted with through several interfaces such as web browser, client apps, console, and dashboards. This means it is impractical to confine users to the usage of only one form of exchange through the broker. However, the propagation of the update from the IaaS to the mobile nodes can only be done by the CSB-UCC.

The test at this stage is carried out by measuring the propagation window for an HTTP GET request from the IaaS cloud to the subscribed mobile devices through the CSB-UCC. As shown in Fig. 4.2, I update the records on the IaaS sources using the HTTP POST method from the PC through the University LAN over a gigabit Ethernet connection. The mobile devices connect to the Internet through the University of Saskatchewan secure Wi-Fi network using 802.11g. The CSB-UCC is hosted on the Amazon EC2 instances as outlined in Table 4.3. The file records, represented as File 1, File 2, and File 3, on the IaaS are newly created records and the aim is to propagate them to the mobile nodes. Practically, users can also update existing records and the updates must be propagated. However, the propagation window is minimal compared to the creation of new records of similar file size from real world use case observations such as Dropbox. Thus, this experiment focuses on the creation of new file records since that can potentially lead to the worst case scenario on file transfer time. The total update propagation window is represented in Fig. 4.3.

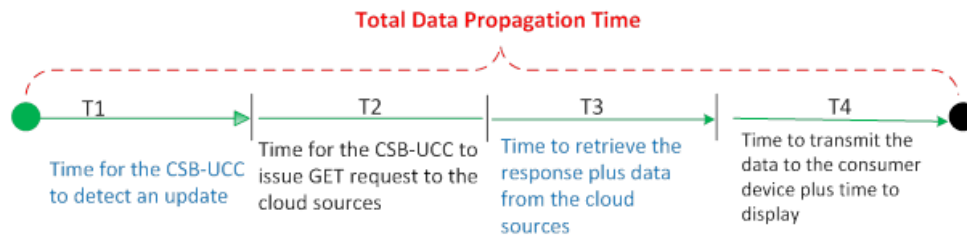
The total propagation window then is:

$$Total\ Time = T1 + T2 + T3 + T4$$

The files used in the experiment are approximately 2 MB each in size. The experiment consists of



**Figure 4.2:** Setup for Testing the File Propagation



**Figure 4.3:** Four-Time Frames for the Total Propagation Time

maximum 2.4 GB file size and this is analyzed across the various devices. In the next sections, the results are reported. For most part of the discussion, the focus is on the determination of the average of the collected dataset. Also, the test is repeated twenty (20) times on each round starting from 100 MB file size to 2.4 GB for all experiments on the update propagation window.

### Experiment 1a: Measuring the Base Propagation Window of Dropbox

In order to evaluate the proposed techniques, there is the need to determine the average propagation window of the various IaaS providers. Starting with Dropbox, the facility has a mobile version that is supported by the client devices under consideration. So, the Dropbox mobile app is installed on the consumer devices for pre-testing. The observer updates the files on the Dropbox server through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. It is important to state that the files are uploaded directly on the Dropbox server through the browser and not through the Dropbox folder on the PC. This is to eliminate the delay/latency when uploading the files to the actual Dropbox

facility which can be dependent on network availability.

The average update propagation window for the various consumer devices is plotted in Fig. 4.4. In this experiment (which is described as the *base result* in this work), the average of the propagation window between the maximum and minimum file sizes (x-axis) is determined for all the devices. Also, the maximum and minimum propagation window (in seconds) are recorded as shown in Table 4.4. The results here form the bases (i.e., the benchmark) for comparison with other results. Since this is the performance of the Dropbox facility in the mobile network, it will mean that any system that is offered must strive to achieve similar/approximate propagation window to guarantee user satisfaction.

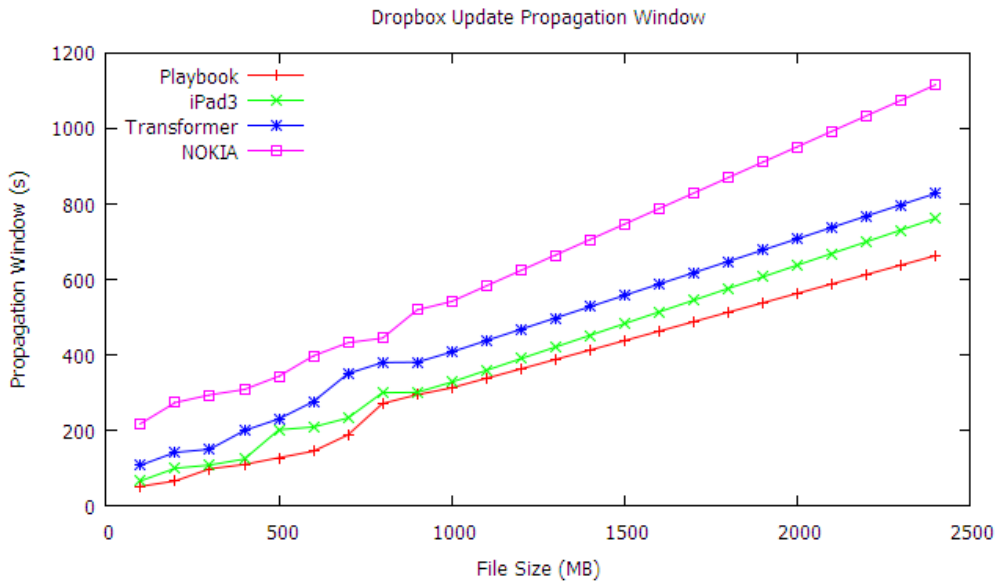


Figure 4.4: Update Propagation Window on Dropbox

Table 4.4: Summary of the Dropbox Result

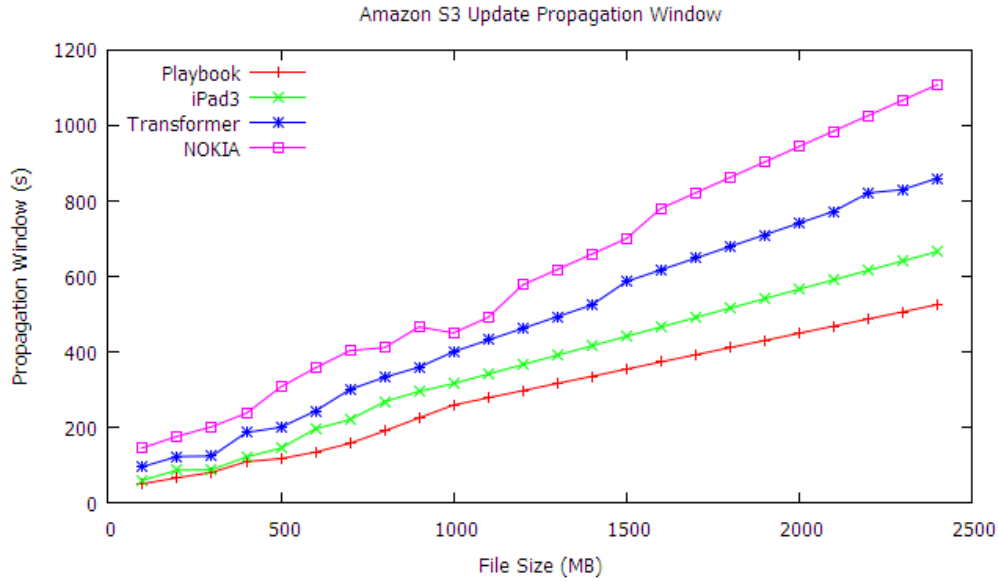
	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	360.79	408.44	478.05	652.12
<i>Maximum Window (s)</i>	662.44	761.35	827.35	1115.23
<i>Minimum Window (s)</i>	51.22	65.56	107.56	217.56

### Experiment 1b: Measuring the Base Propagation Window of Amazon S3

This experiment is similar to the previous but I seek to establish the base results for the Amazon S3 facility. Unlike Dropbox that has mobile apps, Amazon has Software Development Kits (SDKs) specifically for the mobile to enable data access on Amazon S3. Thus, I adapted the Amazon SDK for iOS for the iPad3 and the Android SDK for the Transformer Prime device. Further, the .Net SDK and the Java SDKs are adapted

for the NOKIA and Playbook devices respectively. This is to enable me determine a reasonably sound base file propagation window of the Amazon S3 facility on each device.

The observer updates the files on the Amazon S3 facility through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. The average update propagation window for the various consumer devices is plotted in Fig. 4.5. The average of the propagation window, the maximum, and minimum windows are recorded in Table 4.5.



**Figure 4.5:** Update Propagation Window on Amazon S3

**Table 4.5:** Summary of the Amazon S3 Result

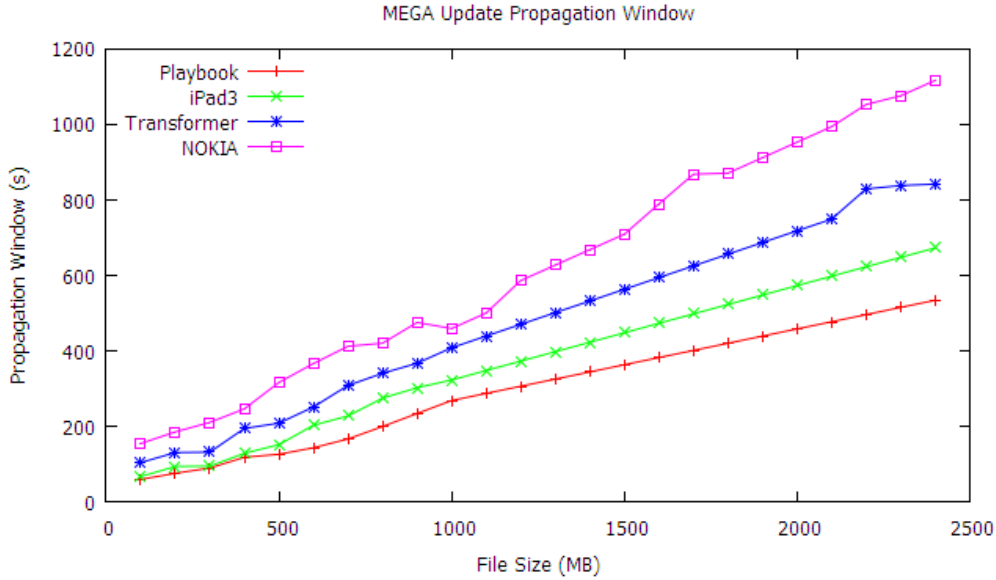
	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	291.93	368.17	480.63	612.27
<i>Maximum Window (s)</i>	524.93	665.66	860.00	1108.00
<i>Minimum Window (s)</i>	50.04	59.44	95.44	145.16

### Experiment 1c: Measuring the Base Propagation Window of MEGA

This experiment is set up to establish the base update propagation window for the MEGA facility. Like Dropbox, MEGA has a mobile App that can be deployed directly on the smartphone and tablet devices with end-to-end encryption. As of the time of performing this experiment, MEGA has Apps for the iOS, Android and the BlackBerry. For the NOKIA phone which is windows powered, I adapted the C# SDK.

Similar to experiments 1a and 1b, the observer updates the files on the MEGA S3 facility through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. The

average update propagation window for the various consumer devices is plotted in Fig. 4.6. The average of the propagation window, the maximum, and minimum windows are recorded in Table 4.6.



**Figure 4.6:** Update Propagation Window on MEGA

**Table 4.6:** Summary of the MEGA Result

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	300.93	375.17	478.51	623.60
<i>Maximum Window (s)</i>	533.93	672.66	842.00	1117.00
<i>Minimum Window (s)</i>	59.04	66.44	103.44	154.16

It is important to state that all the three experiments (experiment 1a, 1b, and 1c) are conducted outside the proposed CSB-UCC. The reason is to establish the benchmark for update propagation time on the devices when any of the IaaS cloud services are employed. There are certain observations made though they are not the focus of this work. For instance, the Amazon S3 facility shows the minimal latency for update synchronization, followed by MEGA and Dropbox shows the highest latency. Also, in terms of the mobile devices, there is a consistency in the output. In the order of retrieving updates, the devices that show the most optimal time are: Playbook, iPad3, Transformer Prime, and NOKIA. The NOKIA device however shows high latency because it is the device with the most feature constraints in comparison to the rest. This means the processing capacity is limited. The phenomenon that is unexplainable is the relatively poor performance of the Android device which has a Quad-core processor.

Now that the benchmark result is observed, the next experiments focus on the employment of the CSB-UCC. The major goal that will be guiding the experimental setups is to determine whether the system aids multiple users to achieve data consistency in soft real-time.

## Experiment 2a: Measuring the Propagation Window of Dropbox (Centralized Broker)

This is the first experiment to test the proposed Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC). The experiment here focuses on the performance of the centralized CSB-UCC in the scenarios where it is used to aggregate data from Dropbox to the mobile nodes. The broker is centralized means it is hosted on a single Amazon EC2 instance as an application server. In this experiment, files are directly placed on the Dropbox facility, but it is the duty of the CSB-UCC to determine the update and push that update to the mobile devices that subscribe to the system. This means the update propagation time follows the diagrammatic illustrations in Fig. 4.2 and Fig. 4.3. As already posited, the importance of the CSB-UCC framework is to allow n-devices to communicate with multi-cloud sources. Hence, the idea here is to test the performance of the framework with regards to Dropbox and subsequently validate the result against the benchmark result of Dropbox.

To perform the experiment, the observer updates the files on the Dropbox server through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. However there is an important observation that needs to be highlighted. That is, since the broker acts as a middleware that handles all incoming and outgoing communications, the broker will always be handling more workloads than just receiving requests from the four mobile devices. So, in this experimental setup, I actually simulate the activities of several users on the broker by using the load generator to mimic actual requesters. The requests are issued to retrieve the files within the same range of 100 MB to 2.4 GB. I observed that for varying number of users, the centralized broker shows resilience for about 3200 users. Within this use case, the result of the centralized broker is within close approximation to the benchmark Dropbox result. For brevity, the summarized outcome is provided in Table 4.7.

**Table 4.7:** Summary of the Dropbox Result (Centralized Broker with Approx. 3200 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	360.37	410.01	475.29	660.99
<i>Maximum Window (s)</i>	662.47	762.92	824.59	1117.93
<i>Minimum Window (s)</i>	61.59	67.13	104.80	220.26

During the experiment, the load generator keeps sending requests to keep the broker busy while I monitor the update propagation window on the actual devices for the requests that they are supposed to receive. When the users of the system go beyond 3200 (representing about 28800 requests) it is observed that the propagation window increases significantly. Also, the maximum number of users I can mimic is 3405 for which the broker becomes non-responsive. The result of the overloaded centralized broker is summarized in Table 4.8.

**Table 4.8:** Summary of the Dropbox Result (Centralized Broker with Over 3200 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	742.29	810.94	932.26	1303.21
<i>Maximum Window (s)</i>	1389.20	1525.07	1640.10	2221.33
<i>Minimum Window (s)</i>	117.45	128.21	199.01	427.08

**Experiment 2b: Measuring the Propagation Window of Amazon S3 (Centralized Broker)**

Using the same centralized broker, I evaluate the performance of the broker in terms of update propagation from the Amazon S3 facility. In this experiment, the files are directly dropped on the Amazon S3 facility and the CSB-UCC propagates the files to the mobile end-points. To perform the experiment, the observer updates the files on the Amazon S3 facility through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. Also, the requests are issued to retrieve the files within the same range of 100 MB to 2.4 GB. I observed that for varying number of users, the centralized broker shows resilience for about 3214 users. This number is just about 14 users more than the Dropbox scenario. Within this use case, the result of the centralized broker is within close approximation to the benchmark Amazon S3 result. The result is tabulated in Table 4.9. Similar to experiment 2a, the load generator keeps sending requests to keep the broker busy (mimicking 3214) users while I monitor the update propagation window on the actual devices for the requests that they are supposed to receive.

**Table 4.9:** Summary of the Amazon S3 Result (Centralized Broker with Approx. 3214 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	324.94	370.38	473.52	603.52
<i>Maximum Window (s)</i>	571.63	667.87	837.93	1064.54
<i>Minimum Window (s)</i>	54.74	61.65	99.37	147.36

When the users of the system go beyond 3214 (representing about 28900 requests) it is observed that the propagation window increases significantly. Also, the maximum number of users I can mimic is 3462 for which the broker becomes non-responsive. The result of the overloaded centralized broker is summarized in Table 4.10.

**Table 4.10:** Summary of the Amazon S3 Result (Centralized Broker with Over 3214 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	582.90	633.96	900.63	1140.76
<i>Maximum Window (s)</i>	1036.36	1142.73	1593.74	2012.19
<i>Minimum Window (s)</i>	95.53	105.61	189.00	278.53

### Experiment 2c: Measuring the Propagation Window of MEGA (Centralized Broker)

The final experimental setup with the centralized broker focuses on the update propagation window from the MEGA facility. In this experiment, the files are directly dropped on the MEGA facility and the CSB-UCC propagates the files to the mobile devices. To perform the experiment, the observer updates the files on the MEGA facility through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. Also, the requests are issued to retrieve the files within the same range of 100MB to 2.4GB. It is observed that for varying number of users, the centralized broker shows resilience for about 3206 users. This number is just about 6 users more than the Dropbox scenario and 8 less than the Amazon S3 scenario. Within this use case, the result of the centralized broker is within close approximation to the base MEGA result. The result is tabulated in Table 4.11. Similar to experiments 2a and 2b, the load generator keeps sending requests to keep the broker busy (mimicking 3206 users) while the update propagation window is monitored on the actual devices for the requests that they are supposed to receive.

**Table 4.11:** Summary of the MEGA Result (Centralized Broker with Approx. 3206 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	329.24	376.97	482.38	617.89
<i>Maximum Window (s)</i>	575.93	674.46	903.33	1068.13
<i>Minimum Window (s)</i>	59.04	68.24	105.77	150.95

When the users of the system go beyond 3206 (representing about 28854 requests) it is observed that the propagation window increases significantly. Also, the maximum number of users I can mimic is 3431 for which the broker becomes non-responsive. The result of the overloaded centralized broker is summarized in Table 4.12.

**Table 4.12:** Summary of the MEGA Result (Centralized Broker with Over 3206 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	631.81	664.40	911.69	1184.36
<i>Maximum Window (s)</i>	1105.20	1281.48	1707.29	2028.38
<i>Minimum Window (s)</i>	113.31	115.33	199.91	286.65

Since the centralized CSB-UCC has shown its capacity to handle concurrent requests from averagely 3200 users, there is the need to explore other techniques to increase this threshold value. Thus, the proposed distributed architecture of the CSB-UCC is tested in the upcoming experiments.

### Experiment 3a: Measuring the Propagation Window of Dropbox (Distributed Broker)

The distributed CSB-UCC architecture is tested by hosting the (i.e., the sub-proxies) on six (6) Amazon EC2 instances. All the amazon EC2 instances are located in North Carolina region. The controller is hosted



on the Windows server in Saskatoon so that it can distribute the load across the various distributed nodes.

Again, the aim is to evaluate the performance of the CSB-UCC in the scenarios where it is used to aggregate the files from Dropbox to the mobile nodes. Identical to the centralized broker, the files are directly dropped on the Dropbox facility and the CSB-UCC is expected to determine the update and push that update to the mobile devices that subscribe to the system. The only difference here is that, the identified update is assigned to a particular node of the broker to handle. Thus, as the requests are increasing, the broker will be distributing the tasks to several of the nodes.

To perform the experiment, the observer updates the files on the Dropbox server through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices. Similarly, the broker is flooded with several requests to keep the system busy. The load generator sends requests to simulate the activities of about 20026 users (representing about 180234 requests). The result is reported in Table 4.13. The experiment is conducted with similar range of file size from 100 MB to 2.4 GB.

**Table 4.13:** Summary of the Dropbox Result (Distributed Broker with Approx. 20026 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	409.44	461.23	540.85	779.07
<i>Maximum Window (s)</i>	750.81	856.83	935.20	1322.83
<i>Minimum Window (s)</i>	71.82	76.85	122.56	254.60

The maximum number of users before the system starts exhibiting signs of higher than accepted latency is 20087.

### **Experiment 3b: Measuring the Propagation Window of Amazon S3 (Distributed Broker)**

This experiment inherits the same setup as that of experiment 3a. The idea is to determine the update propagation window on the Amazon S3 facility when the broker is distributed. To perform the experiment, the observer updates the files on the Amazon S3 facility through the PC, and monitors the propagation window for the update to be synchronized on the consumer devices.

The load generator sends requests to simulate the activities of about 20098 users (representing about 180882 requests). The result is reported in Table 4.14. The experiment is conducted with similar range of file size from 100 MB to 2.4 GB. The maximum number of users before the system starts exhibiting signs of high latency is 20114.

### **Experiment 3c: Measuring the Propagation Window of MEGA (Distributed Broker)**

This experimental setup is similar to experiments 3a and 3b. The idea is to determine the update propagation window on the MEGA facility when the broker follows the distributed architecture. To perform the experiment, the observer updates the files on the MEGA facility through the PC, and monitors the propagation

**Table 4.14:** Summary of the Amazon S3 Result (Distributed Broker with Approx. 20098 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	374.68	444.75	558.34	757.66
<i>Maximum Window (s)</i>	663.30	804.12	991.63	1338.55
<i>Minimum Window (s)</i>	58.55	71.80	113.48	182.90

window for the update to be synchronized on the consumer devices. The load generator sends requests to simulate the activities of about 20077 users (representing about 180693 requests). The result is reported in Table 4.15. The experiment is conducted with similar range of file size from 100 MB to 2.4 GB. The maximum number of users before the system starts exhibiting signs of high latency is 20092.

**Table 4.15:** Summary of the MEGA Result (Distributed Broker with Approx. 20026 Users)

	<i>Playbook</i>	<i>iPad3</i>	<i>Transformer</i>	<i>NOKIA</i>
<i>Average (s)</i>	367.34	427.58	564.23	709.39
<i>Maximum Window (s)</i>	641.16	763.75	1056.74	1222.66
<i>Minimum Window (s)</i>	67.43	78.72	123.60	177.07

## 4.2.2 Discussion of Experiments on the Propagation Window

The results from Experiments 1a – 1c, 2a–2b, and 3a–3c give an idea about how the CSB–UCC influences latency in mobile distributed systems. All the nine (9) experiments focus on the determination of data propagation window in an n–device to multi–cloud ecosystem. Throughout the experiments, the file sizes are kept within the range of 100 MB to 2.4 GB. In experiments, 1a – 1c, I investigate the data propagation window of existing mobile enterprise solutions such as the Dropbox mobile App, the Amazon SDK for mobile, and the MEGA mobile Apps. The reason for the evaluation of these public offerings is to determine the base (benchmark) values with which I can compare/validate the performance of our proposed CSB–UCC architecture.

In experiments 2a – 2c, six (6) results are reported on the employment of the CSB–UCC as a centralized broker. The CSB–UCC is hosted on Amazon EC2 instance and the propagation window is determined using the client devices. With regards to services accessibility on Dropbox, I realized that up to 3200 users (representing approximately 28800 requests) can be supported to retrieve updated data within a time that is close to the base result for Dropbox. The number of requests is the number of files that can be retrieved using the HTTP GET method. However, users beyond 3200 experience a significant delay in the data propagation. Also, it is observed that the centralized broker supports 3214 (representing about 28900 requests) when propagating data from the Amazon S3 facility within close approximation to the base results on Amazon S3 App. Moreover, the centralized broker enables about 3206 users (representing approximately 28854 requests)

to be served when connected to the MEGA facility. Within the stipulated range of supported number of users, we observed that the update propagation window is reasonably close to the base values. The delay is within some insignificant seconds more than the base value.

However, in real world, the users of a system can be more than an average 3200 users. This creates the need to enhance the system capacity to maintain similar propagation window but with bigger user support. Thus, the distributed architecture of the CSB-UCC is tested with six active nodes. I realized that the following number of users can be supported: Dropbox – 20026 users (representing about 180234 requests), Amazon S3 – 20098 users (representing about 180882 requests), and MEGA – 20077 users (representing about 180693 requests). The distributed broker architecture raises the number of supported users to approximately 20000 because each node has the capacity of a centralized server.

For clarity, a summarized overview of the results are presented in Table A.1 in Appendix A. Considering the three public cloud services, I report the results obtained for each client device. The result includes the maximum propagation window within the file size range (Max Window) in seconds, the minimum propagation window (Min Window) in seconds, and the Average (Avg) in seconds. These results are recorded for each set of experiments which includes the base (benchmark) results (Base), the centralized broker within a supported user range (Centralized), the overloaded centralized broker (Overloaded), and the distributed broker architecture (Distributed).

By focusing only on the Avg, I calculate the average propagation window relative to the base result. This is an attempt to determine how slow or fast the proposed system performs against the benchmark results. This is calculated based on the formula:

$$\text{Average Propagation Window Relative to the Base Result} = Avg_B - Avg_M$$

where  $Avg_B$  is the benchmark average result for each IaaS mobile service and  $Avg_M$  is the average result for each experimented broker architecture. So, assuming I want to calculate the average propagation window relative to the benchmark result for the Playbook using Dropbox, the result for the centralized broker is 360.79 - 360.37 which equals 0.42. This means that on average, the update propagation window of the centralized broker is 0.42 seconds faster than the Dropbox App on the Playbook device. Furthermore, the average propagation window relative to the base result for the Playbook using Dropbox when the distributed CSB-UCC is employed is 360.79 - 409.44 which equals -48.65. This means the average propagation window is 48.65 seconds slower when the distributed architecture is employed to propagate the updates from Dropbox in comparison to the Dropbox mobile App on the Playbook.

From the results, it is observed that clearly the delays which are introduced when the centralized server is overloaded can be intolerable. Depending on the domain that will adopt the CSB-UCC, waiting for several minutes to retrieve an update can be time wasting. Especially, in mission critical systems, the luxury of waiting for updates for minutes may not be afforded. Thus, such systems can adopt the distributed architecture approach to ensure load distribution that can lead to reduction in the propagation window. However, the distributed architecture is not a viable approach if the workload is minimal (at least within

the test range of up to 3200 users). In this case, the centralized architecture can be adopted because the propagation window is relatively same. The caveat with the centralized architecture however is the fact that when the broker crashes, the entire system can become non-usable since the broker acts as an application router. A possible solution is to keep a backup copy of the application state that can be employed in the event of failures.

Though the distributed architecture shows great promise, it can also lead to higher cost in terms of expenses to buy several IaaS nodes for hosting, and system management. The distributed architecture means the application states will be hosted on multiple servers and enabling a coordinator to manage the communication flow in decentralized manner. In such cases, the management requires extra efforts when it comes to the detection of faults and maintenance. So, with the pros and cons highlighted, the adoption of the CSB-UCC architecture should be guided by the need of the enterprise regarding latency minimization, the workforce base of the enterprise, and the level of financial/cost commitment.

Next, the proposed best-proximity use case is tested.

### 4.2.3 The Best-Proximity Based Job Assignment

In this section, the proposed best-proximity methodology is evaluated. This experiment is applicable only to the distributed architecture of the CSB-UCC. The position taken in this dissertation is that, the case of assigning jobs to the sub-proxy within the closest proximity in a distributed architecture is not enough reason to attain real time service delivery. Rather, this work argues that the job assignment should be sent to the sub-proxy (or the distributed node) that produces the minimal latency during the execution of system jobs in the cloud. This argument is validated by proposing the best-proximity use case which is dependent on the formula below:

$$\text{Total Time} = \text{Req}_t + \text{Asgn}_t + \text{Proc}_t + \text{Resp}_t$$

Where  $\text{Req}_t$  is the request time from the mobile node,  $\text{Asgn}_t$  is the time taken for the controller to assign the request to the appropriate sub-proxy plus the time taken for the request to reach the sub-proxy,  $\text{Proc}_t$  is the request processing time of the sub-proxy, and  $\text{Resp}_t$  is the response time of the sub-proxy to the mobile node. The formula means the best-proximity is the use case that returns the least total time rather than just assigning requests to nodes that are closest to the mobile participant. For the purpose of later discussions on location-specific tests, Table 4.16 is provided that gives an overview of how the various servers are geographically dispersed. The values are taken from Google Map.

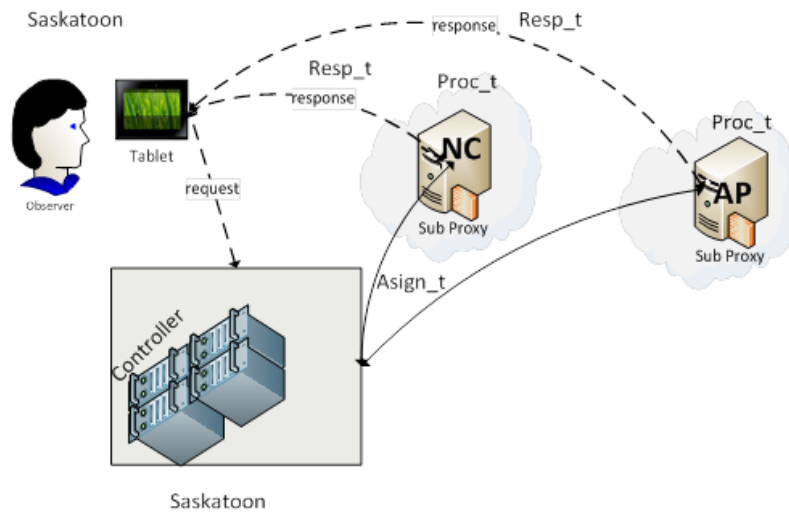
#### Hosting the Controller in Saskatoon

To conduct this experiment, I use the iPad3 mobile device in Saskatoon where I evaluate the Med App system that is deployed based on the distributed CSB-UCC architecture. The controller is also hosted on the internal server (i.e., the windows 7 enterprise system) in Saskatoon. One sub-proxy is hosted on Amazon EC2 instance

**Table 4.16:** Geographical Distance Between the Servers

<i>Location</i>	<i>Destination</i>	<i>Approx. Distance (km)</i>
Saskatoon	North Carolina	3379.50
Saskatoon	North Virginia	3468.90
Saskatoon	Sydney	13682.10
Saskatoon	Tokyo	8322.10
Saskatoon	Singapore	13410.10
North Carolina	Sydney	15465.80
North Carolina	Tokyo	11110.91
North Carolina	Singapore	15868.13
North Carolina	North Virginia	374.33

in North Carolina (NC) which is the closest location to Saskatoon according to Amazon’s recommendation. Then, a second sub-proxy is hosted on the Amazon EC2 instance in Asia Pacific – Sydney (AP) which is farther away from Saskatoon, where both the requester and controller are located. The experimental set-up is shown in Fig. 4.7.



**Figure 4.7:** Set-up for the Evaluation of the Proximity Accessibility

The controller connects to the sub-proxies through the University of Saskatchewan LAN network. In order to achieve almost identical processing time to the data source, CouchDB database (<http://couchdb.apache.org/>) is installed on every EC2 instance where the proxy is located. This means, there will be no variance in the travel time to the data source by the sub-proxy. This also allows for equal payload consideration for all the nodes. A data payload under consideration which is hosted in the CouchDB database is shown in Fig. 4.8. This data is sample survey information stored from the Med App users.

```

HTTP GET Request
GET id=9c0c736e8206577d9acd502592002b02 HTTP/1.1
Host: lomoteyr.usask.ca
Accept: Application/json

Data from CouchDB
HTTP/1.1 200 OK
Server: CouchDB/1.0.2 (Erlang OTP/R13B)
Etag: "9-a981e1f8d8ba2fcf86f088b98119e374"
Content-Type: application/json
Content-Length: 647

{"_id":"9c0c736e8206577d9acd502592002b02","_rev":"1-605728c3ac07c87972dc2b58c6f1c241","Date":"Fri Feb 21 2014 17:49:21 GMT-0600 (CST)","Device_Version":"6.1.3","Device_Model":"iPad3,1","Device_Platform":"IOS","App_Accessibility":4,"Using_App_Injury_Accessibility":"Yes","Using_App_Compression":"No","Using_App_Rest":"Yes","Using_App_Elevation":"Yes","Using_App_Ice":"No","Using_App_Contact":"No","Using_App_Ignore_Injury":"No","Without_App_Compression":"No","Without_App_Rest":"Yes","Without_App_Elevation":"Yes","Without_App_Ice":"No","Without_App_Contact":"Yes","Without_App_Ignore_Injury":"No","Ability_to_Manage_Injury":4,"App_Reusability":5}

```

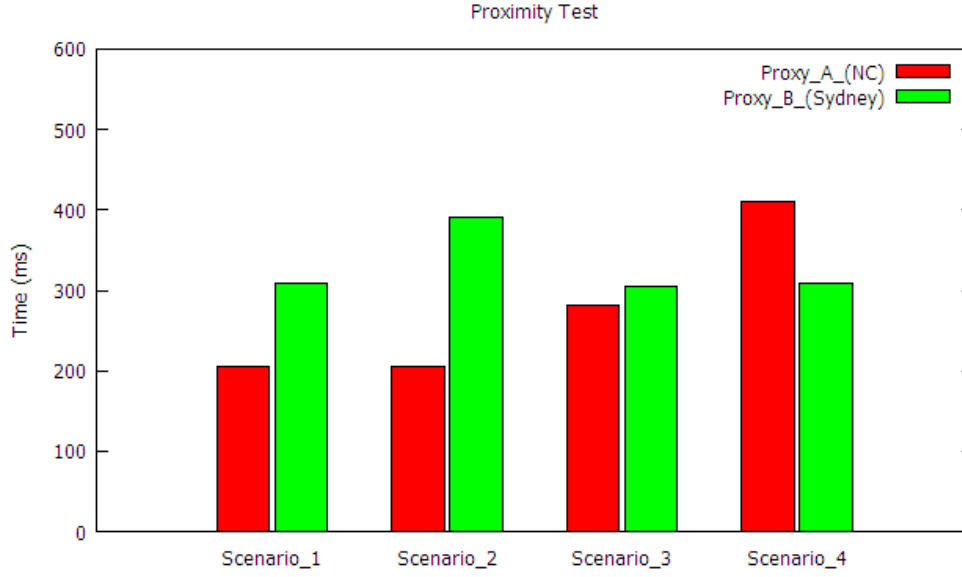
**Figure 4.8:** Request–Response Payload from CouchDB

Since the CouchDB is hosted on the same environment as the sub-proxy, the  $Proc_t$  is considered as the time taken for the proxy to process a request and retrieve the data from the database as well. In the experiment, I issue requests from the mobile to the controller to retrieve the information in the database. The results from the experiment are plotted in Fig. 4.9 and the detail breakdown is provided in Table 4.17. It is important to state that this experiment is repeated using several other nodes in the Asia Pacific (Tokyo) region, Asia Pacific (Singapore) region, and North Virginia. For brevity, the section discusses the experience with the Asia Pacific (Sydney) region in comparison with the North Carolina region, and the other results are provided in Appendix B since the trend is similar.

**Table 4.17:** Breakdown of the Proximity Access (Time in ms) – Sydney

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	21	7	122.33	55	<b>205.33</b>
	Proxy B (Sydney)	21	37	126.67	125	309.67
<i>Scenario 2 (Double Sydney Workload)</i>	Proxy A (NC)	21	7	120.43	57	<b>205.43</b>
	Proxy B (Sydney)	21	37	183.45	149	390.45
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	21	7	190.85	62	<b>280.85</b>
	Proxy B (Sydney)	21	37	127.12	124	309.12
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	21	7	312.71	70	410.71
	Proxy B (Sydney)	21	37	126.38	125	<b>309.38</b>

The experiment is conducted by following four (4) different scenarios. Since it is determined from previous



**Figure 4.9:** The Proximity Test – Sydney

experiments that a single sub-proxy can handle up to about 3200 user requests, I kept the sub proxies busy with 1000 users each in *Scenario 1*. This means both proxies are having equal workloads.

Proxy A (NC) is the proxy in North Carolina which is within the closest proximity per the geographical difference between the mobile requester and the proxy. Proxy B (Sydney) is the proxy hosted in Asia Pacific (Sydney) region. Based on the closest proximity logic, it will be expected that Proxy A will always serve requests in the shortest possible time, an attempt that can further aid services delivery in soft-real time in mobile cloud ecosystems. This is the assumption made in the current state of techniques such as edge computing. Considering the experiment in Scenario 1 where the workload is equal, Proxy A shows the fastest request processing time with the least total time of 205.33ms in comparison to Proxy B which gives total time of 309.67ms. The mobile request time ( $Req_t$ ) is approximately 21ms for all the scenarios because the mobile has the same location distance to the controller. The time for the controller to reach the closest proxy (i.e., Proxy A in North Carolina – NC) is approximately 7ms while Proxy B in Asia Pacific (Sydney) is reachable at approximate time of 37ms. These observations are synonymous with all the scenarios. Again, in Scenario 1, Proxy A is able to send the response to the mobile at approximate time of 55ms while Proxy B takes approximate 125ms to communicate the response to the mobile. These are the determining factors since the processing time are almost identical with the equal workload. In Scenario 1, the controller will assign the job to the proxy within the closest proximity (i.e., Proxy A) because the game changer is the  $Asign_t$  and the  $Resp_t$ . This further validates the existing position taken by some cloud services providers that the requests should be sent to server nodes within the closest proximity.

Next, a second experimental setup is prepared called *Scenario 2* where I double the workload of Proxy B in Asia Pacific (Sydney). Doubling the workload means the user base is increased to 2000 while the user base

for Proxy A remains at 1000. As Scenario 1, all the determinants are the same except for the fact that the processing time (i.e.,  $\text{Proc}_t$ ) of Proxy B has increased. Also, a marginal increase in the response time (i.e.,  $\text{Resp}_t$ ) is observed since the size of data affects the data transfer rate across the bandwidth. But, clearly, the reason for the increase in the total time is because of the significant increase in the processing time of Proxy B which has its workload doubled. At this point, it is clear that all scenarios that involve the increment of the  $\text{Proc}_t$  component on Proxy B will favour Proxy A. Hence, the appropriate node to send the request to in order to achieve optimal request–response time is Proxy A (the closest proxy to the mobile) even though closeness is not the deciding factor in this scenario.

The third set up, which is *Scenario 3*, is the case where the workload is reversed and Proxy A in North Carolina has its workload doubled to 2000 users and Proxy B has 1000 users. Here, it is observed that the  $\text{Proc}_t$  of Proxy A has increased significantly in comparison to Proxy B. Eventually, it has resulted in a high latency for Proxy A which is the closest to the mobile participant. Though the total time is marginally smaller than that of Proxy B, it is obvious the processing time is a major factor that can hamper soft–real time data transfer in mobile ecosystems.

In *Scenario 4*, the workload of Proxy A is quadrupled, and this means it has to serve the maximum capacity of users. Proxy B on the other hand serves 1000 users and now the leap in the total time can be seen as a result of the increased  $\text{Proc}_t$ . It is noteworthy to state that at maximum user capacity, Proxy A sends the "busy" status to the controller so if the close proximity technique is adopted, then the controller has to queue the requests to be served later. This will further increase the processing time as well as the job assignment time. However, in the best–proximity use case, the requests will be sent to the farther server for processing instead of queuing. In this scenario, it will be appropriate to send requests to the far away sub-proxy in North Virginia and still achieve request-response optimality.

However, questions can be asked whether the location of the controller played a key role in the request–response time that eventually translated into the factors influencing latency. This question is key because it can be argued that hosting the controller on a local computer in Saskatoon and making requests to the closest sub-proxy in North Carolina could be the reason and not that in most cases, contextual information such as processing demand of the system should be considered. The quest to answer this question led to the setup of another experiment that has the controller hosted first in North Carolina. The upcoming discussions explain further.

### **Hosting the Controller in North Carolina**

Identical to the earlier setup, the iPad3 mobile device is used to evaluate the Med App system as the distributed version of the CSB–UCC. It is important to point that the user is still in Saskatoon. However, the controller is hosted on one of the Amazon EC2 instances in North Carolina. One sub-proxy is hosted on Amazon EC2 instance in North Carolina (NC) which is the closest location to to the controller. Then, a second sub-proxy is hosted on the Amazon EC2 instance in Asia Pacific – Sydney (AP) which is farther

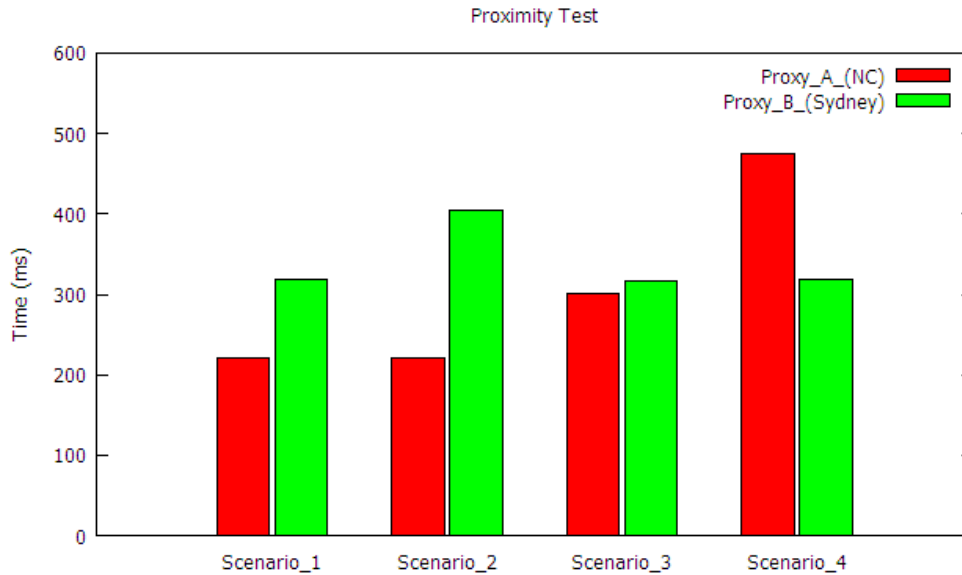


away from both the controller in North Carolina and the mobile user in Saskatoon.

The controller connects to the sub-proxies through an Ethernet connection that guarantees 500 MB/s upload and download. The CouchDB database is also installed on all the sub-proxies required for the experiment. This is to avoid travelling cost regarding data access or better still maintain consistent view of the travel time since none is required. This means, there will be no variance in the travel time to the data source by each sub-proxy under consideration. The same data payload which is shown in Fig. 4.8 is employed which contains sample survey information of the Med App.

Since the CouchDB is hosted on the same environment as the sub-proxy, the  $Proc_t$  is considered as the time taken for the proxy to process a request and retrieve the data from the database as well.

Commencing the experiment, a request is issued from the mobile (while in Saskatoon) to the controller (in North Carolina) to retrieve the information in the database either in North Carolina or Sydney. The results from the experiment are plotted in Fig. 4.10 and the detail breakdown is provided in Table 4.18. It is important to state that this experiment is repeated using several other nodes in the Asia Pacific (Tokyo) region, Asia Pacific (Singapore) region, and North Virginia. The results here will be discussed based on the experiment regarding the Asia Pacific (Sydney) region in comparison with the North Carolina region. In Appendix B, the other results from the other regions are provided. The discussions are sent to the Appendix to avoid repetition of text since the results are mostly identical.



**Figure 4.10:** The Sydney Proximity Test with Controller in North Carolina

The experiment is conducted by following four (4) different scenarios just as done previously with the hosting of the controller in Saskatoon. Since it is determined from previous experiments that a single sub-proxy can handle up to about 3200 user requests, each sub proxy is kept busy with 1000 users each in *Scenario 1*. This is to ensure equal workload on both proxies.

**Table 4.18:** Breakdown of the Proximity Access (Time in ms) – Sydney with Controller in North Carolina

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	34	6	125.12	55	<b>220.12</b>
	Proxy B (Sydney)	34	32	127.33	125	318.33
<i>Scenario 2 (Double Sydney Workload)</i>	Proxy A (NC)	34	6	122.92	57	<b>219.92</b>
	Proxy B (Sydney)	34	32	190.3	149	405.3
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	34	6	199.43	62	<b>301.43</b>
	Proxy B (Sydney)	34	32	126.42	124	316.42
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	34	6	364.25	70	474.25
	Proxy B (Sydney)	34	32	127.96	125	<b>318.96</b>

Proxy A (NC) is the proxy in North Carolina which is within the same geographical location as the controller. Proxy B (Sydney) is the proxy hosted in the Asia Pacific (Sydney) region. Based on the issue being investigated, the focus is on both the influence of the controller and the proximity between requesters. Since the controller and Proxy A are all in North Carolina, it is easy to draw conclusion that requests from that region will be served faster. What even makes this case special is the fact that the requester is also in Saskatoon that is closest to both the controller and the Proxy A. Considering the experiment in Scenario 1 where the workload is equal, Proxy A shows the fastest request processing time with the least total time of 220.12ms in comparison to Proxy B which gives total time of 318.33ms. The mobile request time ( $Req_t$ ) is approximately 34ms for all the scenarios because the mobile requests are coming to the controller from a bounded location (i.e., the University of Saskatchewan Campus). The first observation is that, there is an increase in the  $Req_t$  from 21ms in the previous setup. This is because the controller is now away from Saskatoon so there is a cost associated with the request time. The time for the controller to reach the closest proxy (i.e., Proxy A in North Carolina – NC) is approximately 6ms while Proxy B in Asia Pacific (Sydney) is reachable at approximate time of 32ms. It can be seen that compared to the previous setup, the task assignment time is almost the same for Proxy A but there is a drop in the task assignment task to Proxy B. An explanation for this is the fact that inter-cloud communication is faster compared to hosting the service locally in the other scenarios. Furthermore, in Scenario 1, Proxy A is able to send the response to the mobile at approximate time of 55ms while Proxy B takes approximate 125ms to communicate the response to the mobile. These observation is identical to the previous setup because in the CSB-UCC, proxies send responses directly to the requesters. So, regardless of the time consumed by other system components, the response time will be the same for the same job demand.

In the experiment, it is seen that the main sources of latency are the processing time, the task assignment time, and the response time. But, the crucial factor is the response time and the task assignment durations. The fact that the processing demand is almost similar for both proxies means the controller will assign

incoming tasks to Proxy A since it has the least time to respond to a request. This further validates the assumption that requests should be routed to the proxy within the closest location. Here, it is observed that the location of the controller also plays part because we can say that the time to Proxy A is significantly less than the time to reach Proxy B due to location.

A second experiment is provided called *Scenario 2* where the workload of Proxy B is in Asia Pacific (Sydney) is doubled. Doubling the workload means the user base is increased to 2000 while the user base for Proxy A remains at 1000. As Scenario 1, all the determinants are the same except for the fact that the processing time (i.e.,  $Proc_t$ ) of Proxy B has increased. Also, a marginal increase in the response time (i.e.,  $Resp_t$ ) is observed since the size of data affects the data transfer rate across the bandwidth. But, clearly, the reason for the increase in the total time is because of the significant increase in the processing time of Proxy B which has its workload doubled. At this point, it is clear that all scenarios that involve the increment of the  $Proc_t$  component on Proxy B will favour Proxy A. Hence, the appropriate node to send the request to in order to achieve optimal request–response time is Proxy A (the closest proxy to the mobile) even though closeness is not the deciding factor in this scenario. Another observation is that, apart from the task assignment time that varies in both scenarios, the location of the controller is of no effect. It can happen probably if there time difference is significant as may be seen later.

The third set up, which is *Scenario 3*, is the case where the workload is reversed and Proxy A in North Carolina has its workload doubled to 2000 users and Proxy B has 1000 users. Here, it is observed that the  $Proc_t$  of Proxy A has increased. Considering that all factors are same from the previous two setups, the increment in latency can be attributed to the workload demand.

In *Scenario 4*, the workload of Proxy A is quadrupled, and this means it has to serve the maximum capacity of users. Proxy B on the other hand serves 1000 users and now the leap in the total time can be seen as a result of the increased  $Proc_t$ . The system at maximum capacity sends "busy" state notice so requests will not be served at such saturation level.

In this experiment involving the re–location of the controller to North Carolina, it is seen that not much has changed compared to keeping the controller in Saskatoon. The reason is because, as the user request time has increased, the inter–cloud communication is faster. This also means that any attempt to reduce the task assignment time will give the system consistent performance regardless of the location of the controller. What is seen in the scenarios is the fact that the processing time and the response time should be considered hand–in–hand when building such edge–based architectures.

But, what is worth noting is the fact that the re–location of the controller changed the task assignment time marginally. This means, the location of the controller could prove to be an important factor. However, the question is how far will the performance of the CSB–UCC be influenced by the location of the controller? The previous two experiments evaluate the controller from Saskatoon and North Carolina. Since this locations are relatively close, it is important to examine what happens when the controller is far from the requester. Arguments can be raised that the good performance and the stability of the results in previous setups is due

to the fact that the controller is relatively closer to the requester. To probe this issue further, a third setup is provided that hosts the controller in the Asia Pacific region. The next section discusses this setup.

### Hosting the Controller in Asia Pacific (Sydney)

A third experimental setup is put forward to evaluate the CSB-UCC using the iPad3 mobile device. The Med App service is employed in this experiment. The mobile user resides in Saskatoon just as the case with the other setups. However, the controller is hosted on one of the Amazon EC2 instances in the Asia Pacific region; specifically Sydney. One sub-proxy is hosted on Amazon EC2 instance in North Carolina (NC) which is the closest location to to the mobile user (or requester) while the second sub-proxy is hosted on a Amazon EC2 instance in Asia Pacific – Sydney (AP) which is farther away from the mobile requester but within the same geographical location as the controller.

The controller connects to the sub-proxies through an Ethernet connection that guarantees 500 MB/s upload and download. Similarly, CouchDB database is also installed on all the sub-proxies required for the experiment. this is to avoid travelling cost regarding data access or better still maintain consistent view of the travel time since none is required. This means, there will be no variance in the travel time to the data source by each sub-proxy under consideration. The same data payload which is shown in Fig. 4.8 is employed which contains sample survey information of the Med App. Again, the  $Proc_t$  is considered as the time required to process a request (which includes data retrieval from the database).

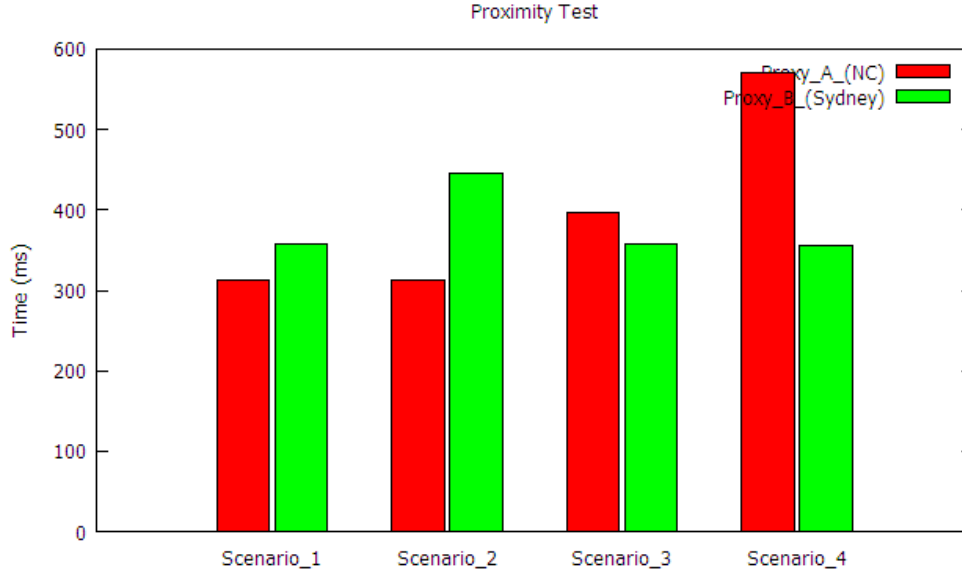
This experiment is unique because the requester is far from the controller (which is the initial access point of the CSB-UCC). To begin with, the user sends a request from Saskatoon using the mobile to the controller in Sydney. This request includes the demand to retrieve an information from the associated CouchDB databases that are hosted on each instance of Amazon EC2.

The results from the experiment are plotted in Fig. 4.11 and the detail breakdown is provided in Table 4.19.

**Table 4.19:** Breakdown of the Proximity Access (Time in ms) – Sydney with Controller in Sydney

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	98	34	125.29	55	<b>312.29</b>
	Proxy B (Sydney)	98	7	126.98	125	356.98
<i>Scenario 2 (Double Sydney Workload)</i>	Proxy A (NC)	98	34	123.06	57	<b>312.06</b>
	Proxy B (Sydney)	98	7	191.34	149	445.34
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	98	34	203.22	62	397.22
	Proxy B (Sydney)	98	7	128.22	124	<b>357.22</b>
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	98	34	369.22	70	571.22
	Proxy B (Sydney)	98	7	126.44	125	<b>356.44</b>

The experiment is conducted based on the four (4) different scenarios proposed in the previous setups but



**Figure 4.11:** The Sydney Proximity Test with Controller in Sydney

the controller in this case is hosted in Sydney. Lessons from past experiences also aided in how the workload should be distributed starting with 1000 to 3200 users for a single node.

Hence, the experiment started with *Scenario 1* that has 1000 users on all nodes to ensure balanced workload.

Proxy A (NC) is the proxy in North Carolina while Proxy B (Sydney) is the proxy hosted in the Asia Pacific (Sydney) region and it is closest to the location of the controller. Again, what is important here is to analyse the impact of the location of the controller on the CSB-UCC. In this setup, it is not easy to guess or assume which sub-proxy will have the best performance result because of the variability of the determining factors. Considering the experiment in Scenario 1 where the workload is equal for both platforms, Proxy A shows the fastest request processing time with the least total time of 312.29ms in comparison to Proxy B which gives total time of 356.98ms. As suspected however, the mobile request time has increased (more than double the other setups) and this cause the request time ( $Req_t$ ) to go up to 98ms. Furthermore, what is surprising even though could be expected is the fact that the task assignment time is almost the same as the case where the controller is hosted in North Carolina. This phenomenon arises because the request is just reversed between the two components and the distance between the two systems will not play any role. What can influence the result will be the processing capacity threshold of each server hosting the controller. In this experiment however, the two servers (i.e., the North Carolina controller hosting sever and the Sydney controller hosting server) have similar capacity. The request time in comparison to the first setup has increased since Sydney is by far a distance from Saskatoon. The time for the controller to reach the closest proxy (i.e., Proxy B in Sydney) is approximately 7ms while Proxy A in North Carolina – NC is reachable at approximate time of 34ms. Furthermore, in Scenario 1, Proxy A is able to send the response

to the mobile at approximate time of 55ms while Proxy B takes approximate 125ms to communicate the response to the mobile. Based on the end result, the CSB-UCC will send the incoming jobs to Proxy A in this case because of the less time required to process a request.

However, looking at the results closely after several repetitions of the experiment (30 times), it can be seen that there is a relative increase in latency compared to hosting the controller closer to the requester. This is because the request time has increased drastically. What is noteworthy however is that, the increased request time affected both setups. In other words, regardless of whether Proxy A is closer to the user or Proxy B, increased request time cause transactions in both setups to increase. So, just as in the case of hosting the controller in North Carolina, the factors that play a major role in increasing the latency are the task assignment time, processing time, and the response time. In the case of Scenario 1, the major reason for choosing North Carolina to send the request is the response time. This can also validate the argument of those proponents who prefer to send requests to edge nodes.

A second experiment is provided called *Scenario 2* where the workload of Proxy B is in Asia Pacific (Sydney) is doubled. By so doing, the workload of Proxy B is increased to 2000 while the user base for Proxy A remains at 1000. As Scenario 1, all the determinants are the same except for the fact that the processing time (i.e.,  $Proc_t$ ) of Proxy B has increased. Also, a marginal increase in the response time (i.e.,  $Resp_t$ ) is observed since the size of data affects the data transfer rate across the bandwidth. But, clearly, the reason for the increase in the total time is because of the significant increase in the processing time of Proxy B which has its workload doubled. If the experiment continues with increasing workload of Proxy B, all latency-sensitive scenarios will favour Proxy A. This is because the  $Proc_t$  component on Proxy B will keep increasing.

The third set up, which is *Scenario 3*, is the case where the workload is reversed and Proxy A in North Carolina has its workload doubled to 2000 users and Proxy B has 1000 users. Here, it is observed that the  $Proc_t$  of Proxy A has increased. Considering that all factors are same from the previous two setups, the increment in latency can be attributed to the workload demand.

In *Scenario 4*, the workload of Proxy A is quadrupled, and this means it has to serve the maximum capacity of users. Proxy B on the other hand serves 1000 users and now the leap in the total time can be seen as a result of the increased  $Proc_t$ . The system at maximum capacity sends "busy" state notice so requests will not be served at such saturation level.

Looking at the results in this experimental setup (where the controller is hosted in Sydney) gives a lot of insight into how the proposed best-proximity use case is relevant. From all the three setups that focus on where the controller should be hosted, it turns out that the location of the controller does not give any special advantage to any of the proxies. In fact, inter-cloud communications are stable to ensure faster information exchanges across system components however, the processing cost and the response time are crucial.

## Discussions and Further Application of the Proximity-Based Technique

From the four scenarios in the three major setups regarding the location of the controller, it can be seen that the issuance of mobile requests to the cloud hosted application server nodes within the closest proximity does not guarantee request–response time optimality. Rather, the proposed best–proximity methodology which is proposed in this dissertation is an efficient way to reduce latency and further minimize the transaction time. The designed distributed architecture of the CSB–UCC therefore follows the best–proximity approach to serve the mobile participants in order to achieve time optimality.

In real world scenarios, there will be instances where requests from certain geographical regions will be more than other regions in competition for services and products accessibility. If the closest proximity between the mobile users and the application server is the de facto logic, the heavy workload can cause significant delays that can defeat the purpose. Thus, this work has made headway by proposing a better approach where the jobs need to be assigned based on the combination of factors such as the request processing time, the request assignment time, and the proximity between the mobile requester and the application server.

Another lesson that is learnt from this section is the fact that translating distributed back end architectures to mobile systems may fail. As seen in the experimental results, regardless of location, inter–cloud communication exchanges are fast but location plays part in mobile communications. For instance, while it took almost the same time for communication exchanges between the server in Saskatoon–to–North Carolina and Saskatoon–to–Sydney, the mobile request time in both cases is doubled.

Now that the best–proximity use case is explained with initial experimental results, the next section will discuss how the approach is adopted to ensure sensor–mobile communication.

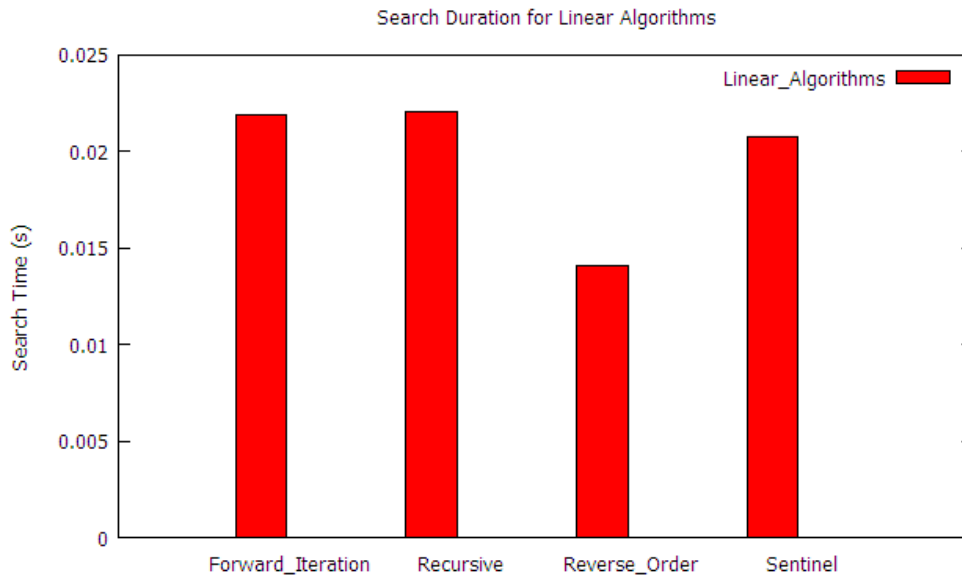
### 4.3 Testing the Sensor–Mobile Communication

The agility of the CSB–UCC is further proven by its adoption to enable mobile–sensor communication. Section 3.9 detailed the design of the mobile–sensor ecosystem. This work is part of a research collaboration with the Environmental Instruments Canada Inc. who are manufactures of sensor devices. The implementation is deployed on a mobile device of the following specifications: iPod Touch – OS: Apple iOS 7.0.4, Resolution: 1136–by–640–pixel resolution at 326 pixels per inch, Processor: ARM Cortex–A9 Apple A5 dual–core 1 GHz, Storage: 32GB, RAM: 512MB. The communication between the sensor devices and the smartphones is Bluetooth. The communication between the smartphones is via Wi–Fi or Bluetooth. In the experiment, we restrict the communication between the mobile hosts (i.e., the smartphones) via Bluetooth for consistency with communications with the sensors. In this experiment, the sensor under consideration is the CT007 and the CC2541 SensorTag.

### 4.3.1 Linear Search Algorithms

This search explores the most optimal linear algorithm for searching through the array list of discoverable mobile hosts. In the field of graph theory and computing, linear search can follow the following designs: forward iteration, reverse order, recursive and sentinel. In order to evaluate the various communication flow patterns for better latency optimality, we first aim to determine the most efficient linear algorithm for the work.

The experiment here evaluates the time cost of searching through the array list of devices on the mobile before making the request to the desired mobile host on the edge. In the experimental setup, we consider fifteen (15) smartphones (actual devices and simulators) and four (4) T1 Sensor Tag devices that can all be discoverable via Bluetooth. This brings the total number of devices to nineteen. The primary goal in this work is to determine the best linear search approach that can further boost the performance of the sequential flow when measured against the other flow patterns. The result for the various linear operations is graphed in Fig. 4.12.



**Figure 4.12:** Search Duration of the Linear Algorithms

In the linear experiment, the index location of a stored item is crucial. In most cases, if the information of the desired mobile host is located in the first index, then the search will exit and the time will be equal for all algorithms. However, the desired terms can also be found in the last index which is the worst case scenario for the forward iteration. Hence, in the test, we put forward three setups. The first involves finding the desired mobile host before the middle index, the second is finding the host at the middle index, and the third is finding the host after the middle index. The array contains list of 19 devices only so this is a manageable situation. The results presented are the overall performance of each algorithm after repeating the experiment for each set-up twenty times.



Starting with the reliability of the linear algorithms, it is observed that the correctness of each of the algorithm is 100% in searching through the 19 discoverable devices. The number of devices is small but in most practical use cases, not too many devices are required to make a Bluetooth connection. It is possible however that if the number of devices increases beyond 19, some errors can occur. Currently, the error is 0% for each of the algorithm. This means, the option on which linear algorithm to employ in the search for mobile hosts in the device list should not be based only on the reliability (in terms of correctness and error) since all the algorithms prove to be reliable.

Since the objective of the work is on latency reduction in the edge-based data sharing sensor-mobile host environment, our observations in the experiment (plotted in Fig. 4.12) are more interesting. After going through the proposed setups, we realized that the reverse order algorithm does better in terms of the search time through the list. The graph shown is the collective average of all the experimental setups. The average time required to go through the list are recorded as follows:

- **Forward Iteration:** 0.0219 seconds
- **Recursive:** 0.0221 seconds
- **Reverse Order:** 0.0141 seconds
- **Sentinel:** 0.0208 seconds

From the result, the reverse order takes approximately 50% less of the time required in the forward iteration and the recursive. Hence, we decided to employ the methodology as the bases for the sequential search in the subsequent experiments.

### 4.3.2 Latency Analysis with the Edge-Based Architecture

In this experimental setup, we seek to evaluate the latency reduction of the proposed flow patterns. In Section 3.9, the following flow patterns are highlighted: sequential, parallel, loop, and choice. The purpose of this experimental setup is to test the validity of the arguments raised throughout the dissertation on the need to focus on the minimization of the total time  $T$  rather than just the distance (i.e., proximity). The flow patterns that we test are the sequential flow (that uses the reverse order search through its list), the parallelism flow, and choice.

Choice in this case represents the existing approaches for edge-based connection that relies on the minimal RTT (i.e., the round-trip time) only to determine the edge to connect to. The parallelism flow issues the request to every discoverable mobile host and sensor the response is prioritized based on the minimal  $T$ , where  $T = RTT + PT$ . The same concept applies to the sequential flow except for the fact that the connection to other nodes is linear.

To perform the experiment, we engage users in the same building but different units (rooms) to enable us have different RTT values. This experiment is also ideal to real world use cases since we can consider the

building walls as obstacles to the data flow in a Bluetooth environment. In Table 4.20, we record the various values we received. The experiment is separated into four groups with five (5) trials each. Each trial is repeated at least twenty times to avoid any bias. In the first group, we considered the ideal case where there is no error and the request at first time receives a response. This is the expectation for any cyber-physical system but the reality is different. The second group is where we disconnect 5 devices out of the 19 through fault injection. The 5 devices broadcast their presence but do not serve the requests so the requester has to issue another request to a different node until the desired data is received. The second group is extended to 10 disconnected nodes and 15 disconnected nodes. In Table 4.20, DN stands for Disconnected Nodes.

**Table 4.20:** Sensor-Mobile Latency Test

State	Sequential			Parallelism			Choice		
	PT (s)	RTT (s)	T (s)	PT (s)	RTT (s)	T (s)	PT (s)	RTT (s)	T (s)
No Error	0.1547	0.0970	0.2517	0.1406	0.0980	0.2386	0.4788	0.0940	0.5728
	0.1592	0.0890	0.2482	0.0984	0.0982	0.1966	0.3377	0.0930	0.4307
	0.1566	0.0910	0.2476	0.0944	0.0934	0.1878	0.4123	0.0980	0.5103
	0.1601	0.0930	0.2531	0.0880	0.0940	0.1820	0.3896	0.0910	0.4806
	0.1655	0.0970	0.2625	0.0996	0.0920	0.1916	0.4479	0.0940	0.5419
		Total	0.2526		Total	0.1993		Total	0.5073
5 DN	0.2366	0.1940	0.4306	0.1211	0.0990	0.2201	0.3871	0.1990	0.5861
	0.2451	0.1880	0.4331	0.1223	0.1070	0.2293	0.4766	0.1820	0.6586
	0.2321	0.1950	0.4271	0.1180	0.1088	0.2268	0.5113	0.1920	0.7033
	0.2373	0.1970	0.4343	0.1178	0.1082	0.2260	0.3561	0.1920	0.5481
	0.2366	0.1770	0.4136	0.1217	0.1067	0.2284	0.4012	0.1730	0.5742
		Total	0.4277		Total	0.2261		Total	0.6141
10 DN	0.4714	0.3130	0.7844	0.1133	0.1510	0.2643	0.7343	0.2640	0.9983
	0.4832	0.3070	0.7902	0.1422	0.1490	0.2912	0.6636	0.2891	0.9527
	0.4836	0.3170	0.8006	0.1210	0.1720	0.2930	0.5731	0.3086	0.8817
	0.4691	0.3420	0.8111	0.1332	0.1440	0.2772	0.5783	0.3012	0.8795
	0.4511	0.4010	0.8521	0.1412	0.1850	0.3262	0.6322	0.2956	0.9278
		Total	0.8077		Total	0.2904		Total	0.9280
15 DN	0.9428	0.5301	1.4729	0.1441	0.1711	0.3152	0.9892	0.5041	1.4933
	0.8931	0.4722	1.3653	0.1623	0.2304	0.3927	1.0340	0.5507	1.5847

Continued on next page

**Table 4.20 – continued from previous page**

<b>Sate</b>	<b>Sequential</b>			<b>Parallelism</b>			<b>Choice</b>		
	0.9022	0.5801	1.4823	0.1522	0.2166	0.3688	1.0440	0.4999	1.5439
	0.9062	0.5011	1.4073	0.1452	0.2271	0.3723	1.1000	0.5939	1.6939
	0.8871	0.5322	1.4193	0.1524	0.2730	0.4254	1.0607	0.5123	1.5730
		Total	1.4294		Total	0.3749		Total	1.5778

In the no error scenario, the average total time,  $T$ , for a request–response scenario is 0.2526 seconds for the sequential flow, 0.1993 seconds for the parallelism flow, and 0.5073 seconds for the choice flow. The parallelism flow proves to be the most efficient while the choice flow is the worst case for the three flow patterns. For the sequential flow and the choice flow patterns, the values of the processing time (PT) are significant because it is the sum of the processing time of the requester and that of the mobile host. With the parallelism, the processing time of the requester is the time required to send the concurrent requests to the discoverable devices. The processing time of the connected mobile host remains the same for any algorithm.

However, the PT of the various algorithms varies due to the processing cost of the requester.

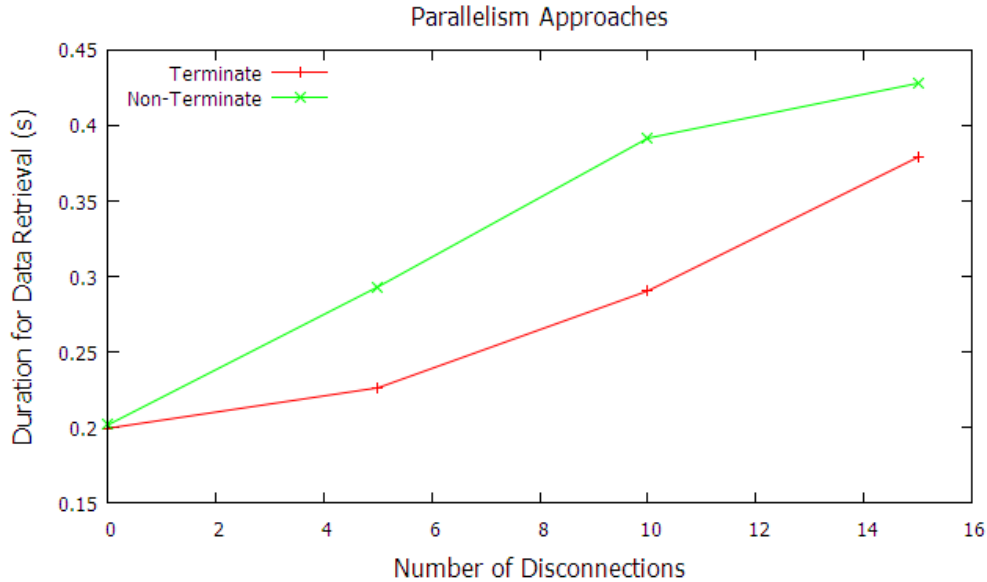
When the results are observed closely, it is realized that the choice flow which emulates the existing approach for edge–based connection demonstrates the worse latency not necessarily because all the factors return higher time intervals. In fact, there are instances, where the RTT (i.e., the request and response travel time) component of the choice flow is better than that of the sequential flow that follows our proposed algorithm. However, the total time,  $T$ , is bigger because the choice flow disregards the need to consider the processing time (PT) of the connected mobile host. Thus, even though some instances show that the RTT of the choice flow pattern is minimal (because it is just focusing on connecting to the closest node on the edge), the time is longer.

The time factor in the parallelism flow pattern is only influenced by the response time and it is only when error occurs that the response order has to change, which can cause an increase in time. Another observation from the result is that, there is a direct proportional relation between the number of disconnected nodes (i.e., representing the number of error) and the request processing time. This observation is seen in the sequential flow and the choice flow scenarios. Thus, as the error on data accessibility from the edges increases, it takes more time to re–route the request to another node until the desired data is retrieved from a mobile host.

However, the parallelism approach exhibits fairly stable time across the categories because there is no extra cost (in terms of time) for processing a request. The result clearly shows that our proposed approach for latency reduction that considers two factors, i.e., RTT and PT, is better than existing approaches that only considers the RTT in edge–based networks. The parallelism approach however shows the most optimal time though the sequential methodology is better than the exiting choice flow pattern.

As also mentioned earlier in the previous section, we studied two design techniques for the parallelism

flow which are terminate and non-terminate techniques. The results discussed so far all follow the terminate technique since it outperforms the non-terminate. In comparison, we present the performances of the two techniques in Fig. 4.13.



**Figure 4.13:** Parallelism Approaches

Considering the nineteen (19) devices for testing, we record the various durations required when there is no error (i.e., zero disconnected devices), 5 disconnections, 10 disconnections, and 15 disconnections. The disconnections are plotted on the x-axis and the duration for receiving the message is on the y-axis. *Terminate* means when the mobile requester receives a desired response based on the priority order (discussed earlier), the connection closes and the requester disregards all the other concurrent requests and the transaction ends. The *Non-Terminate* approach is when the mobile requester sends the request, receives a desired response, and waits for all the other responses before closing the transaction. This also means the transaction only terminates when responses are received for all the concurrent requests. The result (graphed in Fig. 4.13) is simplified in Table 4.21.

**Table 4.21:** Terminate Vrs Non-Terminate Parallelism Flows

<i>State</i>	<i>Terminate (s)</i>	<i>Non-Terminate (s)</i>	<i>Percentage Increase (%)</i>
No Error	0.19932	0.2016	1.14
5 Errors	0.22612	0.2931	29.62
10 Errors	0.29038	0.3916	34.86
15 Errors	0.37888	0.4279	12.94

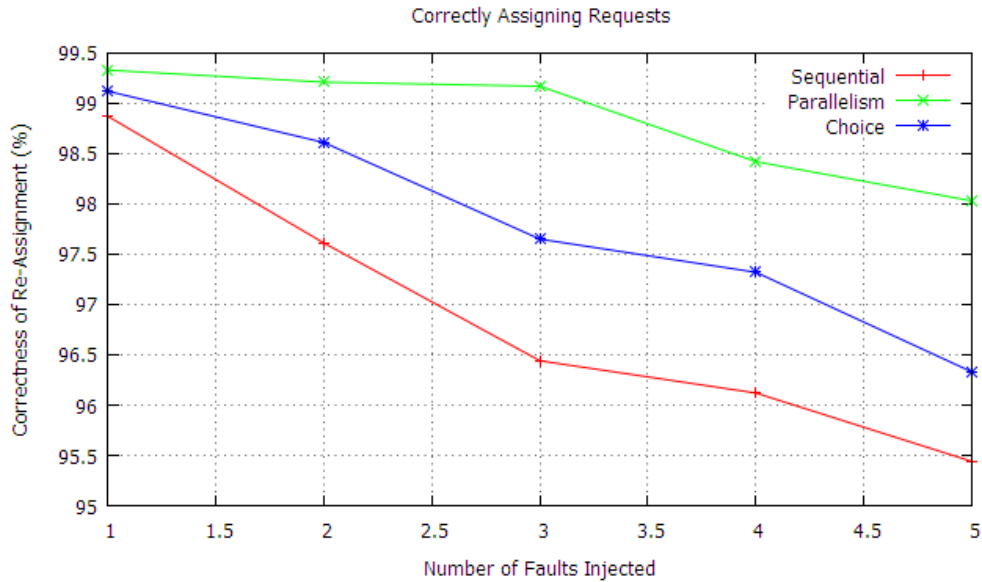
The percentage increase represents the extent to which the non-terminate flow consumes extra time in comparison to the terminate approach.

From the results, we can conclude that the terminate approach saves more time than the non-terminate approach. Thus, it is justifiable to employ the terminate technique as a latency reduction measure in the edge-based mobile hosting environment.

### 4.3.3 Error and Request Re-Assignment

In the last experimental setup, we evaluate the reliability of each flow pattern in terms of accessing data in a group sharing environment when there is failure. Since errors are bound in wireless communication mediums (e.g., Bluetooth), sensor data (e.g., readings from the T1 Sensor Tag) will experience delayed responses or failures. Also, there is no guarantee that communication between the adjacent edges will be seamless since users are mobile within a shared-space.

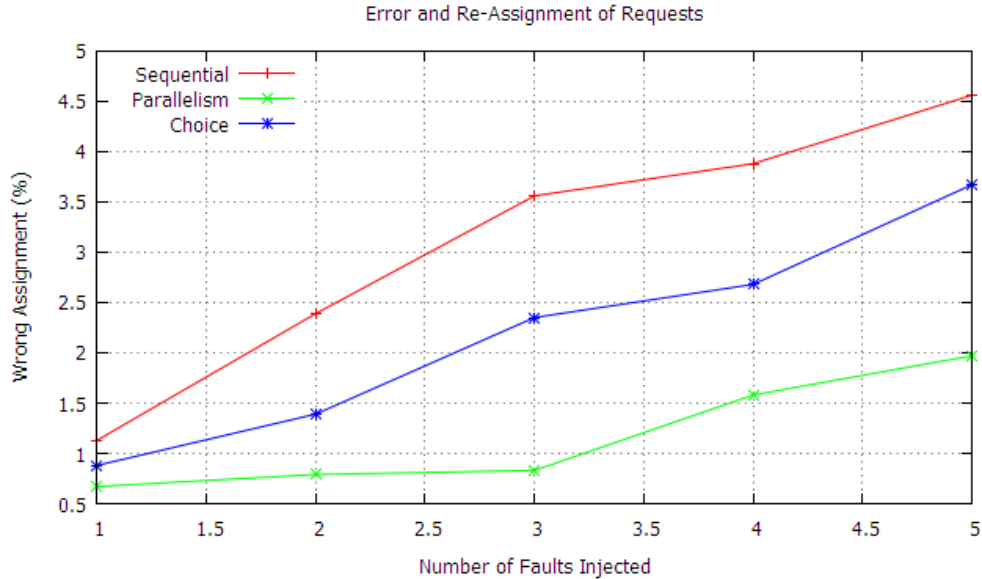
In this experiment, we employed with the fault-injection technique where we evaluate specifically, the accuracy of re-assignment of a transaction to the next desired node. We define transaction within the concept of retrieving sensor data from a mobile host. With fault injection, the idea is to request a data and "kill" the intended host so that another host within the discoverable list of devices can be contacted. We carefully keep terminating the transactions unexpectedly until at least 5 iterations can be made before the desired sensor information is retrieved. The result is presented in Fig. 4.14 and Fig 4.15.



**Figure 4.14:** Correctness of Task Re-Assignment

In Fig. 4.14, we studied the correctness of re-assigning a task to the next desired node following the underlying principle of each flow pattern. We found that the parallelism flow has better accuracy with the task assignment than the other two flow patterns. A reason for this observation is that, the parallelism approach focuses only on the responses since the requests are concurrent. The flow pattern only considers the responses and their order of arrival. Practically, the first response with the desired data is selected so

there is only little room for error. The Choice flow pattern however outperforms the sequential flow in terms of correctly assigning tasks. This happened because the choice flow ignores the processing time of the mobile host (TP) and only focuses on the request-response travel time (RTT). This means, the algorithm focuses on only one factor and the chances of identifying/determining a smaller RTT over another is higher than doing extra calculations.



**Figure 4.15:** Tasks Assignment to Wrongly Identified Mobile Host

The sequential flow sees its correctness value dropping as the fault increases because we are comparing two factors (the RTT and PT) and the number of comparisons are doubled. Another reason is because the PT factor of the host device is not stable and changes can occur between the time a request is issued and a response is received. Future works have to focus deeper on addressing the dynamism of PTs in the entire architecture.

In both Fig. 4.14 and Fig. 4.15, it is observed that the flow patterns that are least in terms of correctly re-assigning a task have higher error rate in terms of wrong task assignment.

#### 4.3.4 Summary

The concept of mobile services hosting is gaining more attention most recently due to the expansion and diversity of macro fields such as Internet of Things (IoT) and Cyber-physical systems. With mobile hosting, services (including data and application states) can be provisioned from mobile devices such as smartphones and tablets; emulating actual servers. This can greatly improve on mobile-to-mobile communication. Moreover, other forms of data (e.g., sensor data) can be collected from sensor devices and sent to the mobile host to be made available to other devices.

The area of mobile hosting has witnessed studies that focus on a single provisioning node. However, today's

cyber-physical systems require group sharing since sensor information can be gathered from heterogeneous sources. The problem in such situations however is the ability to reduce latency through the minimization of communication overhead. Latency reduction is critical because delays in the accessibility of the services in the mobile hosting and sensor system can lead to undesired situations such as information misreading, failed request delivery, wrong ordering of responses, and so on.

This work explores the area of mobile services hosting and the ability to share sensor information especially in a group-sharing scenario. The paper adapted the concept of edge-based services accessibility from distributed cloud computing to create the mobile hosting environment. However, the work advanced on the area by proposing the idea of accessing the services from an adjacent node with the minimal time that considers the total travel time of the request and response (RTT) plus the processing time (PT) of the host. This is the typical scenario that the CSB-UCC is advocating for. Existing works only consider the most optimal travel time of the request between the adjacent nodes on the edge. Furthermore, we studied varied transactional flow patterns that can facilitate better performance regarding the minimization of latency.

The work is evaluated to determine the best approach for achieving low-latency communication and efficient job re-assignment. The preliminary evaluations show that the proposed consideration for the optimal  $RTT + PT$  is better than the existing approaches that evaluate latency solely on the optimal RTT. Also, the results show that the parallelism flow pattern is better than the other two which are the sequential and choice flow patterns. It is important to state that this work in conjunction with the Environmental Instruments Canada Inc. adopted the best-proximity technique for the mobile-sensor-data sharing.

In summary, the contributions in regards to the support of sensor communication of the proposed CSB-UCC includes:

- Proposed mobile hosting architecture for group data sharing. Existing works only focus on the feasibility of the idea while enterprise grade applications on mobile provisioning are few. Existing techniques including the SOPHRA [201] framework do not support sensors.
- Proposed and evaluate different communication flow patterns based on sequential, parallelism, loop, and choice methodologies.
- It is observed that optimal time for a response is not dependent on optimal distance between the adjacent mobile and sensor nodes but factors such as the processing load on the host, and request travel time should be collectively considered.
- Failed communications can be re-routed to the adjacent node that has the next better optimal request-response time.

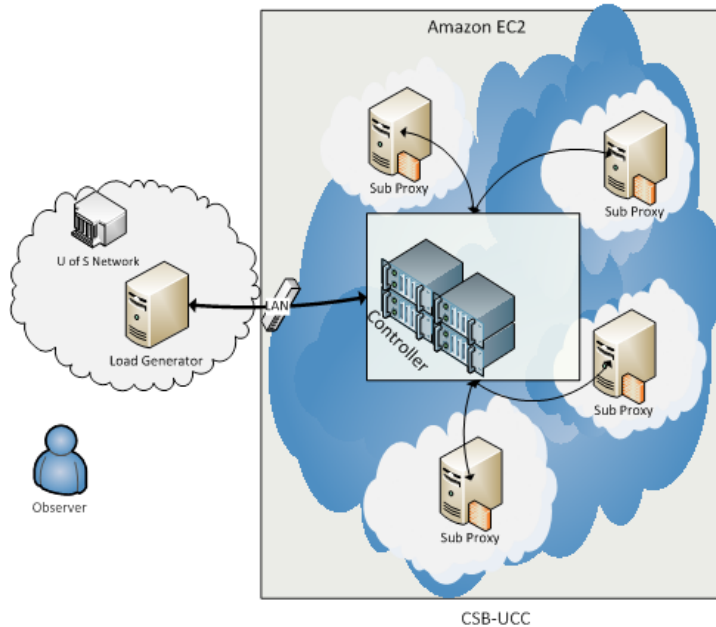
In most of the scenarios, the parallelism flow pattern is better at latency minimization. There is however more room for improvement on the current state of the work. First, there is the need to study more rigorous error management techniques when communications between the mobile host and its consumer

fails. In a mobile hosting environment, errors can be introduced at two levels, communication failure and system failure on the host. Therefore, the two can be studied in detail in another work. Another issue for future consideration is energy conservation on the mobile devices since most applications in mobile hosting environment run constantly either in the foreground or background.

In the next section, the attention of the dissertation will switch to scalability testing of the CSB-UCC. This is another major consideration of the dissertation.

## 4.4 Testing for Scalability

In order to determine the performance of the CSB-UCC during peak loads (i.e., increasing work demand), the scalability test is conducted. In this work, the workload represents the increasing number of mobile users' requests in the system. This experiment requires heavy users and deviates from the previous experiments in terms of the required numbers to determine the actual workload capacity. Thus, a load generating tool, which is built in Erlang and identical to the *Apache Bench software* (<http://httpd.apache.org/docs/2.2/programs/ab.html>), is used as the client to send concurrent HTTP requests to the CSB-UCC for resources at a controlled rate. The load generating tool is installed on the windows 7 enterprise edition server with the specifications in section 4.1 in the laboratory. The set-up for the load capacity testing of the CSB-UCC is illustrated in Figure 4.16.



**Figure 4.16:** Set-up for the Scalability Test

The computer on which the load generator is running connects to the controller component of the CSB-UCC through a Gigabit Ethernet connection. The load generator is configured to simulate the activities of concurrent users of the system in multiples of 30000. The total number of concurrent HTTP requests

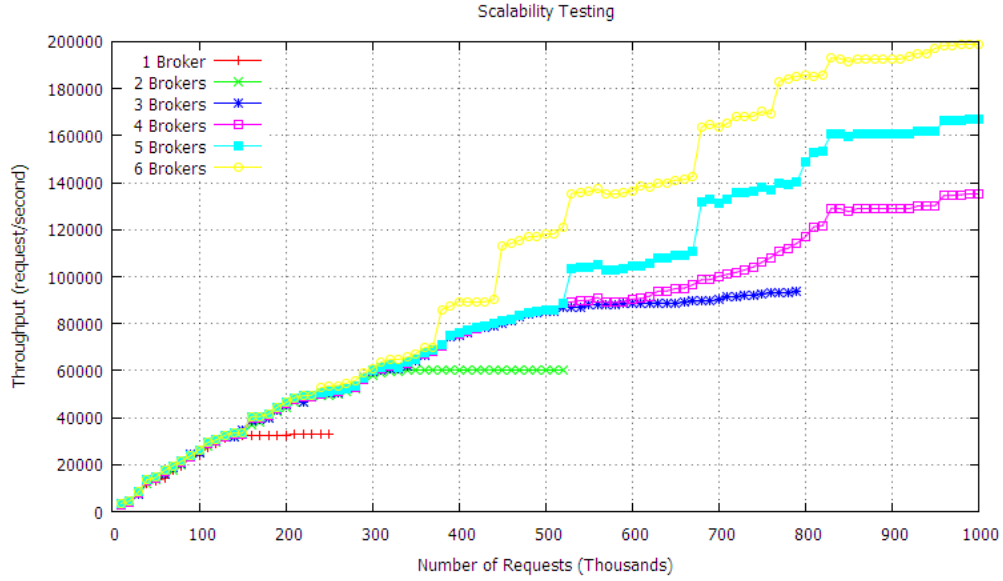


that the users can send ranges from 10000 to 1000000. The load generator is configured to return the mean *throughput* (i.e., the rate at which jobs are completed, measured in request per second) every 10 seconds. In the experiment, the throughput is observed for six different set-ups where the system is centralized (i.e., only 1 broker) and distributed where the brokers are increased from 2 brokers to 6 brokers. Each of the broker (which is the same as sub-proxy) is deployed on an Amazon EC2 instance in North Carolina.

The point here is to get an idea about how the CSB-UCC will behave in different workload scenarios whether centralized or distributed. When the system is distributed, the request handling tasks include the delegation of the load to the various sub-proxies (or brokers) by the controller. The load generator is configured to send concurrent HTTP requests to the CSB-UCC to consume the data that reside on the CouchDB database (as described in section 4.2.3). This experiment is conducted with the data payload from the Clandestine Anomaly Game. It is important to state that the size of the data is not under consideration here because the experiment is not retrieving the actual data from the back-end; rather, only the behavior of the system is being simulated regarding how it cost the CSB-UCC to retrieve the data. Also, the inter-arrival time (which represents the time between successive arrivals) is 10 seconds for 10000 requests. Thus, the arrival rate (which represents the number of jobs arrived per second) is 0.1 per second for every 10000 requests.

The results from the scalability test are graphed in Fig. 4.17 and the summary of values is presented in Table 4.22. The results reported are not single runs, but the averages of the repeated runs. Each experiment is repeated twenty (20) times and the standard deviation between the results within the repeated run is small (2.4 is recorded as the maximum standard deviation value throughout the entire experiment). In the first experiment, I focused on the centralized broker which is shown in the graph as 1 Broker. Starting from 10000 requests, the minimum throughput processed by the centralized broker is 2991 requests/second and the maximum throughput is approximately 32652 requests/second. The average (i.e., the mean) throughput is approximately 24632 requests/second. As shown in the graph, it is observed that the throughput value increases until the system almost stabilizes from 160000 requests. This range shows throughput value of around 32600 until a maximum capacity is reached at 250000 requests. The centralized broker then becomes non-responsive or crashes at this point. This shows that a centralize broker can handle up to about 30000 mobile users if those users are issuing approximately 8.3 requests/second. In the real world use case however, there can be more than this number of users or their request can be more than 8.3 requests/second.

So, the second set-up considers the performance of the system with two sub-proxies (i.e., 2 Brokers). For brevity, the reader is referred to Table 4.22 and Fig 4.17 to check the maximum, minimum, mean capacity, and maximum request capacity of each set-up in order to avoid the repetition of text. With 2 sub-proxies (2 Brokers), it is observed that the throughput value increases until the system almost stabilizes from 410000 requests. This range shows throughput value of about 60300 request/second until a maximum request capacity is reached at 520000 requests. The system then becomes non-responsive or crashes at this point. This performance is approximately double that of the centralized broker assuming we maintain the



**Figure 4.17:** The Scalability Test of the CSB-UCC

**Table 4.22:** The Scalability Testing

	<i>1 Broker</i>	<i>2 Brokers</i>	<i>3 Brokers</i>	<i>4 Brokers</i>	<i>5 Brokers</i>	<i>6 Brokers</i>
Minimum Throughput (request/second)	2991	3022	3122	3002	3504	3507
Maximum Throughput (request/second)	32652	60502	93765	135001	166767	198902
Mean capacity (request/second)	24632	44874	64539	80373	93083	112692
Maximum Request capacity (request/second)	250000	520000	790000	1000000	1000000	1000000

same mobile request rate at 8.3 requests/second.

With 3 Brokers, it is observed that the throughput value increases until the system almost stabilizes from 700000 requests. This range shows throughput value of about 90127 requests/second until a maximum request capacity is reached at 790000 requests. The system then becomes non-responsive or crashes at this point. This performance is approximately triple that of the centralized broker assuming we maintain the same mobile request rate at 8.3 requests/second.

When there are four sub-proxies (i.e., 4 Brokers), it is observed an increase in the throughput until the value stabilizes from 930000 requests. This range shows throughput value of about 130000 requests/second until a maximum request capacity is reached at 1000000 requests. The system then becomes non-responsive or crashes at this point.

In the set-up for 5 and 6 Brokers, the load generator could not issue more than a million requests. So, both experiments are limited to 1000000 requests capacity. I could not conclude this is the maximum number of users that 5 to 6 brokers can serve since the limitation is not from the system but from the experimental set-up. However, even though the maximum value is pegged at 1 million requests, it is observed that the throughput values have improved compared to the initial set-ups. With 5 Brokers, the throughput increases until it stabilizes from 960000 requests. This range shows throughput value of about 166600 requests/second. At the million request mark, the approximate throughput is 166767 requests/second. When the 6 Broker set-up is employed, a million requests shows approximate maximum request rate of 198902 requests/second.

From the entire scalability test, it is observed that for a single broker, approximately 30000 users who own on average 8 mobile devices and can make 1 request/second from each device, to access services can be supported. There is an increase in the number of users by approximately 30000 for each additional sub-proxy that is added to the system. Another important observation from the graph is that, all the six brokers seem to have equal throughput at the beginning of the requests. The reason for this is that, the controller keeps sending the requests to one broker until that broker reaches saturation (where the broker starts sending busy message back to the controller). It is from this point that the controller will start sending the requests to another node. Hence, we realized that if the total request is say 170000, the throughput will be approximately the same for all the 6 brokers since at this number, regardless of the number of active brokers, only one broker will be working.

## 4.5 Fault-tolerance

Finally, I evaluate the performance of the distributed CSB-UCC approach in the event that the sub-proxies are dead and/or non-responsive. This experiment is important to determine the behavior of the system since in reality, some of the sub-proxies will die and will not serve requests. We approach this experiment by adopting the fault injection methodology. In all, there are six sub-proxies and each is designed to handle requests. The sub-proxies are deployed on different Amazon EC2 instances.

Then a constant medical data payload of 100 MB is employed to send messages across the system. The experimental result is shown in Table 4.23. Four separate tests are conducted where the first one involves all six proxies being alive and responsive. The result is similar to the first experiment, and then I intentionally crashed one of the sub-proxies, which put it in a dead state and leaving only 5 alive. It is expected that the time frame will increase but the actual increment is so marginal. I then crashed two sub-proxies and later three sub-proxies are crashed. In each case, there is a marginal increase, which shows the high rate at which the controller re-assigns the tasks of the dead sub-proxy to the other responsive sub-proxies following the best-proximity methodology.

In terms of the overall performance however, when all the sub-proxies are dead leaving one, the performance of the distributed system will be approximately the same as the centralized system. This is true also

**Table 4.23:** Fault Injection Analysis

<i>Number of Requests</i>	<i>Processing Time (ms)</i>			
	<i>All Alive</i>	<i>1 Dead</i>	<i>2 Dead</i>	<i>3 Dead</i>
100	80.40	87.04	94.15	101.56
200	84.20	92.78	102.07	122.33
300	88.04	93.64	112.87	134.88
400	90.23	98.19	128.20	142.74
500	92.32	101.77	133.59	156.73
600	96.34	116.40	144.61	172.01
700	102.33	122.09	166.98	194.09
800	112.87	131.12	173.22	121.55
900	121.33	142.12	181.98	123.14
1000	133.11	149.65	196.55	136.29

for the scalability testing.

## 4.6 Summary

This chapter described the various experiments that were conducted to validate the design choices of the Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC). The experimental set-ups seek to evaluate the performance of the CSB-UCC in line with the research goals on the reduction of the services/data propagation window in an n-device to multi-cloud ecosystem. The first part of the experiment focuses on the minimization of the update propagation window between the mobile devices and the cloud services. Using a Wi-Fi network, the experiments are conducted using the Playbook, iPad3, Transformer prime, and NOKIA Lumia 900. Further, public cloud services providers such as Dropbox, Amazon S3, and MEGA are employed as the services sources. The evaluation of the propagation window of the mobile versions of these products is conducted to determine the base (benchmark) results to which we can compare our own results. The propagation windows of the centralized and distributed architectures of the CSB-UCC are then evaluated.

Also, the proposed best-proximity use case which dictates that distributed loads should not be assigned at all times based on the closeness between the mobile requester and the cloud application node is evaluated. The scalability of the various architectures is also evaluated as well as the fault-tolerance of the CSB-UCC. The summary of the experiments are listed below based on the goals.

### **Goal 1: Soft-real time synchronization**

- *Part 1:* Evaluated the update propagation window of the proposed system. In comparison to the existing mobile versions, it is observed that the centralized architectures can concurrently support 3200 mobile users (who own 4 devices) to propagate files of up to 2.4 GB within reasonable time. Beyond

this number, the distributed architecture has to be adopted which can support up to approximately 20000 users.

- *Part 2:* Evaluated the proposed best-proximity methodology versus the close proximity technique which can be employed to ensure real time services delivery. The evaluation shows that not all cases of assigning tasks to the closest application server minimize latency. Rather, factors such as the request travel time, request assignment time, request processing load, and response time should be considered collectively. The distributed application server node that returns the minimal time when the sum of the time of all the factors are considered should be the most appropriate server to assign the task according to the best-proximity rule

Furthermore, the agility of the CSB-UCC is proven regarding the support for sensor devices. The system supports sensor-mobile data sharing and adopts the proposed best-proximity technique to disseminate information. Test results show that soft real-time data synchronization can be achieved.

**Goal 2: Scalability testing of the brokerage platforms** The CSB-UCC is evaluated based on the centralized architecture and the distributed broker architecture with up to six brokers. It is observed that approximately 30000 users can be supported to issue 8.3 requests/second in the case of the centralized broker. For every additional broker (sub-proxy) which extends the centralized system to a distributed system, the user support base increases by 30000.

**Goal 3: Error tracking and fault recovery** Based on the fault injection technique, an analysis is done on the time within which tasks are re-assigned when some proxies fail. It is observed that the job is re-assigned within a very small window; about 60ms being the highest re-assignment time.

**Goal 4: Audit trail through provenance enforcement** Though this is listed as one of the experiments to be conducted, it is deferred to another work where it will be treated as an independent research.

## CHAPTER 5

# CONCLUSION AND FUTURE WORK

### 5.1 Summary and Contributions

This dissertation presented a cloud brokerage framework called, the *Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC)*, which facilitates n-devices to access multi-cloud services. With the recent advancement in mobile and cloud computing technologies, there are several services that are of interest to consumers at both personal and enterprise levels. These services range from data and file sharing across different cloud platforms, and application sharing. Furthermore, recent product releases such as wearable devices (e.g., Apple Watch) creates the need for ubiquitous access. To facilitate this, several mobile end-points need to be supported to communicate with the multiple cloud services and resources. The era of supporting n-devices to access multi-cloud services is described as the Personal Cloud Computing or the Ubiquitous Cloud Computing (UCC).

As of now, the UCC is in its infancy with not many works seen on its architectural deployment apart from some speculations that brokerage platforms can be adopted to design the architecture. Moreover, the option of supporting the consumerization of multiple cloud services by mobile users especially in the enterprise creates the need to provide services audit trail. Also, open questions to address are how to ensure consumer consistent experience and how to aggregate the n-devices with the multi-cloud sources.

To address the above issues, the proposed Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC) is deployed as a cloud-hosted application brokerage service that integrates the n-devices with the multi-cloud services. The mobile can establish connection to the brokerage platform through HTTP. There are three authentication methodologies that are designed based on 3rd party OAuth 2.0, username/password pairs, and hybrid authentication (i.e., the combination of the other two) to access the cloud services. Further, the broker relies on channels to determine the users, their registered devices, and their subscribed cloud services to propagate updates. The n-devices supported in this work includes smartphones, tablets, notebooks, and sensors.

The CSB-UCC is a distributed broker architecture but can be employed as a centralized broker depending on the workload and user-base. The dissertation studied existing approaches on efficient ways of extending cloud services to mobile devices and the dominant techniques include mobile edge computing and fog computing. With these techniques, existing services providers seek to address the issue of latency in data

propagation through the assignment of tasks to the application broker node within the closest geographical proximity. However, the CSB-UCC deviates from this provision. Rather, the best-proximity use case is put forward which relies on the combination of factors such as request time from the mobile to the broker, the processing time of the broker, and the response time. Subsequently, the evaluations show that the best proximity methodology is better. Also, the CSB-UCC puts forward a provenance mechanism that relies on the combination of mobile context awareness and the role of the user to ensure proactive audit trail. Thus, the proposed provenance considers the location of the user, the time of the request, and the user action before a request is granted. Since several domains have their own requirements, the policy can change.

The evaluations show that the centralized CSB-UCC can support approximately 3200 users to propagate their data from public cloud services such as Dropbox, Amazon S3, and MEGA in a group file sharing scenarios to  $n$ -devices (where  $n=4$  and the file size is up to 2.4 GB). This support base guarantees the same consistent user experience as the available services provided by the cloud owners. The system with six distributed broker nodes can support up to 20000 users with similar user experience without intolerable delay. Also, the experiments on the scalability shows that a single broker node of the CSB-UCC can support up to a maximum of approximately 32300 requests/second. In cases that a distributed node fails, the CSB-UCC takes a relatively short time to re-assign the task to another node.

The contributions and findings of the dissertation are summarized below.

- i An architecture of the mobile ubiquitous cloud computing, called CSB-UCC, is researched and developed which facilitates  $n$ -devices of a single user to access multi-cloud services in groupware and personalized situations.
- ii The CSB-UCC can be adopted as a *centralized* brokerage service or *distributed* brokerage service depending on the size of the users and the system peak load. This further enables agility of the system for different enterprise transactional needs.
- iii When adopted as a distributed architecture, the *best-proximity* use case is proposed that ensures that transactions are assigned to the broker node that can provide latency optimization when factors such as proximity between the mobile and the broker, and the processing time of the broker are considered collectively.
- iv In the distributed architecture, *error and failure* scenarios are accounted for by re-assigning the tasks of failed nodes to other active nodes. In the centralized architecture however, failure is a huge risk but backup of the broker state can be kept.
- v The proposed system supports sensor data sharing which is key in today's ubiquitous system architectures. In this regard, sensors can connect to mobile devices and share information that may be required by users.

- vi To ensure system audit trail, provenance is proposed based on mobile context awareness (i.e., time and location) and user's role.
- vii Several award winning applications and services have adopted the CSB-UCC. Examples include the Hemophilia Injury Recognition Tool (HIRT?), and the Clandestine Anomaly Augmented Reality Game.

### 5.1.1 Designed the Cloud Services Brokerage for Mobile Ubiquitous Computing (CSB-UCC)

There is rapid adoption of cloud computing recently by enterprises to deliver product and services to clients and customers. At the same time, mobile computing technologies are being improved. Thus, some enterprises have discovered the need to combine these two macro fields (i.e., cloud computing and mobile computing) to deliver next generation applications. This is known as Mobile Cloud Computing (MCC). Cloud services providers especially from the commodity cloud suppliers are guaranteeing high services availability for anytime access. The importance of cloud computing for companies and consumers, as detailed in Chapter 2, is enormous. Some of the advantages of cloud computing include: cost management as companies can cut down on their internal IT budget, improved maintenance culture as the task of infrastructure manageability is delegated to the cloud service provider, and soft-real services accessibility.

Some services provisioning models are: Software-as-a-Service (SaaS) - services as applications, Platform-as-a-Service (PaaS) - services for development, testing, interfaces, etc. and Infrastructure-as-a-Service (IaaS) - services that support virtualization such as network, servers, etc.

However, cloud computing alone cannot provide the needed requirements for services delivery to consumers since providers need to ensure anywhere access. This is where the mobile cloud computing comes in handy. With the MCC, the anytime access capability of the cloud can be complemented by the anywhere access feature of mobile computing. But what is more crucial is the recent attitude of services consumers. Most mobile users own multiple devices such as smartphones, tablets, body sensors, and smartwatches. The consumers then expect to have application, services, and data consistency across these devices. As well, users have several cloud services subscriptions and there is the need to facilitate these divergent services to be consumed in a single dimensional workflow. Supporting n-devices to access multi-cloud services is termed Ubiquitous cloud computing (UCC) or the personal cloud.

Currently, there is not enough research on the personal cloud. There has been some studies on Mobile Back-end-as-a-Service (MBaaS) which seek to ensure efficient server-side design for mobile services support. While MBaaS is not directly linked to the personal cloud, the initial designs seek to enable third party services integration into mobile specific applications. This includes the design of back-end layers that can facilitate user authentication from third parties and so on. This again leads to the open gap between research on the personal cloud and the existing up and coming solutions.

Based on the research goal in this dissertation, the designed CSB-UCC offers services convergence solution.



This means when adopted, users can be facilitated to access cross-cloud platform services on their n-devices. In Table 5.1, the key features that distinguish the CSB-UCC from the other existing solutions are highlighted.

**Table 5.1:** Features of the CSB-UCC

Feature	Explanation
<i>Authentication</i>	The CSB-UCC supports hybrid authentication mechanism (i.e. either by social networking or proprietary personal login) based on OAuth 2.0 and password/username usage. This is achieved by proposing a graph technique for mapping users credentials across different cloud services. In this case, two account types are maintained on the brokerage service that checks registered accounts and authorized accounts. Registered accounts are the credentials that are created with the service providers and authorized accounts may be other credentials own by the user which are stored on the broker. The idea of providing the hybrid authentication technique is to help users better manage their credentials.
<i>Aggregation</i>	As of now, the existing frameworks that have been proposed either integrate multiple devices to a single cloud provider (e.g. Dropbox) or aggregate multiple cloud sources to a single user. The CSB-UCC is not aggregating the data on the broker as seen in some frameworks such as RackSpace; rather, the aim is to ensure services selection on the broker and delivering the integrated multi-cloud services directly to the n-devices of the consumer.
<i>Audit</i>	The CSB-UCC is designed to employ provenance techniques to ensure transparent audit trail. Existing services at best allows users to know who made changes to information but are not able to prevent actions that may not be required by other users; especially regarding group sharing scenarios. The proposed provenance technique in the CSB-UCC enables users to perform actions based on contextual data and access level. Context is defined in this work as time and location.
<i>Ubiquitous Device Support</i>	The CSB-UCC is designed to support n-devices such as sensors and smartphones. Most of the services mentioned earlier support only smartphones. However, supporting ubiquitous computing will require consideration for sensors. This is what puts the proposed framework in a pole position for future adoption.
Continued on next page	

**Table 5.1 – continued from previous page**

Features	Explanation
<i>Agility</i>	One of the major issues with enterprises is the ability to accommodate infrastructural changes. Besides, though an enterprise will be offering a single service now (e.g., IaaS), there are chances that the service offering can be expanded in the future. Unlike the previously highlighted services in Table 2.3, the CSB-UCC supports integration with IaaS, SaaS, and PaaS. The Section 3 explains how this is achieved with DropBox, MEGA, Amazon S3, Maqetta, and so on.
<i>Cost</i>	The CSB-UCC can be adopted as a centralized broker or a distributed broker. This is good for managing both small scale and large scale services. Since test results confirm the capacity of each architecture, enterprises can adopt say the centralized architecture to support about 3000 users who may own up to 4 devices with soft real-time need. This is also a good way to manage the cost of buying more virtual servers that may be required for a large scale distributed service.

The proposed CSB-UCC is in use in several real world systems. Since the system promises high scalability, it is adopted by ZenFri Inc. Canada in the deployment of the Clandestine Anomaly augmented reality game. This project is described by the New Media Manitoba as the biggest game property in the history of Manitoba. The project is part of MITACS partnership.

### 5.1.2 Sensor Data Sharing

Today, users are seen with several ubiquitous devices such as smartphones, tablets, smartwatches, body cameras, and sensors. Some sensors can facilitate the detection of bio-hazards that otherwise cannot be detected by other personalized devices. For instance, gamma rays are electromagnetic radiation with a very high frequency that can be biologically hazardous. Most workers in the mining, manufacturing, security, and other industries find themselves in such hazardous environments and governments are trying to contain this issue. While traditionally, high gamma radiation detection sensors have been manufactured to be carried by users, they are not good access point for actual dosage readings. With the recent advancement in mobile technology, the dissertation proposes a mobile hosting architecture to enable mobile-to-sensor communication following the edge-based technique. This means the sensor can detect the radiation and send readings to a smartphone device of the user. All other near-by mobile devices (which are authorized) will receive the notification to alert the people in the hazard zone. A crucial obstacle to overcome is latency reduction and efficient request routing in the mobile-to-sensor eco-system.

In this regard, the CSB-UCC is adopted by the Environmental Instruments Canada Inc. to facilitate the dissemination of sensor data sharing in the n-devices economy. The high dosage readings by the sensor can be transmitted to the nearby mobile so that notifications can be fired up or the data can be sent to the cloud back-end. The proposed work is tested and the results show that detected radiations are sent in soft real-time to the mobile devices.

Furthermore, we can advance on the mobile provisioning architecture by developing a mobile hosting and sensor ecosystem for high radiation detection (e.g., gamma rays). In recent time where workers in different enterprises are exposed to hazardous conditions, the risk can be minimized through sensor technologies. While sensors have limitations on data reporting and proactive action taking, we can combine them with mobile systems to ensure efficient message delivery.

In this dissertation, the sensor can detect high radiation and when the data is sent to the smartphone, the latter can vibrate, push notification, send email or SMS, and make quicker dosage counts which can aid the user to move away from a hazardous zone. The proposed system takes into account the ability to disseminate information to all near-by mobile devices (if there are several users in an enclosed high radiation area). A bigger challenge is how to reduce communication latency and ensure faster information dissemination between the sensor and the mobile. This is the reason the edge-based connection is proposed in an attempt to determine the optimal path between the adjacent mobile hosts. To ensure mobile-to-mobile/sensor service communication, this work therefore proposes an edge-based connection in an attempt to determine the optimal path between the adjacent mobile hosts. Different modes of mobile-to-mobile service communication are designed by adapting the services flow patterns that include sequential, parallelism, loop, and choice approaches. The work is evaluated to determine the best approach for achieving low-latency communication, efficient job re-assignment, and error management when communications between the mobile host and the consumer fails.

Since currently mobile devices are not equipped to determine gamma radiations, we have to rely on specific sensors that can determine the radiation (e.g., CT007 ) in this work. Preliminary evaluations focus on the determination of the best approach for achieving low-latency communication, efficient job re-assignment, and error management when communications between the mobile and the sensor fails.

The work is evaluated to determine the best approach for achieving low-latency communication and efficient job re-assignment within the sensor-mobile ecosystem. The preliminary evaluations show that the proposed consideration for the optimal  $RTT + PT$  is better than the existing approaches that evaluate latency solely on the optimal RTT (where RTT is the request response time and PT is the processing time of a mobile). Also, the results show that the parallelism flow pattern is better than the other two which are the sequential and choice flow patterns. In summary, the dissertation makes the following contributions regarding the use of the CSB-UCC to support sensor data management:

- Proposed mobile hosting architecture for group data sharing.
- Proposed different communication flow patterns based on sequential, parallelism, loop, and choice.

- It is observed that optimal time for a response is not dependent on optimal distance between the adjacent mobile nodes but factors such as the processing load on the host, and request travel time.
- Failed communications can be re-routed to the adjacent node that has the next better optimal request-response time.
- In most of the scenarios, the parallelism flow pattern is better at latency minimization.

In presenting the work on the sensor technology, I will like to acknowledge the efforts of the following individuals who helped with programming, experimentation, data collection, and manufacturing: Sihm Pham (Graduate student at the Department of Computer Science), Kai Kaletsch (Environmental Instruments Canada Inc.), and Prof. Ralph Deters (Department of Computer Science, University of Saskatchewan)

### 5.1.3 Provenance

One of the key requirements in data consumption in mobile ubiquitous computing systems is to maintain audit trail. This is to determine who does what. This question is also important in this dissertation especially when designing a framework such the the CSB-UCC that supports n-devices in a multi-cloud computing environments. One of the enterprises that adopted the CSB-UCC through a research collaboration is the health sector; specifically the Saskatoon Health Region.

In the medical sector, the use of mobile devices promises new opportunities for healthcare delivery. There is increasing number of patient-specific apps that aid the user to perform self-assessments, check eating habits, track physical activities, and so on. On the other hand, the beneficiaries of the mHealth paradigm are physicians and healthcare practitioners. They are able to access the Electronic Health Record (EHR) from their mobile devices and carry out diagnoses when required. This leads to advantages such as remote healthcare delivery, location-independent accessibility of medical data, and fostering strong relationship between patients and physicians. Moreover, the consumer attitude today dictates that users own multiple devices such as smartphones, tablets, notebooks, etc. Furthermore, consumers want to have services and data accessibility across their devices. For instance, at the basic level, consumers want to access their emails across varying devices on the go as well as have services in synchronization across the devices. This requires the deployment of a new paradigm of applications, which is often referred to as N-Screen applications or the ubiquitous cloud computing.

The problem that arises now is the enforcement of security in such n-screen enabled mHealth environment. The fact that the physicians can have multiple mobile devices that house medical records calls for the facilitation of data protection policies. If this is not enforced, there is the risk of losing information to unintended persons. This can lead to breach in privacy, medical data pollution, and attacks on the medical system.

In this work, the methodology of provenance is explored to ensure that data access control in an mHealth environment is provided. Provenance is a methodology that maintains the life-cycle history of processes and

data. The methodology is investigated and a variation of it that blends with policy-based access control in a mobile distributed environment is proposed.

A secure proxy layer that tracks the activities of the physicians and enforces some security policies is proposed. While provenance aids the tracking of the actions and activities of the physicians, the policy aids the enforcement of trust. The policy is enforced based on the combination of factors such as *time, location, and the action that the user wants*. For instance, physicians are required to provide further information if it is detected that they are in a drinking bar at 2:00 am and want to change a patient record on visits.

From the extensive work on provenance, work is yet to be commenced on the adaptation of the idea to solve security issues on N-Screen application support. Primarily, this dissertation believes that any action taken by the physicians should be logged; and this is actually how most Health Information Systems (HIS) operate. The focus therefore is on how secure accessibility of the medical data can be enforced in an N-Screen system, as well as the reduction of the risk of a request being made to the HIS by unwanted users due to an error or misplacement of the mobile device by a physician. Thus, the work explored several questions on how to detect unusual requests, how an attack should be treated, and how to build a reactive n-screen service.

The dissertation proposes three access control factors. There has been successful deployment of context-aware systems that defines system accessibility within a context in previous works. Context can be time-based, location-based, or role-based. Thus, the dissertation proposes a policy that the CSB-UCC runs that is based on factors such as time of information access, location where the request is being made, and the action to be taken by the requester.

The combination of these factors prove to enforce the deployment of a reactive system where if any of the factors are not satisfied can lead to denial of service. In other words, certain requests will be denied based on the time the request is being issued even if the location and action are not suspicious. This applies to any of the factors that fails as well.

The proposed provenance technique can be an extension to the SOPHRA [201] project which seeks to enforce mobile hosting in the medical sector. On-going works on the Haemophilia Injury Recognition Tool can also witness the adoption of the proposed provenance technique.

#### **5.1.4 Best-Proximity Services Delivery**

Most enterprises are eager to bring their digital contents closer to the consumers. This has led to the research on Edge-Computing and Fog Computing. Both approaches seek to extend services to consumers based on their geographical location. The best-proximity is not the same as the closest proximity in this work. Since there are multiple layers in the CSB-UCC, a component is designed called, the controller, which determines the node that can serve a request in the shortest time.

The decision to route a request is based on location, processing time, and request round travel time. The evaluations prove that the proposed approach can be adopted as an extension to Fog and Edge Computing since the results are better at request optimality.

In evaluating the methodology, it is seen that the issuance of mobile requests to the cloud hosted application server nodes within the closest proximity does not guarantee request-response time optimality. Rather, the proposed best proximity methodology in this dissertation is an efficient way to reduce latency and further minimize the transaction time. The designed distributed architecture of the CSB-UCC therefore follows the best-proximity approach to serve the mobile participants in order to achieve time optimality.

If the closest proximity between the mobile users and the application server is the de facto logic, the heavy workload can cause delays that can defeat the purpose. Thus, this dissertation has made headway by proposing a better approach where the jobs need to be assigned based on the combination of factors such as the request processing time, the request assignment time, and the proximity between the mobile requester and the application server.

While the approach proposed is not entirely new in the field of distributed systems, it is a headway in the mobile ubiquitous computing field. One lesson from the experiment is the fact that distributed systems must be mobile-focused to guarantee faster services delivery rather than just taking existing legacy service for wired networks and deploying them for mobile services.

### 5.1.5 Error Tracking and Request Re-assignment

Since the mobile environment can experience errors (e.g., termination of connection), there is the need to investigate the best approaches that can aid the mobile architecture to be resilient. For instance, how does the system recover from connection error? And how does the system determine faults and react accordingly to the error?

The distributed version of the CSB-UCC is built to handle several nodes (called sub-proxies). States are defined for each sub-proxy which ensures that the requests which are sent by the mobile participants (i.e., the users) are processed. When the requests arrive, the controller determines which sub-proxies are in a state that can process a request. There are five states for every sub-proxy which are:

*Dead State*– This means the sub-proxy (or, the application server) is dead and is no longer reachable. This can be due to fatal errors, failures, or crashes. Sub-proxies in this state are not issued requests by the controller. *Alive State*– This means that the sub-proxy is reachable. *Responsive State*: This means the sub-proxy is responding to the communication of the controller. There are cases where the sub-proxies are alive but non-responsive. This can be due to intensive transaction processing. *Busy State*– This is when a sub-proxy notifies the controller and other sub-proxies that it does not want to receive any further requests. *Available State*– This means the sub-proxy is ready to receive new tasks from the broker-layer.

Experimental results show that when a sub-proxy becomes unavailable, the requests of that proxy is sent to another active sub-proxy in soft real-time.

### 5.1.6 Best Paper Awards

Parts of the results in this dissertation have been published and the works have received best paper award recognitions as follows.

- *Best Paper Award* presented to Richard Lomotey and Ralph Deters for their work on “Efficient Mobile Services Consumption in mHealth” at the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), and the International Symposium on Network Enabled Health Informatics, Biomedicine and Bioinformatics (HI-BI-BI 2013), Niagara Falls, Canada, August 2013.
- *Best Paper Award* presented to Richard Lomotey and Ralph Deters for their work on “Topics and Terms Mining in Unstructured Data Stores” at the 2nd IEEE International Conference on Big Data Science and Engineering (BDSE 2013), Sydney, Australia, December 2013.
- *Best Paper Runner-Up Award* presented to Richard Lomotey, Yiding Chai, Shomoyita Jamal, and Ralph Deters for their work on “MobiCrop: Supporting Crop Farmers with a Cloud-Enabled Mobile App” at the 2013 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013), Kauai, Hawaii, USA, December, 2013

### 5.1.7 Some Real-World Applications Built as Part of this Dissertation

- *Hemophilia Injury Recognition Tool (HIRT?)*
- *Clandestine Anomaly Game Project*
- *SSCA Spray Quality*
- *SSCA Tank Mix*

## 5.2 Future Work

### 5.2.1 Cybersecurity and Mobile Data Assurance

This work is just the beginning and the community has a lot more to do. There is the need to explore security issues beyond the concept of just authentication in ubiquitous cloud computing environments. Security involves the detection of impersonations and violations of data and services usage. As mobile devices such as smartphones, tablets, and notebooks are being used to access enterprise services and applications, there is the need to provide resilient security measures. With the recent launch of smart wearable devices such as Apple watch, users will tend to start accessing cloud hosted data on this devices.

While the use of mobile devices provides convenience, it is also the focus of cyber attacks. Cybersecurity measures are being proposed especially regarding information encryption and secure transfer protocols but a lot more is needed.

In future works, this dissertation can be extended to explore mobile context-based information sanitization as an advancement on current standards. Information sanitization has to do with hiding sensitive information from unintended users. When contextual information is considered, even when the device gets into the wrong hands, sensitive information will not be divulged.

This can further assure the confidentiality, integrity, and availability of information exchanged across mobile–cloud services. Moreover, the use of mobile devices to share personalized information such as individual health records (through wearable devices) requires not just data protection and privacy, but the authenticity of the information. For instance, the Apple Watch can be used to send health data such as heart pulse to a user’s physician directly. If a friend or family is wearing the watch, then how does the medical facility detect that the collected data is not from the registered patient but from another user? One way to avoid this is to treat wearable devices as medical kits by users but it is not going to be an efficient approach. This calls for studies on noise detection in data as well as information assurance.

Also, though the dissertation proposed a provenance technique (based on the combination of mobile context and user’s role), there is the need for evaluations to determine its pros and cons. The provenance methodology proposed in this work fits the medical domain but may not be relevant in other domains. The fact that contextual information (such as time and location) and a user’s role aided in auditing medical systems does not mean the same can be applied directly to say mobile banking. For instance, while it is suspicious for medical data to be updated from a drinking bar at midnight, it cannot be same for a user who is making mobile payment from a drinking bar at midnight. In this case, it is impractical to translate a provenance technique directly from one domain to another.

It is therefore desirable that future works should explore domain specific challenges and how they can be generalized to meet cross-enterprise needs. Though the generalization may not be accomplished at the policy level, technical requirements can be studied to offer such cross–enterprise audit trails.

### 5.2.2 Mobile Data Analytic-as-a-Service (AaaS)

There is high–dimensional data at our disposal (known as, “*Big Data*”) today that requires a new look at storage, processing, and data management. While big data currently does not have a universally accepted definition, some works have explored the area as: big transaction data (i.e., exponential increase and diversity in the volume of transaction data), big interaction data (i.e., increase in open data such as social media and device data), and big data processing (i.e., increasing processing demand on high-dimensional data) [156]. However, other works define big data within the concept of the 5V model [157]. The 5V stands for *Volume* – massive data size, *Variety* – unstructured data, *Velocity* – change from batch data to streaming data, *Value* – cost associated with data, and *Veracity* – data pollution that needs cleansing to determine usefulness of



information.

Looking at the 5V model, it is obvious mobile users have key roles in every aspect of big data. The volume, velocity, and variety are all direct results of increasing mobile usage to access any information on the go. This creates the need to start designing enterprise systems that can enhance big data support.

The CSB-UCC can be enhanced to offer supports for the management, storage, and mining of mobile data. Specifically, the framework can be advanced to become mobile data analytic-as-a-service platform. Achieving this will require further studies on different styles of storages (structured vs. unstructured), unstructured data mining, ubiquitous data management (including sensor data), and data stream processing.

The analytic-as-a-service platform can be viewed from different perspectives. Firstly, it can be seen as a platform for big data mining where the focus will be on knowledge discovery and mining processes. So, as information is being received from mobile users from their n-devices, the platform can be doing real-time mining of the streaming data. Secondly, the analytic-as-a-service platform can be used as a data management layer. Data management can be considered from the storage perspective or transactional perspective. Since streams require high performance computing platforms, the distributed CSB-UCC can be adopted to coordinate which information should be processed as streams and which ones can be stored as batch for further mining purposes.

So, what is the state of the Analytic-as-a-Service (AaaS)?

### **Analytics-as-a-Service (AaaS)**

IBM Research has identified AaaS as an area that can offer business value [249]. This is because AaaS is able to aid in the transformation of unstructured data into business creation ventures. In this regard, IBM pushed for the creation of an AaaS platform that can aid end-users and companies to submit their data (in either structured or unstructured format) for analytic purposes. This platform is meant to reduce the financial burden on companies from maintaining in-house data analysts; the same view is shared by EMC [250], SAS [251], Sun et al. [252], and Deepak et al. [253].

The goal of a research conducted by Lomotey and Deters [157] is to build a data analytics tool specifically for terms mining from unstructured data sources. The focus is on document-style NoSQL storage though the tool can be applied to any other form of storage. In order to achieve this goal, there is the need to also understand the intricacies of unstructured data mining. Unfortunately, unstructured data mining is equally a fairly new area which is yearning for research attention. That work further presented an overview of the area on unstructured data mining and how applicable some concepts can be to our work. On the whole, most of the methodologies are based on Natural Language Processing (NLP) which is inherited from Artificial Intelligence, Linguistics, and Ontological Semantics.

To enhance the data mining process, scientist in both the academia and industry are beginning to explore the best methodologies that can aid in the unstructured data mining process; especially in the context of textual data mining. As a result, we have witnessed some methodologies such as: Information Retrieval algo-

rithms based on templates [254], Association Rules [255], Topic tracking and topic maps [256], Term crawling (terms) [257], Document Clustering [258], Document Summarization [259], and Re-usable Dictionaries [260].

Though existing works on the AaaS are focusing on quality of service (QoS) improvement, it is desirable if the architectures will focus on enabling data access from these personalized and detached devices. In this regard, the proposed cloud services brokerage in this dissertation can be enhanced.

The process of enhancing the CSB-UCC has already begun in earnest to achieve the AaaS status. Initial works on unstructured data mining and the architectural design of the AaaS are published in [157] and [261]. It is important to note at this point that the work by Lomotey and Deters [261] received the *Best Paper* award at the 2nd International Conference on Big Data and Engineering (BDSE 2013) in Sydney, Australia.

However, the adaptation of the architecture to meet mobile-specific challenges is still far.

### 5.2.3 Autonomic Computing Architecture

The designed Cloud Services Brokerage for Ubiquitous Cloud Computing (CSB-UCC) employs job re-assignment techniques to maintain system performance when failure occurs at certain nodes. The evaluation of the approach proves effective especially when considering the window within which jobs are re-assigned. However, the current state of the CSB-UCC does not have any provision for automatic system recovery of such failed nodes. This means the maintenance of failed nodes (in dead state as described in the dissertation) requires human input. In mission critical systems and latency-sensitive architectures, this can lead to undesired situations. It is therefore preferred if the CSB-UCC can have some self-healing capabilities.

One approach that can be adopted in this regard is *Autonomic computing* which can be explained as the features of distributed architectures that address self-healing without user inputs anytime there is system failure. Autonomic computing can therefore become a very key part of building mobile ubiquitous architectures that can support seamless user experience. In most cases, mobile ecosystems experience interrupted services due to issues such as: user mobility in geographically unsupported wireless areas, device constraint features that can prevent heavy workload processing, unstable wireless connectivity characterized by fluctuating bandwidth, and limited battery life.

Autonomic computing can therefore be employed in mobile-cloud architectures to guarantee seamless application and data management in case any of the aforementioned issues arises. This can be explored by opting for methodologies such as offloading of maintenance capabilities to the cloud so that whenever client nodes experience failures, they can be re-activated.

So, why autonomic computing?

Recently, researchers have employed autonomic computing to achieve the following: Dynamic scale up of workload distribution in cloud computing environments [262], Ensuring trustworthiness in distributed cloud computing systems [263], and Collaborative Report Management in distributed systems [264].

The advancement on the CSB-UCC to include autonomic computing capabilities can further drive enterprises to manage infrastructure budgets better with little to spend on administration. But more importantly,

it will guarantee services and product re-alignment whereby new products can be deployed without concerns about failures in services delivery.

## REFERENCES

- [1] Li H., Sedayao J., Hahn-Steichen J., Jimison, E., Spence, C., and Chahal, S. Developing an Enterprise Cloud Computing Strategy. *White Paper, Intel Information Technology*, January 2009.
- [2] Majhi, S.K., and Bera, P. Designing an Adaptive Firewall for Enterprise Cloud. In *Proceedings of the 2014 International Conference on Parallel, Distributed and Grid Computing*, PDGC 2014, pages 202,208, 11-13 Dec. 2014. doi = 10.1109/PDGC.2014.7030742
- [3] Schaffer, H. E. X as a Service, Cloud Computing, and the Need for Good Judgment. *IT Professional*, vol.11, no.5, pages 4-5, Sept.-Oct. 2009. doi = 10.1109/MITP.2009.112
- [4] Sanchez, F.D., Al Zahr, S., Gagnaire, M., Laisne, J.P., Marshall, I.J. CompatibleOne: Bringing Cloud as a Commodity. In *Proceedings of the 2014 IEEE International Conference on Cloud Engineering*, IC2E 2014, pages 397-402, 11-14 March 2014. doi = 10.1109/IC2E.2014.62
- [5] Sadiku, M.N.O., Musa, S.M., and Momoh, O.D. Cloud Computing: Opportunities and Challenges. *IEEE Potentials*, vol.33, no.1, pages 34-36, Feb. 2014. doi = 10.1109/MPOT.2013.2279684
- [6] Net-on-the-go.com. How Does Cloud Computing Effect Mobile Phones?. url = <http://www.net-on-the-go.com/future-changes-for-mobile-devices-including-cell-phones-and-tablets/how-does-cloud-computing-effect-mobile-phones/>
- [7] Hazarika, P.; Baliga, V.; Tolety, S. The Mobile-Cloud Computing (MCC) Roadblocks. In *Proceedings of the 2014 Eleventh International Conference on Wireless and Optical Communications Networks*, WOCN 2014, pages 1-5, 11-13 Sept. 2014. doi = 10.1109/WOCN.2014.6923101
- [8] Gibbs, C. The Rise of Tablets in the Enterprise. *GigaOM Pro*, June 2011.
- [9] Mohiuddin, K., Islam, A., Alam, A., and Ali, A. 24X7X365: Mobile Cloud Access. In *Proceedings of the CUBE International Information Technology Conference*, CUBE '12, pages 544-551, 3-5 Sept. 2012. doi = 10.1145/2381716.2381820
- [10] Ma Y., Lu X., Luo, Y., Liu, X. A Graph-Based Approach to Assisting Creation of Mobile Web Applications. In *Proceedings of the 2014 IEEE International Conference on Web Services*, ICWS 2014, pages 728-729, June 27 2014-July 2 2014. doi = 10.1109/ICWS.2014.120
- [11] Yang, Z.; Niyato, D.; Wang, P. Offloading in Mobile Cloudlet Systems with Intermittent Connectivity. *IEEE Transactions on Mobile Computing*, vol.PP, no.99, pages 1-14, 19 February 2015. doi = 10.1109/TMC.2015.2405539
- [12] Plachkinova, M., Andres, S., Chatterjee, S. A Taxonomy of mHealth Apps Security and Privacy Concerns. In *Proceedings of the 2015 48th Hawaii International Conference on System Sciences*, HICSS 2015, pages 3187-3196, 5-8 Jan. 2015. doi = 10.1109/HICSS.2015.385
- [13] Behar, J.; Roebuck, A.; Shahid, M.; Daly, J.; Hallack, A.; Palmius, N.; Stradling, J.; Clifford, G.D. SleepAp: An Automated Obstructive Sleep Apnoea Screening Application for Smartphones. *IEEE Journal of Biomedical and Health Informatics*, vol.19, no.1, pages 325-331, Jan. 2015. doi = 10.1109/JBHI.2014.2307913

- [14] Gad, S. H. Cloud Computing and Mapreduce for Reliability and Scalability of Ubiquitous Learning Systems. *In Proceedings of the Compilation of the Co-located Workshops on Systems, Programming, Languages and Applications: Software for Humanity, SPLASH '11 Workshops*, pages 273-278, 2324 October 2011. doi = 10.1145/2095050.2095096
- [15] Ji, X., Chang, D., Chen, B. The Measure Research on Mobile Commerce Process Based on Users. *In Proceedings of the 2014 11th International Conference on Service Systems and Service Management, ICSSSM 2014*, pages 1-5, 25-27 June 2014. doi = 10.1109/ICSSSM.2014.6943372
- [16] Greer Jr, M. B. and Ngo, J. W. Personal Emergency Preparedness Plan (PEPP) Facebook App: Using Cloud Computing, Mobile Technology, and Social Networking Services to Decompress Traditional Channels of Communication during Emergencies and Disasters. *In Proceedings of the 2012 IEEE Ninth International Conference on Services Computing, SCC '12*, pages 494-498, 24-29 June 2012. doi = 10.1109/SCC.2012.106
- [17] Hollingsworth, J., and Powell, D. J. Requiring Web-based Cloud and Mobile Computing in a Computer Science Undergraduate Curriculum. *In Proceedings of the 49th Annual Southeast Regional Conference, ACM-SE '11*, pages 19-24, 24-26 March 2011. doi = 10.1145/2016039.2016054
- [18] Cantara, M. Hype Cycle for Cloud Services Brokerage, 2012. *Gartner Report*, ID:G00234256, July 2012. url = <http://www.gartner.com/>
- [19] Buchholz, D., Dunlop, J., Ross, A. Improving Security and Mobility for Personally Owned Devices. *Intel White Paper*, February 2012. url = <http://software.intel.com/sites/billboard/sites/default/files/downloads/>
- [20] Janakiram, M.S.V. What Developers should know when choosing an MBaaS Solution. *GIGAOM Research*, Sept 16 2013. url = <http://research.gigaom.com/>
- [21] Riek, M.; Bohme, R.; Moore, T. Measuring the Influence of Perceived Cybercrime Risk on Online Service Avoidance. *IEEE Transactions on Dependable and Secure Computing*, vol.PP, no.99, pages 1-1, 9 March 2015. doi = 10.1109/TDSC.2015.2410795
- [22] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Journal of Future Generation Computer Systems*, Volume 25, Issue 6, pages 599-616, June 2009.
- [23] Dash, S. K., Mishra, D. P., Mishra, R., and Dash, S. Privacy Preserving K-Medoids Clustering: An Approach Towards Securing Data in Mobile Cloud Architecture. *In Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology, CCSEIT '12*, pages 439-443, 26 October 2012. doi = 10.1145/2393216.2393290
- [24] Gonzales, D., Kaplan, J., Saltzman, E., Winkelman, Z., Woods, D. Cloud-Trust - a Security Assessment Model for Infrastructure as a Service (IaaS) Clouds. *IEEE Transactions on Cloud Computing*, vol.PP, no.99, pages 1-14, 30 March 2015. doi = 10.1109/TCC.2015.2415794
- [25] Boonchieng, E. Performance and Security Issue on Open Source Private Cloud. *In Proceedings of the 2014 International Electrical Engineering Congress, iEECON 2014*, pages 1-5, 19-21 March 2014. doi = 10.1109/iEECON.2014.6925941
- [26] Li, S., Zhou, Y., Jiao, L., Yan, X., Wang, X., Lyu, M. Towards Operational Cost Minimization in Hybrid Clouds for Dynamic Resource Provisioning with Delay-aware Optimization. *IEEE Transactions on Services Computing*, vol.PP, no.99, pages 12, 06 February 2015. doi = 10.1109/TSC.2015.2390413
- [27] Gartner Report. Gartner Says the Personal Cloud Will Replace the Personal Computer as the Center of Users' Digital Lives by 2014. *Gartner Newsroom*, March 2012. url = <http://www.gartner.com/it/page.jsp?id=1947315>

- [28] Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., Peterson, R. A., Hong, L., Zheng, X., Musolesi, M., Fodor, K., and Gahng-Seop, A. The Rise of People-Centric Sensing. *IEEE Internet Computing*, vol.12, no.4, 12-21, July-Aug. 2008.
- [29] Liu, Q., Jian, X., Hu, J., Zhao, H., Zhang, S. An Optimized Solution for Mobile Environment Using Mobile Cloud Computing. *In Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCom '09*, pages 1-5 24-26 Sept. 2009. doi = 10.1109/WICOM.2009.5302240
- [30] Intel Solution Brief Report. Secure Big Data Analytics - The Marriage of REST APIs, Security and Big Data. *Intel Expressway Service Gateway*, September 2012.
- [31] Xue, G., Zhu, H., Hu, Z., Yu, J., Zhu, Y., Zhang, G. SmartCut: Mitigating 3G Radio Tail Effect on Smartphones. *IEEE Transactions on Mobile Computing*, vol.14, no.1, pages 169-179, 01 Jan. 2015. doi = 10.1109/TMC.2014.2313339
- [32] Dong, Y., Mao, J., Guan, H., Li, J., Chen, Y. A Virtualization Solution for BYOD With Dynamic Platform Context Switching. *IEEE Micro*, vol.35, no.1, pages 34-43, Jan.-Feb. 2015. doi = 10.1109/MM.2015.3
- [33] Guan, L., Ke, X., Song, M., and Song, J. A Virtual Cloud Computing Provider for Mobile Devices. *In Proceedings of the 2011 10th IEEE/ACIS International Conference on Computer and Information Science, ICIS '11*, pages 387-392, April 2011. doi = 10.1109/ICIS.2011.67
- [34] Huerta-Canepa, G., Lee, D. A Survey of Research on Mobile Cloud Computing. *In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond, MCS '10*, Article 6 5 pages, 15 June 2010. doi = 10.1145/1810931.1810937
- [35] Subpratatsavee, P.; Jantong, N.; Kuha, P.; Chintho, C. HC2D Barcode Reader Using Embedded Camera in Android Phone. *In Proceedings of the 2014 11th International Joint Conference on Computer Science and Software Engineering, JCSSE 2014*, pages 254-257, 14-16 May 2014. doi = 10.1109/JCSSE.2014.6841876
- [36] Bavota, G.; Linares-Vasquez, M.; Bernal-Cardenas, C.E.; Penta, M.D.; Oliveto, R.; Poshyvanyk, D. The Impact of API Change- and Fault-Proneness on the User Ratings of Android Apps. *IEEE Transactions on Software Engineering*, vol.41, no.4, pages 384-407, 01 April 2015. doi = 10.1109/TSE.2014.2367027
- [37] Bahl, P., Han, R. Y., Li, L. E., and Satyanarayanan, M. Advancing the State of Mobile Cloud Computing. *In Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services, MCS '12*, pages 21-28, 25 June 2012. doi = 10.1145/2307849.2307856
- [38] Kim, K.; Cho, B.; Ro, W.; Gaudiot, J. Network Variation and Fault Tolerant Performance Acceleration in Mobile Devices with Simultaneous Remote Execution. *IEEE Transactions on Computers*, vol.PP, no.99, 14 pages, 09 January 2015. doi = 10.1109/TC.2015.2389809
- [39] Ge, Y., Zhang, Y., Qiu, Q., and Lu, Y. H. A Game Theoretic Resource Allocation for Overall Energy Minimization in Mobile Cloud Computing System. *In Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED '12*, pages 279-284, July 30 August 1, 2012. doi = 10.1145/2333660.2333724
- [40] Gember, A., Dragga, C., and Akella, A. ECOS: Leveraging Software-Defined Networks to Support Mobile Application Offloading. *In Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS '12*, pages 199-210, 29 October 2012. doi = 10.1145/2396556.2396598
- [41] La, H. J., Oh, S. H., and Kim, S. D. Methods to Utilizing Cloud Computing in Developing Mobile Internet Device (MID) Applications. *In Proceedings of the 4th International Conference on Ubiquitous Information Management and Communication, ICUIMC '10*, 9 pages, January 1415, 2010. doi = 10.1145/2108616.2108654

- [42] Perera, C.; Jayaraman, P.P.; Zaslavsky, A.; Georgakopoulos, D.; Christen, P. MOSDEN: An Internet of Things Middleware for Resource Constrained Mobile Devices. *In Proceedings of the 2014 47th Hawaii International Conference on System Sciences, HICSS 2014*, pages 1053-1062, 6-9 Jan. 2014. doi = 10.1109/HICSS.2014.137
- [43] Baliga, A., Chen, X., Coskun, B., Reyes, G., Lee, S., Mathur, S., and Van der Merwe, J. E. VPMN: Virtual Private Mobile Network Towards Mobility-as-a-Service. *In Proceedings of the Second International Workshop on Mobile Cloud Computing and Services, MCS '11*, pages 7-12, Jun 28, 2011. doi = 10.1145/1999732.1999735
- [44] Xia, Y., Liu, Y., Tan, C., Ma, M., Guan, H., Zang, B., Chen, H. TinMan: Eliminating Confidential Mobile Data Exposure with Security Oriented Offloading. *In Proceedings of the Tenth European Conference on Computer Systems, EuroSys '15*, 16 pages, April 2015. doi = 10.1145/2741948.2741977
- [45] Bifulco, R., Brunner, M., Canonico, R., Hasselmeyer, P., and Mir, F. Scalability of a Mobile Cloud Management System. *In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12*, pages 17-22, 17 August 2012. doi = 10.1145/2342509.2342514
- [46] Chun, B. G., Curino, C., Sears, R., Shraer, A., Madden, A., and Ramakrishnan, R. Mobius: Unified Messaging and Data Serving for Mobile Apps. *In Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12*, pages 141-154, 25 June 2012. doi = 10.1145/2307636.2307650
- [47] Leners, J. B., Gupta, T., Aguilera, M. K., Walfish, M. Taming Uncertainty in Distributed Systems with Help from the Network. *In Proceedings of the 10th European Conference on Computer Systems, EuroSys '15*, pages 1-16, 21-24 April, 2015. doi = 10.1145/2741948.2741976
- [48] Dong, Y., Zhu, H., Peng, J., Wang, F., Mesnier, M. P., Wang, D., and Chan, S. C. RFS: A Network File System for Mobile Devices and the Cloud. *ACM SIGOPS Operating Systems Review*, Volume 45 Issue 1, pages 101-111, 18 February 2011. doi = 10.1145/1945023.1945036
- [49] Mao, H., Xiao, N., Shi, W., and Lu, Y. Wukong: Toward a Cloud-Oriented File Service for Mobile Devices. *In Proceedings of the 2010 IEEE International Conference on Services Computing, SCC '10*, pages 498-505, 5-10 July 2010. doi = 10.1109/SCC.2010.34
- [50] Koch, C.; Hausheer, D. Optimizing Mobile Prefetching by Leveraging Usage Patterns and Social Information. *In Proceedings of the 2014 IEEE 22nd International Conference on Network Protocols, ICNP 2014*, pages 293-295, 21-24 Oct. 2014. doi = 10.1109/ICNP.2014.51
- [51] Costa, P. M., Pitt, J., Falco e Cunha, J., and Galvo, T. Cloud2Bubble: Enhancing Quality of Experience in Mobile Cloud Computing Settings. *In Proceedings of the Third ACM Workshop on Mobile Cloud Computing and Services, MCS '12*, pages 45-52, 25 June 2012. doi = 1145/2307849.2307860
- [52] Hyhty, M., Palola, M., Matinmikko, M., and Katz, M. Cognitive Engine: Design Aspects for Mobile Clouds. *In Proceedings of the 4th International Conference on Cognitive Radio and Advanced Spectrum Management, CogART '11*, 5 pages, 26 October 2011. doi = 10.1145/2093256.2093288
- [53] Chen, M., Zhang, Y., Li, Y., Mao, S., Leung, V.C.M. EMC: Emotion-aware Mobile Cloud Computing in 5G. *IEEE Network*, vol.29, no.2, pages 32-38, March-April 2015. doi = 10.1109/MNET.2015.7064900
- [54] Burckhardt, S., Fhndrich, M., Leijen, D., and Wood, B. P. Cloud Types for Eventual Consistency. *In Proceedings of the 26th European Conference on Object-Oriented Programming, ECOOP'12*, pages 283-307, 15 June 2012. doi = 10.1007/978-3-642-31057-714
- [55] Lee, G., Ko, H., Pack, S., Kang, C-H. Efficient Delta Synchronization Algorithm in Mobile Cloud Networks. *In Proceedings of the 2014 IEEE 3rd International Conference on Cloud Networking, CloudNet'14*, pages 455-460, 8-10 October 2014. doi = 10.1109/CloudNet.2014.6969037

- [56] Chun, B. G., Ihm, S., Maniatis, P., Naik, M., Patti, A. CloneCloud: Elastic Execution between Mobile Device and Cloud. *In Proceedings of the 6th Conference on Computer Systems*, EuroSys '11, pages 301-314, 10 April 2011. doi = 10.1145/1966445.1966473
- [57] Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. MAUI: Making Smartphones Last Longer with Code Offload. *In Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49-62, 15 June 2010. doi = 10.1145/1814433.1814441
- [58] Ferdous, S.M., Rahman, M.S. A Metaheuristic Approach for Application Partitioning in Mobile System. *In Proceedings of the 2015 International Conference on Networking Systems and Security*, NSysS 2015, pages 1-6, 5-7 Jan. 2015. doi = 10.1109/NSysS.2015.7043520
- [59] Ferber, M., Rauber, T., Torres, M.H.C., Holvoet, T. Resource Allocation for Cloud-Assisted Mobile Applications. *In Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing*, CLOUD 2012, pages 400-407, 24-29 June 2012. doi = 10.1109/CLOUD.2012.75
- [60] Giurgiu, I. Understanding Performance Modeling for Modular Mobile-Cloud Applications. *In Proceedings of the Third Joint WOSP/SIPEW International Conference on Performance Engineering*, ICPE '12, pages 259-262, 22-25 April 2012. doi = 10.1145/2188286.2188333
- [61] Chen, Y., Zhu, Z., Zeng, Y., and He, Z. UbiCloud: A Cloud Computing System for Ubiquitous Terminals Based on End User Virtualization. *In Proceedings of the 2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing*, EUC 2010, pages 363-367, 11-13 Dec. 2010. doi = 10.1109/EUC.2010.150
- [62] Yousfi, A.; de Freitas, A.; Dey, A.; Saidi, R. The Use of Ubiquitous Computing for Business Process Improvement. *IEEE Transactions on Services Computing*, vol.PP, no.99, pages 1-12, 24 February 2015. doi = 10.1109/TSC.2015.2406694
- [63] Kim, K., Lee, S., and Congdon, P. On Cloud-Centric Network Architecture for Multi-Dimensional Mobility. *In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 1-6, 17 August 2012. doi = 10.1145/2342509.2342511
- [64] van der Merwe, J., Ramakrishnan, K.K., Fairchild, M., Flavel, A., Houle, J., Lagar-Cavilla, H.A., and Mulligan, J. Towards a Ubiquitous Cloud Computing Infrastructure. *In Proceedings of the 17th IEEE Workshop on Local and Metropolitan Area Networks*, LANMAN 2010, pages 1-6, 5-7 May 2010. doi = 10.1109/LANMAN.2010.5507151
- [65] Loomba, R.; Lei Shi; Jennings, B.; Friedman, R.; Kennedy, J.; Butler, J. Information Aggregation for Collaborative Sensing in Mobile Cloud Computing. *In Proceedings of the 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, MobileCloud 2014, pages 149,158, 8-11 April 2014. doi = 10.1109/MobileCloud.2014.23
- [66] Oulasvirta, A., and Sumari, L. Mobile Kits and Laptop Trays: Managing Multiple Devices in Mobile Information Work. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1127-1136, 28 April - 3 May 2007. doi = 10.1145/1240624.1240795
- [67] Intel White Paper. A Unified Mobile Architecture for the Modern Data Center. *Intel Expressway Service Gateway*, September 2012.
- [68] Xinogalos, S., Psannis, K. E., and Sifaleras, A. Recent Advances Delivered by HTML 5 in Mobile Cloud Computing Applications: A Survey. *In Proceedings of the 5th Balkan Conference in Informatics*, BCI '12, pages 199-204, 16-20 September 2012. doi = 10.1145/2371316.2371355
- [69] Bouras, C.; Papazois, A.; Stasinou, N. A Framework for Cross-Platform Mobile Web Applications Using HTML5. *In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud*, FiCloud 2014, pages 420-424, 27-29 August 2014. doi = 10.1109/FiCloud.2014.75



- [70] Pearce, J. HTML5 and the Dawn of Rich Mobile Web Applications Pt 1. *Available on slideshare*, Oct 10, 2011. url = <http://www.slideshare.net/jamesgpearce/html5-and-the-dawn-of-rich-mobile-web-applications-pt-1>
- [71] Koukoumidis, E., Lymberopoulos, D., Strauss, K., Liu, J., and Burger, D. Pocket Cloudlets. *In Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2011, pages 171-184, 11 March 2011. doi = 10.1145/1950365.1950387
- [72] Shi, C., Ammar, M. H., Zegura, E. W., and Naik, M. Computing in Cirrus Clouds: The Challenge of Intermittent Connectivity. *In Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 23-28, 17 August 2012. doi = 10.1145/2342509.2342515
- [73] Miluzzo, E., Cceres, R., and Chen, Y. Vision: mClouds - Computing on Clouds of Mobile Devices. *In Proceedings of the 3rd ACM workshop on Mobile Cloud Computing and Services*, MCC '12, pages 9-14, 17 August 2012. doi = 10.1145/2307849.2307854
- [74] Tor-Morten G., Ghinea, G., Younas, M. Context-Aware and Automatic Configuration of Mobile Devices in Cloud-Enabled Ubiquitous Computing. *Personal Ubiquitous Computing*, Vol 18, no 4, pages 883-894, April 2014. doi = 10.1007/s00779-013-0698-3
- [75] Lakshman, T. K., and Thuijs, X. Enhancing Enterprise Field Productivity via Cross Platform Mobile Cloud Apps. *In Proceedings of the 2nd International Workshop on Mobile Cloud Computing and Services*, MCS '11, pages 27-32, 28 June 2011. doi = 10.1145/1999732.1999741
- [76] Seo, S., Kim, M., Cui, Y., Seo, S., Lee, H. SFA-Based Cloud Federation Monitoring System for Integrating Physical Resources. *In Proceedings of the 2015 International Conference on Big Data and Smart Computing*, BigComp 2015, pages 55-58, 9-11 Feb. 2015. doi = 10.1109/35021BIGCOMP.2015.7072851
- [77] Agrawal, R., El-Bathy, N., and Seay, C. Medicaid Fraud Detection Using Data Broker Services. *SIGHT Rec*, vol 2, no 1, pages 25-25, March 2012. doi = 10.1145/2180796.2180817
- [78] Veloudis, S., Friesen, A., Paraskakis, I., Verginadis, Y., Patiniotakis, I. Underpinning a Cloud Brokerage Service Framework for Quality Assurance and Optimization. *In Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, CloudCom 2014, pages 660-663, 15-18 Dec. 2014. doi = 10.1109/CloudCom.2014.146
- [79] Houidi, I., Mechtri, M., Louati, W., and Zeghlache, D. Cloud Service Delivery across Multiple Cloud Platforms. *In Proceedings of the 2011 IEEE International Conference on Services Computing*, SCC '11, pages 741-742, 4-9 July 2011. doi = 10.1109/SCC.2011.107
- [80] Aazam, M., Huh., E. Broker as a Service (BaaS) Pricing and Resource Estimation Model. *In Proceedings of the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, CloudCom 2014, pages 463-468, 15-18 Dec. 2014. doi = 10.1109/CloudCom.2014.57
- [81] Sundareswaran, S., Squicciarini, A., and Lin, D. A Brokerage-Based Approach for Cloud Service Selection. *In Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing*, CLOUD 2012, pages 558-565, 24-29 June 2012. doi = 10.1109/CLOUD.2012.119
- [82] Papaioannou, T. G., Bonvin, N., and Aberer, K. Scalia: An Adaptive Scheme for Efficient Multi-Cloud Storage. *In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, 10 pages, 10 November 2012. doi = 978-1-4673-0806-9/12
- [83] Itani, W., Ghali, C., Bassil, R., Kayssi, A., Chehab, A. BGP-Inspired Autonomic Service Routing for the Cloud. *In Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 406-411, 01 June 2012. doi = 10.1145/2245276.2245356
- [84] Wang, X., Wang, Z., and Xu, X. Price Heuristics for Highly Efficient Profit Optimization of Service Composition. *In Proceedings of the 2011 IEEE International Conference on Services Computing*, SCC '11, pages 4378-385, 4-9 July 2011. doi = 10.1109/SCC.2011.11

- [85] Nair, S. K., Porwal, S., Dimitrakos, T., Ferrer, A. J., Tordsson, J., Sharif, T., Sheridan, C., Rajarajan, M., Khan, A. U. Towards Secure Cloud Bursting, Brokerage and Aggregation. *In Proceedings of the 2010 IEEE 8th European Conference on Web Services, SCC '11*, pages 189-196, 1-3 Dec. 2010. doi = 10.1109/ECOWS.2010.33
- [86] Noreen, Z., Hameed, I., and Usman, A. Development of Database Auditing Infrastructure. *In Proceedings of the 7th International Conference on Frontiers of Information Technology, FIT '09*, 6 pages, Dec 16, 2009. doi = 10.1145/1838002.1838092
- [87] Tao, W. and Steele, R. A Local Broker Enabled MobiPass Architecture for Enhancing Trusted Interaction Efficiency. *In Proceedings of the 31st Australasian Conference on Computer Science, ACSC '08*, pages 55-61, 01 January 2008.
- [88] Li, W., Tordsson, J., Elmroth, E. Modeling for Dynamic Cloud Scheduling via Migration of Virtual Machines. *In Proceedings of the 2011 IEEE 3rd International Conference on Cloud Computing Technology and Science, CloudCom 2011*, pages 163-171, Nov. 29 2011-Dec. 1 2011. doi = 10.1109/CloudCom.2011.31
- [89] Basu, S., Graupner, S., Pruyne, J., Singhal, S. Control Plane Integration for Cloud Services. *In Proceedings of the 11th International Middleware Conference Industrial Track, Middleware Industrial Track '10*, pages 29-34, 21 September 2010. doi = 10.1145/1891719.1891724
- [90] Blum, N., Magedanz, T., Stein, H., Wolf, I. An Open Carrier Controlled Service Environment for User Generated Mobile Multimedia Services. *In Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia, MoMM '09*, pages 137-145, 14 December 2009. doi = 10.1145/1821748.1821779
- [91] Chakraborty, D., Joshi, A., Finin, T., Yesha, Y. Service Composition for Mobile Environments. *Mob. Netw. Appl.*, vol 10, no 4, pages 435-451, August 2005.
- [92] Chen, Y., Schwan, K., and Zhou, D. Opportunistic Channels: Mobility-Aware Event Delivery. *In Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware, Middleware '03*, pages 182-201, 24 June 2003. doi = 10.1007/3-540-44892-6-10
- [93] Li, D., and Anand, M. MaJaB: Improving Resource Management for Web-Based Applications on Mobile Devices. *In Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services, MobiSys '09*, pages 95-108, 22 June 2009. doi = 10.1145/1555816.1555827
- [94] Chauhan, M. A., and Babar, M. A. Cloud Infrastructure for Providing Tools as a Service: Quality Attributes and Potential Solutions. *In Proceedings of the WICSA/ECSA 2012 Companion Volume, WICSA/ECSA '12*, pages 5-13, 20 August 2012. doi = 10.1145/2361999.2362002
- [95] Furtado, P., and Santos, C. Extensible Contract Broker for Performance Differentiation. *In Proceedings of the 2007 International Workshop on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '07*, 8 pages, 20 May 2007. doi = 10.1109/SEAMS.2007.7
- [96] Anastasi, G.F., Carlini, E., Coppola, M., Dazzi, P. QBROKAGE: A Genetic Approach for QoS Cloud Brokering. *In Proceedings of the 2014 IEEE 7th International Conference on Cloud Computing, CLOUD 2014*, pages 304,311, June 27 2014-July 2 2014. doi = 10.1109/CLOUD.2014.49
- [97] Messig, M., and Goscinski, A. Service Migration in Autonomic Service Oriented Grids. *In Proceedings of the 6th Australasian Workshop on Grid Computing and E-research - Volume 82, AusGrid '08*, pages 45-54, 01 January 2008.
- [98] Farooq, U., Parsons, E. W., Majumdar, S. Performance of Publish/Subscribe Middleware in Mobile Wireless Networks. *SIGSOFT Softw. Eng. Notes*, vol 29, no 1, pages 278-289, January 2004. doi = 10.1145/974043.974088
- [99] Ahmed, N., Linderman, M., Bryant, J. PAPA'S: Peer Assisted Publish and Subscribe. *In Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing, MW4NG '12*, 6 pages, 03 December 2012. doi = 10.1145/2405178.2405185

- [100] Oudenstad, J., Eliassen, F., Gjrven, E., and Rouvoy, R. Peer-to-peer Brokering of Planning Meta-Data. *In Proceedings of the 6th International Workshop on Adaptive and Reflective Middleware*, ARM '07, 3 pages, 03 December 2012. doi = 10.1145/1376780.1376786
- [101] Gaddah, A., Kunz, T. A Pro-active Mobility Management Scheme for Pub/Sub Systems Using Neighborhood Graph. *In Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, IWCMC '09, pages 1-6, 21 June 2009. doi = 10.1145/1582379.1582381
- [102] Lundquist, D., and Ouksel, A. An Efficient Demand-Driven and Density-Controlled Publish/Subscribe Protocol for Mobile Environments. *In Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems*, DEBS '07, pages 26-37, 20 June 2007. doi = 10.1145/1266894.1266900
- [103] Malandrino, F., Casetti, C., and Chiasserini, C. Pub/Sub Content Sharing for Mobile Networks Format. *In Proceedings of the 10th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '09, pages 351-352, 18-21 May 2009. doi = 10.1145/1530748.1530809
- [104] Salvador, Z., Lafuente, A., and Larrea, M. Avoiding Mobility-Related Message Flooding in Content-Based Publish/Subscribe. *In Proceedings of the First International Workshop on Algorithms and Models for Distributed Event Processing*, AIMoDEP '11, pages 27-32, 19 September 2011. doi = 10.1145/2031792.2031797
- [105] Chao-Fan, D., Tao, T., Zhang, P., Feng, Y. A Comparison of Data Provenance Systems Based on Processing. *In Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, ICIS 2010, pages 374-379, 29-31 Oct. 2010. doi = 10.1109/ICICISYS.2010.5658641
- [106] Sun, L., Park, J., Nguyen, D., Sandhu, R. A Provenance-aware Access Control Framework with Typed Provenance. *IEEE Transactions on Dependable and Secure Computing*, vol.PP, no.99, pages 1-14, 09 March 2015. doi = 10.1109/TDSC.2015.2410793
- [107] Lucia, B.; Ceze, L. Data Provenance Tracking for Concurrent Programs. *In Proceedings of the 2015 IEEE/ACM International Symposium on Code Generation and Optimization*, CGO 2015, pages 146-156, 7-11 Feb. 2015. doi = 10.1109/CGO.2015.7054195
- [108] Li, T., Liu, L., Zhang, X., Xu, K., Yang, C. ProvenanceLens: Service Provenance Management in the Cloud. *In Proceedings of 2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, CollaborateCom 2014, pages 275-284, 22-25 Oct. 2014. doi = 10.4108/icst.collaboratecom.2014.257339
- [109] Hammad, R., Ching-Seh Wu. Provenance as a Service: A Data-centric Approach for Real-Time Monitoring. *In Proceedings of the 2014 IEEE International Congress on Big Data*, BigData Congress 2014, pages 258-265, June 27 2014-July 2 2014. doi = 10.1109/BigData.Congress.2014.46
- [110] Luo, Z., Xu, G. Research of Provenance Storage in Cloud Computing Environment. *In Proceedings of the 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, TrustCom 2014, pages 660-664, 24-26 Sept. 2014. doi = 10.1109/TrustCom.2014.85
- [111] Zhang, O.Q., Kirchberg, M., Ko, R.K.L., Lee, B. S. How to Track Your Data: The Case for Cloud Computing Provenance. *In Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CloudCom 2011, pages 446-453, Nov. 29 2011-Dec. 1 2011. doi = 10.1109/CloudCom.2011.66
- [112] Davidson, S. B., and Freire, J. Provenance and Scientific Workflows: Challenges and Opportunities. *In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1345-1350, 09 June 2008. doi = 10.1145/1376616.1376772
- [113] Simmhan, Y. L., Plale, B., Gannon, D., and Marru, S. Performance Evaluation of the Karma Provenance Framework for Scientific Workflows. *Provenance and Annotation of Data, Lecture Notes in Computer Science*, Volume 4145, pages 222-236, 2006. doi = 10.1007/11890850-23

- [114] Tylissanakis, G., and Cotronis, Y. Data Provenance and Reproducibility in Grid Based Scientific Workflows. *In Proceedings of the Grid and Pervasive Computing Conference, GPC '09*, pages 42-49, 4-8 May 2009. doi = 10.1109/GPC.2009.16
- [115] Woodman, S., Hiden, H., Watson, P., Missier, P. Achieving Reproducibility by Combining Provenance with Service and Workflow Versioning. *In Proceedings of the 6th Workshop on Workflows in Support of Large-Scale Science, WORKS '11*, pages 127-136, 14 November 2011. doi = 10.1145/2110497.2110512
- [116] Ikeda, R., Salihoglu, S., Widom, J. Provenance-Based Refresh in Data-Oriented Workflows. *In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 1659-1668, 24 October 2011. doi = 10.1145/2063576.2063816
- [117] Knutov, E., Bra, P.M.E. De and Pechenizkiy, M. Provenance meets Adaptive Hypermedia. *In Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, HT'10*, pages 93-98, 13-16 June, 2010.
- [118] Lawabni, A. E., Changjin H., Du, D. H. C., Tewfik, A. H. A Novel Update Propagation Module for the Data Provenance Problem: A Contemplating Vision on Realizing Data Provenance from Models to Storage. *In Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies, CIKM '11*, pages 61- 69, 11-14 April 2005. doi = 10.1109/MSST.2005.2
- [119] Zhang, O.Q., Ko, R.K.L., Kirchberg, M., Suen, C. H., Jagadpramana, P., Lee, B. S. How to Track Your Data: Rule-Based Data Provenance Tracing Algorithms. *In Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012*, pages 1429-1437, 25-27 June 2012. doi = 10.1109/TrustCom.2012.175
- [120] Amsterdamer, Y., Deutch, D., Tannen, V. Provenance for Aggregate Queries. *In Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '11*, pages 153-164, 05 January 2011. doi = 10.1145/1989284.1989302
- [121] Amsterdamer, Y., Deutch, D., Milo, T., Tannen, V. On Provenance Minimization. *Database Syst. ACM Trans*, vol 37, no 4, 36 pages, December 2012. doi = 10.1145/2389241.2389249
- [122] Bao, Z., Khler, H., Wang, L., Zhou, X., and Sadiq, S. Efficient Provenance Storage for Relational Queries. *In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 1352-1361, 29 October - 2 November 2012. doi = 10.1145/2396761.2398439
- [123] Glavic, B., and Alonso, G. Perm: Processing Provenance and Data on the Same Data Model through Query Rewriting. *In Proceedings of the IEEE 25th International Conference on Data Engineering, ICDE '09*, pages 174-185, March 29 2009-April 2 2009. doi = 10.1109/ICDE.2009.15
- [124] She, W., I-Ling, Y., Bastani, F., Tran, B., Thuraisingham, B. Role-Based Integrated Access Control and Data Provenance for SOA Based Net-Centric Systems. *In Proceedings of the 2011 IEEE 6th International Symposium on Service Oriented System Engineering, SOSE 2011*, pages 225-234, 12-14 Dec. 2011. doi = 10.1109/SOSE.2011.6139111
- [125] Tsai, W., Wei, X., Zhang, D., Paul, R., Chen, Y., Chung, J. A New SOA Data-Provenance Framework. *In Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems, ISADS '07*, pages 105-112, 21-23 March 2007. doi = 10.1109/ISADS.2007.5
- [126] Park, J., Nguyen, D., and Sandhu, R. On Data Provenance in Group-Centric Secure Collaboration. *In Proceedings of the 2011 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2011*, pages 221-230, 15-18 Oct. 2011. doi = 10.4108/icst.collaboratecom.2011.247192
- [127] Nguyen, D., Park, J., and Sandhu, R. Integrated Provenance Data for Access Control in Group-Centric Collaboration. *In Proceedings of the 2012 IEEE 13th International Conference on Information Reuse and Integration, IRI 2012*, pages 255-262, 8-10 Aug. 2012. doi = 10.1109/IRI.2012.6303018

- [128] Alharbi, K., and Lin, X. PDP: A Privacy-Preserving Data Provenance Scheme. *In Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops, ICDCSW 2012*, pages 500-505, 18-21 June 2012. doi = 10.1109/ICDCSW.2012.58
- [129] Hasan, R., Sion, R., Winslett, M. Preventing History Forgery with Secure Provenance. *Trans. Storage*, vol 5, no 4, 43 pages, December 2009. doi = 10.1145/1629080.1629082
- [130] Braun, U., Garfinkel, S., Holland, D. A., Muniswamy-Reddy, K., and Seltzer, M. I. Issues in Automatic Provenance Collection. *Lecture Notes in Computer Science on Provenance and Annotation of Data*, vol 4145, pages 171-183, 2006. doi = 10.1007/11890850-18
- [131] Chapman, A. P., Jagadish, H. V., Ramanan, P. Efficient Provenance Storage. *In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08*, pages 993-1006, 09 June 2008. doi = 10.1145/1376616.1376715
- [132] Xie, Y., Feng, D., Tan, Z., Chen, L., Muniswamy-Reddy, K., Li, Y., and Long, D. D. E. A Hybrid Approach for Efficient Provenance Storage. *In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 1752-1756, 29 October 2012. doi = 10.1145/2396761.2398511
- [133] Cheney, J. A Formal Framework for Provenance Security. *In Proceedings of the 2011 IEEE 24th Computer Security Foundations Symposium, CSF '11*, pages 281-293, 27 June 2011. doi = 10.1109/CSF.2011.26
- [134] Reilly, C. F., and Naughton, J. F. Exploring Provenance in a Distributed Job Execution System. *Lecture Notes in Computer Science on Provenance and Annotation of Data*, Volume 4145, pages 237-245, 3-5 May 2006. doi = 10.1007/11890850-24
- [135] Demsky, B. Cross-Application Data Provenance and Policy Enforcement. *ACM Transactions on Information and System Security*, vol 14, no 1, 22 pages, June 2011. doi = 10.1145/1952982.1952988
- [136] Lim, C., Lu, S., Chebotko, A., and Fotouhi, F. Prospective and Retrospective Provenance Collection in Scientific Workflow Environments. *In Proceedings of the 2010 IEEE International Conference on Services Computing, SCC 2010*, pages 449-456, 5-10 July 2010. doi = 10.1109/SCC.2010.18
- [137] Sultana, S., Shehab, M., Bertino, E. Secure Provenance Transmission for Streaming Data. *IEEE Transactions on Knowledge and Data Engineering*, vol.25, no.8, pages 1890-1903, Aug. 2013. doi = 10.1109/TKDE.2012.31
- [138] Genquan, R. Li, Z., Jianmin, W., and Yinbo, L. One Method for Provenance Tracking of Product Lifecycle Data in Collaborative Service Environment. *In Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2011*, pages 347-356, 10-12 October 2011. doi = 10.1109/CyberC.2011.62
- [139] Miles, S., Groth, P., Munroe, S., and Moreau, L. PrIME: A Methodology for Developing Provenance-Aware Applications. *ACM Transactions on Software Engineering and Methodology*, vol 20, no 3, 42 pages, August 2011. doi = 10.1145/2000791.2000792
- [140] Marjit, U., Sarkar, A., and Biswas, U. A Novel Approach to Develop Linked Data with Provenance. *In Proceedings of the 2012 International Conference on Computing, Communication and Applications, ICCCA 2012*, pages 1-5, 22-24 Feb. 2012. doi = 10.1109/ICCCA.2012.6179211
- [141] Sakka, M. A., and Defude, B. A Mediator-Based System for Distributed Semantic Provenance Management Systems. *In Proceedings of the 16th International Database Engineering and Applications Symposium, IDEAS '12*, pages 193-198, 08 August 2012. doi = 10.1145/2351476.2351499
- [142] Filman, R. Reasoning with Worlds and Truth Maintenance in a Knowledge-Based Programming Environment. *ACM Commun*, vol 31, no 4, pages 382-401, April 1988. doi = 10.1145/42404.42405

- [143] Zhuang, C. A Knowledge Base with Dependencies. *In Proceedings of the 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery*, FSKD 2014, pages 403-407, 19-21 Aug. 2014. doi = 10.1109/FSKD.2014.6980868
- [144] Hodnebo O., and Lokketangen, E. The use of an ATMS in Consistency Checking of a Legal Expert System. *In Proceedings of the 4th International Conference on Artificial Intelligence and Law*, ICAIL '93, pages 72-75, 1 Aug. 1993. doi = 10.1145/158976.158985
- [145] Tanaka, M., Murakami, Y., Lin, D, and Ishida, T. A Service Binding Framework for Open Environment. *In Proceedings of the 2012 IEEE Ninth International Conference on Services Computing*, SCC 2012, pages 226-233, 24-29 June 2012. doi = 10.1109/SCC.2012.74
- [146] Ahmed, M.R., Cui, H., Huang, X. Smart Integration of Cloud Computing and MCMC Based Secured WSN to Monitor Environment. *In Proceedings of the 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace and Electronic Systems*, VITAE 2014, pages 1-5, 11-14 May 2014. doi = 10.1109/VITAE.2014.6934449
- [147] Jin, L., Takabi, H., Joshi, J. B. D. Towards Active Detection of Identity Clone Attacks on Online Social Networks. *In Proceedings of the 1st ACM Conference on Data and Application Security and Privacy*, CODASPY11, pages 27-38, 21-23 February, 2011. doi = 10.1145/1943513.1943520
- [148] Lomotey, R. K. and Deters, R. Reliable Consumption of Web Services in a Mobile-Cloud Ecosystem Using REST. *In Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, SOSE 2013, pages 13-24, 25-28 March 2013. doi = 10.1109/SOSE.2013.10
- [149] Lewis, G.; Echeverria, S.; Simanta, S.; Bradshaw, B.; Root, J. Tactical Cloudlets: Moving Cloud Computing to the Edge. *In Proceedings of the 2014 IEEE Military Communications Conference*, MILCOM 2014, pages 1440-1446, 6-8 Oct. 2014. doi = 10.1109/MILCOM.2014.238
- [150] Pacific Northwest National Laboratory. Edge Computing. *Technical Paper*, pages 1, 2013. url = <http://vis.pnnl.gov/pdf/fliers/EdgeComputing.pdf>
- [151] Do, C.T., Tran, N.H., Chuan Pham, Alam, M.G.R., Jae Hyeok Son, Choong Seon Hong. A Proximal Algorithm for Joint Resource Allocation and Minimizing Carbon Footprint in Geo-Distributed Fog Computing. *In Proceedings of the 2015 International Conference on Information Networking*, ICOIN 2015, pages 324-329, 12-14 Jan. 2015. doi = 10.1109/ICOIN.2015.7057905
- [152] Research at Cisco. Fog Computing, Ecosystem, Architecture and Applications. *Technical Paper*, pages 1, 2013. url = <http://www.cisco.com/c/en/us/index.html>
- [153] Lomotey, R. K., Deters, R. Efficient Consumption of the Electronic Health Record in mHealth. *International Journal of Network Modeling Analysis in Health Informatics and Bioinformatics (NetMAHIB)*, Vol 2, no 1, pages 1-12, February 2014. doi = 10.1007/s13721-014-0051-4
- [154] Lomotey, R. K., Mulder, K., Nilson, J., Schachter, C., Wittmeier, K., Deters, R. Mobile Self-Management Guide for Young Men with Mild Hemophilia in Cases of Minor Injuries. *International Journal of Network Modeling Analysis in Health Informatics and Bioinformatics*, Vol 3, no 1, pages 1-12, July 2014. doi = 10.1007/s13721-014-0064-z
- [155] Lomotey, R. K., Deters, R. MUBaaS: Mobile Ubiquitous Brokerage as a Service. *International Journal of the World Wide Web*, Vol 18, no 2, pages 1-15, February 2015. doi = 10.1007/s11280-015-0323-7
- [156] Akerkar, R., Badica, C., Burdescu, C.B. Desiderata for Research in Web Intelligence, Mining and Semantics. *In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, 5 pages, 13 June 2012. doi = 10.1145/2254129.2254131
- [157] Lomotey, R. K., Deters, R. Analytics-as-a-Service Framework for Terms Association Mining in Unstructured Data. *International Journal of Business Process Integration and Management*, Vol. 7, No. 1, pages 49-61, 2014. doi = 10.1007/s11280-015-0323-7

- [158] Lindholm, T., Kangasharju, J., and Tarkoma, S. Syxaw: Data Synchronization Middleware for the Mobile Web. *International Journal of Mob. Netw. Appl.*, Vol. 14, No. 5, pages 661-676, October 2009. doi = DOI=10.1007/s11036-008-0146-1
- [159] Whang, K., Song, I., Kim, T., and Lee, K. The Ubiquitous DBMS. *ACM SIGMOD Record*, Vol. 38, No. 4, pages 14-22, June 2010. doi = DOI=10.1145/1815948.1815952
- [160] Ihm, S-Y., Nasridinov, A., Lee, J-H., and Park, Y-H. Efficient Duality-Based Subsequent Matching on Time-Series Data in Green Computing. *The Journal of Supercomputing*, Vol. 69, No. 3, pages 1039-1053, September 2014. doi = DOI=10.1007/s11227-013-1028-2
- [161] Lv, J., and Zheng, X. Y. Research for a Data Synchronization Model Based on Middleware and Rule Base. *In Proceedings of the 2009 1st International Conference on Information Science and Engineering, ICISE 2009*, pages 2998-3001, 26-28 Dec. 2009. doi = DOI=10.1109/ICISE.2009.904
- [162] Sarwat, M., and Mokbel, M. F. Mobi Social (Mobile and Social) Data Management: A Tutorial. *In Proceedings of the 2015 IEEE International Conference on Cloud Engineering, IC2E 2015*, pages 4, 9-13 March 2015. doi = 10.1109/IC2E.2015.34
- [163] Mokdad, L., and Sene, M. Performance Measures of a Call Admission Control in Mobile Networks Using SWN. *In Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools*, valuetools '06, 7 pages, 11-13 October 2006. doi = 10.1145/1190095.1190177
- [164] Yang, G. Data Synchronization for Integration Systems Based on Trigger. *In Proceedings of the 2010 2nd International Conference on Signal Processing Systems, ICSPS 2010*, pages 310-312, 5-7 July 2010. doi = 10.1109/ICSPS.2010.5555804
- [165] Shun-Yan, W., Luo, Z., and Yong-Liang, C. A Data Synchronization Mechanism for Cache on Mobile Client. *In Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2006*, pages 1-5, 22-24 Sept. 2006. doi = 10.1109/WiCOM.2006.399
- [166] Soon, C. J., and Roe, P. Annotation Architecture for Mobile Collaborative Mapping. *In Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, MoMM '08*, pages 211-218, 24-26 November 2008. doi = 10.1145/1497185.1497230
- [167] Lo, J.T.K., Wohlstadter, E., Mesbah, A. Imagen: Runtime Migration of Browser Sessions for Javascript Web Applications. *In Proceedings of the 22nd International Conference on World Wide Web, WWW '13*, pages 815-826, 13-17 May 2013. doi = 978-1-4503-2035-1/13/05
- [168] Xue, Y. The Research on Data Synchronization of Distributed Real-Time Mobile Network. *In Proceedings of the 2008 International Conference on Computer Science and Software Engineering, CSSE 2008*, pages 1104-1107, 12-14 Dec. 2008. doi = 10.1109/CSSE.2008.1296
- [169] Khan, A. M., Chabridon, S., and Beugnard, A. 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*, pages 99-104, 19-20 September 2007. doi = 10.1145/1326257.1326275
- [170] Savery, C., Graham, N., Gutwin, C., Brown, M. The Effects of Consistency Maintenance Methods on Player Experience and Performance in Networked Games. *In Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW '14*, pages 1344-1355, 15-19 February 2014. doi = 10.1145/2531602.2531616
- [171] Li, N., Zhao, C., Choe, E.K., Ritter, F.E. HHeal: A Personalized Health App for Flu Tracking and Prevention. *In Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '15*, pages 1415-1420, 18-23 April 2015. doi = 10.1145/2702613.2732804
- [172] Lomotey, R., Kazi, R., and Deters, R. Near Real-Time Medical Data Dissemination in m-Health. *In Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES '12*, pages 67-74, 28-31 October 2012. doi = 10.1145/2457276.2457290

- [173] Arunachalan, B., and Light, J. 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, pages 326-329, 27-29 Oct. 2008. doi = 10.1109/DS-RT.2008.50
- [174] Sain, M., Lee, H., and Chung, W. 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, pages 149-154, 24-26 November 2009. doi = 10.1145/1655925.1655953
- [175] Sain, M., Lee, H., and Chung, W. Designing Context Awareness Middleware Architecture for Personal Healthcare Information System. *In Proceedings of the 12th International Conference on Advanced Communication Technology*, ICACT'10, pages 1650-1654, 7-10 February 2010. doi = 978-89-5519-146-2
- [176] Magee, A. Synchronization in Next-Generation Mobile Backhaul Networks. *IEEE Communications Magazine*, vol.48, no.10, pages 110-116, October 2010. doi = 10.1109/MCOM.2010.5594685
- [177] Boukerche, A., and Owens, H. Media Synchronization and QoS Packet Scheduling Algorithms for Wireless Systems. *International Journal of Mobile Networks and Applications*, vol.10, no.1-2, pages 233-249, February 2005. doi = 10.1023/B:MOONE.0000048557.95522.da
- [178] Choi, E., Bae, C. S., and Lee, J. Data Synchronization Between Adjacent User Devices for Personal Cloud Computing. *In Proceedings of the 2012 IEEE International Conference on Consumer Electronics*, ICCE 2012, pages 49-50, 13-16 Jan. 2012. doi = 10.1109/ICCE.2012.6161732
- [179] Gantsou, D. A DSA Model for Data Access in Self-Organizing Systems. *In Proceedings of the 2003 Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and Reliable Software for Real-Time and Distributed Systems Using Ada and Related Technologies*, SIGAda '03, pages 25-28, 7 December 2003. doi = 10.1145/958420.958425
- [180] Choi, M., Kim, Y., and Chang, J. Transaction-Centric Split Synchronization Mechanism for Mobile E-business Applications. *In Proceedings of the International Workshop on Data Engineering Issues in E-Commerce*, DEEC 2005, pages 112- 118, 9 April 2005. doi = 10.1109/DEEC.2005.24
- [181] Choi, M., Park, W., and Kim, Y. A Split Synchronizing Mobile Transaction Model. *In Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, ICUIMC '08, pages 196-201, 31 January 2008. doi = 10.1145/1352793.1352835
- [182] Marforio, C., Ritzdorf, H., Francillon, A., and Capkun, S. Analysis of the Communication Between Colluding Applications on Modern Smartphones. *In Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 51-60, 3-7 December 2012. doi = 10.1145/2420950.2420958
- [183] Gad, S. Data Synchronization Architectural Pattern for Ubiquitous Learning Systems. *In Programming Support Innovations for Emerging Distributed Applications*, PSI EtA '10, 5 pages, 17 October 2010. doi = 10.1145/1940747
- [184] Hansen, F. A. and Grnbk, K. UrbanWeb: A Platform for Mobile Context-Aware Social Computing. *In Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, HT '10, pages 195-200, 13-16 June 2010. doi = 10.1145/1810617.1810651
- [185] Klingelhuber, P., and Mayrhofer, R. Private Notes: Encrypted XML Notes Synchronization and Sharing with Untrusted Web Services. *In Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '11, pages 254-261, 5-7 December 2011. doi = 10.1145/2095536.2095579
- [186] Koskimies, O. Using Data Item Relationships to Adaptively Select Data for Synchronization. *In Proceedings of the 5th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, DAIS'05, pages 220-225, 15-17 June 2005. doi = 10.1007/11498094-20
- [187] Ratner, D., Reiher, P., and Popek, G. J. Roam: A Scalable Replication System for Mobility. *International Journal of Mob. Netw. Appl.*, vol 9, no 5, pages 537-544, October 2004. doi =



- [188] Xie, J., Yu, D., and Ma, S. A Novel Enterprise Mobile E-mail System Using the SyncML Protocol. *In Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science, ICIS '08*, pages 391-396, 2008. doi = 10.1109/ICIS.2008.10
- [189] Lam, F., Lam, N., and Wong, R. Efficient Synchronization for Mobile XML Data. *In Proceedings of the 11th International Conference on Information and Knowledge Management, CIKM '02*, pages 153-160, 4 November 2002. doi = 10.1145/584792.584820
- [190] Veeraraghavan, K., Ramasubramanian, V., Rodeheffer, T. L., Terry, D. B., and Wobber, T. Fidelity-Aware Replication for Mobile Devices. *IEEE Transactions on Mobile Computing*, vol 9, no 12, pages 1697-1712, December 2010. doi = 10.1109/TMC.2010.118
- [191] Veeraraghavan, K., Ramasubramanian, V., Rodeheffer, T. L., Terry, D. B., and Wobber, T. Fidelity-Aware Replication for Mobile Devices. *In Proceedings of the 7th international conference on Mobile systems, applications, and services*, Mobisys 2009, pages 83-94, 22-25 June 2009. doi = 10.1145/1555816.1555826
- [192] Lomotey, R. K. Enabling Mobile Devices to Host Consumers and Providers of RESTful Web Services. *Master of Science Thesis, University of Saskatchewan*, 127 pages, March 2012. url = <http://hdl.handle.net/10388/ETD-2012-03-371>
- [193] Mokarizadeh, S.; Kungas, P.; Matskin, M. Utilizing Web Services Networks for Web Service Innovation. *In Proceedings of the 2014 IEEE International Conference on Web Services, ICWS 2014*, pages 646-653, June 27 2014-July 2 2014. doi = 10.1109/ICWS.2014.95
- [194] Celik, D., and Elgi, A. A Semantic Search Agent Approach: Finding Appropriate Semantic Web Services Based on User Request Term(s). *In Proceedings of the 3rd International Conference on Information and Communications Technology, ITI 2005*, pages 675-687, 5-6 Dec. 2005. doi = 10.1109/ITICT.2005.1609659
- [195] Zhengdong, Z., Yahong, H., Ronggui, L., Weiguo, W., and Zengzhi, L. A P2P-Based Semantic Web Services Composition Architecture. *In Proceedings of the IEEE International Conference on e-Business Engineering, ICEBE '09*, pages 403-408, 21-23 Oct. 2009. doi = 10.1109/ICEBE.2009.63
- [196] Abello, A.; Romero, O.; Bach Pedersen, T.; Berlanga, R.; Nebot, V.; Aramburu, M.J.; Simitsis, A. Using Semantic Web Technologies for Exploratory OLAP: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, vol.27, no.2, pages 571,588, 1 february 2015. doi = 10.1109/TKDE.2014.2330822
- [197] Wicks, G., Van Aerschot, E., Badreddin, O., Kubein, K., Lo, K., and Steele, D. Powering SOA Solutions with IMS. *Publisher: IBM Redbooks Pub.*, Part Number: SG24-7662-00, Pages in Print Edition: 410, 30 March 2009. doi =
- [198] Pham, L., and Gehlen, G. Realization and Performance Analysis of a SOAP Server for Mobile Devices. *In Proceedings of the 11th European Wireless Conference 2005 - Next Generation Wireless and Mobile Communications and Services*, European Wireless 2005, Pages 1-7, 10-13 April 2005. doi = 978-3-8007-2886-2
- [199] Mizouni, R., Serhani, M.A., Dssouli, R., Benharref, A., and Taleb, I. Performance Evaluation of Mobile Web Services. *In Proceedings of the 2011 Ninth IEEE European Conference on Web Services, ECOWS 2011*, Pages 184-191, 14-16 Sept. 2011. doi = 10.1109/ECOWS.2011.12
- [200] Pautasso, C., Olaf, Z., and Leymann, F. RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. *In Proceedings of the 17th International Conference on World Wide Web, WWW'08*, pages 805-814, 21-25 April 2008. doi = 10.1145/1367497.1367606
- [201] Lomotey, R. K., Jamal, S., and Deters, R. SOPHRA: A Mobile Web Services Hosting Infrastructure in mHealth. *In Proceedings of the 2012 IEEE First International Conference on Mobile Services, MS 2012*, pages 88-95, 24-29 June 2012. doi = 10.1109/MobServ.2012.14

- [202] Christensen, J. H. Using RESTful Web-Services and Cloud Computing to Create Next Generation Mobile Applications. *In Proceedings of the Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, OOPSLA 2009, pages 627-633, 25-29 October 2009. doi = 10.1145/1639950.1639958
- [203] Lim, S., and Keum, C. Method of Application Driven QoS Service in Open Service Platform Based on RESTful Web Services. *In Proceedings of the 2014 International Conference on Information and Communication Technology Convergence*, ICTC 2014, pages 632,633, 22-24 Oct. 2014. doi = 10.1109/ICTC.2014.6983235
- [204] Selonen, P., Belimpasakis, P., and You, Y. Experiences in Building a RESTful Mixed Reality Web Service Platform. *In Proceedings of the 10th International Conference on Web Engineering*, ICWE 2010, pages 400-14, 18 December 2009. doi = 10.1007/978-3-642-13911-6-27
- [205] Wang, Q. Mobile Cloud Computing. *M.Sc. Thesis Submitted to the College of Graduate Studies and Research*, Department of Computer Science, University of Saskatchewan, Saskatoon, Canada, pages , January 2011. doi =
- [206] Stirbu, V. 2010. A RESTful Architecture for Adaptive and Multi-device Application Sharing. *In Proceedings of the 1st International Workshop on RESTful Design*, WS-REST 2010. Pages 62-65, 26 April 2010. doi = 978-1-60558-959-6/10/04
- [207] Liu, Y., and Wilde, E. Scalable and Mashable Location-Oriented Web Services. *In Proceedings of the 10th international conference on Web engineering*, ICWE'10. Pages 307-321, 5-9 July 2010. doi = 10.1007/978-3-642-13911-6-21
- [208] McFaddin, S., Coffman, D., Han, J. H., Jang, H. K., Kim, J. H., Lee, J. K., Lee, M. C., Moon, Y. S., Narayanaswami, C., Paik, Y. S., Park, J. W., and Soroker, D. Modeling and Managing Mobile Commerce Spaces Using RESTful Data Services. *In Proceedings of the 9th International Conference on Mobile Data Management*, MDM '08. Pages 81-89, 27-30 April 2008. doi = 10.1109/MDM.2008.38
- [209] Murata, Y. Mobile-NGN Architecture Based on REST Concept. *In Proceedings of the Innovations for Digital Inclusions*, K-IDI 2009. Pages 1-8, Aug. 31 2009-Sept. 1 2009. doi = 92-61-12891-2/CFP0938E
- [210] Chaudry, A. S., Chaudhary, M. A., Aijaz, F., and Walke, B. Mobile-to-Mobile Multimedia Streaming Using REST-Based Mobile Services. *In Proceedings of the 2012 International Conference on Multimedia Computing and Systems*, ICMCS 2012. Pages 681-686, 10-12 May 2012. doi = 10.1109/ICMCS.2012.6320232
- [211] Tsai, C., Chen, H., Huang, J., and Hu, C. Transmission Reduction Between Mobile Phone Applications and RESTful APIs. *In Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11. Pages 445-450, 21 March 2011. doi = 10.1145/1982185.1982280
- [212] Ulmer, C., Serme, G., and Bonillo, Y. Enabling Web Object Orientation with Mobile Devices. *In Proceedings of the 6th International Conference on Mobile Technology, Application and Systems*, Mobility '09. 4 Pages, 2 September 2009. doi = 10.1145/1710035.1710047
- [213] AlShahwan, F., and Moessner, K. Providing SOAP Web Services and RESTful Web Services from Mobile Hosts. *In Proceedings of the 2010 Fifth International Conference on Internet and Web Applications and Services*, ICIW 2010. Pages 174-179, 9-15 May 2010. doi = 10.1109/ICIW.2010.33
- [214] AlShahwan, F., Moessner, K., and Carrez, F. Distributing Resource Intensive Mobile Web Services. *In Proceedings of the 2011 International Conference on Innovations in Information Technology*, IIT 2011. Pages 41-46, 25-27 April 2011. doi = 10.1109/INNOVATIONS.2011.5893861
- [215] Aijaz, F., Ali, S. Z., Chaudhary, M. A., and Walke, B. Enabling Resource-Oriented Mobile Web Server for Short-Lived Services. *In Proceedings of the 2009 IEEE 9th Malaysia International Conference on Communications*, MICC 2009. Pages 392-396, 15-17 Dec. 2009. doi = 10.1109/MICC.2009.5431538

- [216] Aijaz, F., Shaikh, S. J., and Walke, B. Performance Analysis of a Framework for Multi-Interfaced Service Level Agreements on Mobile Devices. *In Proceedings of the Wireless Conference*, European Wireless 2011, Pages 1-8, 27-29 April 2011. doi = 978-3-8007-3343-9
- [217] Stan, J., and Maret, P. Semantic Metadata Management in Web 2.0. *In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, 10 Pages, 13 June 2012. doi = 10.1145/2254129.2254139
- [218] Klein J., and Gorton, I. Design Assistant for NoSQL Technology Selection. *In Proceedings of the 1st International Workshop on Future of Software Architecture Design Assistants*, FoSADA '15, Pages 7-12, 6 May 2015. doi = 10.1145/2751491.2751494
- [219] Bassiliades, N. Agents and Knowledge Interoperability in the Semantic Web Era. *In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, 13 Pages, 15 June 2012. doi = 10.1145/2254129.2254140
- [220] Akerkar, R., Badica, C., and Burdescu, D. B. Desiderata for Research in Web Intelligence, Mining and Semantics. *In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, 5 Pages, 15 June 2012. doi = 10.1145/2254129.2254131
- [221] Jiang, G., Solbrig, H. R., and Chute, C. G. Using Semantic Web technology to Support ICD-11 Textual Definitions Authoring. *Journal of Biomedical Semantics*, vol 4, no 11, Pages 1-9, April 2013. doi = 10.1186/2041-1480-4-11
- [222] Brunetti, J. M., and Garca, R. Information Architecture Automatization for the Semantic Web. *In Proceedings of the 13th IFIP TC 13 International Conference on Human-Computer Interaction*, INTERACT'11, Pages 410-413, 5-9 September 2011. doi = 10.1007/978-3-642-23768-3-45
- [223] Eljinini, M. A. H. Health: Related Information Structuring for the Semantic Web. *In Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*, ISWSA '11, 7 Pages, 18-20 April 2011. doi = 10.1145/1980822.1980828
- [224] Lopes, P., and Oliveira, J. L. COEUS: A Semantic Web Application Framework. *In Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, Pages 66-73, 7-9 December 2011. doi = 10.1145/2166896.2166915
- [225] Frber, C., and Hepp, M. Towards a Vocabulary for Data Quality Management in Semantic Web Architectures. *In Proceedings of the 1st International Workshop on Linked Web Data Management*, LWDM '11, Pages 1-8, 25 March 2011. doi = 10.1145/1966901.1966903
- [226] Garcia, D. Z. G., and Felgar de Toledo, M. B. Semantics-Enriched QoS Policies for Web Service Interactions. *In Proceedings of the 12th Brazilian Symposium on Multimedia and the Web*, WebMedia '06, Pages 35-44, 19-22 November 2006. doi = 10.1145/1186595.1186601
- [227] Ketter, W., Batchu, A., Berosik, G., and McCreary, D. A Semantic Web Architecture for Advocate Agents to Determine Preferences and Facilitate Decision Making. *In Proceedings of the 10th International Conference on Electronic Commerce*, ICEC '08, 10 Pages, 19-22 August 2008. doi = 10.1145/1409540.1409554
- [228] Auer, S., Dalamagas, T., Parkinson, H., Bancilhon, F., Flouris, G., Sacharidis, D., Buneman, P., Kotzinos, D., Stavarakas, Y., Christophides, V., Papastefanatos, G., and Thiveos, K. Diachronic Linked Data: Towards Long-Term Preservation of Structured Interrelated Information. *In Proceedings of the First International Workshop on Open Data*, WOD '12, Pages 31-39, 25 May 2012. doi = 10.1145/2422604.2422610
- [229] Heath, T., and Bizer, C. Linked Data: Evolving the Web into a Global Data Space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 136 Pages, February 2011. doi = 10.2200/S00334ED1V01Y201102WBE001

- [230] Pellegrini, T. Integrating Linked Data into the Content Value Chain: A Review of News-Related Standards, Methodologies and Licensing Requirements. *In Proceedings of the 8th International Conference on Semantic Systems, I-SEMANTICS '12*, Pages 94-102, 5-7 September 2012. doi = 10.1145/2362499.2362513
- [231] Sequeda, J. F., and Miranker, D. P. Linked Data. *Semantic Tech and Business Conference 2012*, Last Access Date: 20 MArch 2015. url = <http://www.slideshare.net/juansequeda/linked-data-tutorial-at-semtech-2012>
- [232] Rodrigues de Oliveira, H., Tavares, A. T., and Lscio, B. F. Feedback-Based Data Set Recommendation for Building Linked Data Applications. *In Proceedings of the 8th International Conference on Semantic Systems, I-SEMANTICS '12*, Pages 49-55, 5-7 September 2012. doi = 10.1145/2362499.2362507
- [233] Halb, W., Stocker, A., Mayer, H., Mulner, H., and Ademi, I. Towards a Commercial Adoption of Linked Open Data for Online Content Providers. *In Proceedings of the 6th International Conference on Semantic Systems, I-SEMANTICS '10*, 8 Pages, 1-3 September 2010. doi = 10.1145/1839707.1839727
- [234] Hartig, O., and Freytag, J-C. Foundations of Traversal Based Query Execution Over Linked Data. *In Proceedings of the 23rd ACM Conference on Hypertext and Social Media, HT '12*, Pages 43-52, 25-28 June 2012. doi = 10.1145/2309996.2310005
- [235] Hausenblas, M. Utilising Linked Open Data in Applications. *In Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS '11*, 4 Pages, 25-27 May 2011. doi = 10.1145/1988688.1988697
- [236] Mendes, P. N., Mhleisen, H., and Bizer, C. Sieve: Linked Data Quality Assessment and Fusion. *In Proceedings of the 2012 Joint EDBT/ICDT Workshops, EDBT-ICDT '12*, Pages 116-123, 30 March 2012. doi = 10.1145/2320765.2320803
- [237] Page, K. R., De Roure, D. C., and Martinez, K. REST and Linked Data: A Match Made for Domain Driven Development? *In Proceedings of the Second International Workshop on RESTful Design, WS-REST '11*, Pages 22-25, March 2011. doi = 10.1145/1967428.1967435
- [238] Paulheim, H. Improving the Usability of Integrated Applications by Using Interactive Visualizations of Linked Data. *In Proceedings of the International Conference on Web Intelligence, Mining and Semantics, WIMS '11*, 12 Pages, 25 May 2011. doi = 10.1145/1988688.1988711
- [239] Le-Phuoc, D., Dao-Tran, M., Parreira, J. X., and Hauswirth, M. A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. *In Proceedings of the 10th International Conference on the Semantic Web, ISWC'11*, Pages 370-388, 23-27 October 2011. doi = 10.1007/978-3-642-25073-6-24
- [240] Urdiales-Nieto, D., and Aldana-Montes, J. F. Towards Designing an Efficient Crawling Window to Analysis and Annotate Changes in Linked Data Sources. *In Proceedings of the 1st International Workshop on Linked Web Data Management, LWDM '11*, Pages 9-16, 25 March 2011. doi = 10.1145/1966901.1966904
- [241] Wood, P. T. Query Languages for Graph Databases. *SIGMOD Rec*, vol 41, no 1, Pages 50-60, April 2012. doi = 10.1145/2206869.2206879
- [242] Ritter, D. From Network Mining to Large Scale Business Networks. *In Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12*, Pages 989-996, 16-20 April 2012. doi = 10.1145/2187980.2188233
- [243] Angles, R., and Gutierrez, C. Survey of Graph Database Models. *ACM Comput. Surv.*, vol 40, no 1, 39 Pages, February 2008. doi = 10.1145/1322432.1322433
- [244] Dries, A., and Nijssen, S. Analyzing Graph Databases by Aggregate Queries. *In Proceedings of the Eighth Workshop on Mining and Learning with Graphs, MLG '10*, Pages 37-45, 24-25 July 2010. doi = 10.1145/1830252.1830258

- [245] Mondal, J., and Deshpande, A. Managing Large Dynamic Graphs Efficiently. *In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, Pages 145-156, 20-24 May 2012. doi = 10.1145/2213836.2213854
- [246] Tribl, S., and Leser, U. Fast and Practical Indexing and Querying of Very Large Graphs. *In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, Pages 845-856, 12-14 June 2007. doi = 10.1145/1247480.1247573
- [247] Lomotey, R. K., Mulder, K., Nilson, J., Schachter, C., Wittmeier, K., Deters, R. Mobile Aid for Haemophilia Injury Recognition. *International Journal of Biomedical Engineering Research (BER)*, Vol. 3, no 4, Pages 96-107, Dec. 2014. doi = 10.5963/BER0304002
- [248] Nilson J, Schachter C, Mulder K, Hahn M, Steele M, Hilliard P, Jarock C. A Qualitative Study Identifying the Knowledge, Attitudes and Behaviours of Young Men with Mild Haemophilia. *Haemophilia*, Vol. 18, no 3, Pages 120-125, May 2012. doi =
- [249] IBM Research. Analytics-as-a-Service Platform. *IBM Publications*, Last Access Date: 03 Deceber 2014 url = <http://researcher.ibm.com/researcher/view-project.php?id=3992>
- [250] EMC. EMC Accelerates Journey to Big Data with Business Analytics-as-a-Service. *EMC Publications*, Last Access Date: 03 Deceber 2014 url = <http://www.emc.com/collateral/white-papers/h11259-emc-accelerates-journey-big-data-ba-wp.pdf>
- [251] SAS. Analytics as a Service: Customer Experiences. *SAS Publications*, Last Access Date: 03 Deceber 2014 url = <http://www.sas.com/offices/europe/uk/resources/brochure/aaas-research-brief.pdf>
- [252] Sun, X., Gao, B., Fan, L., An, W. A Cost-Effective Approach to Delivering Analytics as a Service. *In Proceedings of the IEEE 19th International Conference on Web Services*, ICWS 2012, Pages 512-519, 24-29 June 2012. doi = 10.1109/ICWS.2012.79
- [253] Deepak, P., Deshpande, P.M., Murthy, K. Configurable and Extensible Multi-flows for Providing Analytics as a Service on the Cloud. *In Proceedings of the 2012 Annual SRII Global Conference*, SRII 2012, Pages 1-10, 24-27 July 2012. doi = 10.1109/SRII.2012.11
- [254] Hsu, J.Y., Yih, W. Template-Based Information Mining from HTML Documents. *In Proceedings of the 14th National Conference on Artificial Intelligence and Ninth conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, Pages 256-262, 27 July 1997. doi =
- [255] Chen, Y-C., Peng, W-C., Huang, J-L., Lee, W-C. Significant Correlation Pattern Mining in Smart Homes. *ACM Trans. Intell. Syst. Technol*, vol 6, no 3, 23 Pages, April 2015. doi = 10.1145/2700484
- [256] Tunali, V., Bilgin, T.T. PRETO: A High-performance Text Mining Tool for Preprocessing Turkish Texts. *In Proceedings of the 2012 International Conference on Computer Systems and Technologies*, vol 6, no 3, Pages 134-140, 22-23 June 2012. doi = 978-1-4503-1193-9/12/06
- [257] Janet, B., Reddy, A.V. Cube Index for Unstructured Text Analysis and Mining. *In Proceedings of the 2011 International Conference on Communication, Computing and Security*, ICCCS '11, Pages 397-402, 12-14 February 2011. doi = 10.1145/1947940.1948023
- [258] Han, L., Suzek, T.O., Wang, Y. and Bryant, S.H. The Text-Mining Based PubChem Bioassay Neighboring Analysis. *BMC Bioinformatics*, Vol. 11, No. 549, 9 Pages, 8 November 2010. doi = 10.1186/1471-2105-11-549
- [259] Dey, L., Haque, S.K.M. Studying the Effects of Noisy Text on Text Mining Applications. *In Proceedings of the Third Workshop on Analytics for Noisy Unstructured Text Data*, VAND '09, Pages 107-114, 23-24 July 2009. doi = 10.1145/1568296.1568314
- [260] Chiang, F., Miller, R.J. Automated Dictionary Discovery for the Online Marketplace. *In Proceedings of the 2012 iConference*, iConference '12, Pages 421-422, 7-10 Febraury 2012. doi = 10.1145/2132176.2132233

- [261] Lomotey, R. K., and Deters, R. Topics and Terms Mining in Unstructured Data Stores. *In Proceedings of the 2nd International Conference on Big Data Science and Engineering*, BDSE 2013, 8 Pages, 3-7 December 2013. doi = 10.1109/CSE.2013.129
- [262] Sah, S.K.; Joshi, S.R. Scalability of Efficient and Dynamic Workload Distribution in Autonomic Cloud Computing. *In Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques*, ICICT 2014, Pages 12-18, 7-8 Feb. 2014. doi = 10.1109/ICICT.2014.6781244
- [263] Anisetti, M.; Ardagna, C.A.; Damiani, E. A Certification-Based Trust Model for Autonomic Cloud Computing Systems. *In Proceedings of the 2014 International Conference on Cloud and Autonomic Computing*, ICCAC 2014, Pages 212-219, 8-12 Sept. 2014. doi = 10.1109/ICAC.2014.8
- [264] Stolf, P., Thierry, M. Track Report of Collaborative and Autonomic Green Computing (CAGing 2014). *In Proceedings of the 2014 IEEE 23rd International WETICE Conference*, WETICE 2014, Pages 113-114, 23-25 June 2014. doi = 10.1109/WETICE.2014.48
- [265] AlShahwan, F., Faisal, M. Mobile Cloud Computing for Providing Complex Mobile Web Services. *In Proceedings of the 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, MobileCloud 2014, Pages 77-84, 8-11 April 2014. doi = 10.1109/MobileCloud.2014.12
- [266] Lomotey, R. K., and Deters, R. Synchronization of Medical Data in a Mobile Provisioning Environment: mHealth Use Case. *International Journal of Biomedical Engineering Research*, Vol. 3, no 1, Pages 1-10, March 2014. doi = 10.5963/BER0301001
- [267] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. Section 26.2: The FordFulkerson method. Introduction to Algorithms (Second ed.) *MIT Press and McGrawHill*, Pages 651-664, ISBN 0-262-03293-7. doi =
- [268] Nygren, E., Sitaraman, R. K. and Sun, J. The Akamai Network: A Platform for High-Performance Internet Applications. *SIGOPS Oper. Syst. Rev.*, Vol. 44, no 3, Pages 2-19, August 2010. doi = 10.1145/1842733.1842736

# APPENDIX A

## RESULTS ON THE PROPAGATION WINDOW

**Table A.1:** Overall Summary of the Results on the Propagation Window

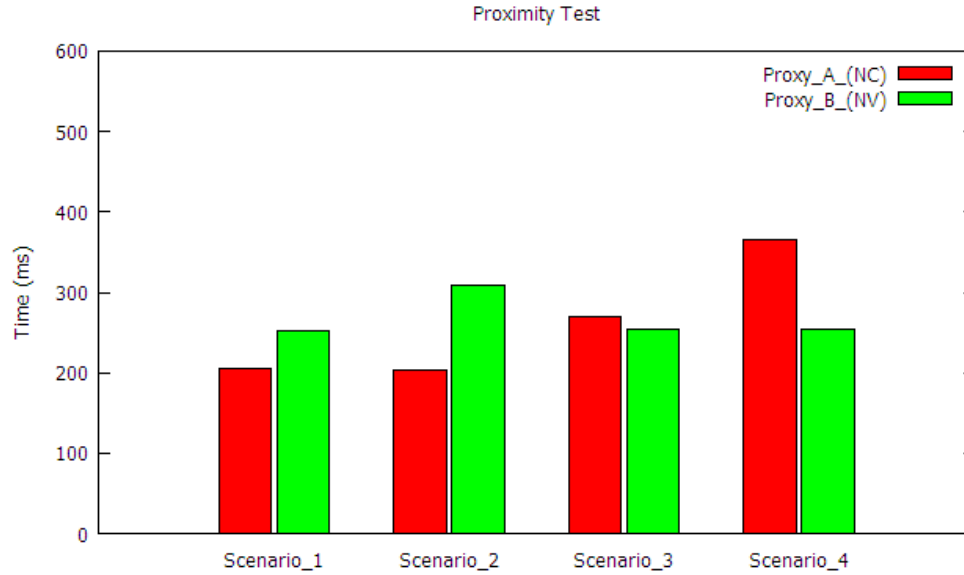
<i>Dropbox</i>					
<i>Device</i>	<i>Set-up</i>	<i>Max Window (s)</i>	<i>Min Window (s)</i>	<i>Avg (s)</i>	<i>Average Propagation Window Relative to the Base Result (s)</i>
Playbook	Base	662.44	51.22	360.79	N/A
	Centralized	662.47	61.59	360.37	0.42 faster
	Overloaded	1389.20	117.45	742.29	381.5 slower
	Distributed	750.81	71.82	409.44	48.65 slower
iPad3	Base	761.35	65.56	408.44	N/A
	Centralized	762.92	67.13	410.01	1.57 slower
	Overloaded	1525.07	128.21	810.94	402.5 slower
	Distributed	856.83	76.85	461.23	52.79 slower
Transformer	Base	827.35	107.56	478.05	N/A
	Centralized	824.59	104.80	475.29	2.76 faster
	Overloaded	1640.10	199.01	932.26	454.21 slower
	Distributed	935.20	122.56	540.85	62.8 slower
NOKIA	Base	1115.23	217.56	652.12	N/A
	Centralized	1117.93	220.26	660.99	8.87 slower
	Overloaded	2221.33	427.08	1303.21	651.09 slower
	Distributed	1322.83	254.60	779.07	118.08 slower
<i>Amazon S3</i>					
Playbook	Base	524.93	50.04	291.93	N/A
	Centralized	571.63	54.74	324.94	33.01 slower
	Overloaded	1036.36	95.53	582.90	290.97 slower
	Distributed	663.30	58.55	374.68	82.75 slower
iPad3	Base	665.66	59.44	368.17	N/A
	Centralized	667.87	61.65	370.38	2.21 slower
	Overloaded	1142.73	105.61	633.96	265.79 slower
	Distributed	804.12	71.80	444.75	76.58 slower
Transformer	Base	860.00	95.44	480.63	N/A
	Centralized	837.93	99.37	473.52	7.11 faster
	Overloaded	1593.74	189.00	900.63	420.00 slower
	Distributed	991.63	113.48	558.34	77.71 slower
NOKIA	Base	1108.00	145.16	612.27	N/A
	Centralized	1064.54	147.36	603.52	8.75 faster
	Overloaded	2012.19	278.53	1140.76	528.49 slower
	Distributed	1338.55	182.90	757.66	144.89 slower

<i>MEGA</i>					
Playbook	Base	533.93	59.04	300.93	N/A
	Centralized	575.93	59.40	329.24	28.31 slower
	Overloaded	1105.20	113.31	631.81	330.88 slower
	Distributed	641.16	67.43	367.34	66.41 slower
iPad3	Base	672.66	66.44	375.17	N/A
	Centralized	674.46	68.24	376.97	1.8 faster
	Overloaded	1281.48	115.33	664.40	289.23 slower
	Distributed	763.75	78.72	427.58	52.41 slower
Transformer	Base	842.00	103.44	478.51	N/A
	Centralized	903.33	105.77	482.38	3.87 slower
	Overloaded	1707.29	199.91	911.69	433.18 slower
	Distributed	1056.74	123.60	564.23	85.72 slower
NOKIA	Base	1117.00	154.16	623.60	N/A
	Centralized	1068.13	150.95	617.89	5.71 faster
	Overloaded	2028.38	286.65	1184.36	560.76 slower
	Distributed	1222.66	177.07	709.39	85.79 slower



# APPENDIX B

## BEST-PROXIMITY EXPERIMENTS



**Figure B.1:** The Proximity Test – North Virginia

**Table B.1:** Breakdown of the Proximity Access (Time in ms) – North Virginia

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	21	7	120.43	56	<b>204.43</b>
	Proxy B (NV)	21	16	121.24	94	252.24
<i>Scenario 2 (Double NV Workload)</i>	Proxy A (NC)	21	7	120.61	54	<b>202.61</b>
	Proxy B (NV)	21	16	174.09	97	308.09
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	21	7	181.98	59	268.98
	Proxy B (NV)	21	16	122.33	94	<b>253.33</b>
<i>Scenario 4 (Triple NC Workload)</i>	Proxy A (NC)	21	7	276.13	61	365.13
	Proxy B (NV)	21	16	121.47	95	<b>253.47</b>

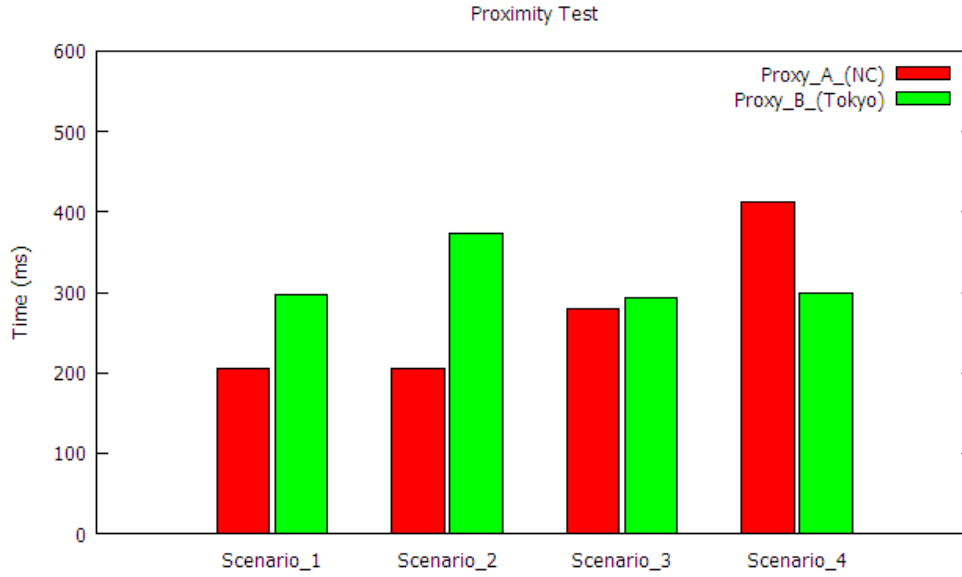


Figure B.2: The Proximity Test – Tokyo

Table B.2: Breakdown of the Proximity Access (Time in ms) – Tokyo

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	21	7	124.19	53	<b>205.19</b>
	Proxy B (Tokyo)	21	34	118.47	120	293.47
<i>Scenario 2 (Double Tokyo Workload)</i>	Proxy A (NC)	21	7	121.33	56	<b>205.33</b>
	Proxy B (Tokyo)	21	34	180.21	136	371.21
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	21	7	192.37	60	<b>280.37</b>
	Proxy B (Tokyo)	21	34	115.87	120	290.87
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	21	7	312.71	72	412.71
	Proxy B (Tokyo)	21	34	119.14	121	<b>295.14</b>

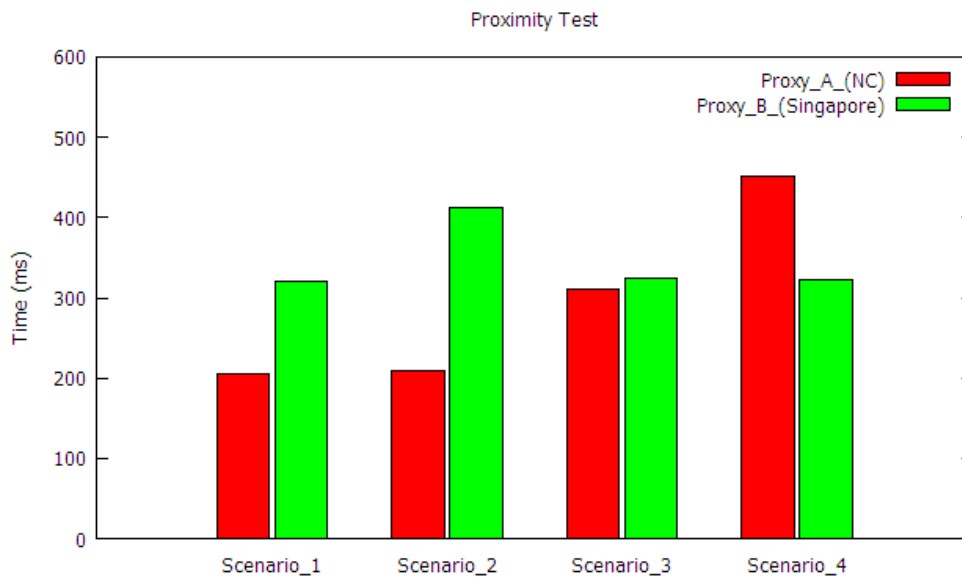
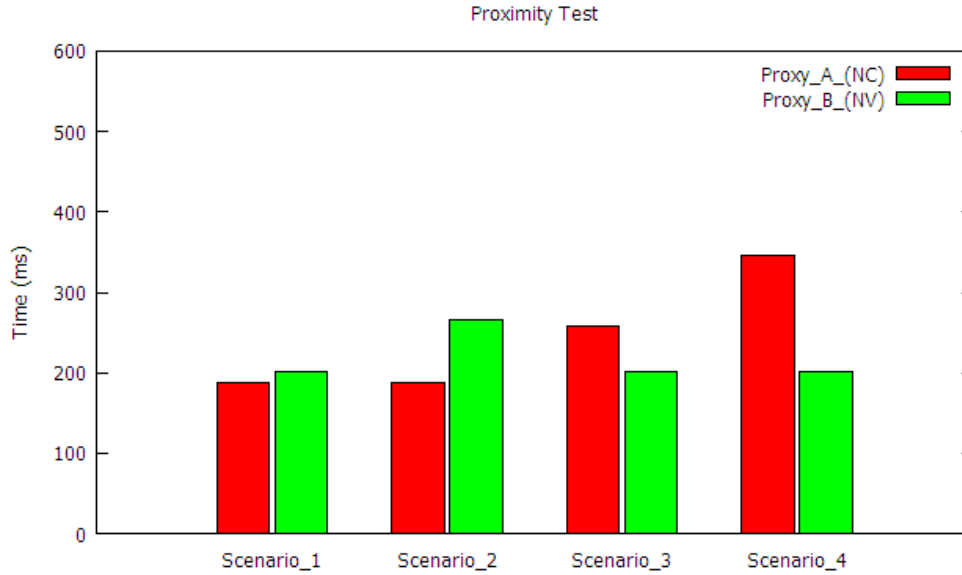


Figure B.3: The Proximity Test – Singapore

**Table B.3:** Breakdown of the Proximity Access (Time in ms) – Singapore

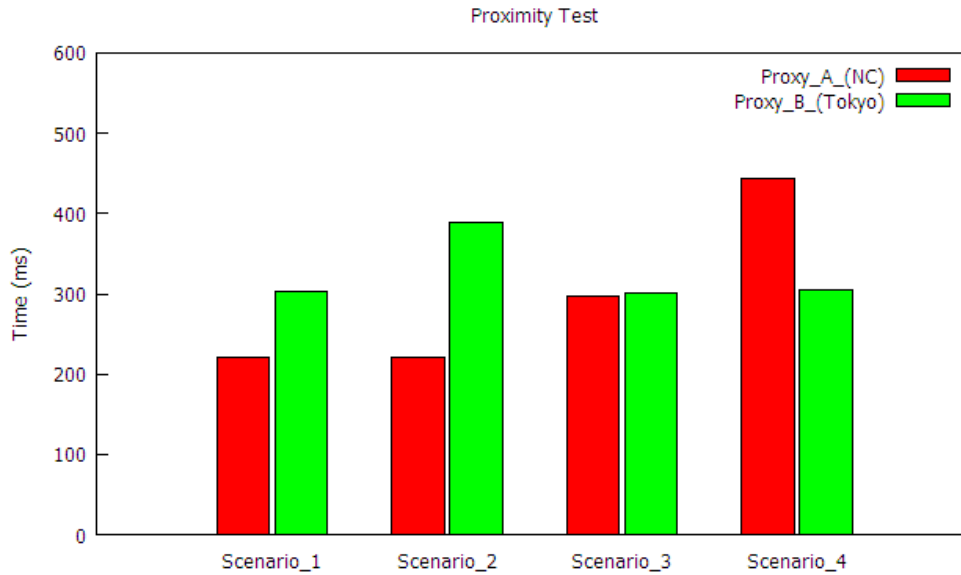
	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	21	7	119.42	57	<b>204.42</b>
	Proxy B (Singapore)	21	40	128.39	131	320.39
<i>Scenario 2 (Double Singapore Workload)</i>	Proxy A (NC)	21	7	122.41	59	<b>209.41</b>
	Proxy B (Singapore)	21	40	212.17	140	413.17
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	21	7	200.46	82	<b>310.46</b>
	Proxy B (Singapore)	21	40	133.67	129	323.67
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	21	7	337.21	87	452.21
	Proxy B (Singapore)	21	40	131.22	130	<b>322.22</b>



**Figure B.4:** The Proximity Test with Controller in North Carolina – North Virginia

**Table B.4:** Breakdown of the Proximity Access (Time in ms) – North Virginia with Controller in North Carolina

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	34	7	122.33	55	<b>187.3</b>
	Proxy B (NV)	34	14	120.99	33	201.99
<i>Scenario 2 (Double NV Workload)</i>	Proxy A (NC)	34	6	121.41	26	<b>187.41</b>
	Proxy B (NV)	34	14	179.44	38	265.44
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	34	6	187.44	31	258.44
	Proxy B (NV)	34	14	120.72	33	<b>201.72</b>
<i>Scenario 4 (Tripple NC Workload)</i>	Proxy A (NC)	34	6	268.41	37	345.41
	Proxy B (NV)	34	14	120.41	33	<b>201.41</b>



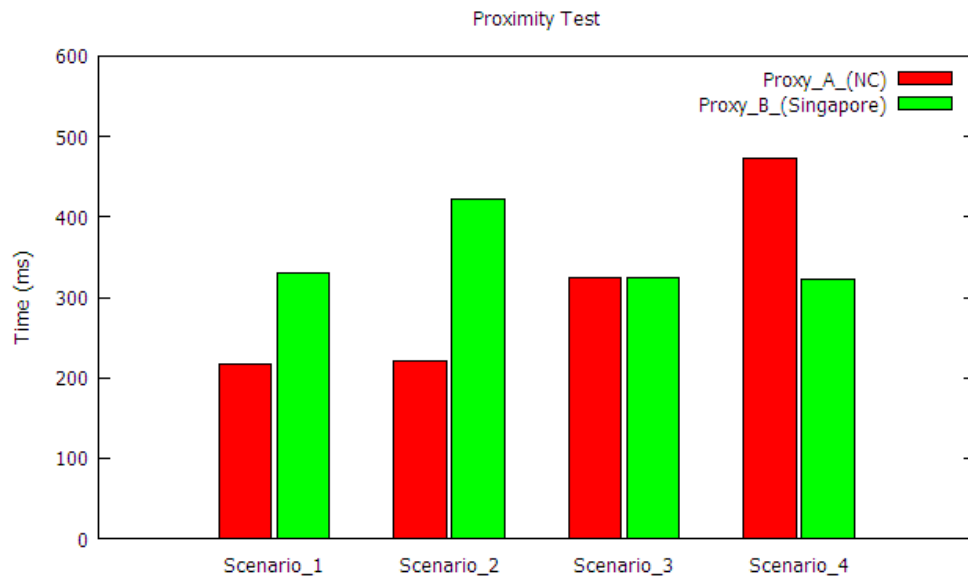
**Figure B.5:** The Proximity Test with Controller in North Carolina – Tokyo

**Table B.5:** Breakdown of the Proximity Access (Time in ms) – Tokyo with Controller in North Carolina

	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	34	6	127.13	53	<b>220.13</b>
	Proxy B (Tokyo)	34	28	120.41	120	302.41
<i>Scenario 2 (Double Tokyo Workload)</i>	Proxy A (NC)	34	6	124.09	56	<b>220.09</b>
	Proxy B (Tokyo)	34	28	191.34	136	389.34
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	34	6	197.43	60	<b>297.43</b>
	Proxy B (Tokyo)	34	28	118.87	120	300.87
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	34	6	330.66	72	442.66
	Proxy B (Tokyo)	34	28	121.13	121	<b>304.13</b>

**Table B.6:** Breakdown of the Proximity Access (Time in ms) – Singapore with Controller in North Carolina

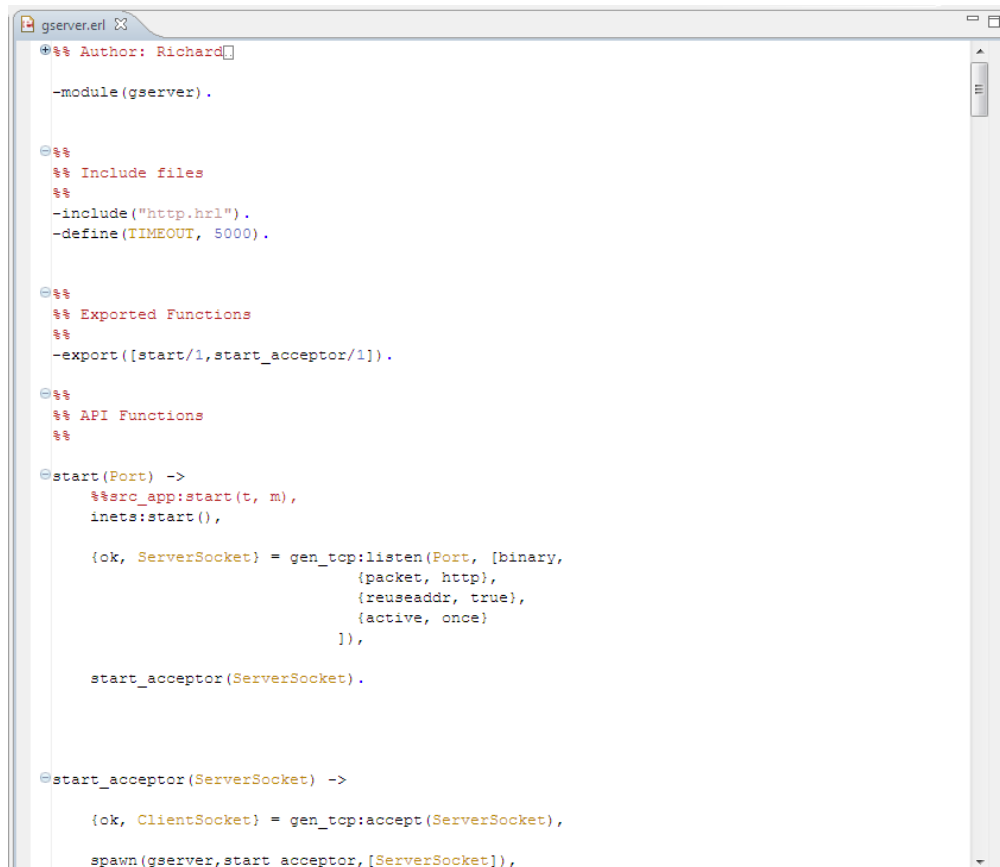
	<i>Proxy</i>	<i>Req<sub>t</sub></i>	<i>Asgn<sub>t</sub></i>	<i>Proc<sub>t</sub></i>	<i>Resp<sub>t</sub></i>	<i>Total</i>
<i>Scenario 1 (Equal Workload)</i>	Proxy A (NC)	34	6	120.54	57	<b>217.54</b>
	Proxy B (Tokyo)	34	38	126.99	131	329.99
<i>Scenario 2 (Double Tokyo Workload)</i>	Proxy A (NC)	34	6	121.43	59	<b>220.43</b>
	Proxy B (Tokyo)	34	38	209.16	140	421.16
<i>Scenario 3 (Double NC Workload)</i>	Proxy A (NC)	34	6	202.18	82	<b>324.18</b>
	Proxy B (Tokyo)	34	38	124.06	129	325.06
<i>Scenario 4 (Quadruple NC Workload)</i>	Proxy A (NC)	34	6	346.18	87	473.18
	Proxy B (Tokyo)	34	38	120.33	130	<b>322.33</b>



**Figure B.6:** The Proximity Test with Controller in North Carolina – Singapore

# APPENDIX C

## ERLANG CODE SNIPPETS OF THE CSB-UCC



```
gserver.erl
%% Author: Richard

-module(gserver).

%%
%% Include files
%%
-include("http.hrl").
-define(TIMEOUT, 5000).

%%
%% Exported Functions
%%
-export([start/1, start_acceptor/1]).

%%
%% API Functions
%%

start(Port) ->
    %%src_app:start(t, m),
    inets:start(),

    {ok, ServerSocket} = gen_tcp:listen(Port, [binary,
                                             {packet, http},
                                             {reuseaddr, true},
                                             {active, once}
                                             ]),

    start_acceptor(ServerSocket).

start_acceptor(ServerSocket) ->
    {ok, ClientSocket} = gen_tcp:accept(ServerSocket),

    spawn(gserver, start_acceptor, [ServerSocket]),
```

Figure C.1: Generic Server (Genserver) Module

```
gserver.erl X
process_request(ClientSocket) ->
    Request_LINE = read_request_line(ClientSocket),
    %% read headers
    Request_HEADERS = read_headers(ClientSocket, Request_LINE),
    %% read body
    Request_BODY = read_body(ClientSocket, Request_HEADERS),
    %%io:write(Request_BODY),
    %%io:write(Request_BODY#http_request.uri),
    URI = atom_to_list(Request_BODY#http_request.uri),
    %%
    %%
    %% [Command|Rest]=Tokenizer,
    io:format(Request_BODY#http_request.body),
    db_req(ClientSocket, comdetect(Request_BODY#http_request.verb,URI,Request_BODY#http_request.bo
    %%{Verb,Path,ContentI,Body) = comdetect(Request_BODY#http_request.verb,URI,Request_BODY#http_
    %%io:format(Path),
    %%
    %% case Command of
    %%     "patientdb"->
    %%         case Rest of
    %%             []->Path="http://128.233.110.216:9090/patientdb/_design/search/_view/all";
    %%             [Patient,"delete"]->
    %%                 %%
    %%                 %% simple_cache:delete("http://128.233.110.216:9090/patientdb/"+++Patient),
    %%                 %% Path="Resource deleted from proxy";
    %%                 %% [Patient]->Path="http://128.233.110.216:9090/patientdb/"+++Patient
    %%             end;
    %%         _->Path="http://128.233.110.216:9090/"+++Command
    %%     %% X=httpc:request(get,{"http://www.cs.usask.ca", [], [], []},
    %%     %% end,
```

Figure C.2: Processing the Request Line

```
gserver.erl X
%%
%% U->
%%   {(_HTTP, _RN, _RT), L, D} = X,
%%   _->{(_HTTP, _RN, _RT), L, D} = Y
%% end,

%% http_write_response(?HTTP_200, ClientSocket, header_to_string(L),D),

%% Get connection
{'Connection', Connection} = get_header('Connection', Request_BODY#http_request.headers),

%% Get version
Version = Request_BODY#http_request.version,

%% Should I continue?
case connection_state(Version,Connection) of

    %% yes
    keep_alive ->

        %% set client socket to read once
        inet:setopts(ClientSocket, [{active,once}, {packet, http}]),

        %% loop
        process_request(ClientSocket);

    %% no
    _other ->
        gen_tcp:close(ClientSocket)
end,
%%after sending the response back to the client, close the socket and, insert the new value
case Flag of
%%   1->simple_cache:insert(Path, Y);
%%   _->
%%       true.
%% end.

%%-----
%% handling the responses including error
```

Figure C.3: Get Connection Details



```

gserver.erl
%%-----
%% handling the responses including error
%%-----

db_req(ClientSocket,{ok,({_HTTP, _RN, _RT}, L, D)}->
    %%http_write_response(?HTTP_200, ClientSocket, header_to_string(L),D);

    %%-----
    %% Encryption Algorithms
    %%-----

    %%Key = <<"abodeefghabodeefgh">>,
    %%IV = <<"12345678abodeefgh">>,

    %%-----AES in Cipher Feedback mode (CFB). IV must be 128 bits (16 bytes)-----

    % Crypt = crypto:aes_cfb_128_encrypt(?Key, ?IV, D),
    % io:format("Crypt is ~p~n", [Crypt]),

    % Decrypt = crypto:aes_cfb_128_decrypt(?Key, ?IV, Crypt),
    % io:format("AES Decrypt is ~p~n", [Decrypt]),
    % http_write_response(?HTTP_200, ClientSocket, header_to_string(L),Crypt):

    %%----- DES in 8-bit CFB mode. Key and IVec must be 64 bits (8 bytes)-----
    Crypt = crypto:des_cfb_encrypt(?Key_DES, ?IV_DES, D),
    io:format("Crypt is ~p~n", [Crypt]),

    % Decrypt = crypto:des_cfb_decrypt(?Key_DES, ?IV_DES, Crypt),
    % io:format("DES Decrypt is ~p~n", [Decrypt]),
    http_write_response(?HTTP_200, ClientSocket, header_to_string(L),Crypt):

    %%-----

db_req(ClientSocket,{error,{connect_failed, _}}->
    http_write_response(?HTTP_404, ClientSocket, "\r\n", "connect_failed");

db_req(ClientSocket,{error,{send_failed, _}}->
    http_write_response(?HTTP_400, ClientSocket, "\r\n", "send_failed");

db_req(ClientSocket,{error,_})->

```

Figure C.4: Managing the Response Including Error

```
gserver.erl x
%%-----
%% RESTful URIs to the database
%%-----

%%{Verb, Path, ContentT, Body}

%%-----Requests to the PlayerDB-----
comdetect('GET', "player/", _) -> http:request(get, {"http://128.233.110.216:9090/patientdb/", []}, [],
%%create request
comdetect('POST', "player/", Body) -> $Decrypt = crypto:aes_ofb_128_decrypt(?Key, ?IV, Body),
    %io:format("Decrypt is ~p~n", [Decrypt]),
    %http:request(post, {"http://128.233.110.216:9090/patientdb/",
    http:request(post, {"http://128.233.110.216:9090/patientdb/",
%%read request
comdetect('GET', "player/" ++ PlayerId, _) -> http:request(get, {"http://128.233.110.216:9090/patientdb/" ++ PlayerId, []}, [],
%%create request
comdetect('POST', "playerunits/", Body) -> http:request(post, {"http://128.233.110.216:9090/patientdb/" ++ PlayerId, Body}, [],
%%read request
comdetect('GET', "playerunits/" ++ UnitId, _) -> http:request(get, {"http://128.233.110.216:9090/patientdb/" ++ PlayerId, UnitId, []}, [],
%%-----
%%-----Requests to the MetricsDB-----
%%create request
comdetect('POST', "playermetrics/", Body) -> http:request(post, {"http://128.233.110.216:9090/patientdb/" ++ PlayerId, Body}, [],
%%read request
comdetect('GET', "playermetrics/" ++ PlayerId, _) -> http:request(get, {"http://128.233.110.216:9090/patientdb/" ++ PlayerId, "playermetrics/" ++ PlayerId, []}, [],
%%read request
comdetect('GET', "gamemetrics/", _) -> http:request(get, {"http://128.233.110.216:9090/patientdb/" ++ PlayerId, "gamemetrics/" ++ PlayerId, []}, [],
%%read request
```

Figure C.5: REST URIs

```
gserver.erl x
%%-----Function Calls-----
%%-----

read_request_line(ClientSocket) ->

%% set client socket to read once
inet:setopts(ClientSocket, [{active,once}, {packet, http}],

%% process data
receive

%% will only deal with http requests
{http, ClientSocket, {http_request, Method, OPath, Version}} ->

%% get path in usable format
Path = get_path(OPath),

%% Split path into URI & ArgumentString
{URI, ArgumentString} = split_path(Path),

%% create tuple list of arguments -> [{arg,val}, ...]
Arguments = create_argument_tuples(ArgumentString),

%% put the data we already have into a request record
Request = #http_request{
    senderPID = self(),

    uri = uri_to_atom(URI),
    arguments = Arguments,

    verb = Method,
    path = Path,
    version = Version
},

Request;

_Else ->
```

Figure C.6: Sample Function Calls

```
gserver.erl X
%Response = "HTTP/1.1 200 OK\r\n" ++ Headers ++ "Content-Length: " ++ erlang:integer_to_list
Response = "HTTP/1.1 " ++ Code++ "\r\n" ++ Headers ++ Data,
gen_tcp:send(Client_Socket, Response).

get_header(Name, []) ->
{Name,undefined};

get_header(Name, [{Name, Value} | _]) ->
{Name,Value};

get_header(Name, [_|List]) ->
get_header(Name, List).

header_to_string(Headers) -> header_to_string(Headers, []).
header_to_string([],Headers) -> Headers++"\r\n";
header_to_string([{Name, Value}|Headers], String) -> header_to_string(Headers, String++Name++": "

%% -----
%%
%%
%% CONNECTION_STATE/1
%%
%% Request: The parsed data of the request
%%
%% ->      close | keep_alive

connection_state({0,9},_) -> close;
connection_state({1,0}, 'Keep-Alive') -> keep_alive;
connection_state({1,0}, _) -> close;
connection_state({1,1}, 'Keep-Alive') -> keep_alive;
connection_state({1,1}, 'keep-alive') -> keep_alive;
connection_state({1,1}, _) -> close;
connection_state(_, _) -> close.

%%
%% Local Functions
%%
```

Figure C.7: Connection States

```
gserver.erl X
%%
%% SPLIT_PATH/1
%%
%% Look for ? and split in before (URI) and after (argument string)
%%
%% Request: The parsed data of the request
%%
%% ->      close | keep_alive

@split_path([47|Path]) ->
    split_path(Path, []);

@split_path(Path) ->
    split_path(Path, []).

@split_path([$?|Arguments], Path) ->
    {lists:reverse(Path), Arguments};

@split_path([H|T], Path) ->
    split_path(T, [H|Path]);

@split_path([], Path) ->
    {lists:reverse(Path), []}.

@uri_to_atom([]) ->
    {error};

@uri_to_atom([47 | _]) ->
    {error};

@uri_to_atom(URI) ->
    list_to_atom(URI).

%% -----
```

Figure C.8: Processing the Requests by Splitting Paths

## APPENDIX D

# THE HAEMOPHILIA INJURY RECOGNITION TOOL (HIRT?)

## APP

### D.1 Overview of HIRT?

This section reproduces the earlier published work by Lomotey et al. [154] [247]. The importance of the section is to highlight the research of the mobile assessment guide in conjunction with the CSB-UCC; and further justify the results presented earlier in the thesis.

### D.2 Awards and Recognition

- Bayer Hemophilia Awards Program (BHAP)
- MITACS Funded in three consecutive Awards
- Showcasing at the World Federation of Hemophilia Conference in Melbourne, Australia
- Featured in the Saskatoon Region Reporter, The StarPhoenix, and CBC News
- Won the BHAP Alumni Global Webinar vote
- Won the Connected to the Community awards (*here: <http://cwta.ca/about-cwta/connected-to-the-community/>*) by the Canadian Wireless Telecommunications Association (CWTA)

### D.3 Overall Scope of the Project

People with Haemophilia are persons (mostly men) who have a blood disorder that prevents blood clotting during injury. The condition can be classified into varying degrees depending on the blood clotting factor such as mild, moderate, or severe.

Across Canada, physiotherapists had seen situations where young men with mild haemophilia waited too long to come in for treatment after an injury. The clinicians and the physiotherapists (i.e., the comprehensive care colleagues) had also noticed this. The question therefore is why this was happening? During interactions with some physiotherapists, there were suggestions for the provision of new booklets to people with mild haemophilia; but, we wondered if that was the best approach. In order to arrive at the best possible solution to provide quality information to the young men with the condition, we followed the following steps.

#### **Understanding the Issue**

In the initial work by Nilson et al. [248], young Canadian men (18–30 years old) with mild haemophilia were interviewed to learn more about their knowledge, attitudes and behaviours. Meetings were also held with the focus groups (i.e., physiotherapists) across Canada. This aided the research group to study what the underlying issues were.

#### **What are the Issues**

From the study of the evidence in step 1, it was realised that paper-based booklets will not just be used by the target group but they may not be considered relevant.

#### **Decision Pathway**

Several workflows are developed based on the interviews with the young men with mild hemophilia.

#### **The Results**

Clearly, there was a need to adopt a more modern approach to aid the young men with mild hemophilia to assess minor injuries when they occur.

## **The Way Forward**

The answer to the final question in step 4 carries a lot of weight. It is one thing developing a mobile application but it is another thing building the application to provide a better option than a booklet. The mobile app must be informative enough to encourage the young men with hemophilia to report the injury to the physiotherapists. Besides, the mobile specific challenges such as intermittent loss of connectivity in distributed architectures must be overcome. Thus, collaboration with computer scientists at the University of Saskatchewan is formed with members of the Canadian Physiotherapists in Haemophilia Care (CPHC) to help develop an electronic and convenient mobile app. This collaboration is sponsored by Bayer Haemophilia Award Program (BHAP) through the Bayer Caregiver Award, and MITACS Accelerate Programme. The partnership has led to the research and development of the Haemophilia Injury Recognition Tool (HIRT).

From the interview conducted for the 25+ young men with mild haemophilia, the message was clear. The young men we spoke with felt that HTC staff "over-reacted" when they came in with an injury. They also felt that most of the existing patient education information did not apply to them. This evidence-based self-management tool helps young men with mild haemophilia assess an injury and decide when to seek medical attention.

It is also important to state that there is a reason why the App is emphasizing on "young men". Our research included young men 18-30(5) years old; therefore, our evidence on these knowledge gaps included this group. The App addresses the issues specifically for this age group. This entire group had haemophilia A or B and were of the Mild form of haemophilia which is 5-50 percent of the clotting factor (which they have a deficit of) circulating in their blood. This study was a Canadian study with participants from different provinces.

## **The Application Process Flow**

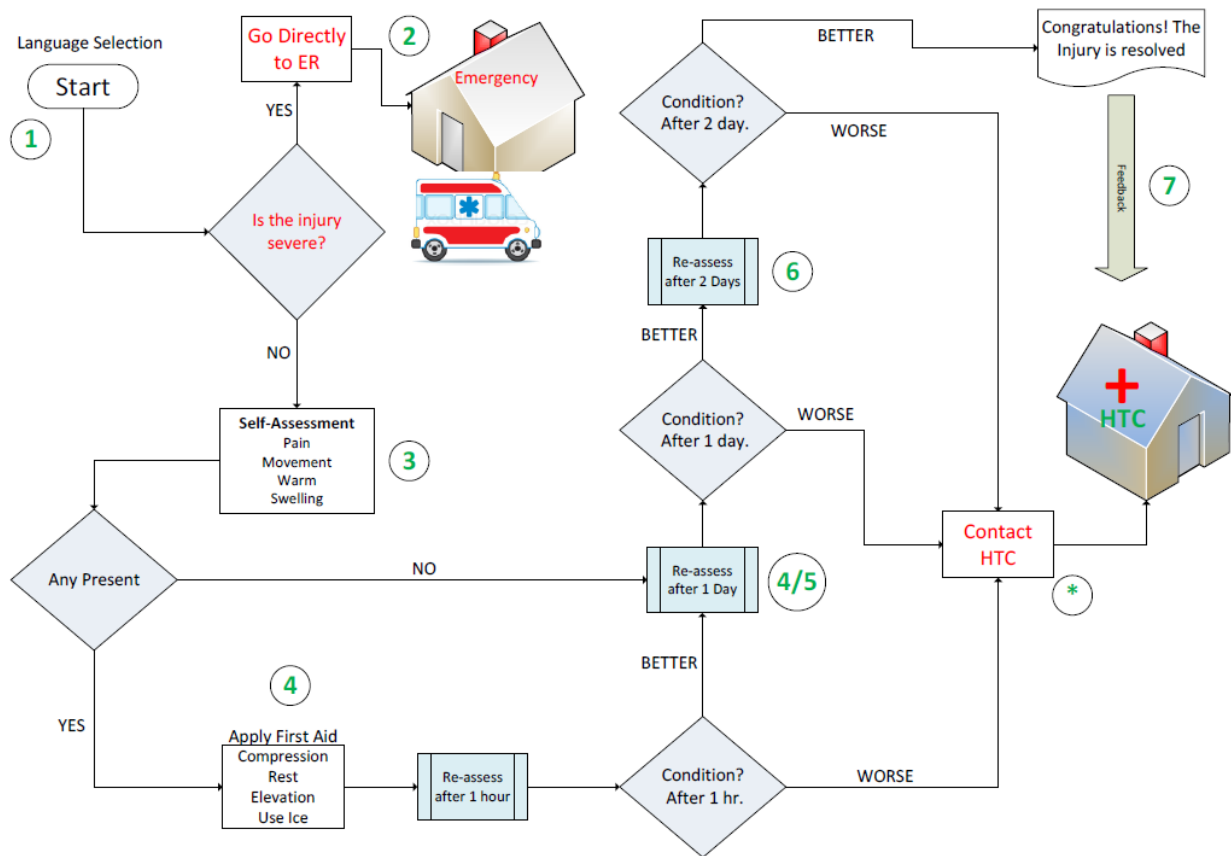
The application process flow that guides the user through steps of activities is illustrated in Fig. D.1. The users commence from Start as the number 1 step where they are guided to select English or French. The user is then taken to the next screen which displays a message asking whether the injury is severe to the head, neck, or abdomen. Injuries to these areas of the body are potentially fatal and must not be ignored. For these injuries, the user is directed to go to the nearest Emergency Room (ER) as the number 2 step. For other injuries of a musculoskeletal nature (i.e., muscle or joint injuries the user is encouraged to use HIRT?. The next step contains information on how to perform the initial self-assessment number 3 step. The self-assessment includes:

- Looking for pain at the injury site, pain at rest, and pain when moving the injured limb or putting weight through the limb,
- Loss of movement,
- Warmth to touch, and
- Swelling at injury site.

In the application, drop-down boxes provide additional details on how to examine any of the symptoms listed. If none of the condition listed above is present, the user will be reminded in 24 hours to do the self-assess again as step 4. If any of the conditions are present, then the user is taken to the next level to apply first aid, this is also part of step 4. The application of first aid is also series of activities which are summarized below:

- Compression: Use tensor bandage or elastic sleeves,
- Rest: Stop activity such as walking, sports, use sling etc.,
- Elevation: keep the injured limb above the level of your heart, and
- Ice: Use crushed ice or gel pack.

After this, the young man will be reminded to re-assess the injury after 1 hour where the step three actions will be repeated. If the condition is getting worse, the user is advised to contact the Hemophilia Treatment Center (HTC). If the condition is getting better, the YMWMH is reminded after 24 hours to



**Figure D.1: Process Flow**

repeat the self-assessment; this is labeled as step 5. After 24 hours, if the condition of the injury is worse, the user is advised to go to the HTC. If there is improvement, the user is reminded to re-assess the injury in 2 days from the previous assessment; this is step 6. If the condition of the injury is worse than the last assessment, the user is advised to go to the HTC. Otherwise we congratulate the user since it appears the injury is resolved.

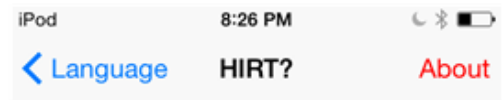
The application contains list of HTCs throughout Canada. The list is organized by province so the users can locate the nearest facility. The HTC list contains phone numbers, and, the users are enabled to dial directly from the application. At the end of the self-assessment, the users are reminded to provide feedback. However, the feedback can be provided anytime during the use of the application; not necessarily when the assessment is over. Thus, the users have quick links screen to provide feedback or dial HTC.

The screen flow of the Haemophilia Injury Recognition Tool (HIRT?) is presented in Figs. D.2, D.3, D.4, and D.5.



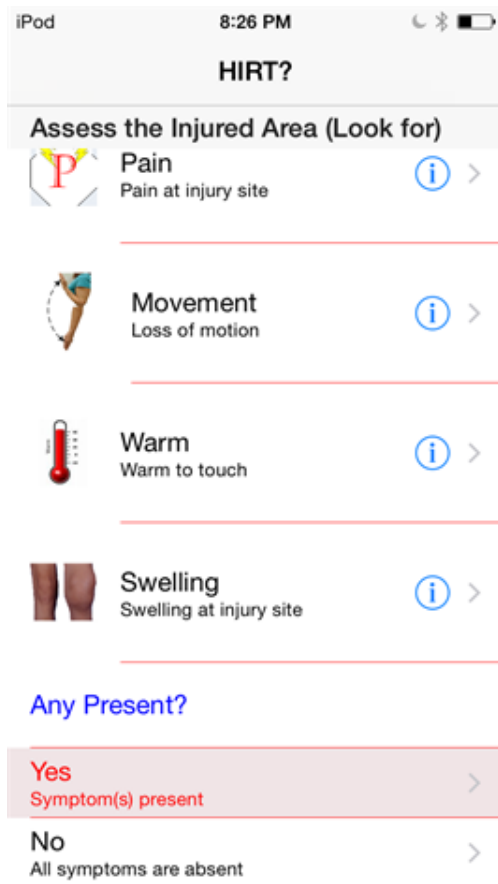


Screen to choose  
Language

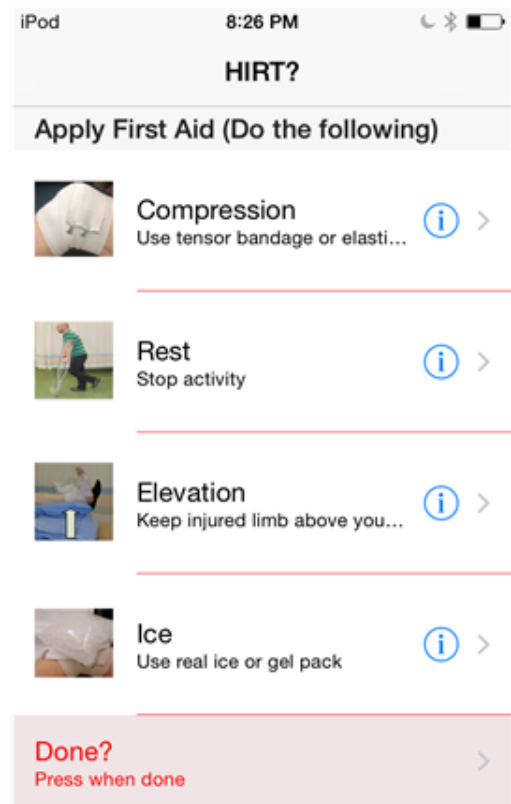
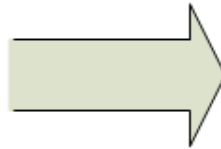


Welcome screen

Figure D.2: Language and Welcome Screen

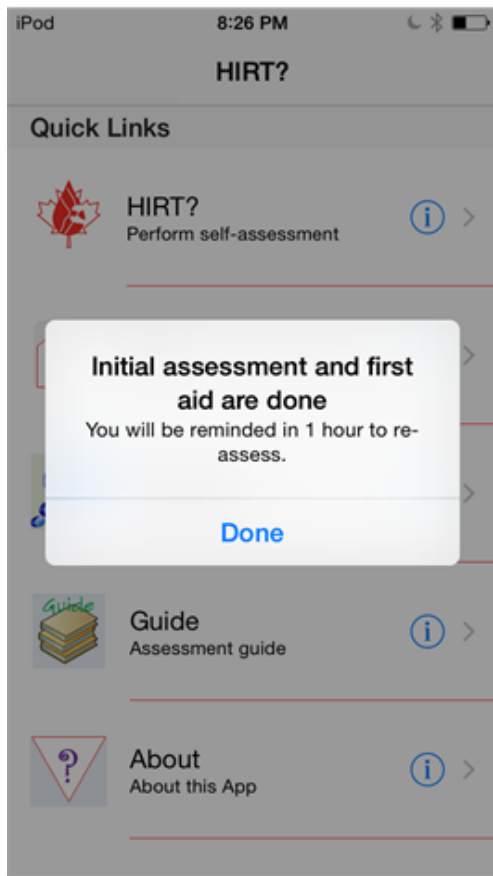


Initial Assessment  
Screen



First Aid Screen

Figure D.3: Initial Assessment and First Aid Screens

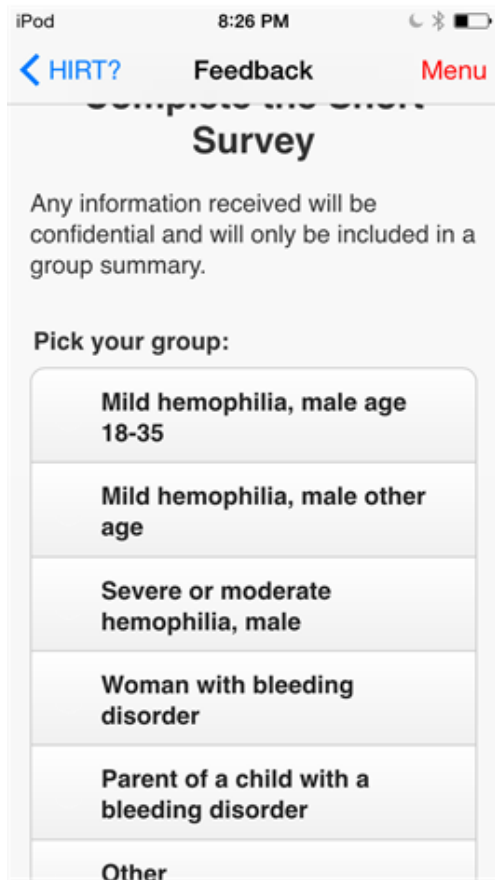


Notice to be Reminded

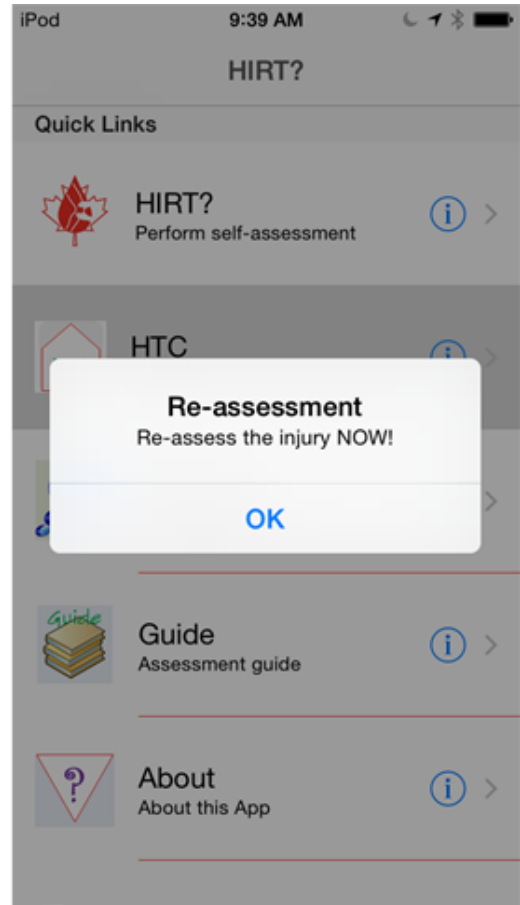
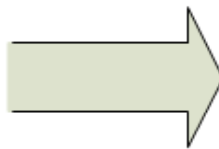


Detailed Assessment Guide

Figure D.4: Notice to be Reminded and Guide



Survey Form

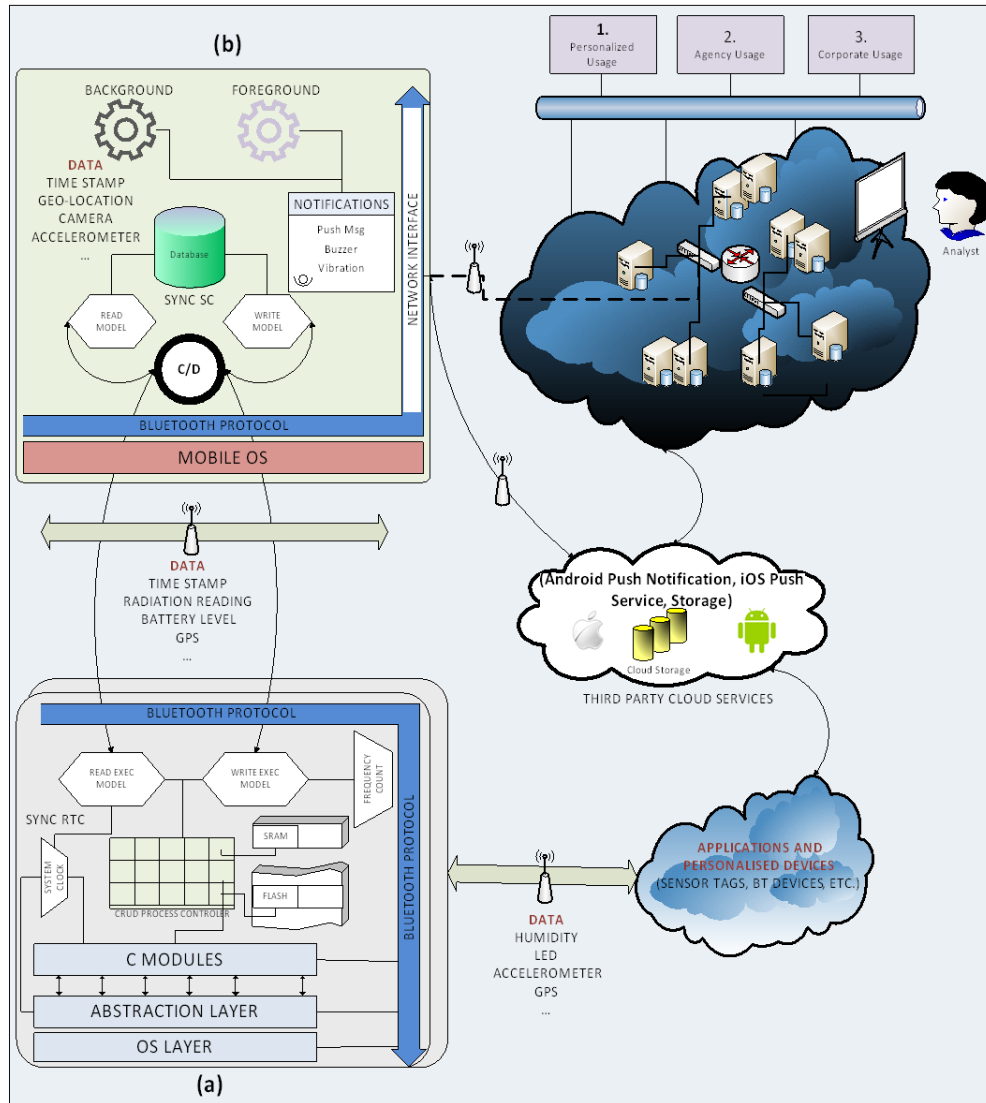


Re-assessment  
Reminder

Figure D.5: Survey Form and Re-Assessment Reminder

# APPENDIX E

## COMPONENTS OF THE MOBILE-SENSOR ARCHITECTURE



**Figure E.1:** Component Architecture of the Mobile-Sensor Environment

### E.1 Sensor Device

There are specific gamma radiation sensor devices such as the one built by the Canadian Environment Instrument, the CT007. This is labelled as (a) in Figure E.1. The sensor device provides the following data: gamma radiation readings, battery level, timestamps of readings, and GPS location. The architecture (in Fig. E.1) provides a good view of the software layer and components. The primary means of communicating with the sensor device is via Bluetooth. Thus, a Bluetooth protocol is built on the OS Layer of the device to enable communication with other devices. On top of the OS layer is the abstraction layer that contains

several software abstractions.

These include system clock, accelerometer, and radiation engine. To use these abstractions, software modules are provided in C programming language. These C modules can be called just as software delegates to access a particular component of the sensor device.

There are two major operations that can be performed on the sensor which are determined by the READ EXEC MODEL and the WRITE EXEC MODEL. The former facilitates the readings of the data from the sensor device by the mobile while the latter enables data to be written to the sensor device. The device has a CRUD PROCESS CONTROLLER which coordinates the request from the mobile or other sources on the type of operation. The CRUD operations are defined as follows: **C**– create new data, **R**– read existing data, **U**– update existing data, and **D**– remove existing data. There are two storage components of the device which are SRAM and FLASH.

We rely on the SYSTEM CLOCK to timestamp the radiation readings. Furthermore, the device has a FREQUENCY COUNT that determines at what intervals radiation requests should be pushed to the mobile. A practical challenge is the determination of the time of readings if the mobile host and the sensor device have different time settings. This issue is handled by the SYNC RTC component. This component synchronizes the real-time clock (RTC) using the timestamp from the perspective of both the mobile host and the sensor device.

## E.2 The Mobile Hosts

The mobile hosts are the smartphone endpoints for the services provisioning. If mobile host, say AH, has the latest reading of data from a near-by sensor device and another mobile device does not have that data, then AH becomes a host and all the other mobile devices from say AH+1, ..., AN become consumers. The main components of the mobile hosts are labelled as (b) in Fig. E.1. For brevity, the implementation details of mobile hosting are omitted.

The sensor data reading on the mobile is done through our implemented mobile client view app. The app runs on the mobile OS and uses multi network interface. The Bluetooth protocol is employed to connect with the sensor device (e.g., CT007) and the Wi-Fi/4G is for interactions with the cloud computing components. The mobile host has a component for detecting connected and disconnected sensor devices (i.e., C/D). When there is disconnect, the mobile re-initiates the connection with the sensors. We ensure that no data is lost within this period since the sensor device will write readings to its memory when the application disconnects.

Furthermore, the mobile app also has the READ MODEL and the WRITE MODEL to ensure that sensor information is retrieved from the mobile database or written to.

The data in the mobile database is synchronized by the SYNC SC (system clock) component. This is to ensure that outdated readings are not confused with new readings in the analysis of the gamma radiation level. Apart from the specific radiation readings from the sensor, there are other readings that are collected from the mobile which otherwise are not derived from the sensor. The mobile specific data that we collect are the geolocation data, camera information, accelerometer readings, and timestamps of such data. Notification services are also integrated to inform the user of high radiations through push notification, buzzer, and vibration.

This message tells the user that the gamma radiation level at the current location is too high. The app works in both the background and foreground modes. This means users can always be informed whether they are actively using the app or not.

## E.3 Cloud Services

### E.3.1 Applications and Personal Devices

Apart from the sensor which reads gamma radiations, the system is designed to interact with other types of sensor devices to read the device specific information. Example is the SensorTag to enable us capture the following data: humidity, LED, accelerometer, and GPS.

### **E.3.2 Push Notification**

These are third party services from Google and Apple that allow us to send messages to the mobile notification center. This enables the user to access the necessary information in real time.

### **E.3.3 Analysts**

This is a back-end layer provided for specific domain users. There are three potential users of sensor devices: personal, agency, and corporate. Personal usage is when the users store history of the sensor readings over a period of time to determine the various places that have certain levels of gamma radiation. This can be within a building or a particular location. Agencies such as homeland security can also use this information from several of their users to perform other tasks based on the analysis. Other corporations (e.g., mining sector) can use the collected information to advise employees. The location where the information is stored is determined by the enterprise that is using the app.

## APPENDIX F

# MY PEER-REVIEWED PUBLICATIONS WITH CONTENTS FROM THIS DISSERTATION

*No. of Book Chapters (BKChp) : 1*

*No. of Journals (J) : 16*

*No. of Conferences (C) : 35*

*Total : 52*

52. BK Chp52: **Lomotey, R. K.**, and Deters, R. 2014. Unstructured Data, NoSQL, and Terms Analytics. Big Data Applications and Use Cases. Springer Book, 2014
51. J51: **Lomotey, R. K.**, and Deters, R. 2015. MUBaaS: Mobile Ubiquitous Brokerage as a Service. World Wide Web (WWW) Journal, 15 Pages, February 2015
50. J50: **Lomotey, R. K.**, and Deters, R. 2015. Mobile Services Provisioning and Sensor Data Sharing. International Journal of Wireless and Mobile Computing, To Appear
49. J49: Pham, S., **Lomotey, R. K.**, Fu, W., and Deters, R. 2015. Peer-to-Peer Mobile Data Flow in a Crop Field. International Journal of Business Process Integration and Management, To Appear
48. J48: **Lomotey, R. K.**, and Deters, R. 2015. Unstructured Data Mining: Use Case for CouchDB. International Journal of Big Data Intelligence, To Appear
47. J47: **Lomotey, R. K.**, and Deters, R. 2015. Terms Analytics Service for CouchDB: A Document-based NoSQL. International Journal of Big Data Intelligence, To Appear
46. J46: **Lomotey, R. K.**, and Deters, R. 2014. Data Mining From Document-Append NoSQL, International Journal of Services Computing (IJSC), 2(2), 2014, pp. 17-29.
45. J45: **Lomotey, R. K.**, and Deters, R. 2014. Exploring Different Architectures to Support Crop Farmers with a Mobile Application on Pesticide Control. International Journal of Cloud Computing and Services Science (IJ-CLOSER), volume 3, number 3, pages 122-132, 2014.
44. J44: **Lomotey, R. K.**, Mulder, K., Nilson, J., Schachter, C., Wittmeier, K., Deters, R. 2014. Mobile Aid for Haemophilia Injury Recognition. International Journal of Biomedical Engineering Research (BER), Dec. 2014, Vol. 3 Iss. 4, PP. 96-107.
43. J43: **Lomotey, R. K.**, Mulder, K., Nilson, J., Schachter, C., Wittmeier, K., Deters, R. 2014. Mobile self-management guide for young men with mild hemophilia in cases of minor injuries. International Journal of Network Modeling Analysis in Health Informatics and Bioinformatics, Volume 3, Issue 1 (NetMAHIB), P 1-12, July 2014, 10.1007/s13721-014-0064-z, Publisher: Springer Vienna.
42. J42: **Lomotey, R. K.**, and Deters, R. 2014. Efficient Consumption of the Electronic Health Record in mHealth. International Journal of Network Modeling Analysis in Health Informatics and Bioinformatics (NetMAHIB), P 1-12, February 2014, DOI: 10.1007/s13721-014-0051-4, Print ISSN: 2192-6662, Publisher: Springer Vienna.
41. J41: **Lomotey, R. K.**, and Deters, R. 2014. N-Screen Provenance-Based Security Enforcement in mHealth. International Journal of Biomedical Engineering Research (BER), June 2014, Vol. 3 Iss. 2, PP. 37-46.



40. J40: **Lomotey, R. K.**, and Deters, R. 2014. Synchronization of Medical Data in a Mobile Provisioning Environment: mHealth Use Case. *International Journal of Biomedical Engineering Research (BER)*, Volume 3, Issue 1, Page 1-10, March 14, DOI:10.5963/BER0301001.
39. J39: **Lomotey, R. K.**, and Deters, R. 2014. Analytics-as-a-Service Framework for Terms Association Mining in Unstructured Data. *International Journal of Business Process Integration and Management (IJBPIIM)*, 2014, Vol. 7, No. 1, 2014 , pp.49-61.
38. J38: **Lomotey, R. K.**, and Deters, R. 2013. RSender: Terms Mining Tool from Unstructured Data Sources. *International Journal of Business Process Integration and Management (IJBPIIM)*, 2013 Vol.6, No.4, pp.298 - 311, DOI: 10.1504/IJBPIIM.2013.059136.
37. J37: **Lomotey, R. K.**, and Deters, R. 2013. Using a Cloud-Centric Middleware to Enable Mobile Hosting of Web Services: mHealth Use Case. *The Personal and Ubiquitous Computing (PUC)*, page 1-14, *Journal of Personal and Ubiquitous Computing*, October 2013.
36. J36: **Lomotey, R. K.**, and Deters, R. 2013. Facilitating Multi-Device Usage in mHealth. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, volume: 4, number: 2, pp. 77-96, June 2013
35. C35: Steven Xue, **Lomotey, R. K.**, and Deters, R. 2015. Enabling Sensor Data Exchanges in Unstable Mobile Architectures. *Proc. of the 2015 IEEE International Conference on Mobile Services (IEEE MS 2015)*, 8 Pages, June 27-July 03, 2015 - New York, NY, USA
34. C34: **Lomotey, R. K.**, Sinh Pham, Wen Fu, and Deters, R. 2015. P2P Key-Value Storage Synchronization Workflow for Agronomic Data Management. *Proc. of the 2015 IEEE International Conference on Mobile Services (IEEE MS 2015)*, 8 Pages, June 27-July 03, 2015 - New York, NY, USA
33. C33: **Lomotey, R. K.**, and Deters, R. 2015. Sensor Data Propagation in Mobile Hosting Networks. *Proc. of the 2015 IEEE International Symposium on Service Oriented and System Engineering*, March 30 - April 3 2015 San Francisco, CA, USA
32. C32: **Lomotey, R. K.**, and Deters, R. 2015. Mobile Hosting and Sensor Eco-System for Gamma Radiation Detection. *Proc. of the 2015 IEEE International Systems Conference*, April 13-April 16, 2015 Vancouver, BC, Canada
31. C31: **Lomotey, R. K.**, and Deters, R. 2014. Data Mining from NoSQL Document-Append Style Storages. *Proc. of the 2014 IEEE International Conference on Web Services (ICWS 2014)*, Pages 385-392, June 27-July 02, 2014 - Anchorage, Alaska, USA
30. C30: **Lomotey, R. K.**, and Deters, R. 2014. Terms Mining in Document-Based NoSQL: Response to Unstructured Data. *Proc. of the 2014 IEEE International Congress on Big Data (BigData 2014)*, Pages 661-668, June 27-July 02, 2014 - Anchorage, Alaska, USA
29. C29: **Lomotey, R. K.**, and Deters, R. 2014. Architectural Designs from Mobile Cloud Computing to Ubiquitous Cloud Computing - Survey. *Proc. of the 2014 IEEE 10th World Congress on Services (IEEE SERVICES 2014)*, Pages 418-425, June 27-July 02, 2014 - Anchorage, Alaska, USA
28. C28: **Lomotey, R. K.**, and Deters, R. 2014. Intrusion Prevention in Asterisk-based Telephony System. *Proc. of the 2014 IEEE International Conference on Mobile Services (IEEE MS 2014)*, Pages 100-107, June 27-July 02, 2014 - Anchorage, Alaska, USA
27. C27: **Lomotey, R. K.**, and Deters, R. 2014. Management of Mobile Data in a Crop Field. *Proc. of the 2014 IEEE International Conference on Mobile Services (IEEE MS 2014)*, Pages 100-107, June 27-July 02, 2014 - Anchorage, Alaska, USA
26. C26: **Lomotey, R. K.**, and Deters, R. 2014. Towards Knowledge Discovery in Big Data. *Proc. of the 8th IEEE International Symposium on Service-Oriented System Engineering (IEEE SOSE 2014)*, Pages 11, April 8-11, 2014 - Oxford, UK

25. C25: **Lomotey, R. K.**, and Deters, R. 2014. Mobile-Based Medical Data Accessibility in mHealth. Proc. of the 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (IEEE MobileCloud 2014). pages: , April 8-11, 2014 - Oxford, UK.
24. C24: **Lomotey, R. K.**, and Deters, R. 2014. Analytics-as-a-Service (AaaS) Tool for Unstructured Data Mining. Proc. of the 2014 IEEE International Conference on Cloud Engineering (IC2E 14). pages: 6, March 10-14, 2014. Boston, Massachusetts, USA.
23. C23: **Lomotey, R. K.**, Chai, Y., Jamal, S., and Deters, R. 2013. MobiCrop: Supporting Crop Farmers with a Cloud-Enabled Mobile App. Proc. of the 2013 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013), pp:8 pages, Kauai, Hawaii, USA, December 16-18, 2013. *Best Paper Runner-Up*
22. C22: **Lomotey, R. K.**, and Deters, R. 2013. Composition of the Electronic Health Record: Mobile Efficiency in mHealth. Proc. of the 2013 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013), pp:8 pages, Kauai, Hawaii, USA, December 16-18, 2013
21. C21: **Lomotey, R. K.**, and Deters, R. 2013. Middleware-Enabled Mobile Framework in mHealth. Proc. of the 6th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2013), pp:8 pages, Dresden, Germany, December 9-12, 2013
20. C20: **Lomotey, R. K.**, Yiding Chai, Kazi Ashik Ahmed, and Deters, R. 2013. Web Services Mobile Application for Geographically Dispersed Crop Farmers. Proc. of the IEEE 16th International Conference on Computational Science and Engineering (CSE 2013), pp:8 pages, Sydney, Australia Dec. 3-5, 2013.
19. C19: **Lomotey, R. K.**, and Deters, R. 2013. Topics and Terms Mining in Unstructured Data Stores. Proc. of the 2nd International Conference on Big Data Science and Engineering (BDSE 2013), pp:8 pages, Sydney, Australia Dec. 3-5, 2013. *Best Paper Award*
18. C18: **Lomotey, R. K.**, and Deters, R. 2013. Consuming Web Services on Mobile Devices for Improved mHealth. Proc. of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2013), pp:4 pages, Orlando, Florida, USA, November 5-8, 2013 (Short Paper)
17. C17: **Lomotey, R. K.**, and Deters, R. 2013. CSB-UCC: Cloud Services Brokerage for Ubiquitous Cloud Computing. Proc. of the ACM International Conference on Management of Emergent Digital EcoSystems (MEDES 13), pp:5 pages, Neumnster Abbey, Luxembourg, October 28th till October 31th, 2013
16. C16: **Lomotey, R. K.**, and Deters, R. 2013. Reliable Services Composition for Mobile Consumption in mHealth. Proc. of the ACM International Conference on Management of Emergent Digital EcoSystems (MEDES 13), pp:5 pages, Neumnster Abbey, Luxembourg, October 28th till October 31th, 2013
15. C15: **Lomotey, R. K.**, Yiding Chai, Kazi Ashik Ahmed, and Deters, R. 2013. Distributed Mobile Application for Crop Farmers. Proc. of the ACM International Conference on Management of Emergent Digital EcoSystems (MEDES 13), pp:5 pages, Neumnster Abbey, Luxembourg, October 28th till October 31th, 2013
14. C14: **Lomotey, R. K.**, and Deters, R. 2013. Supporting N-Screen Medical Data Access in mHealth. Proc. of the IEEE International Conference on Healthcare Informatics 2013 (ICHI 2013), pp:8 pages, Philadelphia, PA, USA, September 9-11, 2013
13. C13: **Lomotey, R. K.**, and Deters, R. 2013. Efficient Mobile Services Consumption in mHealth. Proc. of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), and the International Symposium on Network Enabled Health Informatics, Biomedicine and Bioinformatics (HI-BI-BI 2013), pp:8 pages, Niagara Falls, Canada, August 25-28, 2013. *Best Paper Award*

12. C12: **Lomotey, R. K.**, and Deters, R. 2013. Real-Time Effective Framework for Unstructured Data Mining. Proc. of the 11th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-13), 16-18 July 2013, Melbourne, Australia.
11. C11: **Lomotey, R. K.**, and Deters, R. 2013. Unstructured Data Extraction in Distributed NoSQL. Proc. of the 7th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST-CEE 2013), pp:6 pages, California, USA, 24-26 July 2013
10. C10: **Lomotey, R. K.**, and Deters, R. 2013. RSender: Tool for Topics and Terms Extraction from Unstructured Data Debris. Proc. of the 2013 IEEE International Congress on Big Data, pp:395-402, Santa Clara, California, 27 June-2 July 2013
9. C9: **Lomotey, R. K.**, and Deters, R. 2013. SaaS Authentication Middleware for Mobile Consumers of IaaS Cloud. Proc. of the 2013 IEEE Ninth World Congress on Services, pp:448-455, Santa Clara, California, 27 June-2 July 2013
8. C8: **Lomotey, R. K.**, and Deters, R. 2013. Energy Consumption Analysis in Ubiquitous Cloud Computing. Proc. of the 2013 IEEE Second International Conference on Mobile Services (MS 2013), pp:95-102, Santa Clara, California, 27 June-2 July 2013
7. C7: **Lomotey, R. K.**, and Deters, R. 2013. Cloud Services Brokerage for Ubiquitous Cloud Computing. Proc. of the 2013 IEEE Second International Conference on Mobile Services (MS 2013), pp:70-77, Santa Clara, California, 27 June-2 July 2013
6. C6: **Lomotey, R. K.**, and Deters, R. 2013. Terms Extraction from Unstructured Data Silos. Proc. of the 8th IEEE International Conference on System of Systems Engineering (SoSE), pp.19-24, 2-6 June 2013, Maui, Hawaii, USA.
5. C5: **Lomotey, R. K.**, and Deters, R. 2013. Middleware-Layer for Authenticating Mobile Consumers of Amazon S3 Data. Proc. of the IEEE International Conference on Cloud Engineering (IC2E). pages: 108-113, March 25-28, 2013. San Francisco, California, USA
4. C4: **Lomotey, R. K.**, and Deters, R. 2013. Reliable Consumption of Web Services in a Mobile-Cloud Ecosystem Using REST. Proc. of the 7th IEEE International Symposium on Service Oriented System Engineering (SOSE). pages: 13-24 March 25-28, 2013. San Francisco, California, USA
3. C3: **Lomotey, R. K.**, Kazi, R., and Deters, R. 2012. Near Real-Time Medical Data Dissemination in m-Health. Proc. of the International ACM Conference on Management of Emergent Digital EcoSystems (MEDES), pp:67-74, Addis Ababa, Ethiopia, October 2012
2. C2: **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
1. C1: **Lomotey, R. K.**, and Deters R. 2012. Using a Cloud-Centric Middleware to Enable Mobile Hosting of Web Services. The 9th International Conference on Mobile Web Information Systems, Niagara Falls, Ontario Canada, Procedia Computer Science, Volume 10, 2012, Pages 634-641, ISSN 1877-0509, 0.1016/j.procs.2012.06.081.